# Digitalna transformacija prometnih i logističkih sustava

**Kundih, David**

Sveučilište Sjever

**Diplomski rad br. 116/OMIL/2022**

# Digitalna transformacija prometnih i logističkih sustava

**David Kundih, 2544/336**

Koprivnica, rujan 2022. godine

Sveučilište Sjever

ODJEL ZA LOGISTIKU I ODRŽIVU MOBILNOST

**Diplomski rad br. 116/OMIL/2022**

# Digitalna transformacija prometnih i logističkih sustava


**Student**

David Kundih, 2544/336

**Mentor**

prof. dr. sc. Krešimir Buntak



Koprivnica, rujan 2022. godine

**Sveučilište Sjever**
Sveučilišni centar Koprivnica
Trg dr. Žarka Dolinara 1, HR–48000 Koprivnica

# Prijava diplomskog rada

## Definiranje teme diplomskog rada i povjerenstva

ODJEL Odjel za logistiku i održivu mobilnost

STUDIJ diplomski sveučilišni studij Održiva mobilnost i logistika

PRISTUPNIK David Kundih | MATIČNI BROJ 2544/336

DATUM 14.07.2022. | KOLEGIJ Upravljanje poslovnim procesima u logistici

NASLOV RADA Digitalna transformacija prometnih i logističkih sustava

NASLOV RADA NA ENGL. JEZIKU Digital transformation of traffic and logistics systems

MENTOR dr. sc. Krešimir Buntak | ZVANJE redovni profesor

ČLANOVI POVJERENSTVA
1. doc.dr.sc. Predrag Brlek, predsjednik
2. doc.dr.sc. Ivana Martinčević, članica
3. prof.dr.sc. Krešimir Buntak, mentor, član
4.
5.

## Zadatak diplomskog rada

BROJ 116/OMIL/2022

OPIS

Digitalna transformacija poslovanja postala je imperativ u svim gospodarskim granama i područjima, pa tako i u području prometnih i logističkih sustava.
U radu je potrebno istražiti mogućnosti digitalne transformacije prometnih i logističkih sustava kroz primjenu algoritama kao potpore odlučivanju. Pobliže pojasniti pojmove logistike i prometa, kao i njihov međuodnos. Demonstrirati alate strateškog upravljanja i upravljanja logističkim procesima. Pojasniti temeljne pojmove iz domene prometa i dosege u području znanosti. Prikazati elemente digitalne transformacije kojima se nastoji stvoriti dodana vrijednost prometnih i logističkih procesa. Predstaviti programiranje u programskom jeziku Python kroz pojašnjenje sintakse, tipova podataka i ostalih specifičnosti programskog jezika. Izdvojiti i detaljnije opisati ključne elemente digitalne transformacije. Navesti metode ekstrakcije znanja iz podataka i njihovu primjenu u simulacijskim modelima i algoritmima strojnog učenja. Etički preispitati uvođenje novih tehnologija u sfere ljudskog života i gospodarstva. Objediniti elemente prethodnih poglavlja i prezentirati digitalnu transformaciju procesa primjenom IDEF0 metodologije.

ZADATAK URUČEN 29.8.2022 | POTPIS MENTORA

SVEUČILIŠTE SJEVER

# Predgovor

Ovaj rad dokazao mi je da nikada nije prekasno za naučiti nove vještine i nadograditi postojeće. Predznanje iz programiranja nadogradio sam kroz primjenu Pythona, te povezao područja prometa i računarstva, što mi je prije samo dvije godine djelovalo nezamislivo i teško dostižno.

Godinama se bavim fotografijom i grafičkim dizajnom, a programiranje mi je omogućilo veći doseg u realizaciji kreativnih ideja. Potaknut razmišljanjima o utjecaju umjetne inteligencije na svijet sadašnjosti i budućnosti, pročitao sam knjige koje ne pripadaju stručnoj literaturi, *The age of surveillance capitalism* Shoshanne Zuboff i *1984* Georgea Orwella. Spomenute knjige vjerno prikazuju utjecaj ideologije na čovjeka, pri čemu medijski i tehnološki giganti posebno utječu na javno mnijenje, a da nije bilo ovog rada, upitno je bih li ih ikada pročitao.

Interes za znanošću javlja mi se još u srednjoškolsko doba, odnosno, u Graditeljskoj školi Čakovec. Važnu ulogu u mom osobnom i profesionalnom razvoju imaju profesori spomenute škole koji su me naučili promatrati svijet oko sebe izvan okvira prihvatljivog i predvidivog, pri čemu se ponajviše zahvaljujem Krunoslavu Bediju, Ljiljani Ille, Ivanu Čondoru, Matiji Vargi i mentoru vlastitog završnog rada Dariu Štokiću. Potaknut predavanjima i brojnim inspirativnim mislima profesora matematike te izvrsnog šahista, Franje Jančikića, upisujem fakultet. Odabrano Sveučilište Sjever pokazalo se kao autonomno i suvremeno sveučilište, otvoreno za nove ideje i spoznaje u znanosti gdje je sve podložno preispitivanju i raspravi. Protekle dvije godine studiranja broje puno uspona i padova, no važno je nastaviti boriti se za ono u što čovjek uistinu vjeruje.

Zahvaljujem se svima onima koji su mi posljednje dvije godine studiranja na sveučilišnom diplomskom studiju Održive mobilnosti i logistike učinili nezaboravnim i poučnim iskustvom. Zahvalu zaslužuje i g. Ante Klečina, čija su predavanja van i unutar prostorija Sveučilišta imala veliki utjecaj na moje poimanje održive mobilnosti i prometnih sustava općenito.

Posebno se zahvaljujem mom mentoru završnog i diplomskog rada na Sveučilištu Sjever, profesoru Krešimiru Buntaku koji me je usmjerio na razvoj i predanost radu, te čija predavanja otvaraju sasvim nove vidike. Kroz proteklih pet godina često navodi primjere liderstva, ali smatram da je upravo vlastitim djelovanjem taj pojam najbolje opisao i dao mu utjelovljenje.

Na samome kraju, zahvaljujem se obitelji i rodbini na podršci kroz sve etape dosadašnjeg i budućeg školovanja, jer bez njih ovaj doseg djeluje nezamislivim. Njihova podrška bila je presudna u mojoj ustrajnosti za postizanjem ciljeva i stvaranja novih, još izazovnijih.

# Sažetak

U ovom se radu istražuju mogućnosti digitalne transformacije prometnih i logističkih sustava kroz primjenu algoritama kao potpore odlučivanju. U uvodnom se dijelu rada pobliže pojašnjavaju pojmovi logistike i prometa, kao i njihov međuodnos. Demonstrirani su alati strateškog upravljanja i upravljanja logističkim procesima. Pojašnjeni su temeljni pojmovi iz domene prometa, odnosno, održiva mobilnost, razvoj temeljen na tranzitu i inteligentni transportni sustavi. Nakon uvodnog dijela o logistici i prometu, razrađeno je tematsko područje digitalne transformacije. Prikazani su elementi digitalne transformacije kojima se nastoji stvoriti dodana vrijednost prometnih i logističkih procesa. Za potrebe razumijevanja narednih poglavlja predstavljeno je tematsko područje programiranja u programskom jeziku Python kroz pojašnjenje sintakse, tipova podataka i ostalih specifičnosti programskog jezika. Izdvojeni su i detaljnije opisani ključni elementi digitalne transformacije, odnosno, umjetna inteligencija, strojno učenje i znanost o podacima . U spomenutim poglavljima navedene su metode ekstrakcije znanja iz podataka i njihova primjena u simulacijskim modelima te algoritmima strojnog učenja. Naglašava se potreba etičkog preispitivanja uvođenja spomenutih tehnologija u sfere ljudskog života i gospodarstva. U posljednjem dijelu rada objedinjeni su elementi prethodnih poglavlja i prezentirana je digitalna transformacija procesa utvrđivanja prometnog stanja na raskrižju.



**Ključne riječi:** promet, logistika, inteligentni transportni sustavi, ITS, digitalna transformacija, umjetna inteligencija, algoritmi strojnog učenja, Python

# Abstract

This paper explores the possibilities of digital transformation of traffic and logistics systems, using algorithms to support decision making. The introductory part examines the logistics and traffic terminology, as well as their correlation. The tools of strategic management and the management of logistics processes are demonstrated. Traffic terminology is clarified through the research of sustainable mobility, transit-oriented development and intelligent transport systems. Following the introductory part about the traffic and logistics, the thematic field of digital transformation is being examined and researched. The elements of digital transformation that aim to create an added value of the traffic and logistics processes are presented. For the purpose of better understanding the following chapter contents, Python programming syntax, data types and other programming language specificities are demonstrated in an individual chapter. The key elements of digital transformation, those being artificial intelligence, machine learning and data science, are singled out and described into detail. The mentioned chapters contain the methods of extracting knowledge from the data, as well as the usage of data in simulation models and machine learning algorithms. The usage of mentioned digital technologies is put into the ethical perspective of humankind and the economy. The last chapter of the paper binds all the elements from the previous chapters to present a digital transformation of the process of determining the traffic situation at the observed intersection.



**Keywords:** traffic, logistics, intelligent transport systems, ITS, digital transformation, artificial intelligence, machine learning algorithms, Python

# Popis korištenih kratica

| | |
|---|---|
| **3D** | Trodimenzionalnost |
| **5G** | Mobilna mreža 5. generacije |
| **AI** | Umjetna inteligencija, eng. Artificial intelligence |
| **AR** | Pojačana stvarnost, eng. Augmented reality |
| **BBC** | British broadcasting corporation, britanska televizijska kuća |
| **BCG** | Boston Consulting Group, organizacija |
| **BI** | Poslovna inteligencija, eng. Business intelligence |
| **BMI (ITM)** | Indeks tjelesne mase, eng. Body mass index |
| **CLI** | Command line interface, vrsta aplikacije |
| **CRA** | Chinese room agreement, eksperiment |
| **CRISP-DM** | Cross-industry standard process for data mining, metodologija |
| **CRM** | Upravljanje odnosima s klijentima, eng. Customer relationship management |
| **csv** | Ekstenzija vrijednosti odvojenih zarezom |
| **DS** | Znanost o podacima, eng. Data science |
| **EOQ** | Economic order quantity, pokazatelj |
| **ETL** | Extract, transfer, load, metodologija |
| **ETOP** | Environment, Threat and Opportunity Profile, strateški alat |
| **EUR** | Euro, valuta |
| **GDPR** | General Data Protection Regulation, regulacija o podacima propisana od strane Europske unije |
| **GPL** | Licenca opće namjene, eng. General purpose license |
| **HRK** | Hrvatska kuna, valuta |
| **html** | Hypertext markup language, ekstenzija |
| **HŽ** | Hrvatske željeznice |
| **IaaS** | Infrastruktura kao usluga, eng. Infrastructure as a service |
| **IEC** | Međunarodna komisija za elektrotehniku |
| **IOT** | Internet stvari, eng. Internet of things |
| **ipynb** | Interactive Python notebook, ekstenzija |
| **ISO** | Međunarodna organizacija za standardizaciju |
| **ITS** | Inteligentni transportni sustavi |
| **ITU** | Integrirana teritorijalna ulaganja |
| **jpg** | Joint photographic experts group, ekstenzija |

| | |
|---|---|
| **json** | Ekstenzija programskog jezika JavaScript |
| **kHz** | kilohertz, mjerna jedinica |
| **km/h** | Kilometara po satu, mjerna jedinica |
| **LISP** | List processing, programski jezik |
| **LRT** | Laka željeznica, eng. Light rail transit |
| **md** | Markdown, ekstenzija zapisivanja |
| **MHz** | Megahertz, mjerna jedinica |
| **MIT** | Massachusetts Institute of Technology |
| **ML** | Strojno učenje, eng. Machine learning |
| **MR** | Mješovita realnost, eng. Mixed reality |
| **NaN** | Oznaka vrijednosti koja nije broj, eng. Not a number |
| **NFC** | Komunikacija u bliskom polju, eng. Near-field communication |
| **NFT** | Non-fungible token, varijacija blockchain tehnologije |
| **OOP** | Objektno orijentirano programiranje |
| **OS** | Operacijski sustav |
| **PaaS** | Platforma kao usluga, eng. Platform as a service |
| **PDCA** | Plan-Do-Check-Act, metodologija |
| **pdf** | Ekstenzija portabilnog formata datoteke |
| **PEP 8** | Python enhancement proposal, smjernice za pisanje programskog koda |
| **PEST** | Political, Economic, Social and Technological, strateški alat |
| **PGDP** | Prosječni godišnji dnevni promet, pokazatelj |
| **png** | Portable network graphics, ekstenzija |
| **py** | Ekstenzija programskog jezika Python |
| **PyPi** | Python package index, upravitelj paketima |
| **pyw** | Ekstenzija programskog jezika Python |
| **QR** | Quick response, tehnologija |
| **ReLU** | Rectified linear unit, aktivacijska funkcija |
| **RFID** | Identifikacija radio frekvencija, eng. Radio frequency identification |
| **ROI** | Povrat na investiciju, eng. Return on investment |
| **RPi** | Raspberry Pi, računalo |
| **RTC** | Komunikacija u realnom vremenu, eng. Real-time communication |
| **SAD** | Sjedinjene Američke Države |
| **SARS-CoV-2** | Severe acute respiratory syndrome coronavirus 2, respiratorna bolest |
| **SaaS** | Softver kao usluga, eng. Software as a service |
| **SHA** | Secure hash algorithm, algoritam enkripcije |

| | |
|---|---|
| **SUMP (POUM)** | Plan održive urbane mobilnosti |
| **SWOT** | Strenghts, Weaknesses, Opportunities and Threats, strateški alat |
| **tanh** | Hyperbolic tangent, aktivacijska funkcija |
| **txt** | Tekstualna ekstenzija |
| **TOD** | Razvoj temeljen na tranzitu, eng. Transit-oriented development |
| **TOWS** | Threats, Opportunities, Weaknesses and Strenghts, strateški alat |
| **UI** | Korisničko sučelje, eng. User interface |
| **USD** | Dolar, valuta |
| **UX** | Korisnički doživljaj, eng. User experience |
| **VR** | Virtualna stvarnost, eng. Virtual reality |
| **VRIO** | Value, Rarity, Imitability and Organization, strateški alat |
| **xlsx** | Ekstenzija programa Excel |
| **XR** | Proširena realnost, eng. Extended reality |
| **ZET** | Zagrebački električni tramvaj |

# Sadržaj

# 1. Uvod

Prometni i logistički sustavi sastavni su dio funkcioniranja gospodarstva bilo kojeg područja. Promet omogućava razmjenu dobara iz cijelog svijeta i preduvjet je procesa globalizacije. Uloga prometa posebno je značajna u osiguravanju socijalne inkluzije ranjivih skupina u društvu te kao takav mora biti u interesu nacionalnih, ali i regionalnih te lokalnih politika i strategija.

Održiva mobilnost opisuje razvoj prometnih sustava na održiv i smislen način kroz poticanje integriranog javnog prijevoza, uspostave taktnog voznog reda i promjene modalnih omjera pojedinih vrsta vozila u sustavu. Brojni trendovi današnjice ukazuju na masovno naseljavanje gradova i poticanje divlje urbanizacije, čime se postavljaju izazovi za urbano planiranje u kojem značajnu ulogu imaju prometni inženjeri i stručnjaci. Ključ uspjeha prometnih rješenja i razvoja gusto naseljenih područja leži u suradnji prometnih inženjera i stručnjaka sa urbanistima, građevinskim inženjerima, sociolozima i stručnjacima brojnih drugih struka koji se bave razvojem održivog sustava. Primjer planiranja sustava temeljenog na potrebama za mobilnosti prepoznaje se u razvoju orijentiranom na tranzit (TOD) gradova i regija.

Dugogodišnjim razvojem logistike kao struke i znanstvenog područja, ona nadilazi tradicionalno razmišljanje koje se vezuje uz vojsku i skladištenje. Logistika postaje sastavnim dijelom svih aktivnosti vezanih uz organizaciju i upravljanje procesima. Logistički koncepti primjenjivi su na uređenje sustava i osiguravanje sklada te ujednačenog djelovanja usmjerenog prema zajedničkom cilju. Transcendencija logistike kao znanosti iz originalne prometne domene logičan je slijed njezina razvoja s obzirom na kompleksnost upravljanja operacijama koje je potrebno harmonizirati s ciljem ispunjenja zahtjeva dionika.

Razvoj prometnih i logističkih sustava uključuje i simbiozu s dosezima tehnologija današnjice, pri čemu se prirodno nameće potreba za njihovom digitalnom transformacijom. Uspješnost projekata inteligentnih transportnih sustava prepoznaje se u integraciji usluge javnog prijevoza s naprednim informacijskim rješenjima, kojima korisnik usluge u realnom vremenu prati stanje i putanju vozila koja prometuju na određenoj relaciji. Utvrđivanje prometnog stanja postaje sve dostupnijim zbog primjene algoritama strojnog učenja u detekciji vozila i brojanju prometnih dionika, kao i prepoznavanju njihovog smjera kretanja.

Uvođenje novih tehnologija radikalno mijenja dosadašnji način izvršavanja poslova i postavlja brojna etička pitanja. Postoji bojazan od preuzimanja radnih mjesta od strane robota, čiju opravdanost treba potvrditi, ali nedvojbeno je da se razvojem tehnologije kreiraju i nova radna mjesta te postoji potreba za stručnjacima u poslovima koji do jučer još nisu ni postojali.

Digitalnu transformaciju može se promatrati i kao stvaranje dodane vrijednosti iz uobičajenog rada, što se prepoznaje u ekstrakciji znanja iz prikupljenih podataka poslovnih i ostalih procesa.

## 1.1. Predmet istraživanja

Predmet istraživanja je ispitivanje i dokazivanje mogućnosti primjene digitalne transformacije u kontekstu prometnih i logističkih sustava. Radom se nastoji generirati optimizacijska rješenja i rješenja temeljena na autonomnom djelovanju uređaja, specifičnih za prometne i logističke sustave, prema zahtjevima i izazovima koje postavlja Industrijska revolucija 4.0.

## 1.2. Problem istraživanja

Problem istraživanja prikazuje se kroz identificiranu tromost prometnih i logističkih usluga u Republici Hrvatskoj u odnosu na razvijena gospodarstva koja aktivno primjenjuju znanstvene i istraživačke dosege iz područja digitalne transformacije, čineći ih pritom personaliziranima i relevantnima za krajnje korisnike.

Korisnici usluge javnog prijevoza izvan nacionalnog i većih regionalnih središta u Republici Hrvatskoj bivaju zakinuti u stupnju informiranosti o linijama i aktualnom stanju lokacije vozila koje očekuju na stajalištu, što može rezultirati nezadovoljstvom i neuspjehom pri provođenju strategija uravnoteženog i decentraliziranog razvoja zemlje.

Utvrđivanju prometnog stanja često se pristupa uz nedostatak suvremene tehnologije, a promatranja se vrše izlaskom ljudi na teren. Sposobnost ljudskog zapažanja u odnosu na tehnologiju može biti subjektivno i manje efikasno s obzirom na koncentriranost na ograničen broj parametara koje ljudski mozak može procesuirati u realnom vremenu, dok računalo istovremeno može izvršavati niz operacija i promatranja.

## 1.3. Hipoteza istraživanja

Postavljena je jedna hipoteza rada kojom se nastoji odgovoriti na problem istraživanja.

**H1: Digitalnom transformacijom prometnog ili logističkog procesa generira se širi niz izlaznih vrijednosti u odnosu na klasični proces bez primijenjene digitalne transformacije.**

Hipotezom se nastoji dokazati ili opovrgnuti opravdanost digitalne transformacije procesa, promatrajući pritom obujam i raznolikost izlaznih vrijednosti prije i nakon njegove transformacije.

## 1.4. Izvori podataka

Za potrebe izrade rada koristi se relevantna znanstvena literatura iz područja prometa, logistike, menadžmenta, ekonomije, računarstva i informacijskih tehnologija te iz etike, sociologije i filozofije. U izvore podataka teorijskog dijela ubrajaju se knjige, e-knjige, znanstveni i stručni časopisi te internetski izvori iz repozitorija, članaka i ostalih baza. Za potrebe izrade simulacijskih

modela i modela strojnog učenja u praktičnom dijelu rada koriste se javno dostupni podaci na platformama predviđenim za tu namjenu, a u slučajevima oskudnosti javno dostupnih podataka koriste se samostalno generirani podaci kojima se nastoji reprezentirati realne odnose.

## 1.5. Metode istraživanja

Korištene metode istraživanja u velikoj su mjeri podudarne s onima koje navodi Zelenika (2000), a od kojih su izravno primijenjene metoda analize, metoda sinteze, metoda dokazivanja te induktivna i deduktivna metoda.

## 1.6. Struktura rada

Rad se sastoji od ukupno devet temeljnih poglavlja. Rad započinje uvodnim poglavljem, a završava zaključkom s pripadajućim popisom literature, slika, grafikona i tablica te dodacima izraženim u obliku programskog koda i vizualne prezentacije postupka izrade simulacija i modela strojnog učenja. Prvi dio rada proučava teorijske aspekte prometa i logistike, kao i njihovu međusobnu povezanost te ulogu koju imaju u suvremenom društvu. Navode se znanstveni dosezi u proučavanju prometnih i logističkih sustava te se navode metode za efektivno upravljanje cjelokupnim sustavima i njihovim sastavnim elementima.

Teorijski dio rada obuhvaća i istraživanje o pojmovima digitalne transformacije poput tehnologije lanca blokova, proširene stvarnosti, pametnih tvornica, velikih podataka i sličnog. Kao uvod u praktični dio rada prikazuju se načela i elementi programiranja u programskom jeziku Python s primjerima u području prometnih i logističkih sustava. Iz domene digitalne transformacije posebno se izdvaja umjetna inteligencija i njezini pripadajući elementi, te se oni pobliže opisuju. Istražuje se utjecaj umjetne inteligencije na gospodarstvo i odnose u društvu, izdvajaju se rizici eksploatacije podataka i upotrebe u svrhe masovnog nadzora te potencijalne težnje distopijskoj stvarnosti.

Praktični dio rada objedinjuje primjenu pojedinih tehnologija digitalne transformacije u prometnim i logističkim sustavima. Opisuju se postupci usklađenja, popunjavanja i filtriranja podataka prije upotrebe u simulacijskim rješenjima. Korištenjem programskog jezika Python kreiraju se modeli strojnog učenja i simulacijski modeli procjene opterećenja prometnice u nadolazećem periodu, grupiranja pošiljaka logističkog poduzeća, procjene cijene transporta u odnosu na podatke konkurenata, predviđanja raspona broja vozila na raskrižju u satu i danu godine te pronalaska optimalne putanje pri distribuciji pošiljaka. Zaključno se predlaže rješenje digitalne transformacije procesa utvrđivanja prometnog stanja na raskrižju te se izdvajaju koristi i prepreke nad klasičnim procesom.

# 2. Logistika

U ovom se poglavlju pobliže definira logistika kroz njezinu povijest, značajke, etimologiju i korijen značenja te ulogu u suvremenom društvu i poslovanju. Nastavno na logistiku, detaljno se analizira podjela logističkih sustava i primjena procesnog pristupa pri njihovim upravljanjem.

## 2.1. Povijest i razvoj logistike

Korijen riječi logistike nije sasvim jasno definiran s obzirom na etimološku sličnost pojmova, a prema Kolingeru (2013) postoji niz mogućih korijena riječi 'logistika'; iz francuskog naziva *logistique* što je izvedeno iz naziva dočasničkog čina francuske vojske u 17. stoljeću *Marechal de logis* u čijem je djelokrugu planiranje administrativnih poslova vezanih uz pomak snaga, grčke riječi *logos*, odnosno znanosti o principima i oblicima pravilnog mišljenja i prosuđivanja i *logistikos* što podrazumijeva 'vještine, iskustva i znanja o očuvanju, procjeni i prosudbi svih relevantnih elemenata u prostoru i vremenu potrebnih za optimalno rješavanje strateških i taktičkih zadataka u svim područjima ljudskih aktivnosti' (Kolinger, 2013).

Iz navedene etimološke podjele može se zaključiti kako u svim spomenutim terminima postoji direktna povezanost sa djelatnosti logistike s obzirom na to da pokriva širok djelokrug aktivnosti kojima je potrebno koordinirati kako bi se ostvarili ciljevi uz što veću dodanu vrijednost procesa. Prošlost čovječanstva obilježena je ratovanjem i tendencijama za širenje ili obranu teritorija u čemu logistika ima ključnu ulogu što potvrđuju brojni vojni lideri kroz povijest i stoga američki general Prvog i Drugog svjetskog rata navodi da 'Vojnici pobjeđuju bitke, a logistika pobjeđuje rat' (Perish, 2020), dok su se brojni vojni generali poput Napoleona koji je integrirao upravljanje logistikom kao sastavni dio strategije francuske vojske u 17. stoljeću (Schneid, 2005) i Williama Pagonisa u operaciji *Pustinjska oluja* u 20. stoljeću proslavili po ratnim uspjesima upravo na temelju kvalitetnog upravljanja sustavom logistike.

Brojne vojne operacije orijentirane su na prekid opskrbnih lanaca kako bi se ograničila mogućnost opskrbe izvana što podrazumijeva prekid tokova informacija, namirnica, naoružanja i energije čime se dodatno oslabljuje neprijatelj i postavlja u nepovoljan položaj, a neki od primjera su Lenjingrad (St. Petersburg) u Drugom svjetskom ratu u kojemu su upravo zbog prekinutih lanaca opskrbe civili i vojnici bili u nepovoljnom položaju i vrlo često nastradali od posljedica nedostatka vode ili hrane te vojna invazija Rusije na Ukrajinu u 2022. godini u kojoj su primarne mete napada bile zračne luke u Ukrajini pri čemu je uništen najveći avion na svijetu *Antonov AN-225 Mriya*, a zauzete su nuklearne elektrane u Černobilu i Zaporizhzhiju, dok su vojne operacije usmjerene na okupaciju lučkog grada Mariupola čime se kontrolira pomorski transport i prihvat robe na Crnom moru, kao i funkcionalnost hidroelektrana na rijeci Dnjepar kroz okupaciju grada Khersona.

## 2.2. Domene logistike

Logistika kroz svoj povijesni razvoj nadilazi kontekst vojske i primjenjuje se u organizaciji kompleksnih aktivnosti što se očituje i u primjeni na različite domene i njihove specifičnosti među kojima su neke;

- **Medicinska logistika –** obuhvaća prihvat i zbrinjavanje bolesnika, organizaciju i koordinaciju odjela, zbrinjavanje infektivnog otpada prema posebnim standardima i regulativama, definiranje procesa i načina postupanja u kriznim situacijama, optimizaciju postojećih procesa novim tehnologijama i znanjima.
- Značaj medicinske logistike prepoznat je u razdoblju krizne situacije uzrokovane pandemijom SARS-CoV-2 u kojoj je upravo logistika odigrala značajnu ulogu u očuvanju zdravlja stanovništva kroz organizaciju cijepnih punktova uz vođenje evidencije i zbrinjavanje potrošenog medicinskog materijala poput igli, šprica i vata.
- **Građevinska logistika –** obuhvaća dopremu i prihvat materijala na gradilište i njegovo skladištenje i pozicioniranje prema posebnim zahtjevima, organizaciju otpreme građevinskog otpada s gradilišta te zaštita imovine od krađe. Građevinska poduzeća djeluju na terenu zbog čega je također potrebno osigurati mobilne urede i sobe.
- **Poslovna logistika –** osigurava upravljivost poslovnim procesima organizacije kroz sagledavanje efektivnosti i efikasnosti cjelokupnog sustava kroz postizanje optimuma parcijalnih cjelina što posebno naglašavaju Buntak et al. (2005) kroz primjenu procesnog pristupa u organizacijama, a Braniša (2019) navodi da se 'problemi u vezi s kretanjem dobara i informacija ogledavaju se u njihovoj povezanosti s tokovima vrijednosti unutar poduzeća (sva ulaganja proizvodnih sredstava, kadrova, financija, energije, *know-how* i sl.)' iz čega se može zaključiti da je postojanje ili nepostojanje sustava vrijednosti pokretač pozitivnih ili negativnih promjena u poslovanju organizacija i njihovoj konkurentnosti zbog jasne odnosno nejasne usmjerenosti organizacije na dugoročni uspjeh i stvaranje dodane vrijednosti.
- **Logistika kriznih situacija –** preventivno i reaktivno promišljanje mogućih kriznih situacija s ciljem sprječavanja nastanka havarije u pretkriznoj fazi ili umanjivanje materijalne i financijske štete, odnosno zbrinjavanja unesrećenih ljudi i životinja u kriznoj, odnosno postkriznoj fazi.

  Krizne situacije mogu obuhvaćati prirodne katastrofe poput potresa, erupcija vulkana, požara i poplava u kojima nastaje opća panika i poremećaj dotadašnjeg normalnog odvijanja aktivnosti što podrazumijeva potrebu za organizaciju alternativnih pravaca prometa i opskrbe te osiguranje nužnih službi od potrebe stanovništva što može uključivati improvizirane bolničke centre, škole, kontejnerska naselja, restorane itd. Djelovanje u kriznim situacijama

opisano je Priručniku Sfere (eng. 'The Sphere Handbook') u kojem su definirane dnevne ljudske potrebe za određenim namirnicama poput hrane, vode i lijekova.

- **Skladišna logistika** – posebna poslovna funkcija koja osigurava prihvat, procesuiranje i otpremu robe uz uvažavanje posebnih zahtjeva robe ovisno o njezinim specifičnostima te ustrojstvu poduzeća i njegovoj djelatnosti.

  Poduzeća skladišno poslovanje mogu povjeriti trećim stranama (outsourcing) ili ga zadržati kao sastavni dio vlastitog poslovanja. Zbog potrebe za internacionalizacijom i globalizacijom poslovanja, skladišna logistika nadilazi pojam skladišta i postaje sastavni dio nekih od vrsta logističko-distributivnih centara što nadalje potiče razvoj i digitalnu transformaciju s ciljem procesuiranja većeg obujma robe po jedinici vremena.

- **Logistika popisa stanovništva –** organizacija provedbe popisa stanovništva je zahtjevan poduhvat kojim se utvrđuju relevantni statistički podaci neke države pri čemu logistika ima ključnu ulogu kroz definiranje svih potrebnih ljudskih i materijalnih resursa za provedbu te procesuiranje i validaciju prikupljenih podataka svakog stanovnika neovisno o udaljenosti i nepristupačnosti terena.

  Digitalizacija sustava javne uprave postaje imperativom današnjice, prema čemu prednjači Estonija, no postoje i pozitivni pomaci u Hrvatskoj.

- **Logistika događaja ('Event logistika') –** umijeće i kreativnost u primjeni logistike očituje se u organizaciji događaja zbog brojnih izazova koji se postavljaju pred logistički menadžment, naime pojedini događaji mogu se organizirati u otežanim okolnostima poput nepovoljnog terena i zaštićenih zona uz potencijalne prepreke poput nedostupnosti energetske mreže i potrebe za ograničenjem buke prema zakonskim odredbama, a uz spomenuto potrebno je osigurati dostatnost hrane i pića te hitnih službi koje mogu reagirati u slučaju potrebe.

- **Urbana logistika –** obuhvaća niz strateških i operativnih djelatnosti vezanih uz osiguranje funkcionalnosti i održivog razvoja urbanih područja. Urbana područja specifična su po ograničenosti prostora i velikom broju stanovnika što naglašava potrebu za adekvatnim odgovorom lokalnih vlasti kroz osiguranje komunalne i prometne infrastrukture s ciljem očuvanja okoliša i poboljšanja kvalitete života.

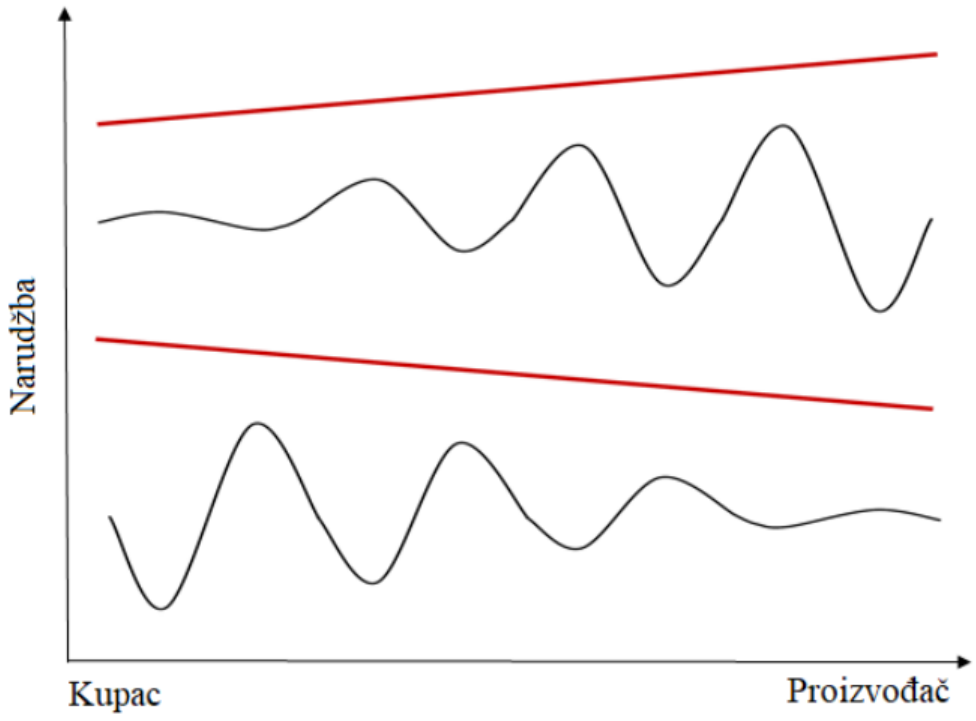
  Uz spomenuto, urbana područja pogodna su za investicije zbog ekonomije obujma i ekonomije razmjera, odnosno obuhvat ponuđenih usluga pokriva veći broj potencijalnih korisnika u odnosu na broj potencijalnih korisnika na jednakoj veličini površine u ruralnim područjima. Ulaskom Republike Hrvatske u Europsku uniju otvorena je mogućnost sufinanciranja projekata iz ITU mehanizma (Integrirana teritorijalna ulaganja) što je pogodno za urbana područja zbog svog obuhvata i utjecaja na gospodarstvo.

## 2.3. Logistika kao konkurentska prednost

Suvremeno društvo naviknuto je na dostatnost roba i usluga u svakodnevnom životu zbog globalizacije i manje ovisnosti o isključivo nacionalnom gospodarstvu, a preduvjet za spomenuto je upravo efektivnost i efikasnost logističkih procesa i upravljanja lancima opskrbe. Brojna poduzeća poput *Amazona* prepoznaju utjecaj logističke usluge na zadovoljstvo korisnika i njihovih specifičnih vremenskih zahtjeva za isporukom, odnosno zaprimanjem naručenih proizvoda u što skorijem roku.

Prema Zekić (2018) uloga menadžera lanaca opskrbe je 'strukturiranje dogovorenog oblika integracije i provedbe upravljačkih aktivnosti da bi se postigla konkurentna brzina odgovora i potrebna fleksibilnost lanca kao poslovnog sustava.'

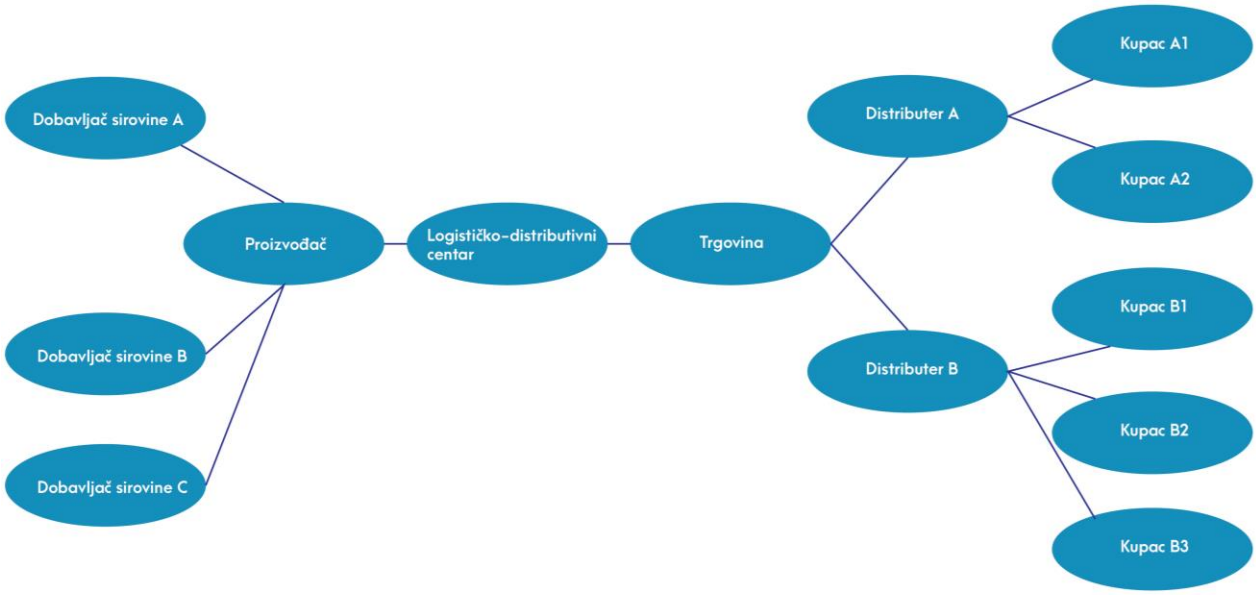
Termin koji opisuje težinu upravljanja lancima opskrbe očituje se kroz tzv. efekt biča (eng. 'bullwhip effect' / 'whiplash effect') kojeg je prema Jakopić (2021) prvi upotrijebio J. W. Forrester u knjizi *Industrial Dynamics* u 1961. godini, a temelji se na tome da su 'podaci podložni varijacijama i distorzijama te se to izobličavanje širi uzvodno u pojačanom obliku što navodi uzvodne članove lanca opskrbe na donošenje loših odluka' (Jakopić, 2021).




*Slika 2.1 'Efekt biča i obrnutog efekta biča',*
*Izvor: https://repozitorij.fsb.unizg.hr/islandora/object/fsb%3A6673/datastream/PDF/view*

Efekt biča također se sagledava i kroz poremećaj ranijih operacija u opskrbnom lancu koje povećavaju neizvjesnost izvedivosti operacija narednih faza lanca opskrbe u planiranim rokovima.

Zbog sve veće kompleksnosti lanaca opskrbe i uključenosti brojnih dionika u proces koristi se i naziv mreža opskrbe. Razliku između lanca opskrbe i mreže opskrbe čini više ulaznih varijabli u svakoj karici uključenoj u logistički proces kretanja dobara i usluga pri mreži opskrbe, dok je lanac opskrbe prikazan linearno sa samo jednom ulaznom varijablom.




*Slika 2.2 'Prikaz pojednostavljene mreže opskrbe',*
*Izvor: Vlastiti rad autora*

## 2.4. Logistički sustavi i okoline

Logistički sustavi opisuju kompleksnost logističkog planiranja i pripadajućih logističkih operacija kroz sustavni pristup koji se temelji na razgranatosti na stupnjevane razine.

Zelenika (2008) prema Vukovskom (2019) logistički sustav opisuje kao 'sustav međusobno, svrsishodno povezanih i međuutjecajnih podsustava i elemenata koji, pomoću logističke infrastrukture, logističke suprastrukture, logističkoga intelektualnoga kapitala i drugih potencijala i resursa, u visokosofisticiranoj logističkoj industriji omogućuju uspješnu, učinkovitu i racionalnu proizvodnju logističkih proizvoda.'

Logistički sustavi temelje se na teoriji sustava, što podrazumijeva sljedivost načela i politika kroz sve razine sustava kako bi se postigla usklađenost i težnja ostvarenju zajedničkih ciljeva. Ključna pretpostavka teorije sustava počiva na pretpostavki jednakih pravila na svim razinama, što znači da pravila više razine automatski podrazumijevaju primjenjivost na nižim razinama kao dijelovima istog sustava.

Sustavi nisu izolirane cjeline koje djeluju u neovisnosti o okolini, već su pod direktnim utjecajem različitih okolina koje mogu promijeniti tijek odvijanja aktivnosti i otežati dugoročno planiranje.

Lozina (1994) za okolinu sustava navodi da ona 'predstavlja onaj element koji može proizvesti značajne promjene unutar sustava, bez obzira o kojem tipu sustava načelno govorimo.'


*Slika 2.3 'Organizacija i organizacijska okolina',*
*Izvor: https://zir.nsk.hr/islandora/object/unin%3A2913/datastream/PDF/view*

Zelenika (2005) vrši podjelu općih (univerzalnih) logistički sustava na:

- megalogistički sustav,
- globalnologistički sustav,
- makrologistički sustav
- mikrologistički sustav,
- metalogistički sustav,
- informacijskologistički sustav,
- interlogistički sustav,
- intralogistički sustav,
- servisnologistički sustav,
- menadžmentskologistički sustav,
- gospodarskologistički sustav
- i ostale logističke sustave.

Buble et al. (2005) opisuju podjelu organizacijskih okolina na:

- internu ili unutarnju okolinu,
- poslovnu okolinu ili okolinu zadatka,
- opću ili socijalnu okolinu.

Unutar konteksta logističkih sustava, okolinu čine svi čimbenici koji okružuju sustav i pripadaju nekoj od spomenutih okolina. Logistička poduzeća imaju potrebu promatrati promjene u okolinama s ciljem adekvatnog reagiranja na moguće rizike koji mogu negativno utjecati na odvijanje aktivnosti. Jedna od metoda preventivnog djelovanja na rizike iz okolina je simuliranje mogućih događaja i priprema kriznog plana kroz preventivnu i reaktivnu fazu mitigacije rizika i njihovih negativnih učinaka.

### 2.4.1. Interna (unutarnja) okolina logističkih sustava

'Interna ili unutarnja okolina još se naziva i „okolinom poduzeća“ jer je, za razliku od eksterne ili vanjske okoline na koju poduzeće redovito ne može djelovati, u potpunosti pod utjecajem poduzeća.' (Buble et al., 2005.)

Interna se okolina odnosi na sustav vrijednosti logističke organizacije koji može djelovati regulatorno ili motivacijski na procese i njihove dionike unutar organizacije. Interna okolina organizacije definira se vizijom, misijom, ciljevima i politikama koje služe kao temelj za upravljanje, planiranje i provedbu organizacijskih zadataka. Sustav vrijednosti kreira sklad unutar sustava i težnju za ostvarenjem zadanih ciljeva, dok organizacija bez sustava vrijednosti nejasno definira svoju ulogu i težnju za razvojem.

### 2.4.2. Poslovna okolina (okolina zadatka) logističkih sustava

'Poslovnu okolinu zadatka (mikrookolinu) čine akteri u neposrednoj okolini poduzeća koji utječu na njegovu sposobnost da tu okolinu opslužuje.' (Buble et al., 2005)

U poslovnu se okolinu ubrajaju svi dionici s kojima organizacija surađuje kako bi ispunila svoje ciljeve i održavala poslovanje stabilnim. Neki od dionika koji se ubrajaju u poslovnu okolinu su kupci, dioničari, dobavljači, financijske institucije i partnerske organizacije. Organizacija je u neposrednom odnosu s dionicima poslovne okoline i teži zadovoljenju njihovih potreba te kreiranju preduvjeta za dugoročnu suradnju.

### 2.4.3. Opća (socijalna) okolina logističkih sustava

Buble et al. (2005) navode da se eksterna okolina vezuje uz pojavu organizacijskog darvinizma, po kojemu se 'opstanak poduzeća vezuje za njegovu sposobnost neprekidnog praćenja promjena i adekvatne prilagodbe promjenama.'

Opća okolina je najširi pojam okoline organizacije koji podrazumijeva niz vanjskih faktora koji mogu utjecati na stabilnost i poslovanje organizacije i na koji organizacija nema izravan utjecaj.

Neki od primjera opće okoline su prirodne katastrofe koje mogu narušiti tijek poslovanja i učiniti raspoložive resurse neuporabljivim, tehnološke promjene koje nameću potrebu za kontinuiranim unaprjeđenjem vlastitih proizvoda i usluga, preferencije potrošača koje se mijenjaju sukladno trendovima i uvjerenjima i slično.

### 2.4.4. Utvrđivanje konteksta organizacije

Utvrđivanje konteksta organizacije vrši se kombinacijom menadžerskih alata kojima se prepoznaju prilike, prijetnje, slabosti i snage organizacije što rezultira informacijama nužnim za kreiranje SWOT matrice i TOWS matrice te odabir adekvatne strategije temeljene na provedenoj analizi. Sam postupak utvrđivanja konteksta organizacije dijeli se na utvrđivanje unutarnjih slabosti i snaga te vanjskih prilika i prijetnji.

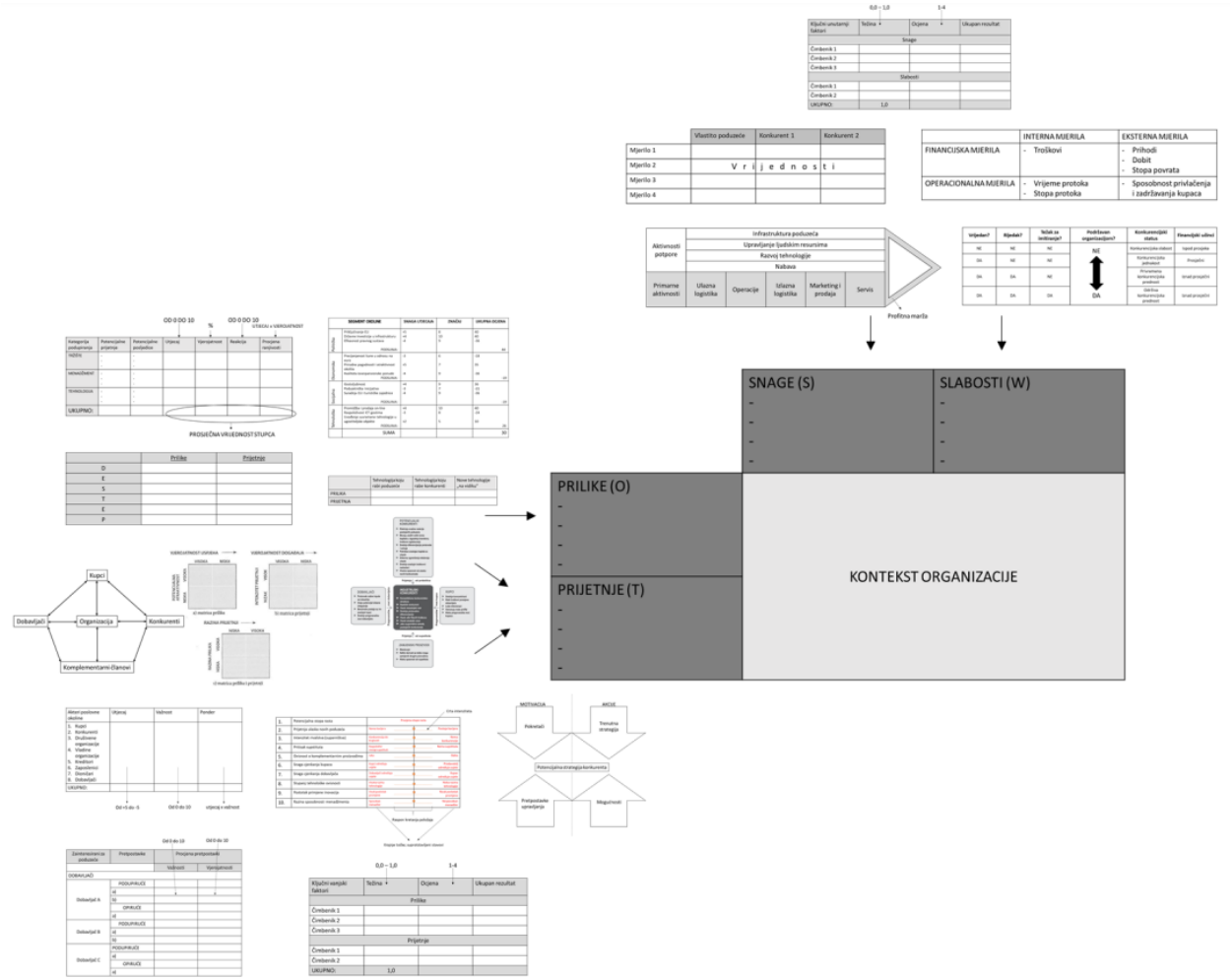
Mutavdžija (2019) navodi sljedeće menadžerske alate kao podlogu za kreiranje SWOT matrice i TOWS matrice te za utvrđivanje konteksta organizacije na temelju analize unutarnjih i vanjskih faktora:

**Unutarnja okolina**

- VRIO okvir,
- BCG matrica,
- 7s model.

**Vanjsko okruženje**

- PEST (PESTEL) analiza,
- Porterovih 5 sila,
- ETOP profil,
- DESTEP analiza.




*Slika 2.4 'Utvrđivanje konteksta organizacije korištenjem menadžerskih alata',*
*Izvor: https://zir.nsk.hr/islandora/object/unin%3A2913/datastream/PDF/view*

Nakon provedene analize menadžerskim alatima, dobivene informacije koriste se za kreiranje SWOT matrice i TOWS matrice kojima se utvrđuju četiri ključne strategije iz kojih proizlazi utvrđivanje konteksta organizacije, a prema Mutavdžiji (2019) to su:

- *Min-Max* strategija (slabosti i prilike),
- *Max-Min* strategija (snage i prijetnje),
- *Max-Max* strategija (snage i prilike),
- *Min-Min* strategija (slabosti i prijetnje).

Navedene strategije opisuju položaj organizacije i ističu poželjan smjer za daljnji razvoj kroz spoznaju o vlastitim mogućnostima sukladno raspoloživim resursima i prilikama iz okoline koje se mogu iskoristiti.

Strateško određivanje organizacije proizlazi iz *Max-Max* strategije, dok se ostale strategije koriste za isticanje mogućih prepreka na koje organizacija može naići pri strateškom određivanju.

## 2.5. Upravljanje poslovnim procesima u logistici

Pretpostavka održivosti bilo kojeg sustava proizlazi iz kvalitete njegova upravljanja. Upravljanje poslovnim procesima u logistici posebno je izazovno zbog zahtjeva nametnutih globalizacijom i potrebom za internacionalizacijom poslovanja što uza sebe nosi određeni stupanj neizvjesnosti i otežanu upravljivost poslovnim procesima.

### 2.5.1. Poslovni procesi i procesna dekompozicija

Poslovni procesi opisuju slijed ponavljajućih aktivnosti koje rezultiraju ostvarenjem zadanih ciljeva korištenjem mehanizama pod određenim pravilima i regulativama s uspostavljenim sustavom mjerenja i definiranim kontrolnim točkama. Poslovne procese nužno je definirati i dokumentirati iz niza razloga koji osiguravaju upravljivost procesima i agilnost organizacije.

Mapiranjem poslovnih procesa jasno se opisuje i definira djelokrug svakog zaposlenika i svakog odjela organizacije kao dijela sustava, što omogućava lakše posredovanje informacijama prema ključnim osobama, definiranje odgovornosti za procese i lakše uvođenje novih zaposlenika u posao.

Mapiranjem poslovnih procesa također se kreira i informacijska podloga koja služi kao polazište nove etape kontinuiranog poboljšanja procesa prema PDCA (*Plan-Do-Check-Act*)[1] načelu, odnosno *Kai-Zen* metodologiji.

Uvođenje procesnog pristupa u poslovanje pokazalo se vrlo uspješnim za vitalnost organizacije i ono je jedan od uvjeta za stjecanje certifikata brojnih normi ISO organizacije.

Drljača (2015) prema Kovačiću (2019) navodi tri vrste poslovnih procesa:

- temeljni procesi,
- upravljački procesi,
- procesi potpore.

**Temeljni procesi –** procesi temeljne djelatnosti organizacije, za koje Buntak et al. (2020) navode da se nazivaju još i radnim procesima, glavnim procesima ili vodoravnim procesima, dok je optimalan broj za organizaciju između pet i osam procesa.

Velik broj temeljnih procesa otežava upravljivost i povećava prisutnost entropije unutar sustava zbog čega je potrebno identificirati, odrediti i dokumentirati sve temeljne procese jer su oni stvarna vrijednost organizacije i za rezultat imaju krajnji proizvod ili uslugu.

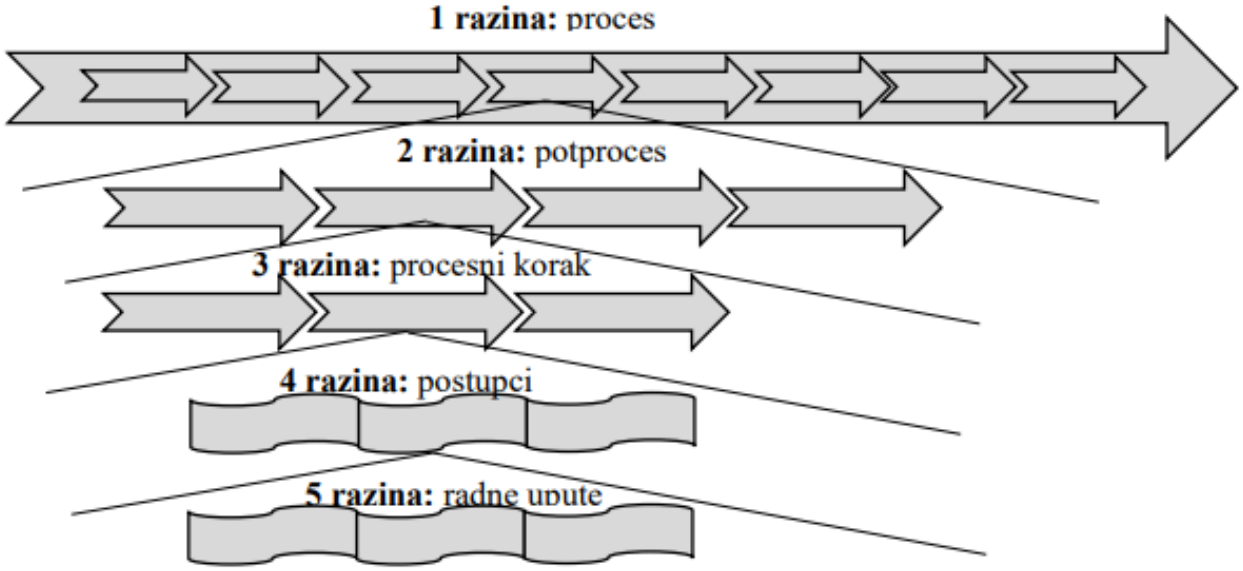
[1] također poznato i kao Demingov krug ili Shewartov ciklus.

**Upravljački procesi –** Prema Buntak et al. (2020) procesi upravljanja utječu na temeljne procese i procese potpore u organizaciji, a kao primjere navode; proces budžetiranja, proces upravljanja razvojem, proces planiranja, upravljanja rizicima i poslovnog odlučivanja.

Upravljački procesi imaju velik utjecaj na temeljne procese i procese potpore jer definiraju okvir poslovanja cjelokupne organizacije te osiguravaju njezinu održivost.

**Procesi potpore –** omogućavaju poslovanje organizacije kroz pružanje podrške u redovitim aktivnostima temeljnog procesa. Buntak et al. (2020) nazivaju ih još i administrativnim procesima, omogućavajućim procesima i resursnim procesima.

Efikasnost i efektivnost temeljnih procesa ovisi o brzini i kvaliteti obavljanja administrativnih poslova unutar djelokruga dionika u procesu potpore.




*Slika 2.5 'Dekompozicija procesa',*

*Izvor: Buntak, K; Sesar,V; Kovačić,M.: Kontroling poslovnih procesa. Quality system condition for successful business and competitiveness. Kopaonik. Srbija, 2016.*

Slika 2.5 prikazuje dekompoziciju procesa na niže razine:

- potprocese,
- procesne korake,
- postupke,
- i radne upute.

Dekompozicijom procesa na niže razine detaljnije se opisuje njegova struktura do razine radnih uputa koje u konačnici doprinose ostvarenju zadanih ciljeva kroz operativnu djelatnost.

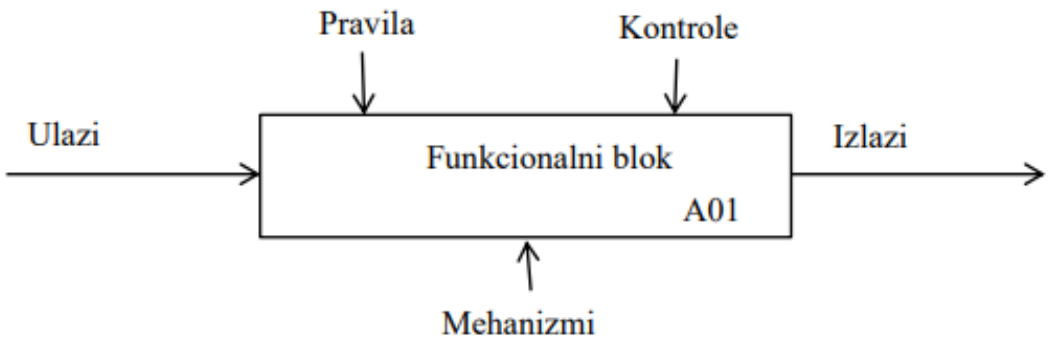
### 2.5.2. IDEF0 dijagram

IDEF0 dijagram jedna je od tehnika kojom je moguće prikazati proces upravljanja poslovnim procesima u logistici uz uvažavanje zakonitosti propisanih samom metodologijom.

Kovačić (2019) navodi da je IDEF0 metodologija 'opće prihvaćeni način prikazivanja i modeliranja procesa koji je standardiziran.'

IDEF0 dijagram opisuje dekompoziciju procesa na niže razine uz uvažavanje pripadajućih:

- **ulaza** – zahtjevi zainteresiranih strana poput zahtjeva menadžmenta i kupaca te općih zahtjeva poput preporuka za poboljšanja radnih uvjeta, ciljeva i sl.
- **pravila –** dokumenti, norme, zakoni, propisi i regulative kojima pojedini proces ili potproces podliježe poput ISO normi, zakona o zaštiti na radu, nacrta, etičkog kodeksa, pravilnika sustava kvalitete, licenci i sl..
- **kontrola** – kontrolne točke koje pružaju povratnu informaciju o stupnju ostvarenja zadanih i propisanih ciljeva procesa ili potprocesa, poput broja obrađenih jedinica u minuti, broju nesukladnosti u danu i sl.
- **mehanizama –** resursi raspoloživi za odvijanje procesa ili potprocesa poput inženjera prometa, laboratorija, obrazaca, infrastrukture, softverskih rješenja i sl.
- **izlaza –** izlazne vrijednosti procesa koje mogu predstavljati dovršen proizvod ili uslugu poput izgrađene ceste ili periodički ostvarene ciljeve kao rezultata potprocesa poput plana izgradnje ceste i ceste u pripremi.

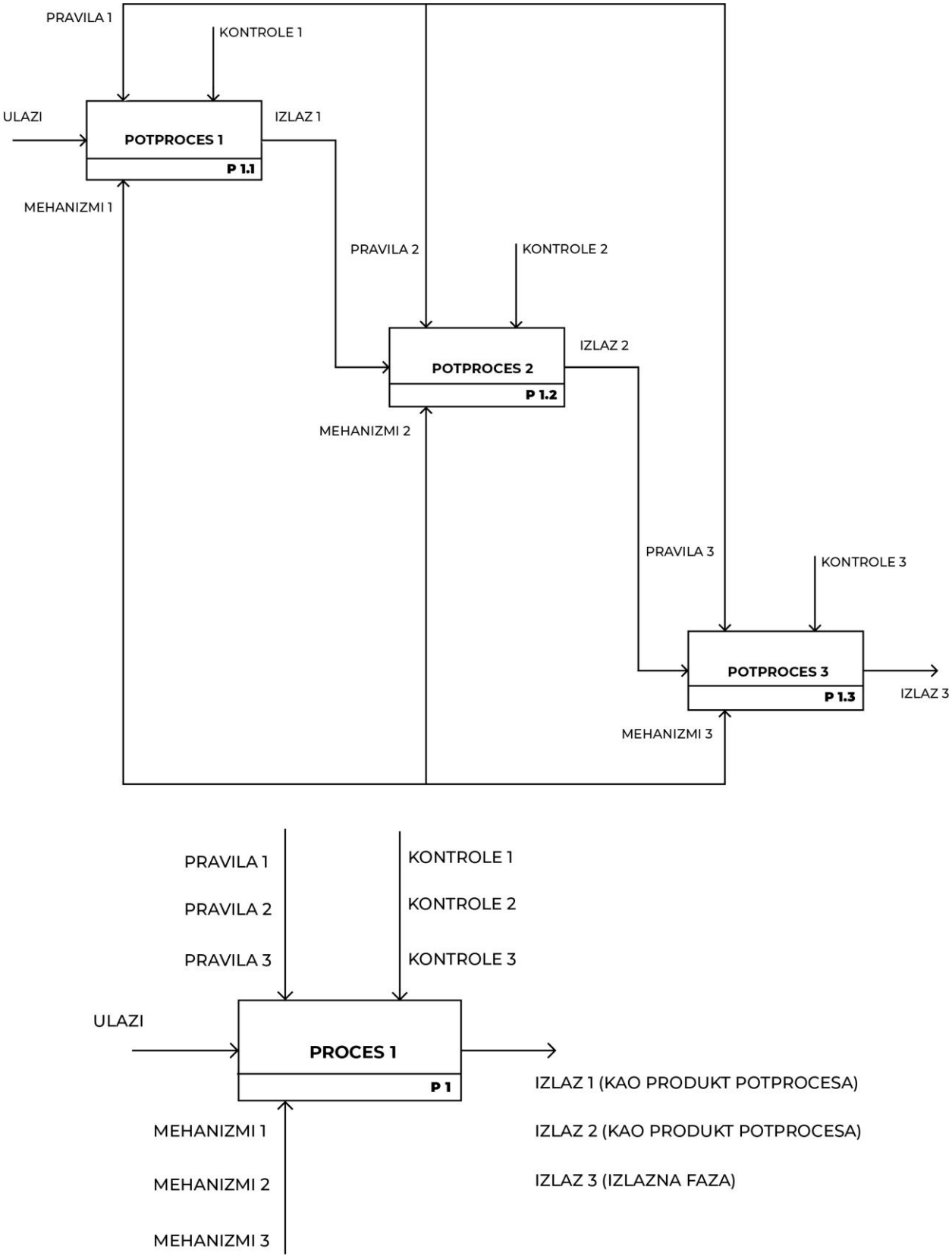



*Slika 2.6 'IDEF0 dijagram konteksta'*

*Izvor: https://zir.nsk.hr/islandora/object/unin%3A2897/datastream/PDF/view*

Slika 2.6 prikazuje shemu dijagrama konteksta kao sastavnog dijela IDEF0 metodologije. Dijagram konteksta vizualno prikazuje sve pripadajuće silnice koje utječu na proces ili potproces. Cjelokupni proces sačinjen od potprocesa prikazuje se dijagramom konteksta, te mu se dodjeljuju sve identificirane potrebe niza potprocesa od kojih je sastavljen.

Slika 2.7 prikazuje postupak dekompozicije procesa primjenom IDEF0 metodologije. Postupak dekompozicije procesa odnosi se na definiranje ulaza, mehanizama, pravila, kontrola i izlaza svakog od potprocesa koji tvore informacijsku podlogu za kreiranje dijagrama konteksta.




*Slika 2.7 'Dekompozicija procesa IDEF0 metodologijom',*

*Izvor: Vlastiti rad autora*

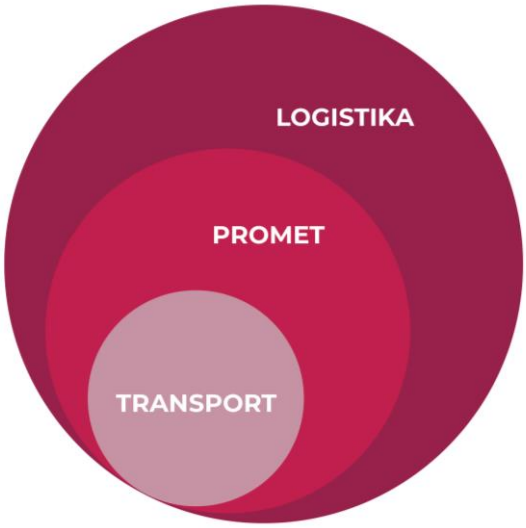
# 3. Promet i mobilnost

U ovom se poglavlju definira promet i njegov međuodnos s logističkim sustavima i ulogom koju ima na gospodarstvo i socijalnu inkluziju. Pobliže su opisani pojmovi integriranog javnog prijevoza, inteligentnih transportnih sustava i održive mobilnosti s konkretnim primjerima.

## 3.1. Prometni sustavi

'Promet je ukupnost različitih prijevoznih, poštansko-telegrafsko-telefonskih usluga, koje kao samostalne gospodarske djelatnosti imaju korisni učinak u premještanju materijalnih dobara, prijevozu ljudi, prijenosu vijesti i izmjeni misli.' (Čavrak, 2003)

Slika 3.1 prikazuje međuodnos logistike, prometa i transporta pri čemu je definirano da su promet i transport sastavni dijelovi logistike i logističkih procesa, a prema Drljači (2020) 'promet obuhvaća transport (prijevoz) robe i putnika pomoću prometne suprastrukture i korištenjem prometne infrastrukture kao i sve operacije i komunikaciju.'



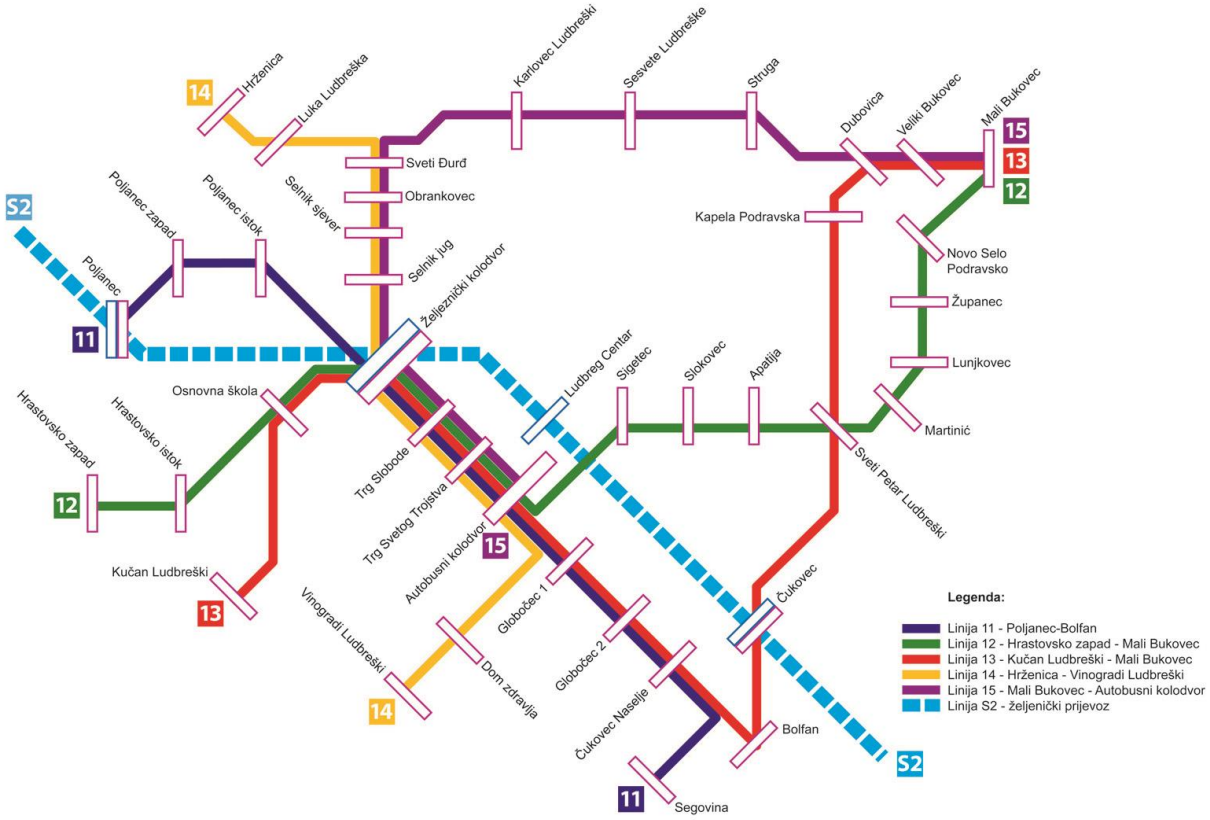

*Slika 3.1 'Međuodnos logistike, prometa i transporta',*
*Izvor: Vlastiti rad autora prema Ivaković Č., Stanković R., Šafran M.(2010): 'Špedicija i logistički procesi', Fakultet prometnih znanosti, Zagreb*

Promet omogućava interakciju međuovisnih elemenata sustava, čime se omogućava usklađenost i usmjerenost prema istom cilju.

Uz prijevozni aspekt prometa koji omogućava mobilnost do zdravstvenih, obrazovnih, kulturnih, zabavnih, sportskih i ostalih usluga, ključan je i informacijsko-komunikacijski promet u koji se također ubraja i internet te ostale telekomunikacijske i poštanske usluge kojima se doseg određene informacije bitno povećava te se smanjuje izoliranost i diskriminacija u ruralnim krajevima kroz povećanje njihove dostupnosti.

## 3.2. Integrirani prijevoz putnika i prostorno-prometno planiranje

Strategije razvoja prometnog sustava na nacionalnoj razini moraju biti usmjerene na smanjenje prometne izoliranosti građana kroz poticanje integriranog javnog prijevoza kombinacijom različitih modova prijevoza. Prometna izoliranost je u velikoj mjeri prisutna u željezničkom prijevozu s obzirom na manju geografsku pokrivenost i dostupnost linija u odnosu na cestovni prijevoz, što za rezultat ima neželjene posljedice na ekonomsku i socijalnu ravnopravnost stanovnika, a posebno je izraženo u ruralnim područjima zbog manje gustoće naseljenosti i slabije financijske isplativosti ulaganja za razliku od urbanih područja u kojima opravdanost investicija proizlazi iz ekonomije razmjera i ekonomije obujma.




*Slika 3.2. 'Prijedlog mreže linija gradsko-prigradskog autobusnog prijevoza za mikroregiju Ludbreg', Izvor: Studija razvoja održivog prometa i mobilnosti grada Ludbrega, 2021.*

Slika 3.2. prikazuje povezivanje usluge željezničkog i autobusnog prijevoza s ciljem integracije prigradskih područja s gradom Ludbregom, kao i primjenu intermodalnosti u prijevozu stanovnika prigradskih područja presjedanjem na putnički vlak na Željezničkom kolodvoru Ludbreg i nastavak putovanja prema regionalnim središtima, Koprivnici i Varaždinu. Spomenuti primjer doprinosi smanjenju prometne izoliranosti i uneravnoteženosti razvoja između gradskih i prigradskih područja kroz povećanje dostupnosti usluga i ostalih sadržaja.

Uspjeh dizajnirane prometne veze integriranog prijevoza putnika u velikoj mjeri ovisi o primjeni taktnog voznog reda s ciljem što kraćeg čekanja na prometnim čvorištima i tečnosti putovanja, neovisno o odabranoj ruti ili modu prijevoza.

Klečina (2020) navodi da 'taktni vozni red i međusobno harmonizirani polasci i dolasci raznih linija i modova prijevoza omogućavaju veliki broj međusobnih veza u sustavu, a iste se, upravo zbog taktnosti ponavljaju u pravilnim razmacima kroz čitav dan.'

Učestali problem lokalnih autobusnih prijevoznika jest financijska neisplativost vožnje na određenim linijama s manjim brojem putnika pri čemu lokalna zajednica mora osigurati subvencioniranje prijevoza kako bi se povećala socijalna inkluzija unatoč pritisku na proračun. Jedan od mogućih problema pri funkcioniranju integriranog prijevoza putnika jest nedostatak suradnje između prijevoznika, što rezultira neskladom u sustavu, različite cijene putovanja i potrebu za kupovinom više karata za realizaciju putovanja korištenjem više modova prijevoza, pri čemu Klečina (2020) identificira potrebu za uvođenjem regionalnih Prometnih uprava koje koordiniraju i upravljaju sustavom na nižim razinama sukladno potrebama lokalnog stanovništva kroz kreiranje prijevozno-tarifnih unija, odnosno sustava u kojem korisnik usluge javnog prijevoza kupovinom jedne karte može ostvariti putovanje između polazišta i odredišta korištenjem različitih modova prijevoza (npr. autobus i vlak ili prigradski vlak i tramvaj).

Za efektivno upravljanje sustavima integriranog prijevoza putnika potrebno je uključiti i aspekt urbanog planiranja, pri čemu Cervero (1998) prema Knowles (2012) predlaže razvoj orijentiran na tranzitu (TOD)[2], te ga definira kao 'proces fokusiranja na razvoj stambenih prostora, zapošljavanja, aktivnosti i javnih usluga oko postojećih ili novih željezničkih stanica koje pružaju željezničke usluge visoke kvalitete i frekvencije linija unutar urbanih područja.'

Uz spomenuto, neophodno je sagledavati i širi kontekst prometnog planiranja urbanih područja zbog dnevnih migracija stanovnika prigradskih naselja koji mogu bitno opteretiti prometne linije i kreirati prometnu potražnju. Zbog guste naseljenosti urbanih područja i velikog broja sadržaja ostvaruju se preduvjeti za visokobudžetne investicije u području mobilnosti poput izgradnje infrastrukture za nadzemnu laku gradsku željeznicu (LRT)[3] ili podzemnu gradsku željeznicu (tzv. metro) koja minimalno uzurpira cestovne prometne pravce koji se nalaze iznad zemlje.

Glavni nedostatak nadzemne lake gradske željeznice jest sudjelovanje u prometu i dijeljena infrastruktura s ostalim nadzemnim modovima prijevoza, što bitno smanjuje brzinu i kvalitetu usluge. Postoji i prijetnja od strujnog udara i uzurpacije zračnog prostora kontaktnom mrežom namijenjenom za napajanje električnom energijom putem pantografa.

[2] eng. 'Transit-oriented development' (TOD).

[3] eng. 'Light rail transit' (LRT).

Glavne mane podzemne gradske željeznice su cijena izgradnje i interakcija s podzemnom prometnom infrastrukturom (plinovod, vodovod itd.), iako se izbjegava interakcija s nadzemnim prometom što rezultira većim brzinama putovanja.

*Slika 3.3 'Prikaz tramvajske infrastrukture u gradu Zagrebu, Hrvatska',*
*Izvor: Fotografirao autor*

Kao mogući smjerovi razvoja infrastrukture prijevoza u urbanim sredinama nameću se 'Hyperloop' sistemi koji se prema Kirschenu i Burnellu (2021) definiraju kao 'koncept sustava masovnog transporta velike brzine koji koristi zatvoreno okruženje niskog tlaka i mala autonomna vozila kako bi se omogućila kombinacija brzih putovanja uz nisku potrošnju energije i ultra-visoku propusnost' te zračni vlakovi koji, nalik na žičaru, putnike prevoze iznad zemlje korištenjem *maglev*[4] tehnologije ili kontaktne mreže integrirane sa infrastrukturom. Korištenjem spomenutih alternativa postojećim oblicima javnog prijevoza ostvaruje se minimalna interakcija sa infrastrukturom ostalih modova prijevoza, karakterističnima za mobilnost u urbanim područjima.

[4]eng. 'magnetic levitation' - magnetska levitacija koja omogućava visoke brzine putovanja korištenjem magnetskih polja čime se postiže levitacija (lebdenje) vlaka u zraku.

## 3.3. Prometni sustavi pametnih gradova

Pojam pametnog grada obuhvaća prikupljanje i primjenu informacija s ciljem povećanja kvalitete života njegovih stanovnika u čemu je prometna komponenta neizostavan element sustava. Primjenom prikupljenih informacija stvaraju se preduvjeti za kreiranje grada po mjeri stanovnika kroz direktnu demokraciju i suodlučivanje građana u temama vezanim za razvoj grada i njegovo određenje, te za optimizaciju usluga u koje se također ubraja i ona prometna.

Rendulić (2022) pametne gradove definira kao 'područje koje koristi senzore za elektroničko prikupljanje podataka koji se nalaze u infrastrukturi, zgradama, vozilima, institucijama i uređajima (internet stvari[5]) za pružanje informacija u stvarnom vremenu o operativnim sustavima gradova.'

Očuvanje zdravlja građana također mora biti u fokusu gradskih vlasti kroz prikupljanje informacija o kvaliteti zraka i štetnih čestica, kao i težnja za kreiranje prometnih politika koje su poticajno nastrojene prema primjeni održivih modova prijevoza.

### 3.3.1. Inteligentni transportni sustavi

Inteligentni transportni sustavi (ITS) su 'upravljačka i informacijsko-komunikacijska nadgradnja klasičnog prometnog i transportnog sustava, tako što se postiže bitno veća propusnost, sigurnost, zaštićenost i ekološka prihvatljivost u odnosu na rješenja bez ITS aplikacija.'

(Krpan, 2017)

Razvojem digitalnih tehnologija uvelike je olakšano nadziranje i upravljanje prometnim sustavima zbog činjenice da se većinom operativno zahtjevnih procesa može upravljati daljinski putem specijaliziranih programskih rješenja, dok su poslovi koji ne pružaju samoaktualizaciju, poput brojanja i nadzora prometa, zamijenjeni naprednim digitalnim rješenjima koja u realnom vremenu primjenom umjetne inteligencije i senzora odašilju povratne informacije bez fizičke prisutnosti na samoj prometnici.

ITS-ima je također omogućeno ažuriranje stanja u prometu informiranjem o prometnim nesrećama i radovima na prometnim dionicama čime se građane pravovremeno usmjeruje na korištenje alternativnih pravaca u planiranju vlastitog putovanja. Usluga javnog prijevoza je također podložna primjeni ITS-a kroz povezanost vozila JGP[6]-a sa informativnim panelima o dolaznim i odlaznim putovanjima, kao i potencijalnim kašnjenjima i zastojima.

[5] eng. 'internet of things' (IoT).

[6] javni gradski prijevoz – sustav gradskih autobusa, tramvaja itd.

Semaforizirana raskrižja u kontekstu ITS-a mogu u realnom vremenu procijeniti prometno stanje i skratiti nepotrebna čekanja na pojedinim prometnim pravcima, posebice u noćnim satima čime mogu dati prednost prolaska pješacima zelenim svjetlom ako senzori ne identificiraju skori dolazak vozila iz bilo kojeg pravca.

ITS-i pronalaze svoju primjenu i u formiranju tzv. 'zelenog vala' u kojem se osigurava protočnost vozila u prometnom toku kroz usklađenost semafora i neometano kretanje u određenom smjeru.

Pavličević (2019) navodi da se 'odlučnost Europske unije da krene s uvođenjem ITS-a vidi iz *Akcijskog plana za uvođenje ITS-a u Europi* kojim se postigao i zakonodavni okvir za odlučniji iskorak u pogledu ITS-a.' u kojem je navedeno 6 područja aktivnosti:

- optimalno korištenje cestovnih, prometnih i putnih podataka,
- neprekinutost ITS usluga za upravljanja prometom i teretom na europskim prometnim koridorima i u gradovima,
- sigurnost na cestama,
- povezivanje vozila i prometne infrastrukture,
- sigurnost i pouzdanost podataka
- i europska suradnja i koordinacija na području ITS-a.

(Narodne novine, 2014 prema Pavličević, 2019)

Preduvjet za uspješnu primjenu ITS-a u prometu je suradnja prometnih stručnjaka s računalnim inženjerima i programerima za cjelovitost programskih i senzorskih rješenja, ali potrebno je uključiti i pravne te urbanističke stručnjake zbog zahtjeva poput regulacija o prikupljanju i obradi podataka, kao i zaštite zgrada od posebnog kulturološkog i tradicionalnog značaja od strane konzervatorskih odjela.

Uvođenje novih tehnologija podrazumijeva kontinuiranu nadogradnju infrastrukture i primjenu najmodernijih rješenja koja nerijetko izazivaju skeptičnost stanovnika zbog potencijalno jačeg elektromagnetskog zračenja koje može nepovoljno utjecati na njihovo zdravlje, što je potrebno uzeti u obzir prilikom planiranja sustava i primjene. U svijetu je sveprisutna primjena najnovije 5G mreže[7], koja ne zaobilazi ni svoju primjenu u ITS-ima, a Pehar (2021) navodi da će 'prema studiji *Qualcomm* iz 2017., 5G mreža ostvariti prihode veće od 2.4 trilijuna američkih dolara što znači da će prometni sektor predstavljati oko 20% cjelokupnog globalnog utjecaja 5G mreže.'

[7] eng. fifth generation (5G) – peta generacija mobilne mreže.

### 3.3.2. Održiva mobilnost

Održiva mobilnost obuhvaća niz politika i mjera kojima se stimulira korištenje održivih modova prijevoza i destimulira korištenje neodrživih modova prijevoza, a ono proizlazi iz strateškog određenja lokalne vlasti i prometnih uprava, ali i iz razine svijesti i sociooloških čimbenika stanovništva nekog područja.

Politike i mjere održive mobilnosti mogu biti usmjerene na:

- izgradnju biciklističke i pješačke infrastrukture,
- uvođenje zona smirenog prometa,
- financijsku stimulaciju korištenja javnog prijevoza,
- uvođenje sustava javnih bicikala ili romobila,
- sprječavanje gomilanja automobila 'kiss-and-ride'[8] regulacijom,
- uvođenje 'park-and-ride'[9] sistema,
- stimuliranje poduzeća na rad od kuće ili netipično radno vrijeme[10],
- primjenu digitalnih tehnologija na parkiralištima,
- proširenje ponude javnog prijevoza,
- car pooling[11] i car sharing[12] sistemi,
- umrežavanje različitih modova prijevoza zajedničkim terminalima
- i redefiniranje sustava naplate parkiranja u užem centru grada itd.

Europska unija obvezuje lokalne vlasti urbanih područja s više od 50 000 stanovnika na kreiranje Planova održive urbane mobilnosti (SUMP[13]) kako bi se prometni sustav orijentirao na doprinos održivim ciljevima i kvaliteti života stanovnika.

Pozitivan primjer usmjerenosti na razvoj održive mobilnosti je grad Koprivnica s postojećim planom, iako ne podliježe obvezi izrade s obzirom na manji broj stanovnika. Grad Koprivnica sustavno ulaže financijska sredstva u proširenje i održavanje pješačke i biciklističke infrastrukture, a uvedeni su i sustavi besplatnih javnih mehaničkih i električnih bicikala, kao i električni autobusi.

---

[8] hrv. 'poljubi i vozi' – omogućava kratko zadržavanje vozila i nastavak putovanja.

[9] hrv. 'parkiraj i vozi' – posebna parkirna mjesta u blizini putničkih terminala s ciljem kombinacije različitih modova prijevoza u jednom putovanju (npr. automobil do parkirališta na obodu grada i tramvaj do centra grada).

[10] netipično radno vrijeme omogućava olakšan protok prometa i manje gužvi u vremenskim periodima koji obuhvaćaju standardno radno vrijeme većine poduzeća poput 7-15h ili 9-17h.

[11] hrv. 'popunjavanje automobila' – organizirani prijevoz više osoba na istoj ili sličnoj ruti.

[12] hrv. 'dijeljenje automobila' – mogućnost korištenja istog automobila po načelu uzmi i ostavi.

[13] eng. Sustainable urban mobility plan.

Glavni izazov u pokušaju implementacije održivih prometnih rješenja u urbanim sredinama je utjecaj na dosadašnje navike stanovništva, pri čemu dominira pretjerana ovisnost o automobilima zbog fleksibilnosti putovanja koju nude u usporedbi s javnim prijevozom koji u većini slučajeva ima fiksno polazište i odredište, te je vremenski i sezonski ograničen.

U Planu održive urbane mobilnosti grada Koprivnice (2015) navedeno je da su 'Planovi održive urbane mobilnosti u Republici Hrvatskoj još uvijek u začetcima izrade i primjene bez postojanja jasnog nacionalnog okvira i smjernica te da je građane potrebno educirati o prednostima koje održivo prometno-prostorno planiranje nudi.'

*Slika 3.4 'Terminal javnih bicikala i električnih autobusa u gradu Koprivnici, Hrvatska',*
*Izvor: Fotografirao autor*

Uz spomenuto, u nekim je zemljama automobil sa sociološke strane pokazatelj uspjeha pojedinca i njegova statusa u društvu, iako brojne razvijenije zemlje (ekonomski i po sustavu vrijednosti) poput Nizozemske i Danske kontinuirano potiču biciklizam kao ekološki prihvatljiv način prijevoza, čime se utječe i na smanjenje eksternih troškova prometa.

Eksterni troškovi prometa obuhvaćaju različite čimbenike koji negativno utječu na stanovništvo, a nastaju zbog omogućavanja mobilnosti. Neki od primjera eksternih troškova su emisije štetnih čestica (zdravstveni troškovi), buka, prometne nesreće, zagađenje tla i slično.

Klečina (2020) napominje da su 'ukupni eksterni troškovi prometa u zemljama Europske unije prema Priručniku za eksterne troškove (2019) Europske komisije tijekom 2016. godine iznosili 841,1 milijardu Eura, a to je 5,7% ukupnog bruto nacionalnog proizvoda čitave EU', a također spominje da se za eksterne troškove u stručnoj literaturi često upotrebljava i izraz 'eksternalije.'

# 4. Digitalna transformacija sustava

U ovom je poglavlju opisan razvoj digitalne transformacije i njezina povezanost s digitizacijom i digitalizacijom. Nabrojani su i opisani svi relevantni elementi digitalne transformacije, kao i njihova uloga u razvoju poslovanja sadašnjosti i usmjerenosti prema budućnosti. U središnjem fokusu poglavlja su podaci, odnosno njihovo prikupljanje, korištenje, zaštita i pohrana u oblaku s ciljem razvoja digitalnih proizvoda koji doprinose kreiranju novog gospodarstva, a radikalno mijenjaju svjetske trendove i kreiraju potrebu za specijaliziranom radnom snagom u određenom području, kao i visok stupanj digitalne i medijske pismenosti svih uključenih dionika u procesu.

Umjetna inteligencija, strojno učenje i znanost o podacima elementi su digitalne transformacije, ali su detaljnije razrađeni u posebnom dijelu rada, odnosno, Poglavljima 6 i 7.

## 4.1. Obilježja i elementi digitalne transformacije

Igrec (2018) navodi da je digitizacija 'proces prijelaza iz analognog u digitalni oblik rada.', dok se digitalizacija odnosi na 'omogućavanje, poboljšanje i transformaciju poslovnih operacija, poslovnih funkcija, modela, procesa i aktivnosti, iskorištavanjem digitalnih tehnologija te šire uporabe i konteksta digitaliziranih podataka.', a digitalna transformacija 'obuhvaća sve aspekte poslovanja, bez obzira radi li se o digitalnom poslovanju ili ne, u vrijeme kada ubrzanje usvajanja tehnologije i promjene dovode do potpuno novog tržišta, kupaca i poslovanja, stvarnosti, prilika i izazova, što u konačnici dovodi do novog gospodarstva.' (Igrec, 2018)

Pavlić (2018) prema Lovrinović (2018) navodi da digitalna transformacija obuhvaća promjenu u 5 različitih područja:

- udaljenost,
- komunikaciju,
- okruženje,
- rad
- i mobilnost.

Težnja za digitalnom transformacijom poslovanja proizlazi iz razvoja gospodarstva koje nameće potrebu za kontinuirano poboljšanje u svim sferama poslovanja zbog očuvanja konkurentnosti i širenja ponude usluga iz postojećih resursa, što obuhvaća i redizajniranje postojećeg poslovanja i usklađivanje s potrebama modernog društva.

Elementi digitalnog poslovanja prisutni su u velikom broju poduzeća, no to ne znači da su ona okrenuta potpunoj digitalnoj transformaciji poslovanja koje bi bilo usmjereno na kreiranje dodane vrijednosti, već na parcijalno pojednostavnjenje određenih poslovnih procesa.

Razvoj novih industrija podrazumijeva i pojavu pojmova koji opisuju njihov djelokrug, no oni vrlo često zvuče futuristički i bivaju teško razumljivi široj javnosti. Pri razumijevanju novih pojmova poglavito su ugrožene starije populacije koje mogu pružati otpor pri uvođenju digitalne transformacije u poslovanje poduzeća u kojem su zaposlene zbog potrebe za dodatnom naobrazbom u području tehnologija koje nisu postojale u vrijeme njihovog obrazovanja, a nužne su za ispunjenje ciljeva menadžmenta pri implementaciji samih tehnologija.

Igrec (2018) navodi neke od elemenata digitalne transformacije:

- veliki podaci[14],
- internet stvari[15],
- pametne tvornice[16],
- računarstvo u oblaku[17],
- upravljanje odnosima s klijentima[18]
- i komunikacija u realnom vremenu[19].

Uz spomenute pojmove, neophodno je spomenuti i one koji imaju značajnu ulogu u funkcionalnosti i diferencijaciji digitalnih proizvoda i usluga poput umjetne inteligencije, strojnog učenja (algoritama strojnog učenja), znanosti o podacima[20], tehnologije lanca blokova[21] te ostalih povezanih pojmova i nižih podjela navedenih u narednim poglavljima.

Hrvatski prijevod mnogih elemenata nije zaživio među stanovništvom zbog čestog upotrebljavanja engleskih naziva, posebice zbog internacionalne suradnje na projektima koji zahtijevaju širok spektar znanja pri razvoju digitalnih rješenja, ali i edukacija koje polaznici pohađaju u stranim zemljama ili primjenom interneta zbog rapidno razvijajuće industrije u kojoj znanja ubrzo zastarijevaju i na tržište se plasiraju sve brža, sigurnija i preciznija digitalna rješenja.

Upravo se kombinacijom više elemenata digitalne transformacije maksimalizira uspješnost funkcioniranja cjelokupnog sustava zbog umrežavanja i kros-funkcionalnosti između aplikacija i digitalnih rješenja za internu upotrebu u poduzeću ili kao proizvod ili usluga koja je namijenjena plasmanu na digitalno tržište.

---

[14] eng. 'big data'.

[15] eng. 'internet of things' (IoT).

[16] eng. 'smart factory'.

[17] eng. 'cloud computing'.

[18] eng. 'customer relationship management' (CRM).

[19] eng. 'real-time communication' (RTC).

[20] često se nazivaju i podatkovnim znanostima.

[21] eng. 'blockchain'.

## 4.2. Podaci i veliki podaci

Podaci su temelj funkcioniranja digitalnog svijeta i preduvjet za orijentaciju prema digitalnoj transformaciji poslovanja.

Veliki podaci (eng. 'Big data') su 'često karakterizirani ekstremnim volumenom podataka, raznolikošću tipova podataka i brzinom kojom se moraju obrađivati.' (Kelleher i Tierney, 2018)

Obujam velikih podataka doprinosi reprezentativnosti istraživanja u kojima se koriste, kao i pouzdanosti predviđanja u algoritmima strojnog učenja.

Sve više poduzeća okreće se prikupljanju i obradi podataka te shvaćaju njihov značaj i ulogu u razvoju poslovanja, što je posebno izraženo u novim poslovnim modelima u kojima podaci postaju roba na tržištu. Osim trgovanja podacima van poduzeća, podaci se koriste i za unaprjeđenje vlastitih proizvoda i usluga te poboljšanju korisničkih sučelja (UI[22]) i korisničkog iskustva (UX[23]).

Bitnu ulogu u razumijevanju i obradi velikih podataka imaju i metapodaci, odnosno podaci koji sadržajno opisuju prikupljene podatke prema parametrima poput vremena nastanka, kreatora podataka, veličine podataka, lokacije, formata i sličnog.

U godišnjem izvješću *AI Data & Analytics Networka* (2021) spominje se niz pojmova vezanih uz podatke i poslovne modele temeljene na njima:

- skladišta podataka[24],
- jezera podataka[25],
- demokratizacija podataka[26],
- vizualizacija podataka[27],
- napredna analitika[28],
- povrat na investiciju u podatke[29] i monetizacija podataka[30]
- i upravljanje podacima[31].

[22] eng. 'user interface' (UI).

[23] eng. 'user experience' (UX).

[24] eng. 'data warehousing'.

[25] eng. 'data lakes'.

[26] eng. 'data democratization'.

[27] eng. 'data visualization'.

[28] eng. 'advanced analytics'.

[29] eng. 'data return on investment (ROI)'.

[30] eng. 'data monetization'.

[31] eng, 'data governance'.

### 4.2.1. Skladište podataka

Skladišta podataka (eng. 'Data warehouses') predstavljaju organizirane skupove podataka, sortirane prema značajkama zbog lakšeg indeksiranja i orijentacije kroz skladište u digitalnom i fizičkom smislu. Sam pojam skladišta podataka proizlazi iz tradicionalnih skladišta u kojima je pohranjena roba, zbog čega vrijede iste zakonitosti poput brzine obrađivanja zahtjeva, grupiranja zahtjeva, sigurnosti robe (podataka) i slično.

Skladište podataka odnosi se na postupak zbrinjavanja i centralizacije podataka, što razvojnim inženjerima, inženjerima strojnog učenja, podatkovnim znanstvenicima i inženjerima te svim ostalim zainteresiranim stranama omogućavaju interakciju s podacima i obradu podataka u realnom vremenu za potrebe razvijanja digitalnih usluga u poduzeću, kao i prikupljanje znanja iz velikih skupova podataka, što rezultira stvaranjem podloge za razvoj inovacija.

Inmon et al. (2008) prema Kopreku (2021) navode da značajnu ulogu u funkcioniranju skladišta podataka ima ETL[32] sustav koji 'izvršava više funkcija u okruženju skladišta podataka, kao što su konverzija podataka, verifikacija domena, konverzija iz jednog sustava za upravljanje bazama podataka u drugi, kreiranje zadanih vrijednosti ako je to potrebno, agregacija podataka, dodavanje vremenskih vrijednosti ključu podataka, restrukturiranje ključa podataka, spajanje zapisa, brisanje nebitnih ili suvišnih podataka.'

'Prije modeliranja i konstruiranja skladišta podataka važno je predočiti si glavne ciljeve skladišta podataka i poslovne inteligencije.' (Koprek, 2021)

Prema istraživanju *AI Data & Analytics Networka* (2021) provedenom na uzorku od 400 poduzeća iz različitih domena i zemalja, 39% ispitanih poduzeća navodi da je u protekloj godini investiralo financijska sredstva u razvoj skladišta podataka u *oblaku*, dok još 29% njih planira investirati u skladišta podataka u *oblaku* u narednom periodu.

### 4.2.2. Jezero podataka

Jezera podataka (eng. 'Data lakes') obuhvaćaju velike skupove podataka za koje ne postoji jasna svrha korištenja, no nagađa se da mogu poslužiti za razvoj u budućem razdoblju. Miloslavskaya i Tolstoy (2016) navode da su jezera podataka građena za zbrinjavanje nadolazećeg velikog volumena podataka u kratkom vremenu i da su sačinjena od svih oblika podataka, pa i sirovih (nestrukturiranih podataka), za razliku od skladišta podataka u kojima su pohranjeni isključivo strukturirani podaci.

[32] eng. 'extract, transfer, load'. – izvuci, prenesi, učitaj.

Prema istraživanju *AI Data & Analytics Networka* (2021) 17% ispitanih poduzeća potvrdilo je da je u protekloj godini investiralo financijska sredstva u jezera podataka, a 15% njih planira učiniti isto u budućem razdoblju.

Tablica 4.1 prikazuje razlike između skladišta i jezera podataka prema vrsti sadržanih podataka, načinu procesuiranja, troškovima održavanja i pohrane, agilnosti, sigurnosti pohranjenih podataka i glavnim korisnicima podataka.

**Tablica 4.1 'Usporedba značajki skladišta i jezera podataka'**

| | SKLADIŠTE PODATAKA | JEZERO PODATAKA |
|---|---|---|
| PODACI | strukturirani, procesuirani podaci | strukturirani, djelomično strukturirani, nestrukturirani, sirovi, neprocesuirani podaci |
| PROCESUIRANJE | pri zapisivanju | pri iščitavanju |
| POHRANA | skupa, ali pouzdana pohrana | jeftina pohrana |
| AGILNOST | manja agilnost, fiksna konfiguracija | visoka agilnost, fleksibilna konfiguracija |
| SIGURNOST | visoka razina sigurnosti | potencijalno niža razina sigurnosti |
| KORISNICI | poslovni profesionalci, donositelji odluka | podatkovni znanstevnici |

**Izvor:** Vlastiti rad autora prema Khine, P.P., Wang, Z.S., (2018): 'Data lake: a new ideology in big data era'., EDP Sciences.

Iz navedenog prikaza i provedenih istraživanja može se zaključiti da su poduzeća sklonija investiranju u skladišta podataka u odnosu na jezera podataka zbog prikupljanja i obrade podataka koje je definirano strategijom poslovne organizacije kao infrastruktura koja dovodi do ostvarenja ciljeva poslovne inteligencije[33] kroz transformaciju prikupljenih podataka u informacije.

Skladišta podataka sadrže strukturirane podatke koji mogu u kraćem vremenskom periodu biti iskorišteni u poslovne svrhe bez dodatnog procesuiranja koje se zahtijeva pri korištenju podataka iz jezera podataka, s obzirom na vrlo visoku vjerojatnost postojanja redundancije (zalihosti) podataka.

Kramberger et al. (2018) navode da 'redundantni podaci znače da se isti podaci pohranjuju na više od jednog mjesta unutar baze podataka', što može dovesti do 'anomalije podataka koja se može dogoditi prilikom unosa, promjene i brisanja podataka.'

[33] eng. 'business intelligence' (BI).

### 4.2.3. Demokratizacija podataka

Demokratizacija podataka (eng. 'Data democratization') odnosi se na proširenje dostupnosti prikupljenih podataka prema zaposlenicima poduzeća ili organizacije uz poštivanje definiranih uvjeta pravednog korištenja uvažavajući sigurnosne i zakonske aspekte. (Awasthi i George, 2020. prema Lebevbre et al. 2021.)

Demokratizacijom podataka bitno se povećava njihova korisnost zbog razvoja novih proizvoda koji mogu biti temeljeni na njima, a također se eliminira i posredovanje podacima na zahtjev između različitih odjela čime se rasterećuju zaposlenici i štedi se vrijeme.

Iako se demokratizacijom podataka podiže razina transparentnosti svih procesa u organizaciji, potrebno je raščlaniti potencijalno osjetljive podatke od ostalih podataka jer mogu biti zlouporabljeni i usmjereni protiv GDPR[34] smjernica.

Kroz demokratizaciju podataka poduzeće se izlaže povišenom riziku od krađe podataka jer se odgovornost dijeli na sve zaposlenike s pristupom koji mogu biti nestručni pri rukovanju podacima i zanemariti sigurnosne aspekte.

Mjere kojima se umanjuje rizik od nastanka krizne situacije nastale zlouporabom ili krađom podataka definiraju se kriznim planom koji obvezuje sve dionike u procesu na djelovanje sukladno propisanim smjernicama s ciljem prevencije nastanka krize u pretkriznom razdoblju i reaktivnim djelovanjem na krizu u kriznom i postkriznom razdoblju prema Coombsovom (2007) modelu životnog ciklusa krize koju navodi Jugo (2017).

'Svrha kriznog plana je promptna reakcija organizacije na krizu kroz predefinirani način postupanja definiranim u obliku dokumenta s jasnim zadaćama, ciljevima i odgovornostima vezanim uz proces kriznog upravljanja.' (Kundih, 2020)

Enkripcija (šifriranje) podataka od presudne je važnosti za poduzeća s postojećim vanjskim korisnicima usluga zbog sprječavanja neovlaštenog pristupa od strane vlastitih zaposlenika i *curenja* podataka u javnost što poduzeće stavlja u nepovoljan položaj i za rezultat ima narušenu reputaciju, kao i moguće sudske tužbe samih korisnika usluga.

Dodatan stupanj zaštite od krađe ili zlouporabe podataka može se postići implementacijom načela normi koje su navedene u ISO/IEC[35] 27000: 'Sustavu upravljanja informatičkom (informacijskom) sigurnosti.'

---

[34] eng. 'General Data Protection Regulation' (Opća uredba o zaštiti podataka) je regulativa propisana od strane Europske unije kojom se uređuju pitanja zaštite podataka i privatnosti. Informacije o regulativi dostupne na: https://eur-lex.europa.eu/legal-content/HR/TXT/?uri=CELEX:32016R0679

[35] eng. International Organization for Standardization (Međunarodna organizacija za standardizaciju) / International Electrotechnical Commission (Međunarodna komisija za elektrotehniku).

Bogati (2011) navodi osam glavnih koraka koje je potrebno slijedno provoditi pri implementaciji norme ISO/IEC 27001 u organizaciju:

- započinjanje projekata,
- definiranje Sustava upravljanja informatičkom (informacijskom) sigurnosti,
- procjena rizikom,
- upravljanje rizikom
- obuka i osvješćivanje,
- priprema za reviziju,
- revizija,
- i neprekidno osvješćivanje.

### 4.2.4. Vizualizacija podataka

Vizualizacijom podataka (eng. 'Data visualization') nastoji se prikazati podatke u interaktivnom sučelju korištenjem programa ili programskog jezika.

Za korisničko sučelje na kojem su prikazani podaci upotrebljava se i naziv upravljačka ploča (eng. 'dashboard'), odnosno, u kontekstu vizualizacije podataka dijeli istu ulogu kao i upravljačka ploča u automobilu kroz pružanje informacija u realnom vremenu. (Blagović, 2016)

Blagović (2016) navodi i da 'neki proizvod može zahvaćati podatke istovremeno sa lokalnog računala iz neke trenutno aktivne aplikacije te sa nekog Web mjesta i kroz upravljačku ploču prezentirati te podatke kao da su došli iz istog izvora.'

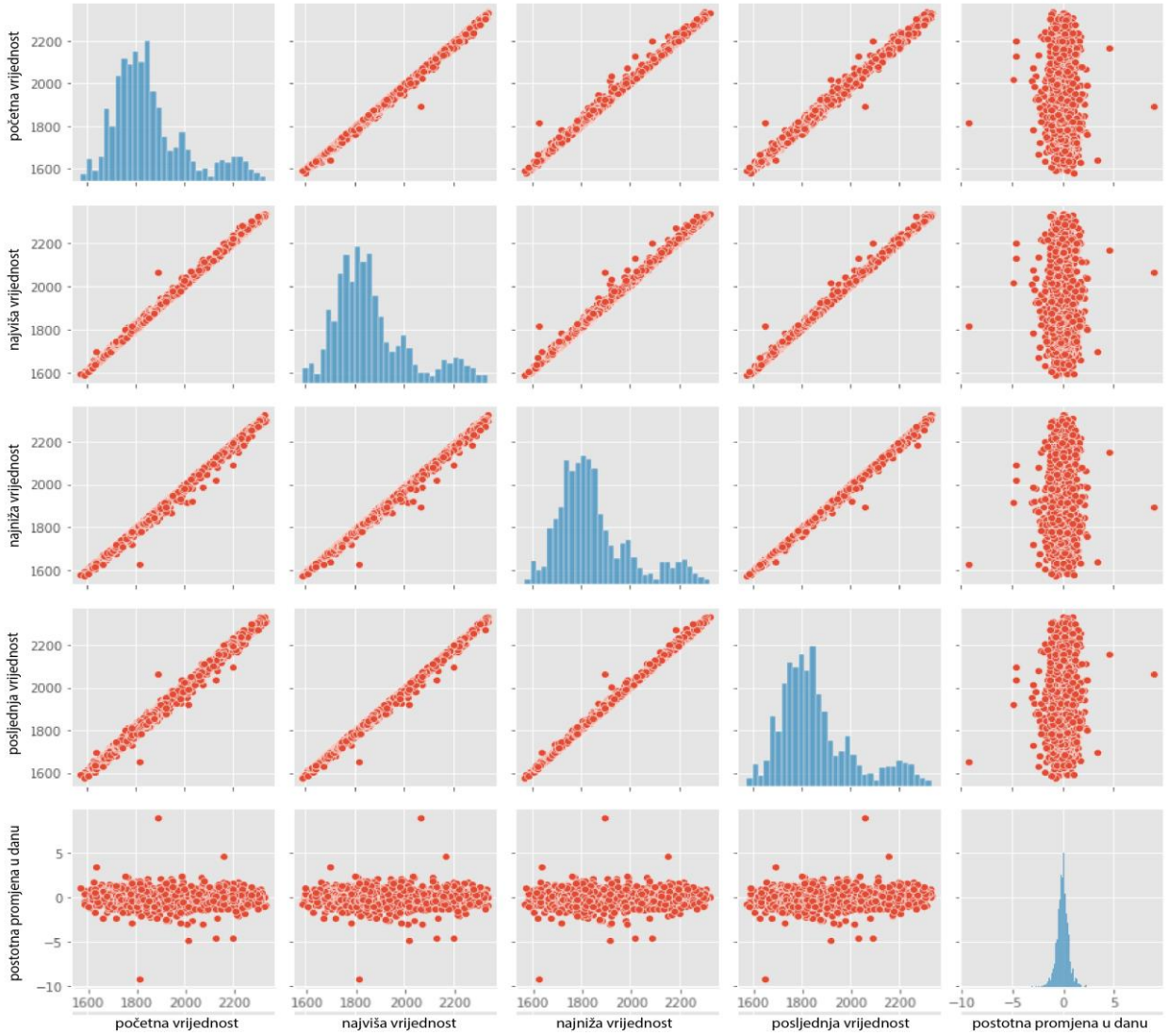
Prema brojnim istraživanjima ljudi najviše informacija primaju upravo vizualno i stoga je razvoj domene vizualizacije podataka logičan slijed u nastojanju poboljšanja korisničkog iskustva mnogih poduzeća koja se strateški određuju prema digitalnoj transformaciji poslovanja. Vizualizacija podataka može se koristiti za interne svrhe zaposlenika, prikaza poslovnih rezultata rukovodećim osoba u poduzeću ili pristup vlastitoj statistici vanjskog korisnika usluge koju poduzeće pruža.

Skladišta podataka i jezera podataka najčešće se sastoje od podataka koji su zapisani različitim simbolima, odnosno slovima i brojevima te mogu djelovati naizgled monotono iako sadrže bitne informacije za poduzeća, a njihovom se vizualizacijom mogu u realnom vremenu prikazati statističke analize i stupanj ispunjenja zadanih ciljeva.

Istraživanje *AI Data & Analytics Networka* (2021) pokazuje da 45% ispitanih poduzeća već koristi neke od metoda i alata vizualizacije podataka, a 56% ih navodi da će uložiti dodatna sredstva u nadolazećem periodu.

Programi koji se često koriste s ciljem vizualizacije podataka su *Microsoft BI*, *Microsoft Excel* i *Microsoft Project*, ali i brojne biblioteke programskih jezika poput *matplotliba*, *seaborna* i *plotlya* te paketa *folium* za vizualizaciju geografskih podataka u programskom jeziku Python 3[36].

Slika 3.1 prikazuje vizualizaciju povijesnih podataka indeksa Zagrebačke burze (CROBEX)[37] korištenjem *seaborn* Python biblioteke i *Microsoft Visual Studio* programa.




*Slika 4.1 'Vizualizacija povijesnih podataka indeksa Zagrebačke burze',*
*Izvor: Vlastiti rad autora korištenjem seaborn biblioteke*

Za potrebe vizualizacije podataka u izvještajima koriste se i računalni programi *Adobe Illustrator* i *Inkscape* za vektorske[38] formate te *Adobe Photoshop* i GIMP za rasterske[39] formate.

[36] detaljnije o paketima programskog jezika Python u poglavlju 5.7.

[37] podaci su javno dostupni na: https://zse.hr/hr/indeks/365?isin=HRZB00ICBEX6&tab=index_history

[38] vektorski formati pamte točke vizualnog rješenja i ne gube na kvaliteti pri promjeni dimenzija.

[39] rasterski formati sastoje se od piksela i mogu izgubiti kvalitetu pri promjeni dimenzija.

### 4.2.5. Napredna analitika

Napredna analitika (eng. 'Advanced analytics') koristi se s ciljem prikupljanja znanja iz skupova podataka, koristeći se pritom algoritmima strojnog učenja. Napredna analitika nadogradnja je osnovnih statističkih modela, a cilj joj je prepoznati međuodnose iz prikupljenih podataka te formulirati i predložiti zaključak.

Primjena napredne analitike posebno je prepoznata u marketinškim istraživanjima tržišta gdje se nastoji predvidjeti stanje kretanja trendova na tržištu što također podrazumijeva i optimizaciju logističkih procesa zbog planiranja potražnje kako bi se pravovremeno osigurali svi resursi nužni za odgovor na potrebe tržišta. Napredna analitika ima svoju ulogu i u medicini, gdje algoritmi strojnog učenja mogu prepoznati faktore koji mogu dovesti do razvoja određenih bolesti kroz prikupljanje podataka i usporedbu s postojećim podacima u bazama podataka poput kreiranja modela predviđanja koji na temelju podataka[40] (Slika 4.2) poput malformacija kože, učestalosti kašlja, spolu, dobi, korištenju alkohola i cigareta te ostalih indikatora prepoznaje potencijalno ugrožene pacijente s visokom vjerojatnosti razvoja kancerogenih bolesti.

| SPOL | DOB | PUŠAČ | ŽUTI PRSTI | ANKSIOZNOST | KRONIČNE BOLESTI | UMOR | ALERGIJA | OTEŽANO DISANJE | ALKOHOL | KAŠALJ | KRATKOĆA DAHA | OTEŽANO GUTANJE | BOL U PRSIMA | RAK PLUĆA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M | 69 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | DA |
| M | 74 | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | DA |
| Ž | 59 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 2 | NE |
| M | 63 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | NE |
| Ž | 63 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | NE |
| Ž | 75 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 1 | DA |
| M | 52 | 2 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | DA |
| Ž | 51 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 1 | DA |
| Ž | 68 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | NE |
| M | 53 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | DA |
| Ž | 61 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 2 | 1 | DA |
| M | 72 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | DA |
| Ž | 60 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | NE |
| M | 58 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | DA |
| M | 69 | 2 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | NE |
| Ž | 48 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 1 | DA |
| M | 75 | 2 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | DA |
| M | 57 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | DA |
| Ž | 68 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | DA |
| Ž | 61 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | NE |
| Ž | 44 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | DA |
| Ž | 64 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | DA |
| Ž | 21 | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | NE |
| M | 60 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | DA |
| M | 72 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | DA |
| M | 65 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | DA |
| Ž | 61 | 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 2 | DA |
| M | 69 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 2 | NE |
| Ž | 53 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 2 | DA |
| M | 55 | 1 | 2 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | NE |
| Ž | 57 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | NE |
| M | 62 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | DA |
| M | 56 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | NE |
| Ž | 67 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | DA |
| M | 59 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | NE |
| Ž | 59 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | DA |

*Slika 4.2 'Prikaz podataka pacijenata oboljelih od raka pluća',*
*Izvor: Adaptirano prema datoteci https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer?resource=download*

[40]navedeni podaci su u javnoj domeni – cjelovita datoteka je dostupna na platformi Kaggle: https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer?resource=download.

Kao i u ostalim primjenama algoritama strojnog učenja, za kreiranje modela u službi napredne analitike potrebno je uzeti u obzir značajke koje uistinu doprinose povećanju razine uspješnosti predviđanja. Iz identificiranih odnosa (korelacija) u podacima može se računski prikazati njihova međuovisnost kroz formulu, a jedan on primjera navode Kelleher i Tierney (2018) kroz kreiranje formule za izračun vjerojatnosti nastanka dijabetesa u odnosu na ITM[41] pacijenta (4.1):

$$\text{Dijabetes} = -7.38431 + 0.55593 * \text{BMI} \quad (4.1)$$

BMI – indeks tjelesne mase.

Dijabetes – varijabla Y.

Navedena formula proizlazi iz modela predviđanja vrijednosti zavisne varijable u odnosu na nezavisnu korištenjem algoritma linearne regresije[42].

'Vrijednost parametra nagiba $w_1 = 0.55593$ ukazuje na to da za svako jedinično povećanje indeksa tjelesne mase BMI, procijenjenu vrijednost izgleda za dijabetes model povećava za nešto više od pola posto.

Kelleher i Tierney (2018)

Za uspješnost modela predviđanja potrebno je razmotriti implementaciju različitih algoritama strojnog učenja i primijeniti onaj s najboljim performansama, a pri samom odabiru kompleksnost algoritma nije uvijek garancija veće pouzdanosti pri čemu Kaisler et al. (2014) navode da je od kompleksnosti algoritma važnija inteligencija koja je ugrađena u njega, a uz to i posebna znanja iz promatrane domene u kojoj se on primjenjuje.

Navedeni zaključak proizlazi iz različitih potreba i specifičnosti svakog od projekata modeliranja, kao i volumena podataka te odabranih značajki koje se uzimaju u obzir pri treniranju i testiranju algoritma.

Upravo je prediktivna analiza prepoznata kao glavni nositelj napredne analitike, čemu u prilog ide i podatak da je u protekloj godini 54% poduzeća investiralo financijska sredstva u neki od njezinih oblika u vlastitom poslovanju. (Izvješće *AI Data & Analytics Networka*, 2021)

Napredna analitika sastavni je dio brojnih digitalnih usluga i proizvoda, a često je orijentirana na personalizaciju sadržaja prema korisniku kroz identifikaciju njegova ponašanja i preferencija pri upotrebi.

Web aplikacije poput *Youtubea*, *Spotifya* i *Netflixa* kroz vrijeme uče o interesima korisnika i sukladno tome kreiraju personalizirano sučelje, a reklamni sadržaj koji je ugrađen u digitalnu uslugu može biti relevantan ukoliko se ne koriste digitalna rješenja koja ga blokiraju.

---

[41] eng. BMI – 'body mass index' – indeks tjelesne mase.

[42] detaljnije o algoritmima strojnog učenja u poglavlju 7.

### 4.2.6. Monetizacija podataka i povrat na investiciju (ROI) u podatke

Neiskorišteni podaci mogu predstavljati mrtvi kapital poduzeća, zbog čega je potrebno kreirati strategiju prikupljanja i korištenja podataka za poslovne namjene kako bi ono bili svrhovito i doprinosilo optimizaciji postojećih i budućih digitalnih rješenja.

Pojmovi monetizacije podataka i povrata na investiciju u podatke proizlaze iz ekonomske (financijske) terminologije te opisuju odnos investicija i očekivane koristi iz poslovnih modela koji se temelje na podacima. Korist koja proizlazi iz takvih poslovnih modela ogleda se u financijskom aspektu, ali i kvaliteti proizvoda i usluga, kao i njihove konkurentnosti na tržištu.

Neki od primjera poduzeća koja koriste poslovne modele koji se temelje na podacima su globalno prepoznatljivi *Google, Facebook (Meta)* i *Microsoft.*

Većina usluga koje nude spomenuta poduzeća su besplatna, ali ta ista poduzeća imaju izuzetno visoke troškove poput održavanja, dostupnosti, kontinuiranog razvoja i primjena novih tehnologija te infrastrukture i plaća zaposlenika, što samo po sebi nameće pitanje isplativosti jer poduzeća su uglavnom profitno orijentirana, a ne altruističke organizacije koje svjesno preuzimaju troškove bez ikakvog benefita.

Odgovor na to pitanje leži upravo u monetizaciji podataka, odnosno prikupljanju podataka i prodaji zaključaka ostalim poslovnim subjektima koja ovise o oglašavanju na društvenim mrežama te o analizi postojećih i nadolazećih trendova na tržištu, što bitno mijenja marketinške strategije samih poduzeća i njihove proizvode i usluge.

Zuboff (2019) takve poslovne modele opisuje pojmom *nadzornog kapitalizma*[43] i kao primjer navodi uređaje poduzeća Google, suprotstavljajući pritom dva moguća scenarija:

- **scenarij 1** – prodaja uređaja koji na temelju prepoznatljivog brenda *Google* može razviti percepciju pouzdanosti i postići višu cijenu, odnosno maržu po prodanom uređaju.
- **scenarij 2** – prodaja Googleovog uređaja po što jeftinijoj cijeni kako bi se postigla što veća prodaja, odnosno, što veća baza korisnika od kojih se mogu prikupljati podaci.

Prema Zuboff (2019) *Google* se orijentirao na poslovni model baziran na scenariju 2.

Efekti te odluke mogu se prepoznati i pri kupovini većine *Androidovih* mobilnih uređaja koji u sebi sadrže sve više *Googleovih* usluga i aplikacija poput *Google asistenta, Google Playa, Google tražilice* kao zadane pri pretraživanju, pa sve do zamjene aplikacije za fotografije (galerije) aplikacijom *Google fotografije* kod mobilnih uređaja poput američke *Motorole*.

[43] eng. 'surveillance capitalism'.

Iz spomenutog primjera, uočljivo je da su upravo prikupljanje i korištenje korisničkih podataka postali instrumenti globalno-političkog utjecaja velikih poduzeća i država iz kojih dolaze zbog mogućnosti kontrole i prikupljanja podataka kroz naizgled bezazlene i vrlo često zabavne sadržaje kroz koje se dolazi do podataka.

Prema Nanaj (2020), 15. svibnja 2019. američki predsjednik Donald Trump najavio je zabranu djelovanja korporacija koje mogu predstavljati prijetnju za nacionalnu sigurnost, što je bila svojevrsna poruka kineskom poduzeću *Huawei Telecommunication Company* kojem je nadalje onemogućeno prisustvovanje na američkom tržištu i kao kupcu i kao prodavaču.

Postavlja se pitanje je li *Huawei Telecommunication Company* uistinu poduzeće koje predstavlja prijetnju nacionalnoj sigurnosti SAD-a[44] ili se samo radi o eliminaciji konkurencije američkih telekomunikacijskih mastodonata i ojačavanju njihove pozicije u SAD-u i pristup podacima korisnika usluga koje vrlo vjerojatno koristi i sama država.

Navedeni niz događaja najvjernije se može opisati pojmom *stanja trajnog rata* knjige *1984.* Georgea Orwella u kojem je sveprisutna vanjska prijetnja koja opravdava određene odluke vlasti s ciljem zaštite vlastitih građana.[45]

Prikupljanje podataka od strane korisnika s ciljem njihove monetizacije često se vrši na perfidan i neizravan način kroz skočne prozore (eng. 'pop-up') s beskrajnim linijama teksta u kojima korisnik mora potvrditi da je suglasan s prikupljanjem podataka, a u protivnom postoji mogućnost blokiranja sadržaja i nemogućnosti pristupa.

Manipulaciji i pristanku na prikupljanje podataka kroz vizualno primamljiva i poticajna sučelja posebno su izložene osobe s nedostatkom medijske pismenosti i djeca. Nerijetko je slučaj da upravo gumb za suglasnost biva obojen uočljivom bojom poput crvene, dok je gumb za odbijanje monotono dizajniran i ne iskače iz cjeline. Korisnici digitalnih usluga djelomično su navikli na spomenute skočne prozore i već nesvjesno daju suglasnost kako bi pristupili sadržaju, a ako se i odluče usprotiviti sakupljanju podataka, moraju pojedinačno isključiti svaku od značajki bez opcije da se to učini jednim klikom - što djeluje demotivirajuće na korisnika da učini isto.

---

[44] SAD – Sjedinjene Američke Države

[45] U kontekstu knjige '1984.' Georgea Orwella pojam *stanje trajnog rata* obuhvaća niz mjera vlasti kojima se opravdavaju nestašice, neprestano se ulažu financijska sredstva u vojsku (prikazano kroz tzv. *plovne tvrđave*), provode se *dvominutne mržnje* kako bi se ojačala netrpeljivost prema neprijatelju, ali izravni sukobi velikih sila su rijetki jer je upravo 'stanje trajnog rata' način na koji one egzistiraju kroz prizmu sijanja straha, a ako do sukoba i dođe, Peričić (2014) u pogovoru navodi da se on očituje kroz tzv. zamjenske ratove (eng. 'proxy wars') u kojem se velike sile neizravno sukobljavaju preko trećih strana zbog rizika od nastanka nuklearnog rata. Prikupljanje podataka i konstantan nadzor u nekim od država koje uvode sisteme društvenog bodovanja (eng. 'social credit system') može se poistovjetiti s upotrebom *telekrana* u samoj knjizi.

### 4.2.7. Upravljanje podacima

Upravljanje podacima (eng. 'Data governance') odnosi se na kreiranje konteksta, odnosno na definiranje strategije, politika, općih i operativnih ciljeva vezanih uz prikupljanje podataka, korištenje podataka i sigurnost podataka.
Jašarević (2019) navodi da programi upravljanja podacima imaju utjecaj na stratešku, taktičku i operativnu razinu menadžmenta kroz niz ciljeva koji se odnose na:

- minimiziranje opasnosti,
- određivanje pravila za uporabu podataka unutar organizacije,
- implementaciju zakonskih regulativa,
- poboljšanje unutarnje i vanjske komunikacije,
- povećanje vrijednost podataka,
- olakšavanje administracijskih poslova,
- smanjenje troškova
- i osiguranje stabilnosti organizacije.

Zakonske regulative vezane uz prikupljanje podataka rapidno se mijenjaju zbog brzog razvoja novih industrija vezanih uz digitalnu transformaciju i informacijsko-komunikacijsku infrastrukturu što predstavlja rizik za krajnje korisnike i njihovu privatnost, a poduzeća moraju pratiti promjene u općem okruženju (političkom, gospodarskom, društvenom, tehnološkom, okolišnom i pravnom) i prilagoditi mu svoje poslovanje zbog ograničenog utjecaja na više razine sustava kojima je podložno.


*Slika 4.3 'Krivulja vrijednosti informacije',*
*Izvor: Varga, M. (2021): 'Upravljanje podacima', Sveučilište u Zagrebu, Zagreb, p 10.*

Slika 4.3 opisuje kretanje vrijednost podataka i informacija kroz vrijeme. Za razumijevanje prikaza potrebno je raščlaniti pojam podatka i informacije, pri čemu Kramberger et al. (2018) navode da je podatak 'skup simbola ili znakova koji su pohranjen u memoriji', a informacija 'obrađeni podatak koji sadrži znanje koje primatelju opisuje nove činjenice, otklanja neizvjesnost i služi kao podloga za odlučivanje.' Iz navedenog se zaključuje da podatak postaje informacija svojim ulaskom u eksploatacijsko razdoblje.
Varga (2021) prikazane zone tijeka vrijednosti informacija opisuje kao:

- **Zona pasivnosti podataka**

Prikazuje negativnu vrijednost podataka za organizaciju zbog postojećih troškova vezanih uz pohranu i njihovo prikupljanje, kao i uvođenje sustava sigurnosti od zlouporabe i krađe podataka te izdvajanje sredstava za plaće informatičkih stručnjaka i razvojnih inženjera.

- **Zona prvog povrata vrijednosti podataka**

Definira se kao prelazak podataka iz pasive u aktivu kroz upotrebu u poslovnim aktivnostima, analizama i planiranju budućeg poslovanja. U ovoj fazi podaci prelaze iz negativne vrijednosti za organizaciju u generiranje pozitivne vrijednosti.

- **Zona poslovne inteligencije**

Zona u kojoj se podaci demokratiziraju na horizontalnoj i vertikalnoj osi organizacije i postaju dostupni širem broju odjela za potrebe poslovanja, razvoja novih digitalnih rješenja te poslovnih projekcija i analiza. Zona poslovne inteligencije omogućava internu transparentnost poslovanja što zahtijeva visoku razinu povjerenja u odgovorno djelovanje svih obuhvaćenih zaposlenika.

- **Zona podatkovnog proširenja organizacije**

Podatkovno proširenje organizacije opisuje širenje upotrebe prikupljenih podataka iz organizacije van nje same, pri čemu vanjski dionici (kupci, dobavljači, poslovni partneri) postaju aktivni korisnici istih, što može doprinijeti harmonizaciji suradnje i zajedničkog poslovanja, kao i eksterne transparentnosti.

- **Zona prodaje informacija**

Prodajom informacija, odnosno znanja prikupljenog iz podataka, organizacije stvaraju dodatni prihod, a prodajom sirovih podataka otvaraju prilike za razvoj novih poslova budućnosti (poglavito vezane uz znanost o podacima, umjetnu inteligenciju i strojno učenje koji ovise o podacima). Važno je naglasiti da je vrijednost podataka dugotrajnija od vrijednosti informacija jer se iz podataka mogu izvoditi novi zaključci, a informacija je relevantna kroz kratko vrijeme.

## 4.3. Internet stvari

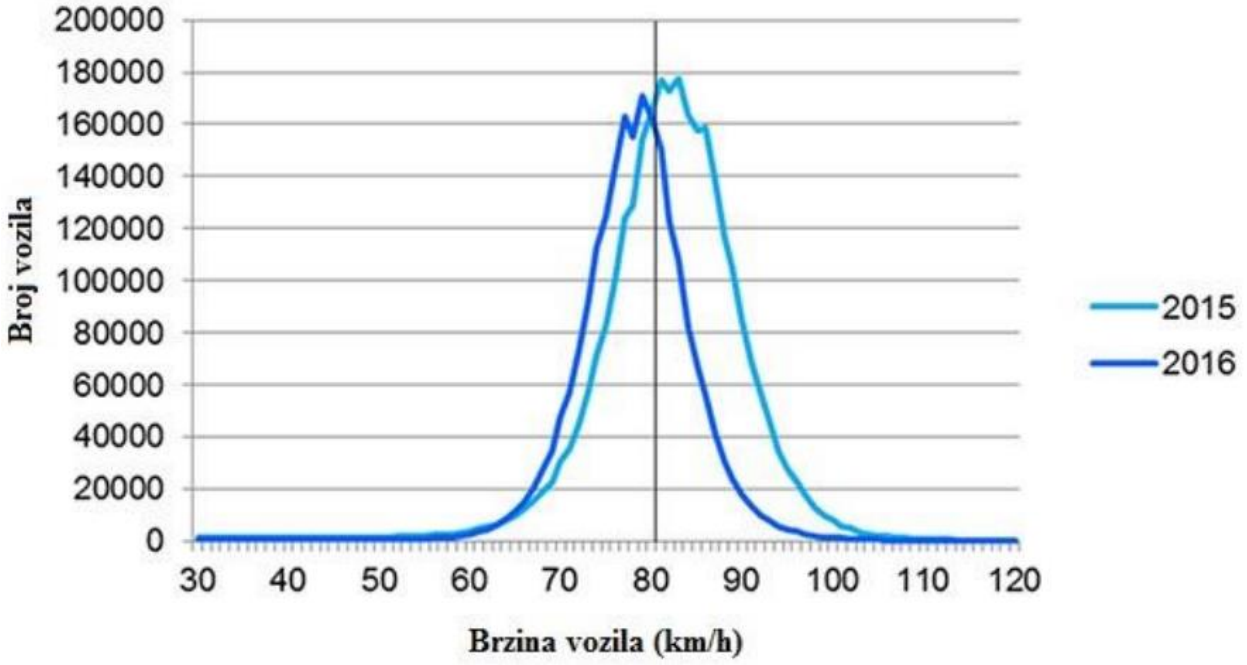
Internet stvari (eng. 'Internet of Things', IoT) opisuje proces umrežavanja uređaja i njihovu međusobnu interakciju korištenjem digitalnih komunikacijskih i mrežnih tehnologija.

Bitar (2018) navodi da internet stvari obuhvaća niz pametnih uređaja koji aktivno sudjeluju u komunikaciji te kao primjere navodi senzore, računalne resurse, uređaje za pohranu podataka i uređaje za komunikaciju s korisnicima.

Primjena interneta stvari sveprisutna je u mnogim industrijskim granama, a svoju primjenu nalazi i u području prometa i logistike gdje omogućava interakciju između uređaja u realnom vremenu čime se eliminira nepotrebna interakcija i fizički prijenos podataka.

U području prometa, internet stvari predstavlja temeljni strukturni element uspostave inteligentnih transportnih sustava (ITS) na nekom području kroz umrežavanje vozila i prometne infrastrukture. Primjeri upotrebe interneta stvari u prometu su automatska regulacija semafora kroz detekciju opterećenja prometnih tokova, ažuriranje voznog reda usluge javnog prijevoza sukladno zastojima i kašnjenju, automatska naplata cestarine, regulacija dozvoljene brzine kretanja putem digitalnih panela kroz praćenje vremenskih uvjeta koji mogu narušiti sigurnost vožnje i slično. Uvođenje digitalnih tehnologija u područje prometa pokazalo se kao efektivan način uspostave sigurnijeg sustava kao što je i prikazano na Grafikonu 4.1 kroz promatranje brzine vozila prije uvođenja sustava automatske kontrole brzine (2015) i nakon uvođenja spomenutog (2016).

**Grafikon 4.1 'Utjecaj sustava automatskog provođenja kontrole brzine na brzinu vožnje'**




**Izvor:** Malin F. (2017): 'Dense automatic speed enffectively reduces speeding', VTT Technical Research Centre of Finland, Oulu, Finska, prema Vrbanić (2017).

Tehnologija interneta stvari koristi se i u operativnom djelovanju autonomnim viličara koji odrađuju operativne zadatke prihvata i otpreme robe izbjegavajući koliziju s ostalim dionicima u procesu korištenjem senzorske tehnologije i algoritama strojnog učenja koji u realnom vremenu detektiraju moguće točke kolizije i sukladno tome redefiniraju rutu kojom se vozila kreću. Primjer spomenutih autonomnih viličara prikazan je na Slici 4.4.

*Slika 4.4 'Autonomni roboti u području logistike',*
*Izvor: https://www.debug.hr/gideon-brothers-jedan-je-od-vodecih-inovatora-u-podrucju-industrijske-robotike/*

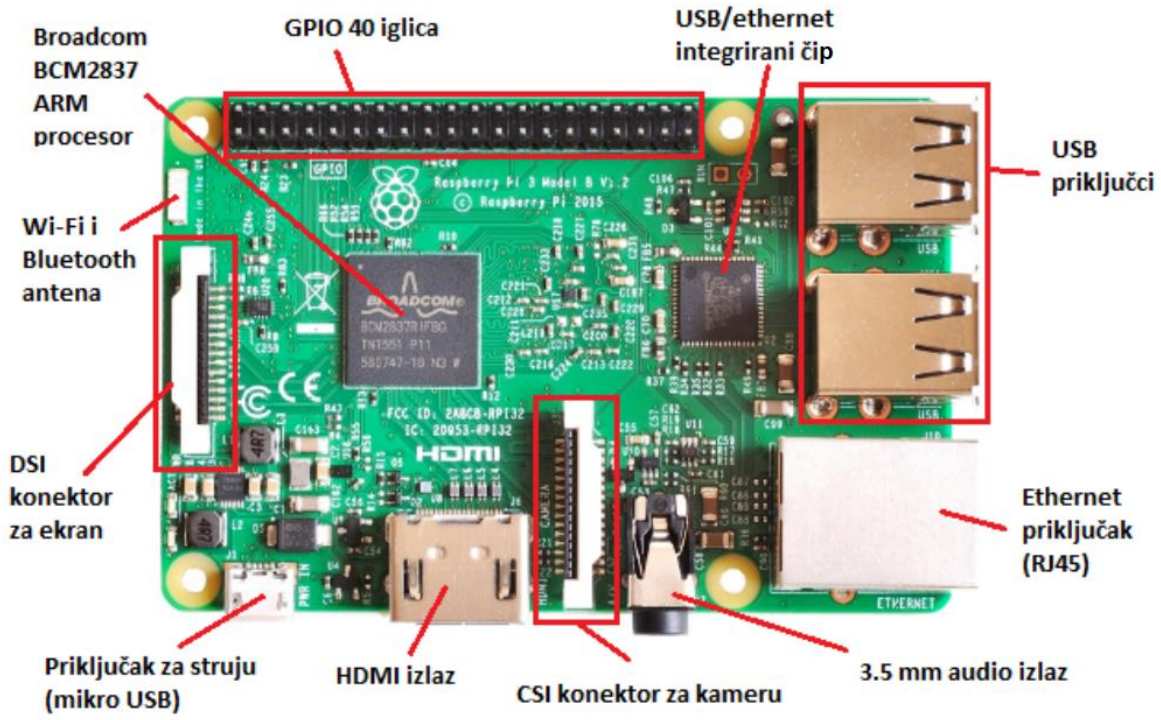
Glavni benefit uvođenja autonomnih viličara je neovisnost o zaposleniku koji upravlja standardnim viličarom. Zaposlenik na viličaru može se ozlijediti ili prekasno uočiti ostale zaposlenike u skladištu koji mu se mogu naći van polja preglednosti, odnosno u mrtvom kutu, dok autonomni viličar kontinuirano analizira stanje oko sebe i može reagirati u gotovo istom trenutku ako prepozna prijetnju nekim od senzora. Uvođenje autonomnih vozila i zamjena tradicionalnih radnih mjesta rješenjima koje koriste umjetnu inteligenciju postavlja brojna etička pitanja, što je obrađeno u posebnom dijelu rada, odnosno poglavlju 6.

Razvoj interneta stvari otvara put brojnim inovacijama, pri čemu se posebno ističu *Pametne kuće* u kojima se prema analizi okoline automatski izvršavaju operacije poput spuštanja i podizanja roleta, uključivanja sigurnosnog sustava pri napuštanju doma, automatske regulacije topline i vlažnosti, zalijevanja biljki, regulacije svjetlosti i slično. Područja robotike, elektrotehnike i strojarstva također doživljavaju svojevrsnu revoluciju zbog razvoja tehnologije interneta stvari jer su potrebe korisnika za umrežavanjem vlastitih uređaja i kontrolom putem jednog klika postale normativ, čemu se poduzeća u spomenutim područjima moraju prilagoditi.

Funkcionalnost uređaja interneta stvari ovisi o njegovom programskom rješenju i odabranom hardverskom uređaju. Dva primjera minijaturnih računala koja omogućavaju vlastitu izradu uređaja temeljenih na tehnologiji interneta stvari su *Raspberry Pi* (RPi) i *Arduino*.

Perko (2019) za *Raspberry Pi* navodi da je ono 'računalo razvijeno u Cambridgeshireu u Velikoj Britaniji od strane *Raspberry Pi Foundationa*' pri čemu se 'za kontrolu resursa najčešće koristi *Linux* operativni sustav, a preporučen je i *Raspbian OS'* za koji Redžić (2018) navodi da je nastao na temelju *Debian* operativnog sustava.

*Raspberry Pi* podupire korištenje programskog jezika Python pri programiranju uređaja čime se bitno olakšava proces izrade programskog rješenja zbog velikom broja raspoloživih biblioteka.




*Slika 4.5 'Raspberry Pi uređaji njegovi elementi',*
*Izvor: https://zir.nsk.hr/islandora/object/mathos%3A192/datastream/PDF/view*

Živković (2017) navodi da *Raspberry Pi* uređaj ima gotovo identične komponente kao i standardno računalo, ali na samo jednoj pločici veličine bankovne kartice.

Prednost veličine *Raspberry Pi* uređaja je svakako mogućnost integracije u različite uređaje s značajkama koje su dostupne i na osobnom računalu koje zauzima veću površinu. Sam uređaj posjeduje konektor za kameru, audio ulaz, *Wi-Fi* antenu i *Bluetooth* antenu, što ga čini podložnim za programiranje i integraciju elemenata umjetne inteligencije poput prepoznavanja objekata, dešifriranja glasovnih poruka i detekcije ljudskog pokreta. U kontekstu prometnog sustava može se koristiti za simuliranje semaforske regulacije prometa na minijaturnom modelu prije konačne implementacije rješenja u svakodnevnom prometu.

## 4.4. Pametne tvornice

Pametne tvornice (eng. 'Smart factories') predstavljaju 'fleksibilan sustav koji može sam optimizirati značajke širom mreže, samostalno se prilagođavati i učiti iz novih uvjeta u stvarnom ili skoro stvarnom vremenu te autonomno pokretati čitave proizvodne procese.' (Burke et al., 2017 prema Mrkonji, 2020)

Pametne tvornice objedinjuju više različitih tehnologija Industrije 4.0 poput umjetne inteligencije, strojnog učenja, interneta stvari, računarstva u *oblaku*, komunikacije u realnom vremenu i velikih podataka u svrhu optimizacije radnih procesa i kreiranja podloge za izradu novih digitalnih usluga i proizvoda. Prikupljeni podaci iz radnih procesa vrlo su često zanemareni kapital organizacije, ali mogu poslužiti za razvoj informacijskog sustava ljudskih potencijala organizacije, optimalnu narudžbu repromaterijala i planiranje proizvodnje.
Buntak (2016) navodi da se kompetentnost organizacije bazira na:

- kompetentnosti ljudskog potencijala,
- tehničkoj kompetentnosti,
- tehnološkoj kompetentnosti
- i strukturnoj kompetentnosti.

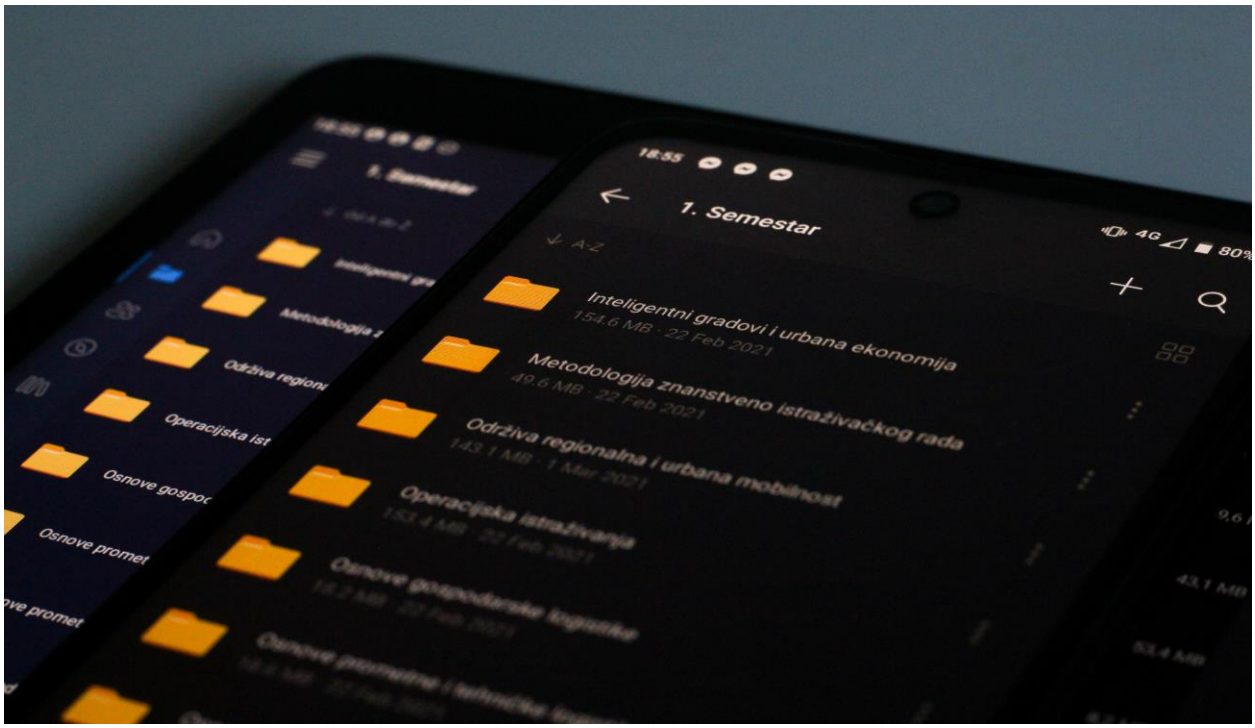
Iz navedene raščlambe kompetentnosti organizacije zaključuje se da koncept pametne tvornice mora uključivati niz znanja i kompetencija svojih zaposlenika uz pripadajuću tehniku, tehnologiju i strukturnu kompetentnost kako bi polučio maksimalni učinak jer svaki sustav ovisi o pripadajućim vanjskim i unutarnjim čimbenicima koji utječu na efektivnost pri ostvarenja ciljeva. Transformacija tradicionalne tvornice, ili organizacije bilo kojeg tipa, u pametnu tvornicu zahtijeva visoku razinu ulaganja u informacijski sustav, digitalne tehnologije i uređaje te obrazovanje zaposlenika. Transformacija postojećeg poslovnog modela može se negativno percipirati zbog potrebe usvajanja novih znanja kod zaposlenika koji nisu skloni cjeloživotnom obrazovanju, ali ona je imperativ za opstanak organizacije na tržištu i njezin daljnji razvoj.
Veliku ulogu u stvaranju preduvjeta za transformaciju tvornica ima i lokalna vlast o kojoj često ovisi i izgradnja infrastrukture koja omogućava prijenos informacija u realnom vremenu korištenjem optičke mreže i 5G tehnologije što je posebno izazovno u ruralnim krajevima.

Kao glavna pozitivna značajka pametnih tvornica spominje se njezina agilnost i mogućnost odgovora na promjene na tržištu, pri čemu Oulovsky (2018) navodi da 'prava snaga pametne tvornice leži u njenoj sposobnosti da se razvija i raste zajedno s promjenjivim potrebama organizacije, bilo da se mijenja potražnja kupaca, širenje na nova tržišta, razvoj novih proizvoda ili usluga, predvidljiviji i osjetljiviji pristup poslovanju i održavanju, ugradnja novih procesa ili tehnologija, ili promjene u proizvodnji.'

## 4.5. Računarstvo u oblaku

Računarstvo u *oblaku* (eng. '*Cloud* computing') je usluga koja podrazumijeva različite vrste interakcije (pohrane, preuzimanja, rada u realnom vremenu) sa sadržajem koji je povezan sa infrastrukturom (serverom) ponuditelja usluge mrežnim putem.

Kurelović et al. (2014) navode da su 'stručnjaci odabrali pojam *oblak* jer se koriste resursi virtualnih računala, točnije mrežnih poslužitelja čija točna lokacija nije poznata' pri čemu je '*oblak* ujedno i metafora za internet.'




*Slika 4.6 'Primjer umrežavanja mobilnog uređaja i tableta korištenjem tehnologije računarstva u oblaku i aplikacije OneDrive za potrebe fakultetskog obrazovanja',*
*Izvor: Fotografirao autor*

Usluga računarstva u *oblaku* omogućava pohranu sadržaja na internetskoj platformi ili u kombiniranom obliku koji istovremeno sadržaj pohranjuje i u fizičkom obliku na vlastitom računalu. Primjeri usluge računarstva u *oblaku* su *Google Disk, Microsoft One Drive, Dropbox, The Box, Moodle, iCloud* i *Amazon Web Services.*

Upotrebom usluge računarstva u *oblaku* smanjuje se potreba za fizičkim prijenosom podataka putem eksternih tvrdih diskova ili memorijskih štapića jer je sadržaj u realnom vremenu dostupan putem platforme uz postojanje stabilne i brze internetske mreže. Pristup sadržaju na platformi moguće je postići i korištenjem pametnih telefona i tableta čime korisnik usluge postaje neovisan o poslovnom računalu i proširuju mu se opcije za mobilnost i putovanje.

Razvojem računarstva u *oblaku* pojavljuju se i platforme za *e-učenje* koje svojim polaznicima edukacijskih programa nude mogućnost pristupa nastavnom sadržaju i prilaganje rješenja problemskih zadataka i zadaća za koje se vrlo često dodjeljuju *e-certifikati* kao dokaz o polaganju i ispunjenju svih potrebnih zahtjeva za njihovim stjecanjem.

Značaj primjene usluga računarstva u *oblaku* posebno je prepoznat u vrijeme pandemije SARSa-COV-2 zbog nametnutih restrikcija kojima se ograničava kretanje stanovništva i potiče rad od kuće. Brojna poduzeća su zbog spomenutih okolnosti morala pronaći alternativne načine poslovanja kako bi očuvala vlastitu vitalnost zbog otežanog pristupa podacima i interakcije između zaposlenika, pri čemu se usluga računarstva u *oblaku* nameće kao logičan izbor.

Korištenjem usluge računarstva u *oblaku* organizacije se mogu izložiti rizicima od hakerskih napada koji rezultiraju krađom ili zloupotrebom podataka zbog čega je potrebno sustavno ulagati u sigurnosne standarde i protokole korištenja usluge od strane svih dionika u procesu.

Zovko (2017) navodi da postoje tri različita modela usluga računarstva u *oblaku*:

- **Softver kao servis (eng. 'Software-as-a-Service', SaaS)**
  korisnik *oblaka* kontrolira samo konfiguracije aplikacija,
- **Platforma kao servis (eng. 'Platform-as-a-Service', PaaS)**
  korisnik *oblaka* također kontrolira hosting okruženja,
- i **Infrastruktura kao servis (engl. 'Infrastructure-as-a-Service', IaaS)**
  korisnik *oblaka* kontrolira sve osim infrastrukture podatkovnih centara. (Zovko, 2017)

CERT CARNet (2010) prema Kurelović et al. (2014) navode da svaki od spomenutih modela usluga može biti realiziran korištenjem nekog od četiri raspoloživih vlasničkih tipova *oblaka*:

- javni *oblak*,
- privatni *oblak*,
- hibridni *oblak*
- i zajednički *oblak*.

Spomenute vrste računarstva u *oblaku* razlikuju se po sigurnosti pristupa podacima, održavanju infrastrukture i vlasništvu nad infrastrukturom i resursima. Chen et al. (2010) prema Zovko (2017) javni *oblak* definiraju kao *oblak* koji je dostupan široj javnosti, zajednički *oblak* kao *oblak* koji koristi nekoliko različitih organizacija, privatni *oblak* koji je ograničen samo na jednu organizaciju i hibridni *oblak* koji je mješavina različitih *oblaka*. Iz navedenog se zaključuje da se organizacije odlučuju za različite tipove *oblaka* uvjetovanih njihovim sigurnosnim politikama, financijskom sposobnosti investiranja u digitalne tehnologije i stručnosti vlastitog kadra koji je zadužen za implementaciju usluge i njezino održavanje.

## 4.6. Upravljanje odnosima s klijentima

Upravljanje odnosima s klijentima (eng. 'Customer relationship management', CRM) odnosi se na evaluaciju, održavanje i poboljšanje odnosa s ključnim dionicima poduzeća kao preduvjeta za održivo poslovanje i preciznost poslovnog planiranja.

Buckingham (2011) prema Dukić i Gale (2015) navodi da postoji sedam razloga zbog kojih je potrebno graditi partnerski odnos s klijentima:

- ne treba tražiti nove kupce,
- prodaja se povećava,
- jača se tržišna pozicija,
- povećava se vjernost potrošača,
- smanjuju se poslovni troškovi,
- povećava se dobit
- i povećava se uživanje i zadovoljstvo svakodnevnog posla.

Kumar i Reinartz (2018) navode da lojalni klijenti poduzeća često eksploatiraju svoj položaj, tražeći pritom posebne pogodnosti i tretman, akcijske cijene i kontinuiranu komunikaciju što u konačnici rezultira većim troškovima od stvarne vrijednosti poslovnog odnosa.

Poduzeća kontinuirano evaluiraju klijente s kojima surađuju kako bi prilagodila svoj odnos prema istinski vrijednim partnerskim odnosima pri čemu digitalne tehnologije omogućavaju neposredni kontakt i povratnu informaciju u realnom vremenu.

U digitalne usluge nerijetko se implementiraju računalni *botovi* koji odgovaraju na najčešće postavljena pitanja, a ako klijent odluči potražiti dodatne informacije, spomenuti *bot* može ga usmjeriti prema specijaliziranom zaposleniku ili odjelu za korisničku podršku koji detaljno obrađuju zahtjev klijenta. Spomenuta procedura bitno smanjuje opterećenje odjela za korisničku podršku, ali ono može izazvati percepciju zakinutosti kod klijenta jer se eliminira izravni kontakt između njega i poduzeća.

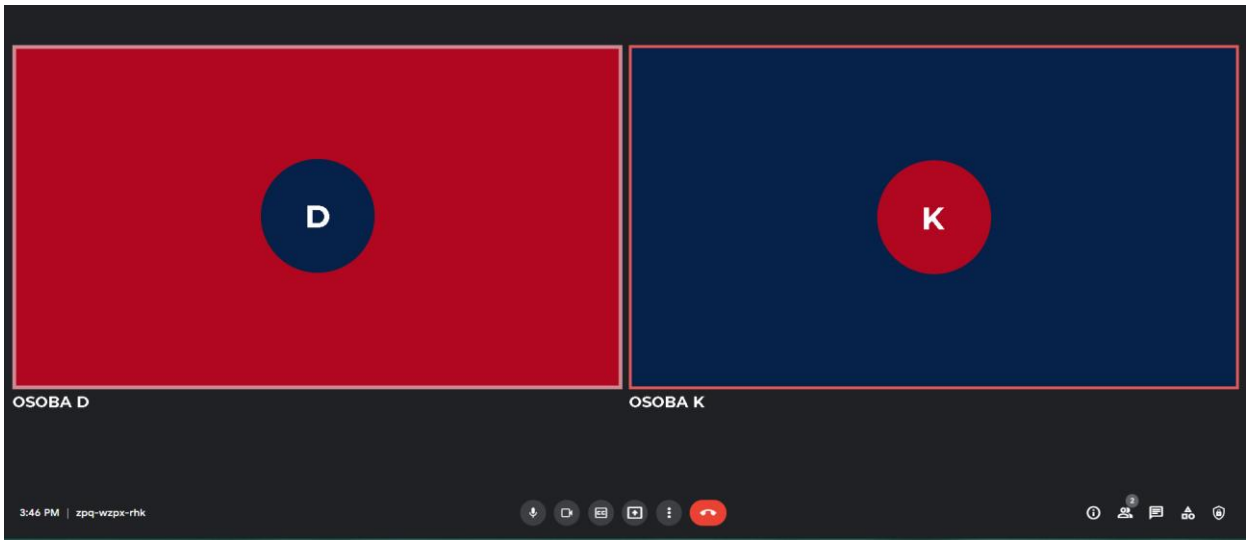
Digitalizacijom usluga koje poduzeće pruža širi se njegova dostupnost i mogućnost interakcije s klijentima, pri čemu se individualizira odnos sukladno njegovim preferencijama. Individualizirani odnos prema klijentu može izazvati pozitivne emocije i osjećaj posebnosti, razvijajući pritom dugotrajan i stabilan poslovni odnos koji je manje podložan raskidanjem suradnje u slučaju manjih propusta.

Za osiguranje efikasnosti i efektivnosti digitalnog rješenja koje identificira preferencije klijenata potrebno je kontinuirano prikupljati podatke i sukladno tome ažurirati modele predviđanja temeljnih na algoritmima strojnog učenja kako bi predloženi sadržaj bio sezonski prilagođen, bolji od konkurentske ponude i relevantan krajnjem korisniku usluge ili proizvoda.

## 4.7. Komunikacija u realnom vremenu i virtualni sastanci

Komunikacija u realnom vremenu (eng. 'Real-time communication', RTC) popularizirana je pojavom aplikacije *Skype*, ali svoj puni potencijal i prepoznatljivost u poslovnom i edukacijskom segmentu, kao i prethodno spomenuta usluga računarstva u *oblaku*, doživljava pojavom pandemije SARSa-COV-2 zbog ograničenja kretanja i interakcije između ljudi.

Neki od primjera programa koji se koriste u komunikaciji u realnom vremenu su *Google Meet, Zoom, Microsoft Teams* i *Discord.*




*Slika 4.7 'Primjer sučelja za komunikaciju u realnom vremenu - platforma Google Meet', Izvor: Vlastiti rad autora – adaptirani snimak zaslona*

Slika 4.7 prikazuje simuliranu komunikaciju u realnom vremenu između Osobe D i Osobe K korištenjem platforme *Google Meet*. Virtualni sastanci sve se češće upotrebljavaju u poslovnom svijetu zbog geografske neovisnosti o lokaciji njihovih održavanja i nepostojanja putnih troškova, ali uz preduvjet postojanja stabilne i brze internetske veze te uređaja s kamerom i mikrofonom. Pojavom virtualnih sastanaka postavlja se pitanje subvencije energenata i priključka internetskoj vezi od strane poduzeća prema zaposlenicima zbog korištenja njihovih vlastitih resursa.

Uz brojne prednosti virtualnih sastanaka bitno je naglasiti i negativne značajke poput otežanog dešifriranja poruka i percipiranja neverbalne komunikacije, društvene izolacije, tehničkih poteškoća, odvlačenja pažnje sadržajem na uređaju i slično. Virtualno održavanje nastave na daljinu posebno je štetno za psihičko i fizičko zdravlje mlađih naraštaja koji se kroz društvenu interakciju uče biti aktivnim sudionicima društva. Karl et al. (2022) provedenim istraživanjem o virtualnim sastancima dolaze do zaključka da mnogim zaposlenicima nedostaje svijest o ozbiljnosti sastanaka te da su nepripremljeni za rad u virtualnom okruženju te da organizacije moraju predstaviti svoja očekivanja od planiranih sastanaka, kao i norme ponašanja.

Istraživanje Karla et al. (2022) pokazuje da korisnici ističu druženje, postojanje opcije tekstualnog razgovora i dublje upoznavanje s okruženjem ostalih zaposlenika (sobe, obitelji) kao pozitivne karakteristike, a kašnjenje, nedostatak dnevnog reda, dugi sastanci i višezadaćnost[46] kao negativne karakteristike provođenja virtualnih sastanaka.


*Slika 4.8 'Kretanje vrijednosti dionice poduzeća Zoom Video Communications INC',*
*Izvor: Vlastiti rad autora – adaptirani snimak zaslona prema Google Finance prikazu*

Virtualni sastanci pojavom pandemije izazvane SARSom-COV-2 postaju uobičajeni oblik komunikacije u realnom vremenu, a to se značajno odražava na vrijednost dionica poduzeća koja nude spomenute usluge. Slika 4.8 prikazuje primjer rasta vrijednosti dionice poduzeća *Zoom Video Communications INC* gdje točka *T1* označava vrijednost dionice u vremenu proglašenja globalne pandemije (107,47 USD / 753,82 HRK), a točka *T2* najvišu dosegnutu vrijednost dionice od 559 USD, odnosno 3920.94 HRK[47].

Vrijednost dionice spomenutog poduzeća u sedam je mjeseci narasla za više od pet puta u odnosu na inicijalnu vrijednost zabilježenu na početku pandemije.

Evans (2020) prema Karl et al. (2022) navodi da platforma *Zoom Video Communications INC* u prosincu 2019. godine bilježi oko 10 milijuna svakodnevnih korisnika, a u travnju 2020. godine taj je broj narastao na preko 300 milijuna svakodnevnih korisnika.

[46] eng. 'multitasking' – pojam koji opisuje odrađivanje nekoliko aktivnosti u isto vrijeme.

[47] prema tečaju 1 USD = 7.01 HRK.

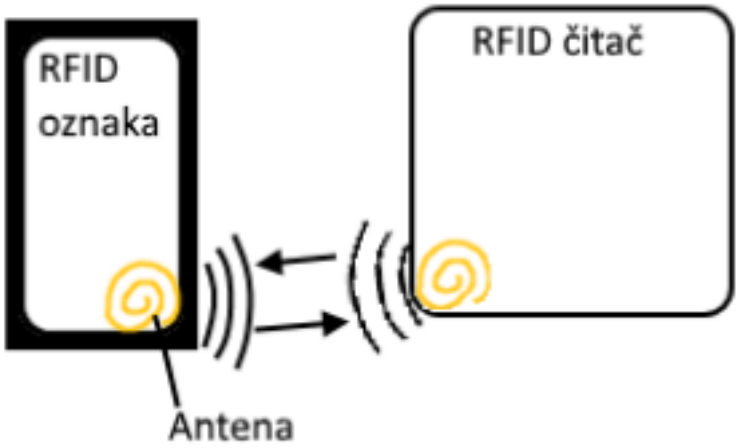
## 4.8. RFID i NFC tehnologija

RFID (eng. 'Radio Frequency Identification')[48] je 'takozvana *not in line of sight* tehnologija koja omogućava identifikaciju objekata u blizini koji ne moraju nužno biti vidljivi.' (Tomić, 2021)

Tomić (2021) također navodi da je RFID tehnologija slična barkod tehnologiji, uz mogućnost identifikacije nekolicine objekata istovremeno, kao i ažuriranja podataka što kod barkod tehnologije nije slučaj.

Primjer korištenja RFID tehnologije u području poboljšanja usluge prometa je uvođenje RFID kartica koje omogućavaju olakšanu identifikaciju putnika, uvid u njihovu razinu subvencija (studenata, učenika, umirovljenika i slično) te prikaz putničke karte koja je unesena na blagajni.

RFID čitač putem radio valova vrši interakciju sa objektima koji sadržavaju RFID oznaku, pri čemu oni sami ne moraju nužno biti vidljivi. Slika 4.9 prikazuje interakciju RFID čitača i RFID oznake koja sadrži antenu.




*Slika 4.9 'RFID identifikacija',*
*Izvor: https://zir.nsk.hr/islandora/object/etfos%3A2743/datastream/PDF/view*

Područje logistike posebno je plodno za implementaciju RFID tehnologije zbog kontrole unosa robe u skladišni prostor i pohrane informacija o njezinom vlasništvu te roku isporuke.

RFID tehnologijom omogućava se nadzor nad protokom robe unutar i van skladišta što doprinosi automatskom ažuriranju podataka bez potrebe ljudskog ophođenja uz preduvjet dizajniranja infrastrukture s visokom pouzdanosti detekcije i usklađenja emitiranih radio valova kojima RFID čitač detektira objekte u okolini. U području prometa RFID čitač pri semaforskoj regulaciji može identificirati hitne službe ili vozila JGP-a na putanji i dati im prednost prolaska.

[48] hrv. 'identifikacija radijske frekvencije'.

Zrnić (2020) navodi da se RFID oznake dijele na aktivne i pasivne, a kao ključne razlike navodi izvor napajanja i razinu dometa pri čemu pasivne oznake sadrže integrirani sklop i antenu te se aktiviraju energijom elektromagnetskog polja čitača, dok aktivne oznake posjeduju vlastiti izvor napajanja što im omogućava veći domet, ali i potencijalno manju ekonomičnost i neprikladnu veličinu u odnosu na pasivne oznake.

Prema Zrniću (2020) za prijenos informacija aktivnih oznaka koriste se frekvencije od 433 MHz ili 915 MHz, a za prijenos informacija pasivnih oznaka koriste se:

- niska frekvencija od 125 do 134 KHz,
- visoka frekvencija ili NFC[49] 13.56 MHz (najraširenije korištena)
- i ultra visoka frekvencija od 865 do 960 MHz.

## 4.9. Tehnologija lanca blokova

Tehnologija lanca blokova (eng 'Blockchain') je 'zajednička, raspodijeljena knjiga koja olakšava proces evidentiranja transakcija i praćenja imovine u poslovnim mrežama' (Gupta, 2017 prema Pejčić, 2020). Tehnologija lanca blokova bilježi rapidni rast u svojoj primjeni u brojnim industrijama poput financija, sigurnosti i digitalne umjetnosti.

Živković (2018) navodi da postoje tri vrste mreže lanca blokova:

- javna mreža,
- privatna mreža
- i konzorcijska mreža.

Javna mreža lanca blokova široko je dostupna i svatko tko ima internetski pristup može joj pristupiti (koriste je kriptovalute *Blockchain* i *Ethereum*), privatna mreža lanca blokova ograničavajuća je jer traži odobrenje administratora dok konzorcijsku mrežu lanca blokova čini niz različitih organizacija koja se njome koriste. (Živković, 2018)

Razlog široke primjene tehnologije lanca blokova leži u osiguranju podataka od krađe i zlonamjernog djelovanja na podatke, a Šimunjak (2021) opisuje proces njezina funkcioniranja kao linearno i kronološko pohranjivanje blokova koji su teško promjenjivi zbog posjedovanja *hash* koda koji je povezan s blokom prije njega i njegovom vremenskom oznakom.

Frankenfield (2022) *hash* kod definira kao funkciju koja zadovoljava zahtjeve procesa lanca blokova, a njome se postiže potrebno računanje u postupku dešifriranja zahtjeva te je gotovo nemoguće pogoditi duljinu samog *hash* koda.

[49] eng 'near-field communication' – komunikacija u bliskom polju (očitavanje s bliske udaljenosti).

### 4.9.1. Kriptovalute i NFT

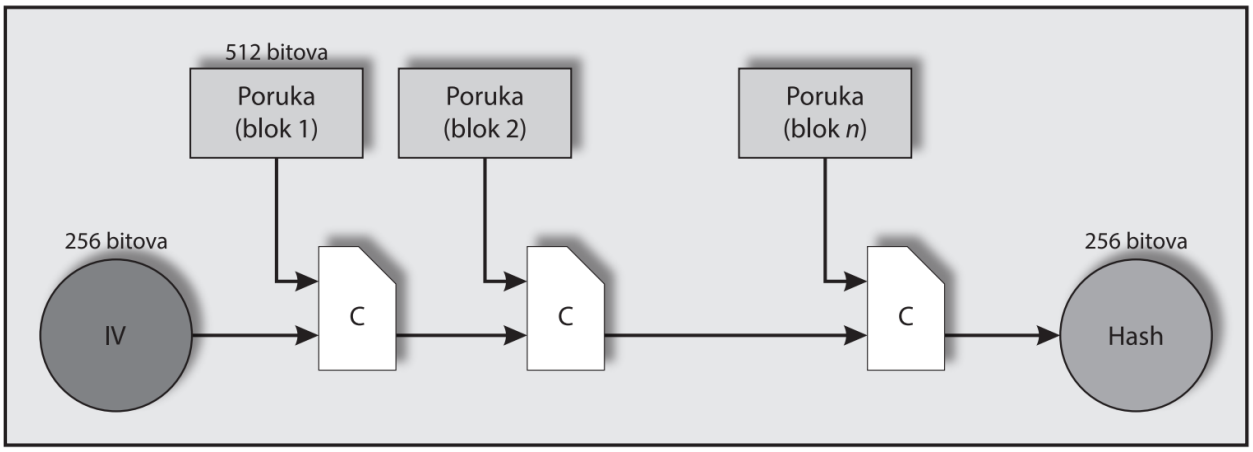
Pojam tehnologije lanca blokova široj je javnosti populariziran pojavom kriptovaluta[50] (*Bitcoin, Ethereum, Dogecoin, Terra Luna*) zbog njihovog koncepta koji se bazira na posjedovanju imovine u virtualnom svijetu, koristeći se pritom najvišim sigurnosnim standardima koji se baziraju na enkripciji podataka.

'Kriptovaluta je ime dano nekom sustavu koji upotrebljava kriptografiju kako bi omogućio siguran transfer i razmjenu digitalnih tokena na distribuiran i decentraliziran način, pri čemu se ti tokenu mogu mijenjati za standardne valute po njihovim uobičajenim tržišnim vrijednostima.'

(Dourado i Brito, 2014 prema Turudić et. al, 2017).

Pojavom kriptovaluta otvara se prilika za novi način ulaganja koji konkurira kupovini i iznajmljivanju nekretnina i pokretnina te ostalim oblicima investiranja na tržištu kapitala poput trgovanja dionicama. Brojna poduzeća omogućavaju plaćanje u kriptovalutama, ali postoje i primjeri odustajanja od spomenutog zbog ekoloških prijetnji koje proizlaze iz tzv. *rudarenja* kriptovaluta s obzirom na potrošnju energije i povećanje cijena računalnih komponenti.

*Rudarenje* kriptovaluta doprinijelo je nestašici grafičkih kartica na tržištu u 2021. godini.




*Slika 4.10 'SHA-256 protokol',*
*Izvor: https://hrcak.srce.hr/file/282079*

Slika 4.10 prikazuje pozadinu funkcioniranja lanca blokova u kojem se koristiti često upotrebljavani SHA[51]-256 protokol enkripcije, za koji Turudić et al. (2017) navode da 'omogućava usporedbu trenutačnog i prijašnjeg stanja podataka kako bi se spriječila neželjena modifikacija od strane neautoriziranih aktera.

[50] eng. 'cryptocurrency'.

[51] eng. 'secure hash algorithm' – algoritam sigurnog 'hasha'.

Turudić et al. (2017) spominju ulogu ključeva u posjedovanju određenog segmenta lanca blokova pri čemu 'ključ dodjeljuje vlasništvo svakog para ključeva, ili ‚kovanice', osobi koja je u posjedu privatnog ključa, a parovi ključeva su pohranjeni u datoteci imena *wallet.dat*, koja egzistira u uobičajenom skrivenom direktoriju na tvrdom disku.'

Razvoj tehnologije i primjena protokola dovodi do pojave NFT[52] načina akvizicije imovine u digitalnoj sferi pri čemu se ostvaruje vlasništvo nad dijelom lanca blokova nekog digitalnog proizvoda poput virtualnih umjetnina, fotografija, dizajna i sličnog. (Franceschet et al., 2020 prema Wang et al., 2021)

Wang et al. (2021) navode da je glavna razlika između kriptovaluta poput *Bitcoina* i NFT-a to da su standardne kriptovalute nerazlučive i ujednačene, dok je NFT jedinstven i nije zamjenjiv, što ga čini prikladnim za povezivanjem i indeksiranjem s nekim oblikom digitalne imovine.

## 4.10. Proširena stvarnost

Proširena stvarnosti (XR)[53] obuhvaća niz tehnologija koje za svrhu imaju adaptaciju i nadogradnju postojećeg stanja okoline i njezina transferiranja u digitaliziranu ili djelomično digitaliziranu sferu, a Kaplan et al. (2021) navode da je ona krovni pojam koji objedinjuje virtualnu stvarnost, pojačanu stvarnost i mješovitu stvarnost, iako se pojam proširene stvarnosti često poistovjećuje s pojačanom stvarnosti.

Virtualna stvarnost (VR)[54] digitalna je tehnologija koja se bazira na kreiranju virtualnog prostora u kojem se korištenjem uređaja vrši interakcija s okolinom. Uređaji koji se koriste u procesu bitno utječu na kvalitetu doživljaja virtualne okoline pri čemu se koriste različiti modeli VR naočala koji u kombinaciji s pomagalima omogućavaju i interakciju sa elementima virtualnog prostora, odnosno personaliziran doživljaj simulacije u virtualnom prostoru.

Tehnologija virtualne stvarnosti često se koristi u medicini kroz simuliranje operacija i obuku medicinskog osoblja, u muzejima kroz virtualnu šetnju, u obogaćivanju turističke ponude kroz retrospektivni pogled u prošlost nekog područja te za detaljno prezentiranje dizajnerskog rješenja prenamjene postojeće ili izgradnje nove građevine i njezina interijera.

Optimizacija logističkih sustava može se postići korištenjem *FlexSim* programskog rješenja koje omogućava virtualnu šetnju u skladišnom prostoru i interakciju s proizvodima na proizvodnim trakama, a nakon čega je moguće pristupiti statističkim podacima vezanim uz obavljene operacije koji služe kao podloga za normizaciju i uređenje pitanja radnog opterećenja.

---

[52] eng. 'non-fungible token' – token koji se ne može zamijeniti (vrsta kriptovalute).

[53] eng. 'extended reality'.

[54] eng. 'virtual reality'.

U prometnim se sustavima testiraju rješenja vezana uz optimizaciju prometa i povećanje sigurnosti njegovih dionika poput semaforske regulacije kojom je potrebno prilagoditi vrijeme prolaska automobila, bicikala i pješaka preko prometne infrastrukture u virtualnom okruženju prije implementacije rješenja u realno okruženje. Ovom se metodom vrši mitigacija velikog broja rizika jer se sustav detaljno promišlja iz strane korisnika umjesto prometnog planera i to prije same izrade, odnosno, osposobljavanja infrastrukture i njezinog puštanja u rad.

Pojačana stvarnost (AR)[55] nadogradnja je stvarne okoline dodavanjem virtualnih elemenata kojima se pristupa korištenjem uređaja. Specifičnost pojačane stvarnosti je njezina agilnost i široka mogućnost primjene u realnom svijetu s obzirom na to da joj se može pristupiti korištenjem mobilnog uređaja skeniranjem QR[56] koda i u realnom vremenu adaptirati okolinu dodatnim informacijama, zabavnim, turističkim, edukativnim i ostalim sadržajem.

Razvojem pojačane stvarnosti omogućena je navigacija kroz prostor u realnom vremenu s implementacijom virtualnih znakova koji usmjeravaju korisnika pri kretanju prema željenoj destinaciji, a upozoravaju ga i o potencijalnim rizicima i preprekama na odabranoj ruti.

Pojačana i virtualna stvarnost razlikuju se u vrsti okoline, odnosno, virtualna stvarnost pruža iskustvo korisniku u virtualnoj okolini, dok je pojačana stvarnost nadogradnja realne okoline.

Mješovita stvarnost (MR)[57] prema Kaplan et al. (2021) obuhvaća kombinaciju proširene i mješovite stvarnosti, odnosno, udruživanje elemenata stvarnog i virtualnog svijeta.

Društvene mreže i digitalni sadržaj poprimaju novu dimenziju poslovanja zbog pojave spomenutih oblika stvarnosti kojima se trebaju prilagoditi. *Metaverzum*[58] je pojam koji opisuje kombinaciju više oblika proširene stvarnosti i iz sadašnje perspektive djeluje futuristički, iako već postoje koncepti eksploatacije virtualnog prostora za potrebe oglašavanja. *Metaverzum* omogućava isprobavanje odjeće i obuće, što značajno povećava doseg i pouzdanost online trgovina koje ih prodaju, a planira se i korištenje tehnologija za simulaciju životnih događaja koje je moguće ponovno doživjeti u prostoru, poput primjerice vjenčanja ili koncerata.

Korištenjem tehnologija proširene stvarnosti također se doprinosi mogućnostima primjene gemifikacije koja za rezultat može imati izbjegavanje monotonosti okoline. Općeprihvaćena definicija gemifikacije prema Seabornu i Felsu (2015) je 'korištenje elemenata igre van njezina uobičajenog konteksta'. Nelson (2012) prema Seabornu i Felsu (2015) navodi da se u praksi koristi već dugi niz godina i da za rezultat ima poboljšanje produktivnosti i razvoj međuljudskih odnosa.

[55] eng. 'augmented reality'.

[56] eng. 'quick response'. – brzi odgovor (reakcija).

[57] eng. 'mixed reality'.

[58] eng. 'metaverse'.

# 5. Programski jezik Python

Ovaj rad temelji se na primjeni programskog jezika Python u procesu digitalne transformacije prometnih i logističkih sustava. U svim programskim rješenjima ovog rada korištenja je Python verzija 3.9.2 te programi *JupyterLab i Visual Studio Code*.

Pojašnjena je Python sintaksa i strukture podataka koji se koriste u programiranju. Prikazani stil oblikovanja koda uglavnom se temelji na PEP 8 preporukama, aktu *Zen Pythona* i načelima objektno orijentiranog programiranja.

Opisan je proces instalacije biblioteka korištenjem *pip* i *anaconda* upravitelja paketima korištenjem terminala računala te unos biblioteka, modula i objekata u programsko sučelje.

Prikazane su naredbe koje se koriste pri postavljanju uvjeta i obradi grešaka.

Navedeni su alati i biblioteke programskog jezika Python potrebni za programiranje algoritama strojnog učenja, kao i ostale biblioteke potrebne za izradu *CLI* aplikacija (programskog sučelja za automatizaciju rutinskih poslova u poslovanju korištenjem skripta koje se pokreću naredbom), te za vizualizaciju podataka na grafikonima i njihov prikaz u matričnom i tabličnom obliku.

Prezentirana je izrada *CLI* aplikacije bibliotekom *argparse,* te izrada iste vlastitom bibliotekom imena *duality* u terminalu uređaja korištenjem *dekorator-funkcija*.

Detaljna primjena programskog jezika Python i nekih od pripadajućih biblioteka navedenih u ovom poglavlju prikazana je u poglavlju 7.

U narednim poglavljima pojam 'programski jezik Python' oslovljava se korištenjem pojednostavljenog izraza 'Python'.

## 5.1. Uvodno o Pythonu

Šantić (2017) navodi da Python nastaje kasnih osamdesetih godina 20. stoljeća od strane nizozemskog programera Guida van Rossuma prema viziji o stvaranju novog programskog jezika koji je vizualno ugodan i jednostavan za korištenje. Lutz (2013) navodi da je Python nazvan po komediji BBC-a, *Letećeg cirkusa Montyja Pythona*.

Programska rješenja kreirana u Pythonu prepoznaju se po različitim ekstenzijama, među kojima se ističu standardna *.py* datoteka, *.pyw* datoteka i *.ipynb*[59] datoteka.

Python je programski jezik koji se često koristi u znanosti, automatizaciji, projektima umjetne inteligencije i strojnog učenja te pri rješavanju problemskih zadataka. Popularan je zbog lakoće čitanja programskog koda i široke baze korisnika koji mogu pružiti podršku.

[59] eng. 'interactive Python notebook' – interaktivna Python bilježnica.

## 5.2. Zen Pythona i PEP 8 smjernice

*Zen Pythona*[60] ugrađen je pri instalaciji programskog jezika na računalo, a sastoji se od načela pravilnog programiranja prema zajednici, i to kako slijedi:

'Lijepo je bolje nego ružno.

Eksplicitno je bolje nego implicitno.

Jednostavno je bolje nego složeno.

Kompleksno je bolje nego komplicirano.

Ravno je bolje nego ugniježđeno.

Prorijeđeno je bolje od gustog.

Čitljivost se računa.

Posebni slučajevi nisu dovoljno posebni da opravdaju kršenje pravila.

Iako praktičnost pobjeđuje čistoću.

Pogreške nikada ne bi trebale proći potiho.

Osim uz izričito ušutkavanje.

Suočavanjem s dvosmislenošću, odbijte napast za pogađanjem.

Trebao bi postojati jedan - i po mogućnosti samo jedan - očit način za učiniti to.

Iako taj način u početku možda nije očit, osim ako ste Nizozemci.

Sada je bolje nego nikad.

Iako je *nikad* često bolje od *upravo sada*.

Ako je implementaciju teško objasniti, radi se o lošoj ideji.

Ako je implementaciju lako objasniti, možda se radi o dobroj ideji.

Prostori za imena sjajna su ideja – napravimo ih još i više!'

(prevedeno prema Timu Petersu, 1999)

Navedenim smjernicama može se pristupiti korištenjem programskog koda:

```
import this
```

Detaljne smjernice programiranja u Pythonu propisane su PEP 8[61] dokumentom koji uređuje pitanja pravilnog programiranja poput korištenja tabulatora i razmaka za strukturiranje razina koda, korištenja pravilnih metoda u danom scenariju i za imenovanje objekata. Ujednačenim stilom programiranja postiže se jasnoća koda koji je univerzalno prepoznatljiv.

[60] eng. 'The zen of Python'.

[61] eng. 'Python enhancement proposal'.

## 5.3. Tipovi podataka u Pythonu

Prema Jalswal (2017) tipovi podataka u Pythonu dijele se na primitivne i neprimitivne. Primitivni tipovi podataka jednostavniji su od neprimitivnih i mogu biti njihovim sastavni dijelom, primjerice, lista može biti sačinjena skupa brojeva ili slova. Python je programski jezik koji se, poput *Jave* i *Rubya*, bazira na objektno orijentiranom pristupu što znači da se svi elementi smatraju objektima. Tipovi podataka prema podjeli Jalswal (2017) prikazani su u narednim poglavljima.

### 5.3.1. Osnovni primitivni tipovi podataka

Slika 5.1 prikazuje dodjeljivanje vrijednosti primitivnog tipa podataka varijabli. Analiza tipa varijable (objekta) vrši se ugrađenom funkcijom:

```
type()
```

```
[1]: obj_1 = 23
[2]: type(obj_1)
[2]: int
[3]: obj_2 = 23.6
[4]: type(obj_2)
[4]: float
[5]: obj_3 = 'Promet'
[6]: type(obj_3)
[6]: str
[7]: obj_4 = True
[8]: type(obj_4)
[8]: bool
```

*Slika 5.1 'Primitivni tipovi varijabli',*
*Izvor: Vlastiti rad autora*

Varijabli obj_1 dodijeljena je vrijednost 23 koja predstavlja cijeli broj, iskazan kraticom *int (integer).* Varijabli obj_2 pridodana je vrijednost 23.6, čime ona postaje broj s decimalom, prikazan pojmom *float*. Varijabla obj_3 označava 'Promet', što se smatra tekstom, a iskazuje se kraticom *str (string)*. Varijabli obj_4 dodijeljena je vrijednost *True* (hrv. Istina) što naslućuje da se u ovom primjeru radi o ispitivanju istinitosti varijable uz postojanje i suprotne vrijednosti *False* (hrv. Neistinito), a tip tih podataka prikazan je kraticom *bool (boolean).*

### 5.3.2. Rječnik

Rječnik je neprimitivni tip podataka kojim se pohranjuju vrijednosti u paru, a sastoji se od ključa i vrijednosti. Korisni su dodatak programskom jeziku jer omogućavaju pohranu podataka s pripadajućim jedinstvenim ključem, što sprječava kolanje istih stavki i probleme s indeksiranjem.

```
[9]: obj_1 = {}
     obj_2 = dict()

[10]: type(obj_1)

[10]: dict

[11]: type(obj_2)

[11]: dict
```

*Slika 5.2 'Primjer inicijalizacije rječnika',*
*Izvor: Vlastiti rad autora*

Slika 5.2 prikazuje dva različita postupka inicijalizacije rječnika nad varijablom. Prvi način inicijalizacije rječnika vrši se pridodavanjem vitičastih zagrada ('{}') nakon znaka jednakosti, a drugi način je pozivanje funkcije:

```
dict()
```

```
[24]: # kreiranje rječnika (podložno i učitavanju s računala).
      Dostava = {
      'Bazen': '16.04.2022.',
      'Sobna lampa': '05.05.2022.',
      'Ručni sat': '16.05.2022.',
      'Naočale' : '28.05.2022.',
      }

[25]: # primjer odabira proizvoda 'Sobna lampa' iz rječnika i pristup datumu.
      objekt = 'Sobna lampa'
      datum = Dostava[objekt]

[26]: # izvještavanje o datumu dostave željenog objekta.
      f'Proizvod {objekt} mora biti isporučen na dan {datum}'

[26]: 'Proizvod Sobna lampa mora biti isporučen na dan 05.05.2022.'
```

*Slika 5.3 'Primjer kreiranja i korištenja rječnika',*
*Izvor: Vlastiti rad autora*

Slika 5.3 prikazuje primjer definiranja rječnika koji sadrži objekte u skladišnom prostoru s pripadajućim datumom isporuke. Na ovaj se način može pohraniti niz različitih varijabli u jednom prostoru, a pristupa im se unosom ključne riječi, odnosno:

```
rječnik['ključna riječ'] # '' popuniti željenom ključnom riječi.
```

### 5.3.3. Lista

Lista je neprimitivni tip podataka koji omogućava kreiranje niza vrijednosti. Razlikuje se od rječnika po tome što se određenoj vrijednosti niza pristupa unosom rednog broja u listi (indeksa).

```
[12]: obj_3 = []
      obj_4 = list()

[13]: type(obj_3)

[13]: list

[14]: type(obj_4)

[14]: list
```

*Slika 5.4 'Primjer inicijalizacije liste',*

*Izvor: Vlastiti rad autora*

Slika 5.4 prikazuje dva načina kreiranja liste u Pythonu, korištenjem otvorene i zatvorene uglate zagrade ('[]') te inicijalizaciju liste korištenjem ugrađene funkcije:

```
list()
```

Slika 5.5 prikazuje korištenje liste na realnom primjeru selekcije vlakova s obzirom na vrstu pogona, pri čemu HŽ elektromotorni vlakovi počinju brojkom 6, a dizel-motorni brojkom 7.

```
[144]: # kreiranje liste.
       Vlakovi = ['HŽ 6111', 'HŽ 6112', 'HŽ 7122', 'HŽ 7022', 'HŽ 7023', 'HŽ 7121']

[145]: # primjer lociranja vlaka na prvoj poziciji unutar liste. (u Pythonu se odbrojava od 0)
       Vlakovi[0]

[145]: 'HŽ 6111'

[146]: # identifikacija vrste motora vlaka.
       elektromotorni = [vlak for vlak in Vlakovi if vlak[3] == '6']
       dizelmotorni = [vlak for vlak in Vlakovi if vlak[3] != '6']

[147]: # prikaz elektromotornih vlakova.
       elektromotorni

[147]: ['HŽ 6111', 'HŽ 6112']

[148]: # prikaz dizel-motornih vlakova.
       dizelmotorni

[148]: ['HŽ 7122', 'HŽ 7022', 'HŽ 7023', 'HŽ 7121']

[149]: # sortirani prikaz dizel-motornih vlakova.
       dizelmotorni = sorted(dizelmotorni)
       dizelmotorni

[149]: ['HŽ 7022', 'HŽ 7023', 'HŽ 7121', 'HŽ 7122']
```

*Slika 5.5 'Primjer kreiranja i korištenja liste',*

*Izvor: Vlastiti rad autora.*

### 5.3.4. Set

Set je neprimitivni tip podataka koji ima svojstva jednaka listi, ali bez mogućnosti imanja istih vrijednosti (duplikata) u skupu podataka i mogućnosti pristupa pomoću rednog broja (indeksa).

```
[18]: obj_5 = set()

[19]: type(obj_5)

[19]: set
```

*Slika 5.6 'Primjer inicijalizacije seta',*

*Izvor: Vlastiti rad autora*

Slika 5.6 prikazuje primjer inicijalizacije seta vrijednosti korištenjem ugrađene funkcije: `set()`

Set je moguće kreirati i unosom vrijednosti u vitičaste zagrade ('{}'), nalik na rječnik.

Slika 5.7 prikazuje postupanje skretničara vlakova prema popisu dolaznih i odlaznih vlakova.

```
[147]: # kreiranje seta automobila koji prolaze kroz raskrižje.
       Dolazni_vlakovi = {'Koprivnica', 'Koprivnica', 'Golubovec', 'Zagreb GK', 'Zagreb GK', 'Koprivnica'}
       Odlazni_vlakovi = {'Zagreb GK', 'Zagreb GK', 'Koprivnica'}

[148]: # primjer ispisa jedinstvenih relacija iz kojih vlakovi dolaze.
       Dolazni_vlakovi

[148]: {'Golubovec', 'Koprivnica', 'Zagreb GK'}

[149]: # primjer ispisa jedinstvenih relacija prema kojima vlakovi odlaze.
       Odlazni_vlakovi

[149]: {'Koprivnica', 'Zagreb GK'}

[150]: # početno stanje skretnice.
       manevriranje_skretnice = False

[151]: # donošenje odluke o pomaku skretnice sukladno prometnim trasama vlakova.
       if 'Čakovec' not in Dolazni_vlakovi and 'Čakovec' not in Odlazni_vlakovi:
           manevriranje_skretnice = False
       elif 'Čakovec' in Dolazni_vlakovi or 'Čakovec' in Odlazni_vlakovi:
           manevriranje_skretnice = True

[152]: # stanje skretnice nakon analize.
       manevriranje_skretnice

[152]: False

[153]: # ispis odluke o pomaku skretnice prema Čakovcu.
       if manevriranje_skretnice == True:
           print('Potrebno je manevrirati skretnicom prema Čakovcu.')
       elif manevriranje_skretnice == False:
           print('Nije potrebno manevrirati skretnicom prema Čakovcu.')

       Nije potrebno manevrirati skretnicom prema Čakovcu.
```

*Slika 5.7 'Primjer kreiranja i korištenja seta',*

*Izvor: Vlastiti rad autora.*

### 5.3.5. Uređena lista

Uređena lista (eng. 'tuple') lista je koju nije moguće mijenjati. Omogućava lociranje sadržaja i grupiranje sadržaja kroz pokretanje generatora koji sortira parove atributa.

```
[20]: obj_6 = ()
      obj_7 = tuple()

[21]: type(obj_6)

[21]: tuple

[22]: type(obj_7)

[22]: tuple
```

*Slika 5.8 'Primjer inicijalizacije uređene liste',*

*Izvor: Vlastiti rad autora*

Slika 5.8 prikazuje primjer kreiranja uređene liste, korištenjem zagrada () ili funkcijom:

```
tuple()
```

Slika 5.9 prikazuje primjenu uređenih lista pri grupiranju istih atributa kupaca 1 i 2.

```
[119]: # kreiranje uređene liste.
       Kupac_1 = ('Marko', (45.82, 15.97), 'Sobna lampa')
       Kupac_2 = ('Ana', (43.82, 16.97), 'Bazen')

[120]: # raspakirani podaci.
       ime_1, lokacija_1, proizvod_1 = Kupac_1
       ime_2, lokacija_2, proizvod_2 = Kupac_2

[121]: # primjer proizvoda prvog kupca.
       proizvod_1

[121]: 'Sobna lampa'

[122]: # grupiranje svih kupaca.
       grupiranje = list(zip(Kupac_1, Kupac_2))
       grupiranje

[122]: [('Marko', 'Ana'), ((45.82, 15.97), (43.82, 16.97)), ('Sobna lampa', 'Bazen')]

[123]: # pristup grupiranim lokacijama.
       lokacije = grupiranje[1]
       lokacije

[123]: ((45.82, 15.97), (43.82, 16.97))

[124]: # pristup zemljopisnoj širini prvog kupca unutar lokacije.
       zemljopisna_širina = lokacije[0][0]
       zemljopisna_širina

[124]: 45.82
```

*Slika 5.9 'Primjer kreiranja i korištenja uređene liste',*

*Izvor: Vlastiti rad autora*

### 5.3.6. Datoteka

Python omogućava unos, kreiranje i izmjenu datoteka korištenjem programskog koda, što ga čini često korištenim pri automatizaciji i ažuriranju niza datoteka primjenom skripti.

```
[25]: obj_8 = open('primjer.txt', 'w')

[26]: type(obj_8)

[26]: _io.TextIOWrapper
```

*Slika 5.10 'Primjer inicijalizacije tekstualne datoteke',*
*Izvor: Vlastiti rad autora*

Slika 5.10 prikazuje primjer kreiranja tekstualne datoteke (ekstenzije *.txt*) u modu za pisanje prikazanog argumentom *w* (eng. 'write').

```
[64]: # unos biblioteke za unos vremena.
      from datetime import datetime

[66]: # zapisivanje narudžbi i vremena.
      with open('Narudžbe.txt', 'w', encoding = 'utf-8') as f:
          vrijeme = datetime.now()
          f.write('Sobna lampa' + ' ' + str(vrijeme))
          f.write('\nBazen' + ' ' + str(vrijeme))
          f.write('\nRučni sat' + ' ' + str(vrijeme))
          f.write('\nNaočale' + ' ' + str(vrijeme))

[67]: # otvaranje datoteke s narudžbama.
      datoteka = open('Narudžbe.txt', 'r', encoding = 'utf-8')
      datoteka.read().splitlines()

[67]: ['Sobna lampa 2022-06-27 13:50:39.698633',
       'Bazen 2022-06-27 13:50:39.698633',
       'Ručni sat 2022-06-27 13:50:39.698633',
       'Naočale 2022-06-27 13:50:39.698633']

[68]: #zatvaranje datoteke.
      datoteka.close()
```

*Slika 5.11 'Upis narudžbi prema vremenu u tekstualnu datoteku',*
*Izvor: Vlastiti rad autora*

Slika 5.11 prikazuje unos narudžbi logističkog poduzeća u tekstualnu datoteku uz navođenje vremena u kojem su zaprimljene. Za prikupljanje informacija o vremenu korištena je funkcija *now()* modula *datetime*, istoimene biblioteke *datetime*[62]. Uz datoteke ekstenzije *.txt*, postoji niz datoteka kojima se može upravljati u Pythonu poput onih s ekstenzijama *.xlsx, .pdf, .csv, .json, .md, .jpg, .png, .html* i slično.

[62] biblioteke, moduli i funkcije detaljnije su pojašnjeni u poglavlju 5.7.

## 5.4. Naredbe u Pythonu

U ovom su potpoglavlju prikazane ključne naredbe programiranja u Pythonu. Naredbama se prema određenim uvjetima vrši manipulacija varijablama s namjerom postizanja ciljeva programskog rješenja.

### 5.4.1. *If* petlja

*If* petlje, nazivane i *if-then*[63] petljama, uvjetne su naredbe te su sastavni dio programiranja u brojnim programskim jezicima pri čemu ni Python nije iznimka. Njima se postiže ispitivanje uvjeta i određivanje slijeda operacija ovisno o zadovoljenju, odnosno nezadovoljenju postavljenih uvjeta.

```
[55]: # kreiranje podatkovnog tipa.
      from typing import Union
      broj = Union[float, int]

[56]: # definiranje varijable x.
      x : broj = 5

[57]: # ispitivanje veličine varijable x.
      if x > 5:
          print(f'Broj {x} je veći od 5!')
      elif x < 5:
          print(f'Broj {x} je manji od 5!')
      elif x == 5:
          print(f'Broj {x} je jednak 5!')
      else:
          print('Unos nije prihvatljivog podatkovnog tipa!')

      Broj 5 je jednak 5!

[58]: # definiranje prazne liste.
      vrijednosti : list = []

[59]: # ispitivanje postojanja sadržaja u listi.
      if not vrijednosti:
          print('Lista je prazna!')
      elif vrijednosti:
          print('Lista sadrži vrijednosti!')

      Lista je prazna!
```

*Slika 5.12 'Prikaz upotrebe if petlje u Pythonu',*
*Izvor: Vlastiti rad autora*

Slika 5.12 prikazuje dva slučaja primjene *if* petlji u Pythonu. Prvim primjerom se ispituje veličina varijable *x* u odnosu na broj 5 s ispisom predloženih poruka za svaki scenarij. Drugi primjer prikazuje ispitivanje postojana sadržaja u kreiranoj listi naziva *vrijednosti*.

[63] hrv. 'ako-onda'.

### 5.4.2. *While* petlja

*While* petlje, u različitim varijantama poput *do-while*[64] oblika koriste se za izvršavanje operacija dokle vrijedi zadani uvjet. U određenim prigodama ispitivanja istinitosti uvjeta mogu se koristiti nalik na *if* petlje, ali specifične su po sposobnosti kontinuiranog ispitivanja istinitosti pri svakoj novoj iteraciji u nizu i prekidanju petlje ako uvjet nije zadovoljen.

```
[43]: # broj slobodnih mjesta.
      slobodna_mjesta = 100

[44]: # tijek brojanja slobodnih mjesta u vlaku
      while slobodna_mjesta > 0:
          u_broj = int(input('Broj putnika koji ulaze:'))
          i_broj = int(input('Broj putnika koji izlaze:'))
          stanje = u_broj - i_broj
          slobodna_mjesta -= stanje
          if slobodna_mjesta >= 0:
              print(f'\n>>> Broj slobodnih mjesta u vlaku sada je {slobodna_mjesta}. <<<')
      print('\n>>> U vlaku više nema mjesta. <<<')

      Broj putnika koji ulaze: 42
      Broj putnika koji izlaze: 0

      >>> Broj slobodnih mjesta u vlaku sada je 58. <<<
      Broj putnika koji ulaze: 36
      Broj putnika koji izlaze: 7

      >>> Broj slobodnih mjesta u vlaku sada je 29. <<<
      Broj putnika koji ulaze: 29
      Broj putnika koji izlaze: 0

      >>> Broj slobodnih mjesta u vlaku sada je 0. <<<

      >>> U vlaku više nema mjesta. <<<

[48]: # početna vrijednost brojača i unos biblioteke time.
      import time
      brojač = 5

[49]: # odbrojavanje do polaska vlaka.
      while brojač > 0:
          print(f'Vlak Zagreb GK-Koprivnica kreće za {brojač} sekundi')
          time.sleep(1)
          brojač -= 1
      print('\n>>> Vlak Zagreb GK-Koprivnica krenuo je s kolodvora. <<<')

      Vlak Zagreb GK-Koprivnica kreće za 5 sekundi
      Vlak Zagreb GK-Koprivnica kreće za 4 sekundi
      Vlak Zagreb GK-Koprivnica kreće za 3 sekundi
      Vlak Zagreb GK-Koprivnica kreće za 2 sekundi
      Vlak Zagreb GK-Koprivnica kreće za 1 sekundi

      >>> Vlak Zagreb GK-Koprivnica krenuo je s kolodvora. <<<
```

*Slika 5.13 'Prikaz upotrebe while petlje u Pythonu',*
*Izvor: Vlastiti rad autora*

Slika 5.13 prikazuje primjenu *while* petlje u dva slučaja. Prvi slučaj upotrebe petlje prikazan je pri ispisu broja raspoloživih mjesta u vlaku s obzirom na izmjenu broja putnika, dok drugi slučaj prikazuje odbrojavanje do polaska vlaka i promjenu informacije na info panelu kolodvora.

[64] hrv. 'izvršavaj-dok'.

### 5.4.3. *For* petlje

*For*[65] petlje koriste se pri izvršavanju naredbi nad nizom podataka (n >= 2). Omogućavaju iterativni proces u Pythonu, a bitan su element automatizacije kroz primjenu skripti.

```
[40]: # inicijalni prikaz statusa semafora.
semafori = {
    'Semafor 1' : 'Upaljen',
    'Semafor 2' : 'Upaljen',
    'Semafor 3' : 'Upaljen',
    'Semafor 4' : 'Upaljen',
    'Semafor 5' : 'Upaljen',
}

[41]: # automatska regulacija svih semafora.
for semafor, status in semafori.items():
    semafori[semafor] = 'Ugašen'

[42]: # status semafora nakon regulacije.
semafori

[42]: {'Semafor 1': 'Ugašen',
 'Semafor 2': 'Ugašen',
 'Semafor 3': 'Ugašen',
 'Semafor 4': 'Ugašen',
 'Semafor 5': 'Ugašen'}

[43]: # popis vozila
vozila = ['ČK-478-FZ', 'ČK-288-JK', 'VŽ-921-TN', 'ZG-9212-HF', 'KC-242-TI', 'ST-202-HI']

[46]: # testiranje dohvaćanja grada s registracije.
vozila[2][0:2]

[46]: 'VŽ'

[47]: # ispis vozila koja nisu registrirana u KC-KŽ županiji.
for vozilo in vozila:
    if vozilo[0:2] != 'KC' and vozilo[0:2] != 'KŽ':
        print(f'Vozilo {vozilo} nije registrirano u Koprivničko-križevačkoj županiji.')

Vozilo ČK-478-FZ nije registrirano u Koprivničko-križevačkoj županiji.
Vozilo ČK-288-JK nije registrirano u Koprivničko-križevačkoj županiji.
Vozilo VŽ-921-TN nije registrirano u Koprivničko-križevačkoj županiji.
Vozilo ZG-9212-HF nije registrirano u Koprivničko-križevačkoj županiji.
Vozilo ST-202-HI nije registrirano u Koprivničko-križevačkoj županiji.
```

*Slika 5.14 'Prikaz upotrebe for petlje u Pythonu',*

*Izvor: Vlastiti rad autora*

Slika 5.14 prikazuje dva primjera upotrebe *for* petlje. Prvi primjer prikazuje primjenu petlje nad rječnikom koji sadrži statuse semafora na raskrižju *x*. Korištenjem spomenute petlje moguće je promijeniti status svih podataka neke liste ili rječnika jednom operacijom. Drugi primjer prikazuje dohvaćanje kratice grada s registracije evidentiranih vozila i filtriranje onih koji nisu registrirani na području Koprivničko-križevačke županije.

[65] hrv. 'za'.

### 5.4.4. Sažimanje liste s više naredbi

Sažimanje liste, u Pythonu zvano *list comprehension*[66] koristi se pri programiranju u jednom retku. Programiranje u jednom retku alternativni je način programiranja klasičnih petlji, a postiže se kolanjem naredbi poput *for*, *if* i *while* petlje unutar liste. Kolanje naredbi metodom sažimanja liste prikazano je popisom registracija vozila koja prekoračuju toleranciju brzine (Slika 5.15).

```
[9]: # prikaz vozila i zabilježenih brzina.
     brzine_vozila = {
         'ČK-478-FZ' : 85,
         'ČK-288-JK' : 105,
         'VŽ-921-TN' : 88,
         'ZG-9212-HF' : 94,
         'KC-242-TI' : 101,
         'ST-202-HI' : 90,
     }

[10]: # registracije vozila koja su prekoračili dopuštenu toleranciju prekoračenja brzine.
      [vozilo for vozilo, brzina in brzine_vozila.items() if brzine_vozila[vozilo] > 99]

[10]: ['ČK-288-JK', 'KC-242-TI']
```

*Slika 5.15 'Prikaz upotrebe sažimanja listi u Pythonu',*
*Izvor: Vlastiti rad autora*

### 5.4.5. *Try-except* blok

*Try-except* blok, također znan i kao *try-except-finally*[67], a u nekim programskim jezicima i *try-catch*[68] blok koristi se u izmjeni uvjeta pri detekciji neočekivanog ponašanja funkcije.

```
[50]: # hvatanje pogreške.
      try:
          funkcija()
      except:
          # alternative print funkciji su Except i assert naredbe s djelotvornim svojstvima.
          print('Funkcija nije izvršena jer ne postoji.')

      Funkcija nije izvršena jer ne postoji.
```

*Slika 5.16 'Prikaz upotrebe Try-except bloka u Pythonu',*
*Izvor: Vlastiti rad autora*

Slika 5.16 prikazuje upotrebu *try-except* bloka na primjeru nepostojeće funkcije. Primjenom bloka daje se povratna informacija o pogrešci, a moguće je nametnuti i prekid funkcije.

---

[66] hrv. doslovno prevedeno kao 'razumijevanje liste.'

[67] hrv. 'pokušaj-osim ako-naposlijetku.'

[68] hrv. 'pokušaj-uhvati.'

## 5.5. Funkcije u Pythonu

Postoje dva osnovna načina definiranja funkcija u Pythonu. Prvi način definiranja jednostavnih funkcija odnosi se na korištenje *lambda* izraza, dok se drugi način odnosi na definiranje opsežnijih funkcija primjenom *def* izraza.

Funkcije koje se definiraju u programskom jeziku identične su standardnim matematičkim zapisima istih. Primjer dodjeljivanja funkcije s argumentima *a* i *b* varijabli *y* prikazuje se, uz prethodno kreiranje funkcije imena *zbrajanje* korištenjem *def* ili *lambda* izraza, na sljedeći način:

```
a = 5
b = 10
y = zbrajanje(a, b)
print(y)
```

```
>>> 15
```

Funkcijama se mogu dodijeliti početni argumenti pri njihovom definiranju, pri čemu obvezni argumenti moraju prethoditi opcionalnim argumentima. Obvezni argumenti najčešće bitno utječu na funkcionalnost, dok se opcionalni argumenti često koriste za izmjene uvjeta unutar funkcije.

### 5.5.1. Funkcije primjenom *lambda* izraza

*Lambda* izrazi se najčešće koriste za jednokratne potrebe. Praktičan su način iskazivanja funkcije s argumentima u jednom retku.

```
[247]: # prikaz modela vlakova bez informacije o prijevozniku.
       Vlakovi = {
           'Model' : [6112, 7022, 7023]
       }

[248]: # kreiranje lambda funkcije.
       funkcija = lambda tekst, rječnik: [tekst + str(x) for x in rječnik]

[249]: # primjena lambda funkcije.
       funkcija('HŽ ', Vlakovi['Model'])

[249]: ['HŽ 6112', 'HŽ 7022', 'HŽ 7023']
```

*Slika 5.17 'Prikaz upotrebe lambda izraza nad rječnikom u Pythonu',*
*Izvor: Vlastiti rad autora*

Slika 5.17 prikazuje primjenu *lambda* izraza kroz ažuriranje elemenata u rječniku pridodavanjem željene riječi uz svaki element. U konkretnom slučaju prikazano je dodjeljivanje informacije o prijevozniku svim postojećim vlakovima u rječniku. Prikazanom metodom promijenjen je i tip podataka iz brojčanog (*int*) u tekstualni (*str*).

### 5.5.2. Funkcije primjenom *def* izraza

*Def* izrazi koriste se pri definiranju složenijih funkcija u Pythonu. Razlikuju se od *lambda* izraza po strukturi i mogućnostima upotrebe. Za razliku od *lambda* izraza, *def* izrazi nakon izvršene radnje vraćaju vrijednost koristeći se *return* izrazom. *Def* izrazi omogućavaju ugnježđivanje funkcija u funkcije, čime se postiže stupnjevanje tijeka izvršavanja pojedinih funkcija. Ugnježđivanjem *def* izraza razvile su se tzv. *dekorator-funkcije* koje se pozivaju korištenjem simbola @ nad funkcijom definiranom *def* izrazom.

```
[16]: # unos biblioteke za praćenje vremena
      from datetime import datetime

[17]: # unos linija.
      dubrava_prečko = ['09/07/2022 17:05:00','09/07/2022 17:29:00', '09/07/2022 17:42:00']
      borongaj_prečko = ['09/07/2022 17:11:00','09/07/2022 17:40:00', '09/07/2022 17:45:00']

[18]: # formatiranje linija u vremenski oblik.
      dubrava_prečko = [datetime.strptime(dolazak, '%d/%m/%Y %H:%M:%S') for dolazak in dubrava_prečko]
      borongaj_prečko = [datetime.strptime(dolazak, '%d/%m/%Y %H:%M:%S') for dolazak in borongaj_prečko]

[19]: # kreiranje funkcije koja predlaže prvi nadolazeći tramvaj prema stanici Prečko.
      def predloži_liniju_za_prečko(čvor):
          vrijeme = datetime.now()
          najbrži_dubrava = min([dt for dt in dubrava_prečko if dt > vrijeme])
          najbrži_borongaj = min([dt for dt in borongaj_prečko if dt > vrijeme])
          prikaz = {
              'Borongaj-Prečko' : f'Tramvaj dolazi na stajalište {čvor} iz smjera Borongaj u {najbrži_borongaj}.',
              'Dubrava-Prečko' : f'Tramvaj dolazi na stajalište {čvor} iz smjera Dubrava u {najbrži_dubrava}.',
          }
          odabir = prikaz['Dubrava-Prečko'] if najbrži_dubrava < najbrži_borongaj else prikaz['Borongaj-Prečko']
          print('Trenutno vrijeme:', vrijeme, '\n')
          return print(odabir)

[20]: # korištenje funkcije predlaganja prvog nadolazećeg tramvaja prema stanici Prečko.
      predloži_liniju_za_prečko('Vjesnik')

      Trenutno vrijeme: 2022-07-09 17:17:28.092044

      Tramvaj dolazi na stajalište Vjesnik iz smjera Dubrava u 2022-07-09 17:29:00.
```

*Slika 5.18 'Prikaz upotrebe def izraza nad rječnikom u Pythonu',*
*Izvor: Vlastiti rad autora*

Slika 5.18 prikazuje upotrebu *def* izraza na primjeru predlaganja najrelevantnije nadolazeće tramvajske linije koja prolazi kroz stajalište Vjesnik i kreće se prema stajalištu Prečko. Za usporedbu razdoblja dolazaka uzete su dvije tramvajske linije na kojima se prometuje po pravcu koji obuhvaća putanju Vjesnik-Prečko, a to su Dubrava-Prečko (linija 5) i Borongaj-Prečko (linija 17). Za potrebe demonstracije *def* izraza u obzir su uzete opcije bez presjedanja. Vremenski periodi dolazaka tramvaja na stanicu simulirani su za potrebe kreiranja i izvršavanja funkcije, te ne prikazuju realno stanje dolazaka i odlazaka definiranog voznog reda kreiranog od strane ZET-a.

*Def* izrazom kreirana je funkcija *predloži_liniju_za_prečko(čvor)* koja pohranjuje stajalište kroz argument *čvor*. Pokretanjem funkcije ispisuje se trenutno vrijeme i vrijeme prvog nadolazećeg tramvaja koji prolazi stajalištem, a vodi prema stajalištu Prečko.

## 5.6. Objektno orijentirano programiranje

Objektno orijentirano programiranje (OOP) jedan je od modela programiranja koji je, uz Python, karakterističan i za programski jezik Javu te brojne druge.

Klasama se u objektno orijentiranom programiranju nazivaju dijelovi programskog koda koji služe kao predlošci iz kojih se kreiraju instance objekta (Šantić, 2017). U nekim se programskim jezicima umjesto naziva klasa upotrebljava naziv prototip.

Šantić (2017) kao element objektno orijentiranog programiranja također navodi i metode, a kao primjer navodi konstruktorsku *__init__()* metodu.

### 5.6.1. Izrada klase i instance objekta

Izrada klase u Pythonu započinje upisivanjem riječi *class*, kojoj se pridodaje ime i dvotočka. Za razliku od funkcija, klasa najčešće ne poprima vrijednost u zagradi, ali ono također može biti sastavni dio klase ako se primjenjuje *metaklasa* prema kojoj se gradi instanca klase i definira se u zagradi nakon imena klase, a prije dvotočke.

```
[19]: # izrada klase.
      class Vlak:

          ''' Klasa prema kojoj se kreira instanca objekta. '''

          def __init__(self, proizvođač, oznaka, brzina):
              self.proizvođač = proizvođač
              self.oznaka = oznaka
              self.brzina = brzina
```

*Slika 5.19 'Kreiranje klase vlaka',*
*Izvor: Vlastiti rad autora*

Slika 5.19 prikazuje izradu klase imena Vlak s pripadajućim varijablama. Korištenjem *__init__()* konstruktorske metode, pri kreiranju instance objekta na temelju klase automatski se inicijaliziraju i pripadajuće varijable. Primjetno je kako *def* izraz *__init__() konstruktorske metode* ne poprima izraz *return* na kraju, kao što je to najčešće slučaj kod ostalih metoda i funkcija općenito. Slika 5.20 prikazuje kreiranje instance objekta s pripadajućim varijablama.

```
[21]: # kreiranje instance.
      HŽ_7022 = Vlak('Gredelj', 7022, 160)

[22]: # potvrda o pripadnosti klasi.
      HŽ_7022

[22]: <__main__.Vlak at 0x193f7b7c790>
```

*Slika 5.20 'Kreiranje instance objekta',*
*Izvor: Vlastiti rad autora*

### 5.6.2. Izrada i korištenje metoda klase

Metode klase imaju ulogu funkcija, odnosno, omogućavaju manipulaciju podacima kreirane instance objekta. Razlika klasičnih funkcija i metoda klase je korištenje izraza *self* unutar zagrade, što simbolizira pripadnost klasi. Izlaznim podacima te varijablama pojedine metode također se može pristupiti ako se one definiraju korištenjem *self* izraza i to na način:

```
self.naziv_metode()#naziv_metode zamijeniti željenim nazivom.
```

```
self.varijabla = varijabla #varijabla zamijeniti željenim nazivom.
```

```
[23]: # izrada klase.
class Vlak:

    ''' Klasa prema kojoj se kreira instanca objekta. '''

    def __init__(self, proizvođač, oznaka, brzina):
        self.proizvođač = proizvođač
        self.oznaka = oznaka
        self.brzina = brzina

    def prikaži(self):
        return f'Vlak proizvođača {self.proizvođač}, oznake {self.oznaka} i brzine {self.brzina}km/h.'

    def uspori(self, količina):
        self.brzina -= količina
        return self.brzina

[24]: # kreiranje instance.
HŽ_7022 = Vlak('Gredelj', 7022, 160)

[26]: # prikaz detalja o vlaku.
HŽ_7022.prikaži()

[26]: 'Vlak proizvođača Gredelj, oznake 7022 i brzine 160km/h.'

[27]: # dohvaćanje brzine.
HŽ_7022.brzina

[27]: 160

[28]: # usporavanje vlaka za 20.
HŽ_7022.uspori(20)

[28]: 140

[29]: # prikaz detalja o vlaku.
HŽ_7022.prikaži()

[29]: 'Vlak proizvođača Gredelj, oznake 7022 i brzine 140km/h.'
```

*Slika 5.21 'Izrada i korištenje metoda klase na primjeru klase Vlak',*
*Izvor: Vlastiti rad autora*

Slika 5.21 prikazuje izradu metoda klase. Metoda *prikaži()* ispisuje detalje o kreiranoj instanci objekta u rečenici, dok se korištenjem metode *uspori(količina)* mijenja inicijalno definirana brzina za iznos prikazan u zagradi, a što u konkretnom slučaju znači smanjenje sa 160 na 140 km/h.

## 5.7. Python biblioteke i instalacija paketa

Python omogućava unos vanjskih biblioteka, čime se proširuju mogućnosti programera s obzirom na to da nije potrebno započeti projekt iz samog temelja. Unos biblioteka može se izvršiti ručno i kloniranjem repozitorija, ali preuzimanju i ažuriranju biblioteka najčešće se pristupa korištenjem naredbi u terminalu (naredbenom retku) računala. Često se koriste alati koji dohvaćaju raspoložive biblioteke poput *anaconde* i *PyPi*[69]*-a*. Primjer instalacije biblioteke *ime_paketa* na računalo korištenjem *anaconde* vrši se naredbom:

```
conda install ime_paketa
```

dok se instalaciji paketa korištenjem *PyPi-a* pristupa naredbom:

```
pip install ime_paketa
```

Unos biblioteke u programsko sučelje vrši se naredbom:

```
import ime_biblioteke
```

Unos modula ili objekta iz biblioteke može se izvršiti korištenjem naredbi:

```
import ime_biblioteke.ime_modula
from ime_biblioteke import ime_modula
```

### 5.7.1. Biblioteke strojnog učenja - *TensorFlow, scikit-learn* i *keras*

*TensorFlow, scikit learn*[70] i *keras* biblioteke su kojima se omogućava programiranje algoritama strojnog učenja korištenjem njihovih dostupnih značajki. U većini slučajeva, spomenute se biblioteke koriste u kombinaciji jedna s drugom.

Primjer kombiniranja spomenutih biblioteka očituje se u modeliranju algoritma umjetnih neuronskih mreža, gdje se korištenjem biblioteke *sklearn* vrši podjela seta podataka na setove za treniranje, testiranje i validaciju, korištenjem *keras* biblioteke unose se optimizacijska rješenja koja doprinose efektivnosti modela dok se bibliotekom *TensorFlow* vrši proces treniranja modela i procjena uspješnosti nad neviđenim podacima.

Uz model programiranja baziranog na *TensorFlow* platformi, postoji i okvir programiranja[71] koji se bazira na primjeni biblioteke *PyTorch* s pripadajućim optimizacijskim rješenjima.

---

[69] eng. 'Python package index' – hrv. indeksiranje Python paketa.

[70] naziva se još i 'sklearn'.

[71] eng. 'framework'.

### 5.7.2. Biblioteka za kompleksne matematičke operacije – *numpy*

Biblioteka *numpy* sastavni je dio prethodno navedenih biblioteka strojnog učenja jer se izlazni podaci najčešće prikazuju u obliku vektora, matrica i tenzora. Biblioteka *numpy* omogućava zapis vrijednosti u višedimenzionalnom prostoru te oblikovanje elemenata linearne algebre i izračunavanje njihovih međuodnosa. Bibliotekom se omogućava indeksiranje (lociranje) podataka u vektorima, matricama i tenzorima, zbog čega ona postaje sastavnim dijelom ostalih biblioteka koje se koriste pri unosu datoteka tabličnog tipa. Sastavni je dio modeliranja stohastičkih modela jer omogućava kreiranje slučajnih (pseudoslučajnih) vrijednosti.

Slika 5.22 prikazuje primjenu *numpy* biblioteke u preoblikovanju lista u grupirani niz podataka s kojima se nadalje vrše standardne matematičke operacije. Prikazano je filtriranje redaka u kojima postoji negativna vrijednost drugog retka, transponiranje te množenje s jediničnom matricom.

```
[75]: # unos biblioteke.
      import numpy as np

[76]: # kreiranje početnih listi.
      x = [20, 24, 14, 13, 23, 56, 12]
      y = [-40, -54, -13, 33, 25, -56, 34]
      z = [10, 46, 26, 13, 45, -52, 14]

[77]: # pretvorba listi u numpy niz.
      podaci = np.vstack([x, y, z])

[78]: #prikaz numpy niza.
      podaci

[78]: array([[ 20,  24,  14,  13,  23,  56,  12],
             [-40, -54, -13,  33,  25, -56,  34],
             [ 10,  46,  26,  13,  45, -52,  14]])

[79]: #uklanjanje redaka gdje drugi redak ima negativnu vrijednost.
      podaci = podaci[:, podaci[1] > 0]

[80]: # prikaz podataka nakon uklanjanja redaka gdje drugi redak ima negativnu vrijednost.
      podaci

[80]: array([[13, 23, 12],
             [33, 25, 34],
             [13, 45, 14]])

[81]: podaci = podaci.T

[82]: # prikaz podataka nakon transponiranja.
      podaci

[82]: array([[13, 33, 13],
             [23, 25, 45],
             [12, 34, 14]])

[84]: # množenje s jediničnom matricom.
      podaci * np.identity(3)

[84]: array([[13.,  0.,  0.],
             [ 0., 25.,  0.],
             [ 0.,  0., 14.]])
```

*Slika 5.22 'Primjer upotrebe numpy biblioteke u Pythonu',*

*Izvor: Vlastiti rad autora*

### 5.7.3. Biblioteka za rad u tabličnom prikazu – *pandas*

Biblioteka *pandas* omogućava kreiranje okvira podataka (eng. 'dataframe') u tabličnom prikazu. Neizostavan je element brojnih projekata znanosti o podacima zbog ugrađenih metoda kojima se vrši indeksiranje i dohvaćanje podataka prema ključnoj riječi ili uvjetu.

```
[7]: # unos biblioteke.
     import pandas as pd
```

```
[8]: # set podataka od 4 autoceste.
     # podaci iz: Ključne brojke (2018), Hrvatska udruga koncesionara za autoceste s naplatom cestarine.
     autoceste = pd.DataFrame({
         'Autocesta' : ['A1', 'A2', 'A3', 'A4'],
         'Čvor1' : ['Zagreb', 'Zagreb', 'Bregana', 'Zagreb'],
         'Čvor2' : ['Ploče', 'Macelj', 'Lipovac', 'Goričan'],
         'Nadimak' : ['Dalmatina', 'Zagorska autocesta', 'Posavska autocesta', 'Varaždinska autocesta'],
         'Duljina' : [480.15, 60, 306.4, 96.9],
     })
```

```
[9]: #prikaz autocesta.
     autoceste
```

[9]:

| | Autocesta | Čvor1 | Čvor2 | Nadimak | Duljina |
|---|---|---|---|---|---|
| 0 | A1 | Zagreb | Ploče | Dalmatina | 480.15 |
| 1 | A2 | Zagreb | Macelj | Zagorska autocesta | 60.00 |
| 2 | A3 | Bregana | Lipovac | Posavska autocesta | 306.40 |
| 3 | A4 | Zagreb | Goričan | Varaždinska autocesta | 96.90 |

```
[10]: # dohvaćanje informacija retka 4 (brojanje započinje od 0, indeks je 3).
      autoceste.iloc[3]
```

```
[10]: Autocesta                     A4
      Čvor1                     Zagreb
      Čvor2                    Goričan
      Nadimak      Varaždinska autocesta
      Duljina                    96.9
      Name: 3, dtype: object
```

```
[11]: # prikaz autocesta kojima je početni čvor u Zagrebu.
      autoceste[autoceste['Čvor1'] == 'Zagreb']
```

[11]:

| | Autocesta | Čvor1 | Čvor2 | Nadimak | Duljina |
|---|---|---|---|---|---|
| 0 | A1 | Zagreb | Ploče | Dalmatina | 480.15 |
| 1 | A2 | Zagreb | Macelj | Zagorska autocesta | 60.00 |
| 3 | A4 | Zagreb | Goričan | Varaždinska autocesta | 96.90 |

*Slika 5.23 'Primjer upotrebe pandas biblioteke pri kreiranju tabličnog prikaza podataka', Izvor: Vlastiti rad autora*

Slika 5.23 prikazuje kreiranje okvira podataka četiriju hrvatskih autocesta. Izrada okvira podataka bazira se na shemi izrade Python rječnika. Prikazanim primjerom izvršeno je dohvaćanje informacija o pojedinom retku okvira podataka (redak 4), kao i filtriranje podataka prema ključnoj riječi, što je prikazano na primjeru podataka gdje je početni čvor identičan argumentu *Zagreb*.

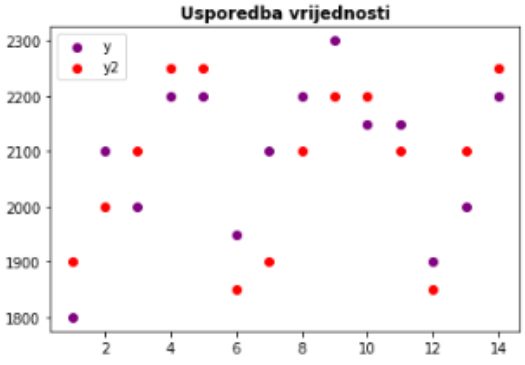
Stalne promjene okvira podataka vrše se pozivanjem i definiranjem argumenta *inplace = True*.

### 5.7.4. Biblioteka za vizualizaciju podataka – matplotlib.pyplot

Modulom *pyplot* biblioteke *matplotlib* vizualno se prikazuju podaci. Primjenom modula pristupa se brojnim varijantama iskazivanja grafova, od prikaza točkama do linijskog prikaza, *boxplot* prikaza i kreiranja histograma.

```
[9]: # unos biblioteke i modula.
     import matplotlib.pyplot as plt

[10]: # kreiranje testnog seta.
      y = [1800, 2100, 2000, 2200, 2200, 1950, 2100, 2200, 2300, 2150, 2150, 1900, 2000, 2200]
      y2 = [1900, 2000, 2100, 2250, 2250, 1850, 1900, 2100, 2200, 2200, 2100, 1850, 2100, 2250]

[11]: # prikazi grafova.
      plt.title('Usporedba kretanja vrijednosti', fontweight = 'bold')
      plt.plot(y, label = 'y')
      plt.plot(y2, label = 'y2')
      plt.legend()

[11]: <matplotlib.legend.Legend at 0x23f4dfa84f0>
```


```
[12]: plt.title('Usporedba vrijednosti', fontweight = 'bold')
      plt.scatter(y = y, x = range(1,15), color = 'purple', label = 'y')
      plt.scatter(y = y2, x = range(1,15), color = 'red', label = 'y2')
      plt.legend()

[12]: <matplotlib.legend.Legend at 0x23f4e09b040>
```




*Slika 5.24 'Primjer upotrebe matplotlib.pyplot biblioteke pri iscrtavanju grafa i točaka', Izvor: Vlastiti rad autora*

Ucrtavanje točaka i linija definiranih vrijednosti modulom *pyplot* prikazano je na Slici 5.24. Prvi graf prikazuje primjenu metode *plot* s pripadajućim argumentima za prikaz, dok su na drugom grafu prikazani međuodnosi između vrijednosti primjenom metode *scatter*.

### 5.7.5. Biblioteke za izradu *CLI* aplikacija u naredbenom retku – argparse i duality

*CLI*[72] aplikacije tip su aplikacije koja se pokreće u terminalu (naredbenom retku) računala. Njima se omogućava pokretanje skripti i inicijalizacija programskog sučelja. Benefiti *CLI* aplikacija prepoznaju se u bržem pokretanju s obzirom na manje vizualnih značajki od onih koje se pokreću kod standardnih aplikacija, dok se nedostaci očituju u manje atraktivnom vizualnom prikazu i korištenju naredbi umjesto miša.

Za potrebe demonstracije, prikazane su dvije varijante izrade *CLI* aplikacija, bibliotekom *argparse*, te samostalno izrađenom bibliotekom *duality*[73].

Prikazana je izrada kalkulatora EOQ pokazatelja. Prema Petar i Matajčić (2021) EOQ pokazateljem se izračunava optimalna količina robe koju je potrebno naručiti za popunjavanje zaliha i to formulom:

$$EOQ = \sqrt{\frac{2ca}{h}} \quad (5.1)$$

EOQ – ekonomična količina narudžbe.

c – trošak naručivanja po narudžbi.

a – godišnja potrošnja (u jedinici proizvoda).

h – troškovi skladištenja po jedinici proizvoda.

Optimalan broj narudžbi prema Petar i Matajčić (2021) izračunava se formulom:

$$\frac{a}{EOQ} \quad (5.2)$$

a – godišnja potrošnja (u jedinici proizvoda).

EOQ – ekonomična količina narudžbe.

Za potrebe demonstracije izračuna EOQ pokazatelja korištena je samostalno izrađena biblioteka *vandal* s pripadajućim objektom koji omogućava izračun.

Aplikacija se pokreće unosom sljedeće naredbe u naredbeni redak:

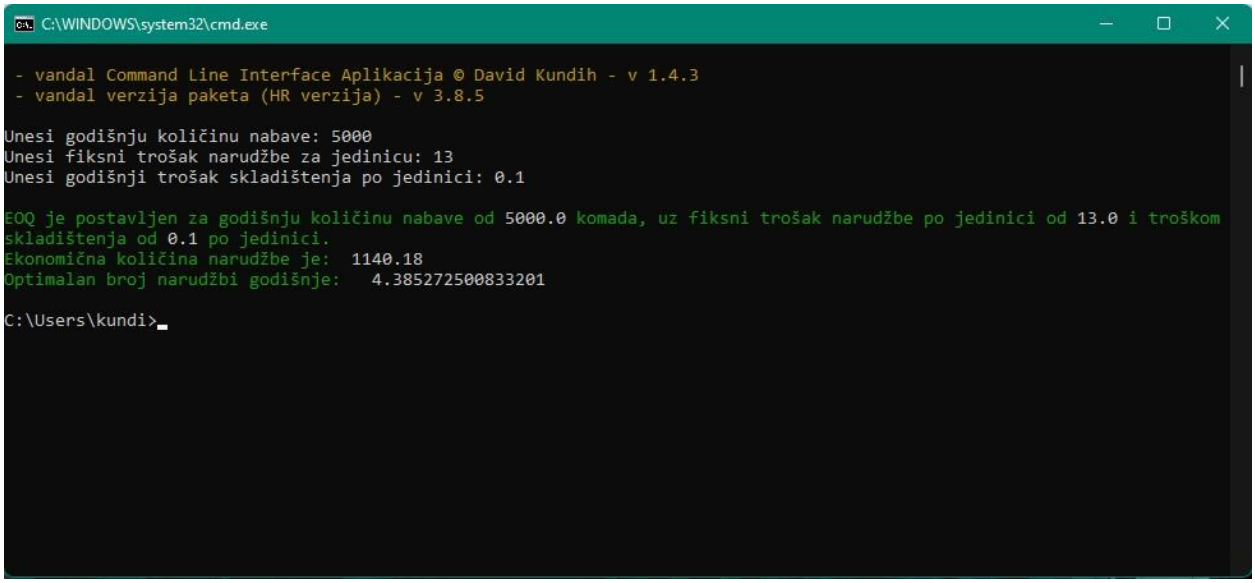
```
python -m vandal -e eoq
```

Spomenutom naredbom, biblioteka *argparse* povezuje Python datoteku biblioteke *vandal* u kojoj je stacioniran programski kod za izračun EOQ pokretanjem biblioteke kao datoteke, što je postignuto kreiranjem posebnog modula *__main__.py* unutar same biblioteke.

[72] eng. 'command line interface'.

[73] programski kod i detaljna razrada dostupni u Prilozima.

Nakon pokretanja naredbe u naredbenom retku prikazuje se vizualno sučelje aplikacije u kojoj se najprije unose parametri za izračun pokazatelja, nakon čega aplikacija daje povratnu informaciju o izračunu ekonomične količine narudžbe i pripadajuće optimalne količine narudžbe kao što je prikazano na Slici 5.25




*Slika 5.25 'CLI aplikacija korištenjem biblioteke argparse',*
*Izvor: Vlastiti rad autora*

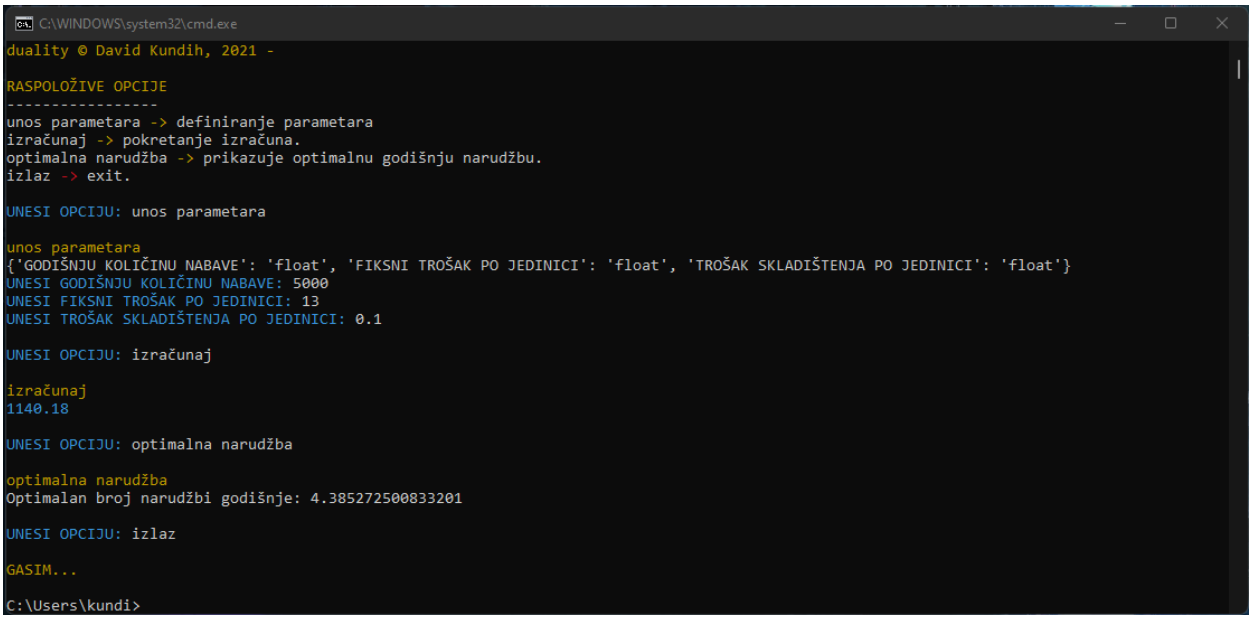
Dio programskog koda kojim se pristupa željenom modulu unutar aplikacije (prilagođenim na hrvatski jezik) korištenjem biblioteke *argparse*:

```
if __name__ == '__main__':
     import argparse
     import vandal
     from colorama import Fore, init
     init()
     parser = argparse.ArgumentParser()
     parser.add_argument('-u', '--ulaz', help = pokretanje željene
     aplikacije.', type = str,
     choices = ['montecarlo', 'eoq'])
     args = parser.parse_args()

    if args.entry == 'montecarlo':
        print(Fore.YELLOW + '=== ULAZIM U MODUL IZRAČUNA MONTE CARLO
SIMULACIJE... ===\n', Fore.RESET)
        vandal.objects.montecarlo.MCapp()
    elif args.entry == 'eoq':
        print(Fore.YELLOW + '=== ULAZIM U MODUL IZRAČUNA EOQ
POKAZATELJA... ===\n', Fore.RESET)
        vandal.objects.eoq.EOQapp()
```

Programski kod prikazuje unos biblioteke za izračun pokazatelja i primjenu boje u tekstualnom ispisu. Korištenjem naredbe *if __name__ == '__main__':* modul se pokreće kao datoteka.

Drugi način kreiranja *CLI* aplikacije izvršava se korištenjem samostalno izrađene biblioteke *duality*. *Duality* biblioteka koristi *dekorator-funkcije* nad metodama i povezuje ih s aplikacijom.

Ključna prednost *duality* biblioteke nad *argparse* bibliotekom jest mogućnost kreiranja niza funkcija što omogućava rad s objektima (Slika 5.26). Uz spomenuto, omogućen je i unos ulaznih vrijednosti u zaseban rječnik, a one se pokreću zajedno s odabirom željene funkcije.




*Slika 5.26 'CLI aplikacija korištenjem biblioteke duality',*

*Izvor: Vlastiti rad autora*

Primjer primjene *dekorator-funkcije* i *duality* aplikacije prikazan je programskim kodom:

```
import duality
app = duality.DualityApp()

class EOQ:
    @app.entry(option_name = 'unos parametara', option_description =
'definiranje parametara')
    def __init__(
        self,
        godišnja_količina_nabave : NumberType = app.store(variable =
'godišnja_količina_nabave', type = 'float', overwrite = 'GODIŠNJU
KOLIČINA NABAVE),
        fiksni_trošak_po_jedinici : NumberType = app.store(variable =
'fiksni_trošak_po_jedinici', type = 'float', overwrite = 'FIKSNI
TROŠAK PO JEDINICI'),
        trošak_skladištenja_po_jedinici : NumberType =
app.store(variable = 'trošak_skladištenja_po_jedinici', type =
'float', overwrite = 'TROŠAK SKLADIŠTENJA PO JEDINICI')

if __name__ == '__main__':
    app.wheel(type = 'dynamic', display_headline = 'RASPOLOŽIVE
OPCIJE', display_message = 'UNESI OPCIJU: ', output_message = 'ODABRAO
SI: ', exit_message = 'GASIM...', enter_message = 'UNESI ', break_key
= 'izlaz')
```

# 6. Umjetna inteligencija

U ovom je poglavlju izdvojen element umjetne inteligencije kao sastavni dio digitalne transformacije prometnih i logističkih sustava. Ovo poglavlje definira umjetnu inteligenciju kao krovni pojam nad strojnim učenjem i dubokim učenjem. Detaljnija razrada elemenata umjetne inteligencije i korelacija s znanosti o podacima prikazana je u poglavlju 7.

Najprije se istražuje povijest i nastanak umjetne inteligencije kao pojma te se definira njezina uloga u svijetu današnjice i budućnosti. Objašnjava se etimološka pozadina njezina naziva te pojava programskog jezika namijenjenog za realizaciju njezinog operativnog djelovanja.

Umjetna inteligencija stavlja se u kontekst Industrije 4.0 i Industrije 5.0. kroz uspostavu vizije daljnjeg razvoja tehnologije i znanosti koji dovodi do njezine šire primjene i pojavu novog gospodarstva koje se temelji na digitalnim inovacijama.

Naveden je i detaljno prikazan proces dizajniranja proizvoda s elementima umjetne inteligencije kroz četiri koraka prema metodologiji američkog MIT-a, a to su redom: inteligencija, poslovni proces, tehnologija i poboljšanja.

Stavlja se naglasak na etičke aspekte umjetne inteligencije i preispitivanje njezine uloge u društvu. Detaljno se analiziraju rizici operativnog djelovanja algoritama strojnog učenja kao sastavnog dijela umjetne inteligencije na primjeru autonomnog vozila i njegovog međuodnosa sa ostalim sudionicima prometnog sustava. Analizira se mogućnost uvođenja univerzalnog temeljnog dohotka kao odgovora na rapidne promjene na tržištu rada. Izdvaja se primjer upotrebe umjetne inteligencije u svrhu nadzora stanovništva kroz uvođenje sustava društvenog bodovanja u Narodnoj Republici Kini i autoritarnim režimima.

Podjela umjetne inteligencije na slabu i jaku s obzirom na ulogu svijesti u proizvodu umjetne inteligencije opisana je u posljednjem dijelu ovog poglavlja.

## 6.1. Pojam umjetne inteligencije i njezina povijest

Umjetna inteligencija radikalno mijenja dosadašnje funkcioniranje sustava i preduvjet je konkurentnosti digitalnih i fizičkih proizvoda u kontekstu razvoja Industrije 4.0 i Industrije 5.0.

Umjetna inteligencija se prema Pristeru (2019) koristi za 'označivanje svojstva svakog neživog sustava koji pokazuje inteligenciju (inteligentni sustav); obično su to računalni sustavi, dok se izraz katkad neutemeljeno primjenjuje na robote, koji nisu nužno inteligentni.'

Nastanak umjetne inteligencije logičan je slijed razvoja računalnih znanosti i težnje pridodavanju ljudskih karakteristika strojevima i uređajima, no ona seže bitno dalje i nadilazi čovjekove sposobnosti u određenim elementima poput prepoznavanja malignih i benignih tumora

na temelju unesenih parametara, a može djelovati i preventivno kroz ukazivanje visokog stupnja rizika razvoja bolesti na temelju navika osobe[74].

Prema Valerjevu (2006) promišljanje o umjetnoj inteligenciji seže daleko u prošlost kroz razvoj filozofije i logike te se 1940-ih godina pojavljuju znanstveni radovi koji se mogu prepoznati kao neki od oblika umjetne inteligencije. Valerjev (2006) navodi da se umjetna inteligencija službeno počinje nazivati na taj način 1956. godine kada je 'na Darthmouth Collegeu, na dvomjesečnoj radionici za 10 polaznika, McCarthy skovao termin *artificial intelligence*', na hrvatskom jeziku danas poznatu kao umjetna inteligencija, te da 1958. godine razvija dominantan programski jezik u području umjetne inteligencije skraćenog naziva LISP[75].

Nakon pridodavanja imena i definiranja konteksta umjetne inteligencije, intenzivno se provode istraživanja i razvijaju modeli njezine primjene u praksi kao sastavnog elementa suvremenog svijeta, ali nedvojbeno i budućnosti zbog masovnog ulaganja u razvoj tehnologije koju podupire i Europska unija svojim zelenim i digitalnim ciljevima.

## 6.2. Umjetna inteligencija u kontekstu Industrije 4.0 i Industrije 5.0

Razvoj Industrije 4.0 i težnja budućoj Industriji 5.0 značajno doprinose interesu poslovnih subjekata, između kojih i logističkih poduzeća, za primjenu tehnologije umjetne inteligencije kao potpore postojećim procesima u poslovanju. Naziv *Industrija 4.0* prvi puta se spominje 2011. godine od strane njemačke poslovne, političke i znanstvene zajednice. (Skobeley, 2017)

Sastavnice Industrije 4.0 prema Pristeru (2019) su:

- umjetna inteligencija,
- robotika,
- nanotehnologija,
- internet stvari (IoT),
- autonomna vozila,
- kvantna računala
- i 3D tisak (ispis u tri dimenzije).

Definirana podjela Industrije 4.0 prikazuje tehnologije od kojih se sastoji, ali autonomna vozila i robotika također obuhvaćaju primjenu elemenata umjetne inteligencije i interneta stvari u svom djelovanju te su njihova svojevrsna nadogradnja, a nisu nužno izdvojeni elementi.

[74] detaljnije razrađeno u poglavlju 4.2.5 'Napredna analitika'.

[75] izvedeno iz eng. naziva 'list processing' – procesiranje liste/i.

Tranzicija iz Industrije 4.0, koja se bazira na povezivanju i interakciji između uređaja, u Industriju 5.0 očituje se u tome što Industrija 5.0 podrazumijeva i interakciju između uređaja i ljudi. Rapidni razvoj tehnologije već danas omogućava interakciju između uređaja i ljudi, ali postavlja se pitanje komuniciranja između uređaja i ljudske misli umjesto detekcije govora, pokreta, otiska prsta ili prepoznavanja lica korištenjem algoritama strojnog učenja. Prema Skobelevu (2017) Vijeće za znanost, tehnologiju i inovacije Vlade Japana umjesto naziva Industrija 5.0 predlaže naziv Društvo 5.0 (SuperPametno društvo)[76].

Skobley (2017) navodi da se Industrija 5.0, za razliku od Industrije 4.0, ne odnosi samo na proizvodnju, već i na rješavanje društvenih problema u kojima se naprednim tehnologijama, internetom stvari, robotima, umjetnom inteligencijom i proširenom stvarnosti doprinosi kvaliteti različitih usluga (spominju se i javne usluge) kroz personaliziran pristupu korisniku.

## 6.3. Dizajniranje proizvoda umjetne inteligencije

Proizvod umjetne inteligencije rezultat je spremnosti poslovne organizacije na iskorak prema digitalnoj transformaciji poslovanja. Inicijativa za kreacijom proizvoda umjetne inteligencije proizlazi iz *bottom-up*[77], *top-down*[78] ili horizontalnog pristupa. *Bottom-up* pristup označava inicijativu nižih organizacijskih razina koje prepoznaju prilike za implementaciju tehnologije umjetne inteligencije u poslovanje ili dizajn proizvoda iz prikupljenih resursa uobičajenog poslovanja prema višim razinama koje inicijativu odobravaju. *Top-down* pristup odnosi se na strateško određenje organizacije prema implementaciji i proizvodnji digitalnih proizvoda. Pokretanje inicijative na istoj organizacijskoj razini rezultat je horizontalnog pristupa.

Subirana (n.d.) spominje da se dizajniranje proizvoda umjetne inteligencije sastoji od četiri elemenata koje je potrebno uzeti u obzir:

- inteligencije,
- poslovnog procesa,
- tehnologije,
- i poboljšanja.

Svaki element dizajniranja proizvoda sastoji se od dva elementa niže razine koji pobliže objašnjavaju njegov kontekst.

[76] eng. 'Society 5.0 – (SuperSmart society)'.

[77] hrv. od nižih razina prema višim razinama.

[78] hrv. od viših razina prema nižim razinama.

### 6.3.1. Inteligencija

Prema Subirani (n.d.) prvi element dizajniranja proizvoda umjetne inteligencije je inteligencija.


*Slika 6.1 'Element 1 - Inteligencija',*
*Izvor: Adaptirano prema Subirana, B. (n.d.): 'AI and Machine Learning for Everyone Module 1, Introduction to the Artificial Design Process', MITxPRO module, Emeritus*

Sam pojam umjetne inteligencije u svom nazivu sadrži inteligenciju, a upravo je ona inspiracija za kreiranje novog proizvoda. Inteligenciju kao pojam teško je opisati jer i sama metakognicija čovjeka ne može do kraja objasniti razvoj i pojavu određene misli u ljudskome mozgu, a isprepleteni emocionalni i racionalni segmenti njegova djelovanja još više otežavaju repliciranje spomenutih značajki u uređaje i strojeve.

Jedan od elemenata inteligencije prema promatranom modelu je mjerenje performansi. Mjerenje performansi odnosi se na uspostavljanje metrike u fazi planiranja proizvoda umjetne inteligencije. Kao neki od primjera mjerenja uspješnosti modela uzimaju se brzina detekcije objekata, pogreške u prepoznavanju i agilnost samog modela u uvjetima koji nisu idealni. Uspješnost modela od 99% teško može biti dostižna i u većini slučajeva smatra se prihvatljivom, ali u sustavima koji mogu narušiti čovjekovo zdravlje i život, poput primjerice autonomnih vozila, ona nije nužno prihvatljiva i potrebno je implementirati dodatne sigurnosne elemente.

Drugi element inteligencije u promatranom modelu je djelokrug. Djelokrug opisuje kontekst unutar kojeg se planira proizvod, odnosno što se njime želi postići i na koji način se ono odražava na okolinu u kojoj djeluje. Primjer definiranja djelokruga je određivanja relevantnih podataka koje je potrebno prikupiti sofisticiranim uređajima za osposobljavanje algoritama strojnog učenja. Organizacija koja provodi projekt optimizacije prometnog sustava određuje specifične elemente koje algoritam strojnog učenja detektira na samoj lokaciji, a ovisno o ciljevima projekta, to se može odnositi na brojanje prometnih prekršaja, stupanj zagušenja prometnih tokova i slično.

### 6.3.2. Poslovni proces

Prema Subirani (n.d.) element dizajniranja proizvoda umjetne inteligencije koji slijedi nakon inteligencije je poslovni proces.


*Slika 6.2 'Element 2 – Poslovni proces',*
*Izvor: Adaptirano prema Subirana, B. (n.d.): 'AI and Machine Learning for Everyone Module 1, Introduction to the Artificial Design Process', MITxPRO module, Emeritus*

Utjecaj primjene umjetne inteligencije u poslovanju i dizajniranju proizvoda radikalno mijenja postojeće poslovne procese i kreira temelje za sve poslovne procese budućnosti. Prema promatranom modelu, nužno je obratiti pozornost na strateške i operativne segmente poslovnih procesa kako bi se postigla cjelovitost projekta na svim njegovim razinama.

Hax (n.d) prema Subirani (n.d) predlaže primjenu 'Delta modela' pri strateškom planiranju poslovnih procesa, a on se sastoji od 3 ključne strategije kojima se poduzeće može voditi:

- dizajniranje najpreciznijeg i najpouzdanijeg proizvoda,
- dizajniranje personaliziranog proizvoda prema zahtjevima klijenta
- i dizajniranje proizvoda s najširom bazom podataka/korisnika.

Operativni segment poslovnih procesa odnosi se na procjenu učinka tehnologije umjetne inteligencije na uobičajeno poslovanje. Obuhvaća detaljno opisivanje načina funkcioniranja procesa prije i nakon implementacije proizvoda umjetne inteligencije, uvažavajući pritom zahtjeve tehničke, tehnološke, kadrovske i strukturne[79] kompetentnosti, koji se stavljaju pred poduzeće.

[79] prema modelu organizacijske kompetentnosti koji navodi Buntak, K. (2016) – detaljnije u poglavlju 4.4.

### 6.3.3. Tehnologija

Prema Subirani (n.d.) treći element dizajniranja proizvoda umjetne inteligencije je tehnologija.


*Slika 6.3 'Element 3 - Tehnologija',*
*Izvor: Adaptirano prema Subirana, B. (n.d.): 'AI and Machine Learning for Everyone Module 1, Introduction to the Artificial Design Process', MITxPRO module, Emeritus*

Tehnologija umjetne inteligencije obuhvaća elemente koji su nužni za ostvarenje strateških i operativnih ciljeva poslovnih procesa. Ona kreira kontekst i definira ostvarivi doseg proizvoda umjetne inteligencije sukladno infrastrukturnim, financijskim, pravnim, kadrovskim i ostalim mogućnostima poslovne organizacije.

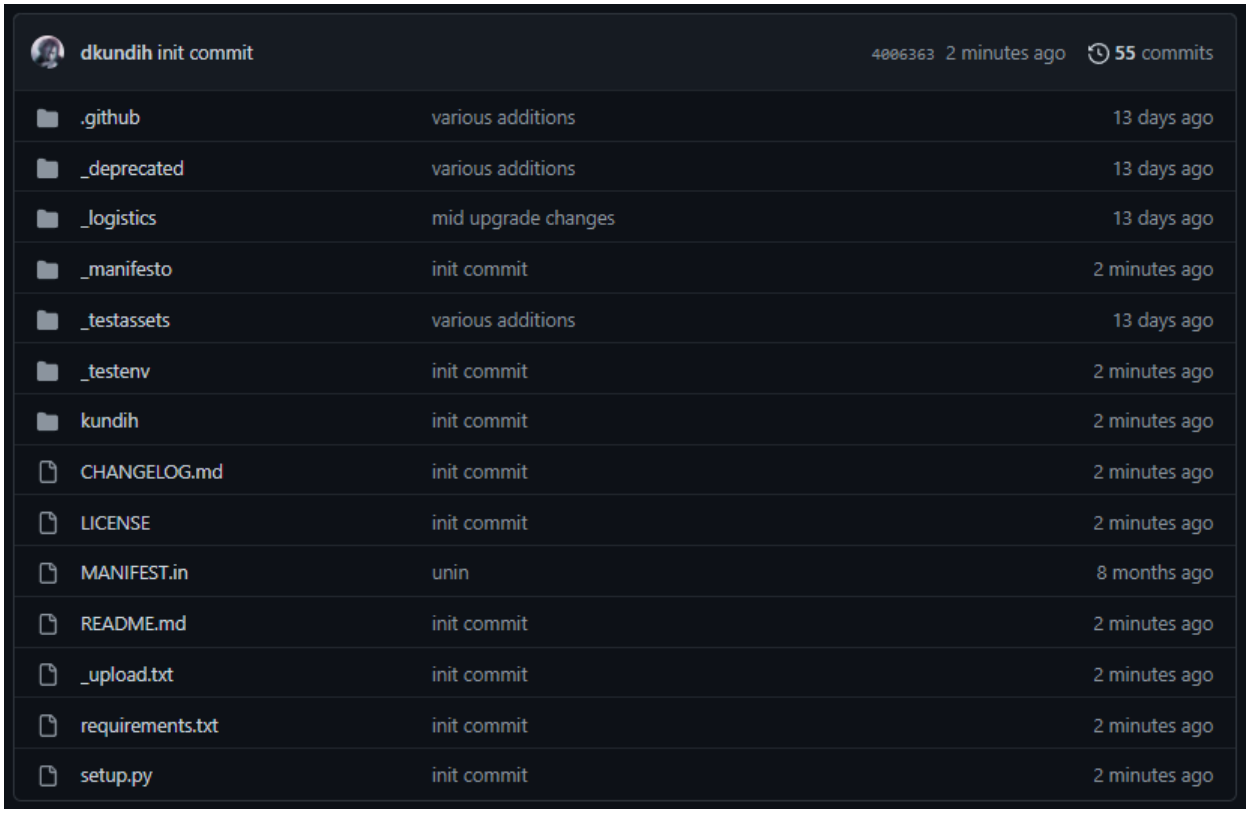
Intelektualno vlasništvo jedan je od elemenata tehnologije. Ono se odnosi na ispunjenje uvjeta za legalno korištenje javno dostupnih tehnologija poput otvorenog koda[80] algoritama koji se koriste u proizvodu, te na zaštitu vlastite inovacije i definiranje uvjeta korištenja. Postoji velik broj javno dostupnih besplatnih algoritama, ali potrebno je poštovati uvjete propisane licencom, odnosno, učiniti licencu proizvoda umjetne inteligencije kompatibilnom s njezinim pripadajućim dijelovima. Oblici licence koji se često koriste u programiranju su MIT[81], Apache verzija 2.0 i GNU GPL[82] verzija 3. Iako postoji opravdana bojazan od javne objave programskog koda proizvoda umjetne inteligencije, ono također omogućava doprinos javnosti novim idejama i integraciju u druge proizvode.

[80] eng. 'open source'.

[81] eng. 'Massachusetts Institute of Technology'.

[82] eng. 'General purpose license' – licenca opće namjene.

Na taj se način može potaknuti i personalizacija sadržaja, kao i njegova proširena upotreba. Platforma *GitHub* korisnicima omogućava pohranu programskog koda i ostalih dokumenata poput licence, manifesta, povijesti verzija i marketinških materijala besplatno, uz uvjet da su objavljeni javno, u suprotnom postoji destimulirajuće ograničenje pohrane. Alternativa platformi *GitHub* sa sličnim značajkama je *GitLab*, dok se instalacija i ispostava paketa može vršiti uz pomoć platformi *Docker* i *PyPi*[83] (Python).




*Slika 6.4 'Prikaz GitHub repozitorija',*
*Izvor: www.github.com/dkundih/kundih– snimak zaslona*

Pristup podacima[84], i općenito upravljanje podacima, drugi je element tehnologije u promatranom modelu. Planiranje prikupljanja podataka neophodno je zbog osiguranja dovoljnog kapaciteta skladišta ili jezera podataka, kao i njihove sigurnosti. Prikupljanje podataka esencijalno je za efikasnost i efektivnost algoritama strojnog učenja jer oni doprinose njihovoj preciznosti. Poslovna organizacija odlučuje na koji način što bolje konstruirati proizvod umjetne inteligencije koji u realnom vremenu koristi prikupljene podatke i integrira ih u postojeće algoritme strojnog učenja, čime se osigurava kontinuitet i relevantnost usluge. Pristup podacima podliježe brojnim pravnim i moralnim preprekama, stoga sigurnost korisničkih podataka postaje imperativ.

[83] eng. 'Python package index' – hrv. indeksiranje Python paketa.

[84] detaljnije opisano u poglavlju 4.2.

### 6.3.4. Poboljšanja

Prema Subirani (n.d.) četvrti i posljednji element dizajniranja proizvoda umjetne inteligencije su poboljšanja.


*Slika 6.5 'Element 4 - Poboljšanja',*
*Izvor: Adaptirano prema Subirana, B. (n.d.): 'AI and Machine Learning for Everyone Module 1, Introduction to the Artificial Design Process', MITxPRO module, Emeritus*

Održivost proizvoda odražava se u njegovoj agilnosti i otpornosti na promjene u okolini. Kontinuirano poboljšanje preduvjet je za dugotrajnost proizvoda i njegovu prilagodbu na moguće oscilacije na tržištu, posebice u rapidno razvijajućim industrijama današnjice. Poboljšanja se prema promatranom modelu granaju na razvoj softverskog rješenja te identifikaciju rizika i prepreka u procesu planiranja, kao i izradu i održavanje proizvoda umjetne inteligencije.

Razvoj softverskog rješenja počiva na potrebama korisnika proizvoda ili usluge. Korisnik očekuje stabilnost i efektivnost proizvoda umjetne inteligencije, pri čemu se od razvojnih inženjera očekuje održavanje sustava, ažuriranje sustava korištenjem naprednijih tehnologija i algoritama, ažuriranje baza podataka (ili skladišta podataka/jezera podataka), usklađenje s ažuriranim verzijama povezanih programskih rješenja i slično.

Element rizika i prepreka proizvoda umjetne inteligencije sagledava se kroz preventivno i reaktivno djelovanje na rizike, što propisuju i brojne tehnike upravljanja rizicima prema normi ISO 31010, primjerice *leptir-mašna* (Kundih, 2020). Postupak donošenja odluke algoritma strojnog učenja može biti pod utjecajem pristranosti, zbog čega je potrebno proizvod umjetne inteligencije podvrgnuti intenzivnom testiranju prije ispostave korisniku. Poželjno je održavati kontakt s korisnikom kako bi se otklonile poteškoće koje nastaju pri primjeni proizvoda.

## 6.4. Etika umjetne inteligencije

Pojam etike umjetne inteligencije može se sagledavati kroz procjenu ispravnosti njezina odlučivanja u operativnom smislu i na njezin utjecaj na društvo općenito.

Doseg tehnologije umjetne inteligencije bitno se povećava iz godine u godinu i ona biva sve preciznija u svojim operacijama. Dugi niz godina postoji bojazan od dominacije robota nad čovječanstvom, gubitaka radnih mjesta i smisla ljudskog postojanja ako je robot sposoban učiniti nešto brže, preciznije i jeftinije koristeći se vlastitom inteligencijom. Pojam umjetne inteligencije često se povezuje s vizijom distopijske budućnosti u kojoj se postiže masovna kontrola i monitoring ljudskih aktivnosti korištenjem naprednih tehnologija koje su u službi vlasti.

Iako spomenuta viđenja umjetne inteligencije mogu zvučati paranoično i radikalno, ona mogu ukazivati na potencijalne probleme koje nove tehnologije donose. Sporost pravne regulacije novih tehnologija doprinosi dugim eksploatacijskim razdobljima poduzeća koja djeluju bez nadzora i svojevoljno, profitirajući pritom na potencijalno nemoralan način.

'Ljudi žele stvoriti računala koja su pametnija od čovjeka, ali to za sobom povlači rizik da će računala proizvesti nova računala koja će postajati sve pametnija, s konstantom rasta sposobnosti i inteligencije koja će uvelike nadmašiti ljudsku. Teško je u tom slučaju zamisliti da bi strojevi koji su milijune puta pametniji od najpametnijeg čovjeka, ljudima služili kao robovi.'

(Puzek, 2018)

### 6.4.1. Umjetna inteligencija i uvođenje univerzalnog temeljnog dohotka

Algoritmi strojnog učenja u kombinaciji sa znanjima iz robotike mogu uvelike utjecati na strukturu zaposlenosti stanovništva. Radna mjesta koja se ne baziraju na visokoj kompleksnosti operacija već danas bivaju zamijenjena standardnim digitalnim tehnologijama koje i ne sadrže elemente umjetne inteligencije, primjerice, kupovina karata za vlak vrši se online putem ili korištenjem specijaliziranih uređaja na kolodvorima umjesto obavljanja istog na šalteru. Sposobnost strojeva s elementima tehnologije umjetne inteligencije još je i veća zbog mogućnosti učenja i personalizirane primjene. Nastavno na ovu problematiku u javnosti se često spominje pojam univerzalnog temeljnog dohotka koji bi osigurao stabilne prihode svim stanovnicima, između ostalog i onima koji su na radnim mjestima zamijenjeni digitalnim tehnologijama.

Tipurić et al. (2020) definiraju univerzalni temeljni dohodak kao 'koncept kojeg izvorno obilježavaju tri svojstva: univerzalnost (prema svima), bezuvjetnost (nema ograničenja) i davanje isključivo u novcu' te navodi da ga 'ta obilježja razlikuju od zajamčenog minimalnog dohotka čija pokrivenost je točno određena i uvjetovana krivuljom distribucije dohotka.'

Bruun i Duka (2018) navode da pojam univerzalnog temeljnog dohotka nije novina, već da seže u 1795. godinu kada englesko-američki politički aktivist i filozof Thomas Paine povodom uvođenja sustava zemljišnog posjedovanja predlaže osnivanje posebnog proračuna iz kojeg bi se plaćao poseban porez kao naknada za gubitak prirodnog nasljedstva oštećenih vlasnika zemlje, i to u doživotnom periodu za svakog od njih.

Brunn i Duka (2018) također predlažu uvođenje univerzalnog temeljnog dohotka zbog djelovanja umjetne inteligencije na promjenu cijelog spektra poslova, smatrajući da se tehnologija umjetne inteligencije bitno razlikuje od prethodnih inovacija poput parnih strojeva i električne energije jer povezuje visoku brzinu i efikasnost stroja sa sposobnošću i kreativnošću koja je na razini samog čovjeka.

Nedvojbeno je da umjetna inteligencija mijenja sve segmente ljudskog života pri čemu ni radna mjesta nisu pošteđena od implementacije iste. Daljnjim razvojem umjetne inteligencije politika će morati pristupiti spomenutom problemu i zajedno sa stručnjacima kreirati sustav koji omogućava tehnološki razvoj uz eliminaciju štetnog učinka na čovječanstvo.

U kontrastu mogućeg nestajanja pojedinih radnih mjesta koja bivaju zamijenjena tehnologijom, javlja se i potreba za specijaliziranim stručnjacima u određenom polju znanosti koje proizlazi kao rezultat tehnološkog napretka.

Spomenuti model univerzalnog temeljnog dohotka može podsjećati na propale pokušaje socijalističkih uređenja u kojima se nastojala održati ravnoteža i raspodjela bogatstva, ponekad i neovisno o radnom učinku i doprinosu individue.

Peričić (2014) u pogovoru hrvatskog izdanja knjige '1984' Georgea Orwella spominje frazu *Ako je sve podjednako vrijedno, što je doista vrijedno?*, a to je također primjenjivo i na moguće uvođenje univerzalnog temeljnog dohotka, koji bi tretirao svakog podjednako i možebitno demotivirao stvaranje dodane vrijednosti.

Poticanje inflacije kroz pretjeranu ovisnost tržišta o državi također je valjan rizik koji je potrebno uzeti u obzir pri implementaciji modela. Jedan od problema uvođenja univerzalnog temeljnog dohotka također je i nemogućnost samoaktualizacije zaposlenika kroz rad jer, koliko god pojedini poslovi bili monotoni, oni također mogu biti i veza zaposlenika s vanjskim svijetom i prilika za socijalnu interakciju te uključenje u društvena događanja.

Iako je ideja univerzalnog temeljnog dohotka osigurati stabilne prihode kojima se financiraju osnovne životne potrebe, što je samo po sebi prihvatljivo zbog ispunjenja stavki nižih razina Maslowljeve hijerarhije potreba, ono također može djelovati destimulirajuće na potragu za dodatnim poslom i polučiti kontraefektom.

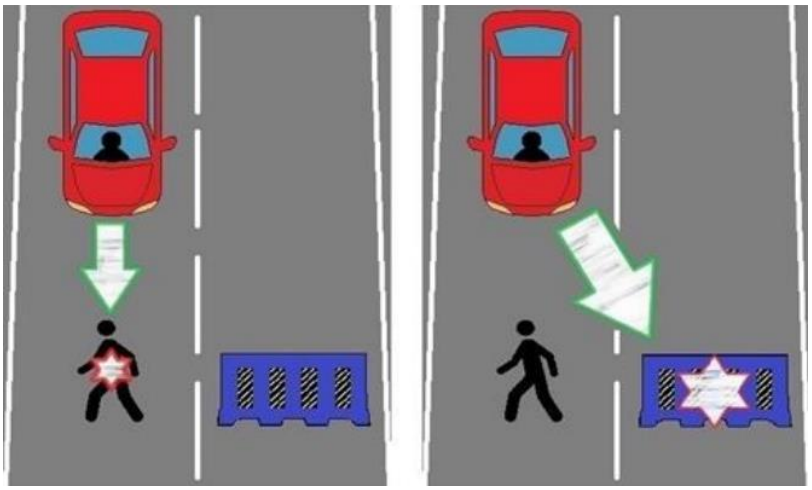
Prototip primjene modela univerzalnog temeljnog dohotka od 560 eura (4213 HRK)[85] mjesečno proveden je u dvogodišnjem periodu (2017 i 2018) u Finskoj i to na uzorku od dvije tisuće ljudi u rasponu dobi od 25-58 godina. (Kangas et al., 2019)

Prema istraživanju Kangasa et al. (2019) provedenom na uzorku od 2 000 pripadnika testne grupe (korisnika univerzalnog temeljnog dohotka) i 173 000 pripadnika kontrolne grupe (opće javnosti), uvođenje univerzalnog temeljnog dohotka podupire 85% korisnika testne grupe i 75% ispitanika kontrolne grupe. Rezultati istraživanja pokazuju da se glavni benefiti uvođenja univerzalnog temeljnog dohotka očituju kroz manje zdravstvenih problema, nižu razinu stresa i poboljšanje koncentracije, ali i na pozitivniji pogled na budućnost ispitanika u testnoj grupi, što je objašnjivo kroz izglednu stabilnost prihoda, neovisno o stanju zaposlenosti. (Kangas et al., 2019)

### 6.4.2. Rizici operativnog djelovanja umjetne inteligencije

Kao i ljudsko djelovanje, djelovanje algoritama strojnog učenja u sklopu umjetne inteligencije također nije nepogrešivo. Pokušaj preslikavanja ljudske inteligencije na neki uređaj ili stroj kompleksna je aktivnost, primarno zbog otežanog definiranja što inteligencija uistinu jest i iz čega proizlazi. Čovjek je svjestan vlastitog postojanja, ali način na koji dolazi do misli teško je dokučiv jer misao sama po sebi ne može se locirati, a isto vrijedi i za pamćenje iste. Upravo je to polazište proučavanja umjetne inteligencije i njezinog djelovanja s obzirom na to da čovjek može djelovati racionalno ili emocionalno, što je teško zapisati u obliku računalnog koda.

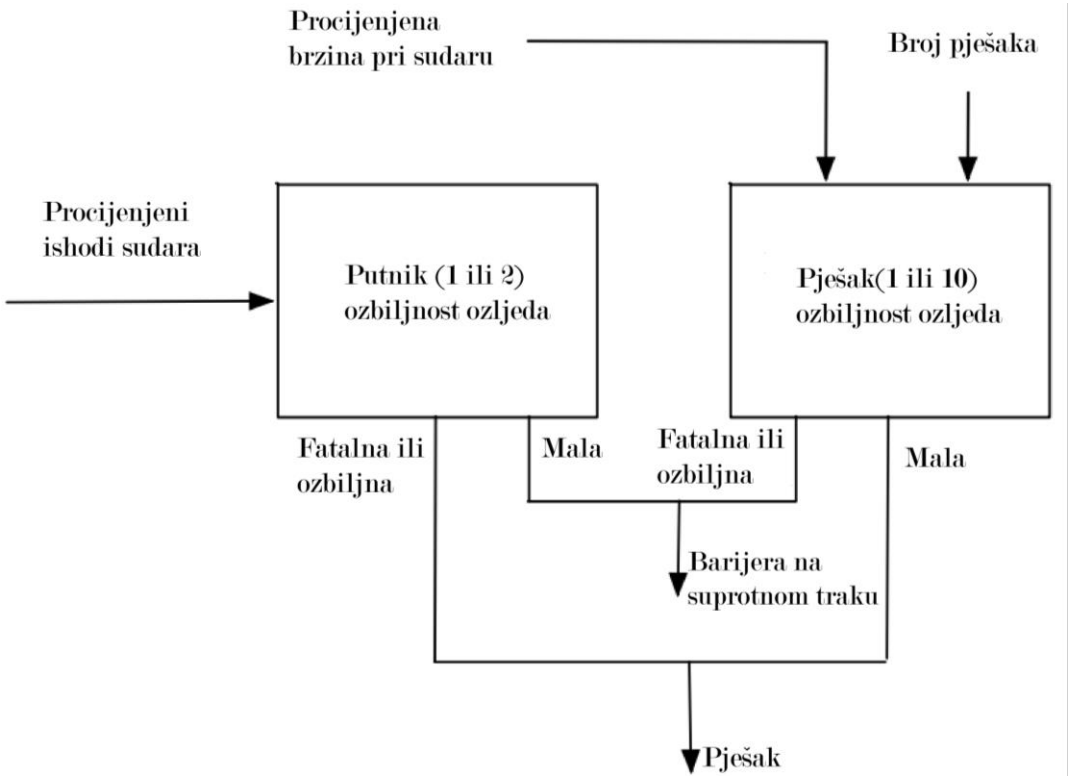
Slika 6.6 prikazuje primjer etične dileme autonomnog vozila u slučaju sigurnog udarca u nailazećeg pješaka uz postojanje prepreke na suprotnom prometnom traku.



*Slika 6.6 'Etična dilema autonomnog vozila u nailasku pješaka uz postojanje druge prepreke', Izvor: https://www.sciencedirect.com/science/article/pii/S2405896319304136*

[85] prema tečaju 1 EUR = 7.53 HRK.

Rizici koji se vezuju uz operativno djelovanje umjetne inteligencije prepoznaju se pri pokušaju izrade potpuno autonomnih vozila koja bez vozača u realnom vremenu procjenjuju prometno stanje i sudjeluju u prometu sa ostalim vozilima. Vožnja automobila rizična je operacija zbog nepredvidivosti prometnog stanja, ali i zbog ostalih uključenih sudionika poput pješaka, biciklista te ostalih vozila koja sudjeluju unutar prometnog sustava. Postavlja se pitanje etične ispravnosti djelovanja algoritma strojnog učenja u situacijama koje nose potencijalne ljudske žrtve poput mogućeg udarca autonomnog vozila u nekog od sudionika prometnog sustava koji se iznenadno i nepropisno uključuje u promet, a pri čemu ga nije moguće pravovremeno zaobići. Na algoritam strojnog učenja postavlja se dilema izravnog udarca u tog sudionika prometnog sustava ili naglo skretanje autonomnog vozila kako bi se izbjegao izravan udarac, ali i ozlijedio korisnik samog autonomnog vozila. Bitno je spomenuti da je navedena situacija moguća i pri vožnji klasičnog automobila u kojem vozač donosi odluku instinktivno, pri čemu je također teško procijeniti koja je odluka u tom trenutku prikladnija, iako su obje nepoželjne i nose sa sobom ozbiljne posljedice.

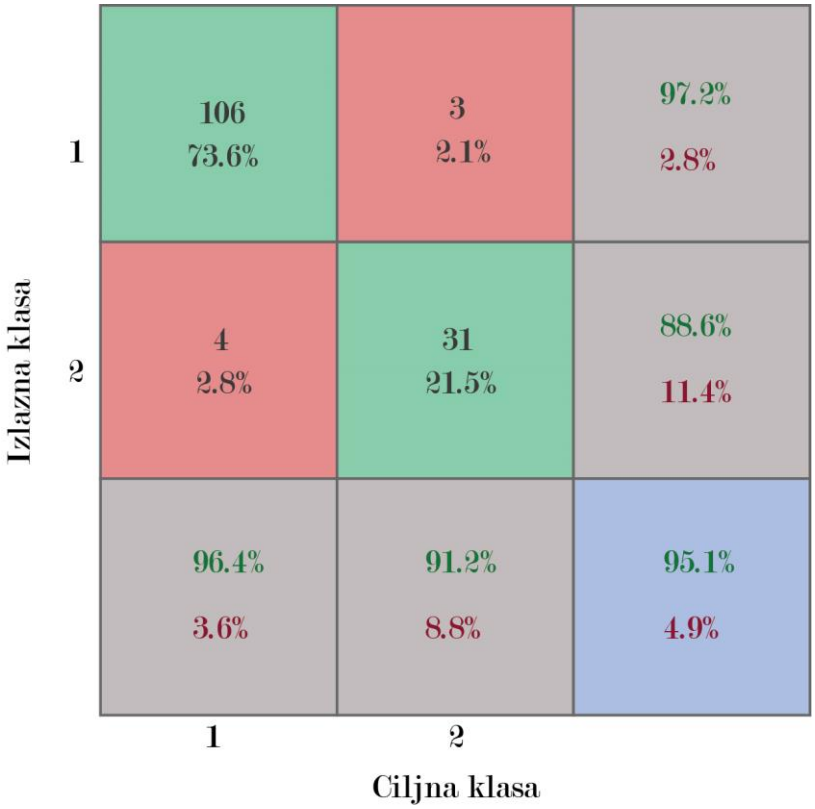



*Slika 6.7 'Postupak odlučivanja autonomnog vozila u situaciji sigurnog udara',*
*Izvor: Adaptirano prema https://www.sciencedirect.com/science/article/pii/S2405896319304136*

Pickering et al. (2019) kreiraju model (Slika 6.7) prema kojem autonomna vozila mogu procijeniti varijantu koja donosi manje štete u situaciji etične dileme s izglednim posljedicama.

Prikazana shema je orijentacijska iz razloga što je teško predvidjeti kretanje pješaka koji može skočiti u neku od strana kako bi izbjegao sudar, a ključnu ulogu pri odlučivanju autonomnog vozila imaju brzina vožnje i vremenski uvjeti, ali i broj pješaka na koje samo autonomno vozilo nailazi.

Algoritam strojnog učenja također može i krivo identificirati sudionike u prometu kroz primjenu detekcije objekata, ne uočavajući pješaka koji je obučen na atipičan način, ili ako kamera koja raspoznaje objekte radi otežano zbog vremenskih uvjeta, slabe vidljivosti u noći ili pak direktnog zasljepljenja svjetlom drugog automobila koje dolazi u susret. Nadogradnja sustava detekcije ostalih prometnih sudionika kroz primjenu umjetne inteligencije može se postići ugradnjom različitih senzora u autonomno vozilo kao dodatnog načina podizanja sigurnosti.




*Slika 6.8 'Matrica zabune na primjeru raščlambe dvije klase',*
*Izvor: Adaptirano prema Visa, S., Ramsay, B., Ralescu, A.L., Van Der Knaap, E., (2011): 'Confusion matrix-based feature selection', MAICS, 710(1), pp.120-127.*

Slika 6.8 prikazuje matricu zabune[86] koja proizlazi kao rezultat ocjene uspješnosti algoritma strojnog učenja, sastavnog dijela autonomnih vozila kojim se postiže klasifikacija objekata u okolini i planira interakcija sa subjektima prometnog sustava prema prometnim propisima.

Zelenom bojom označena su polja ispravne klasifikacije, tzv. istinski pozitivna i istinski negativna klasifikacija. Crvenom bojom označena su polja neispravne klasifikacije, tzv. neistinita pozitivna i neistinita negativna klasifikacija. Sivim poljima prikazane su horizontalna i vertikalna analiza uspješnosti, odnosno neuspješnosti, klasifikacije dok je plavom bojom prikazana cjelokupna uspješnost modela.

[86] eng. 'confusion matrix'.

### 6.4.3. Zlouporaba umjetne inteligencije i težnja distopijskoj stvarnosti

Brojne ljudske inovacije bivaju iskorištene u negativne svrhe, od izuma dinamita Alfreda Nobela koji postaje oružje, do napredovanja u znanosti koje je doprinijelo izradi atomske bombe. Umjetna inteligencija također predstavlja opasnost od eksploatacije u negativne svrhe, poglavito u ratnom djelovanju zbog manjeg izlaganja ljudi na bojišnici i preciznosti koja se može postići sofisticiranim i naprednim algoritmima u kombinaciji s robotikom.

Sloboda čovjeka također biva narušena zbog kontinuiranog motrenja, identifikacije lica i praćenja korištenjem algoritama strojnog učenja namijenjenih za detekciju objekata. Brojne autoritarne zemlje već koriste tehnologiju umjetne inteligencije kao element nadzora i upravljanja društvenim procesima. Shen (2019) navodi da se prema planu Narodne Republike Kine iz 2014. godine nastoji uvesti sustav društvenog bodovanja[87] koji se sastoji od 3 ključne značajke:

- **Upravljan Vladom, građen od strane naroda**
  primjenjuje se na raznolik sadržaj usluga od interesa za stanovnike,
- **Kreiranje zajedničkog sustava razmjene podataka**
  bilježenje, dijeljenje i transfer podataka između poduzeća i ministarstava,
- **Nagrađivanje i penalizacija**
  bodovanje uzornih građana i institucija uz pogodnosti poput medijske promocije, prioritetni položaj za usluge države. Penalizacija se očituje u javnoj osudi, uvrštavanje u *crnu listu* i eliminaciju s tržišta.

(Shen, 2019)

Spomenuti prijedlog sustava društvenog bodovanja egzistira u suživotu s tehnologijom, a zbog masovnosti sustava i broja stanovnika Narodne Republike Kine, upravo je ona temelj na kojem počiva izvedivost projekta jer se tehnologija umjetne inteligencije nameće kao korektiv u društvu uz bok represivnim i sigurnosnim službama.

Creemers (2018) navodi da se jasno propisuje namjera proširenja upotrebe automatiziranih sustava baziranih na prikupljenim podacima, i to ne samo u planu društvenog bodovanja, već i u nacionalnim strategijama vezanih uz velike podatke i umjetnu inteligenciju.

Knight i Creemers (2021) stavljaju sustav društvenog bodovanja u kontekst kineskog pristupa upravljanju pandemijom SARS-COV-2, pri čemu navode da je sustav regionalnim vlastima omogućio regulaciju cijena medicinskih proizvoda, penalizaciju za izbjegavanje propisane samoizolacije te za konzumaciju divljih životinja.

[87] eng. 'social credit system'.

## 6.5. Slaba i jaka umjetna inteligencija

Relevantna istraživanja vezana uz umjetnu inteligenciju pojavljuju se tek unazad nekoliko desetaka godina, iako ideja o autonomnim objektima navođenih vlastitom inteligencijom postoji puno duže. Upravo iz tog razloga nije sasvim jasan smjer daljnjeg razvoja umjetne inteligencije, poglavito zbog mogućnosti pojave pravnih regulativa koje mogu ograničiti njezin razvoj i primjenu u određenim segmentima ljudskog života.

Searle (1980) prema Liu (2021) navodi da američki filozof John Searle prvi vrši podjelu umjetne inteligencije na slabu umjetnu inteligenciju i jaku umjetnu inteligenciju, nastalu misaonim eksperimentom kineskog sporazuma o sobi[88].

Cole (2020) navodi da misaoni eksperiment kineskog sporazuma opisuje; Zamišljanje Searlea samog u sobi s računalom koje je osposobljeno za odgovaranje na poruke koristeći se kineskim znakovima. Ispod vrata dobiva ugovor o sobi na kineskom jeziku. Searle ne razumije kineski, ali uz pomoć računala uspijeva odgovoriti odgovarajućim nizom simbola i znakova na kineskom jeziku pri čemu se dovodi do pogrešne pretpostavke osobe s druge strane vrata da se radi o kineskom govorniku u sobi. Poveznica eksperimenta i prikazane podjele umjetne inteligencije očituje se u pretpostavci da računalo prepoznaje jezik, ali bez pravog razumijevanja istog. (Cole, 2020). Ovaj je misaoni eksperiment, dakako, definiran daleko prije pojave internet prevoditelja.

Puzek (2018) opisuje podjelu slabe i umjetne inteligencije na:

- **slabu umjetnu inteligenciju**
  strojevi koji se ponašaju inteligentno.
- i **jaku umjetnu inteligenciju**
  strojevi koji se ponašaju inteligentno i doista misle.

Prema podjeli umjetne inteligencije Searlea (1980) zaključuje se da je glavna razlika između slabe i jake umjetne inteligencije ta da se slaba umjetna inteligencija prepoznaje kroz operativno izvršavanje operacija temeljenih na prethodno utreniranom scenariju, dok jaka umjetna inteligencija ima sposobnost vlastitog promišljanja i odlučivanja, odnosno, naziva se takvom zbog postojanja vlastite svijesti i sposobnosti podučavanja same sebe.

Flowers (2019) također navodi podjelu umjetne inteligencije na slabu i jaku, a kao ključnu razliku između pojmova izdvaja sposobnost jake umjetne inteligencije na razumijevanje vlastitih emocija te postojanje svijesti i mašte u kontrastu sa slabom umjetnom inteligencijom koja ima sposobnost izražavanja osjećaja, ali bez posjedovanja svijesti o njihovom značenju.

[88] eng. 'Chinese room agreement', (CRA).

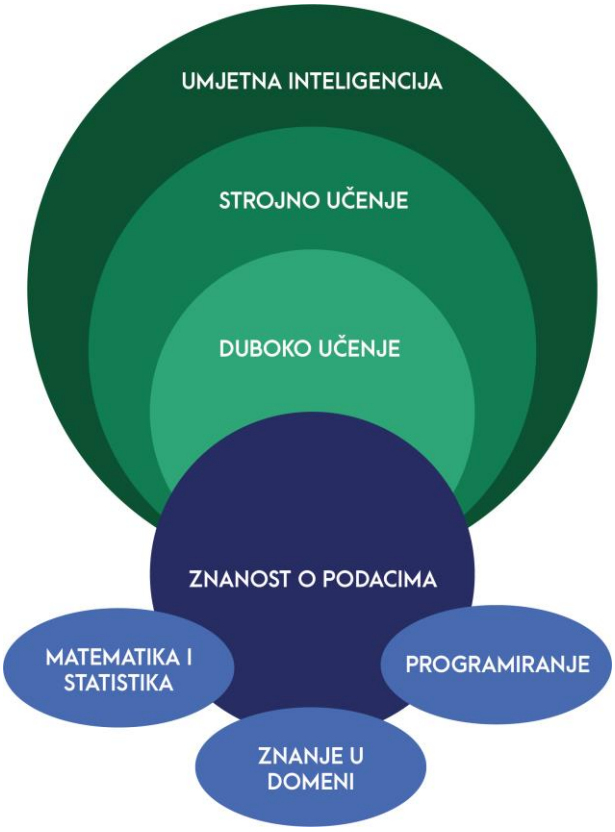
# 7. Primjena temeljnih elemenata umjetne inteligencije u prometnim i logističkim sustavima

Ovo poglavlje definira temeljne elemente umjetne inteligencije, odnosno, strojno učenje i duboko učenje te njihovu povezanost s znanosti o podacima.

Nastavno na uvodni dio o pozadini funkcionalnosti umjetne inteligencije u poglavlju 6, detaljnije je obrađena tema znanosti o podacima koja služi kao podloga za funkcioniranje algoritama strojnog učenja. Obrađene su metode zamjene podataka koji nedostaju, filtriranja podataka i skaliranja podataka korištenjem Tukeyevih ljestvi moći.

Povezani su elementi linearne algebre s opisivanjem međuodnosa podataka u prostoru korištenjem skalara, vektora, matrica i tenzora.

Tematsko područje strojnog učenja odnosi se na podjelu nadziranih, nenadziranih i djelomično nadziranih algoritama od kojih su neki detaljnije razrađeni i opisani. Algoritmi strojnog učenja jezgra su umjetne inteligencije, njima robot ili programsko rješenje prepoznaje određene međuovisnosti između podataka i daje zaključak (Kundih, 2022). Programiranje algoritama strojnog učenja u Pythonu prikazano je na primjerima prometnih i logističkih sustava.




*Slika 7.1 'Umjetna inteligencija i povezani pojmovi',*
*Izvor: Adaptirano prema Himanshu, V. (n.d.): 'AI & ML for Everyone', Office Hours with Himanshu Vaidya, Emeritus, studijska grupa veljača, 2022.*

## 7.1. Znanost o podacima

Znanost o podacima, često nazivana i terminom *podatkovne znanosti,* se prema Kelleheru i Tierneyu (2018) razlikuje od *rudarenja* podataka s obzirom na imanje većeg opsega, a bavi se 'strukturiranim i nestrukturiranim (velikim) podacima, koristeći se znanjima iz niza područja uključujući strojno učenje, statistiku, podatkovnu etiku i zakonsku regulativu te računalstvo visokih performansi.'

Kelleher i Tierney (2018) na jednostavan način opisuju pojam znanosti o podacima 'kao partnerstvo između podatkovnog znanstvenika i računala'.

O'Neil i Schutt (2013) prema Bašić (2020) posao podatkovnog znanstvenika definiraju kao 'posjedovane raznih vještina, od društvenih znanosti do biologije, te rad s velikom količinom podataka i suočavanje se s problemima vezanim uz strukturu, veličinu, red i složenost prirode podataka.' Uz razvoj tehnologije, područje znanosti o podacima uz prethodno spomenutu biologiju (i bioinformatiku općenito), širi se i na ostala područja znanosti poput medicine, istraživanja tržišta (maketing), prometa, planiranja potražnje (logistika) i dr.

Uspjeh projekta iz područja znanosti o podacima ovisi o:

- relevantnosti prikupljenih podataka,
- volumenu podataka,
- odabranim atributima.
- ažuriranju postojećih modela novim podacima,
- vještinama podatkovnog znanstvenika,
- odabranim metodama usklađenja i filtriranja podataka
- i brzini izvođenja operacija računala koje se koristi u procesu.

Kao što je prikazano na uvodnoj slici poglavlja 7, znanost o podacima u nekim se segmentima vezuje uz umjetnu inteligenciju i njezinu funkcionalnost, ali vještine koje se pritom koriste nadilaze samu podjelu.

Matematika i statistika, programiranje te znanje u domeni potrebne su vještine za rad u području znanosti o podacima (Himanshu, n.d). Matematika i statistika potrebne su za proces planiranja izrade modela i njegovog zapisivanja, evaluaciju značaja atributa koji se koriste u modelu i kreiranje algoritma kojim se može automatizirati proces filtriranja podataka.

Programiranje je *produžena ruka* matematike i statistike, ono omogućava izvedivost definiranih operacija na velikom volumenu podataka u prihvatljivoj brzini.

Znanje u domeni (primjerice promet i logistika) također je bitan element znanosti o podacima zbog utvrđivanja međuodnosa između atributa i postavljanja hipoteze istraživanja.

### 7.1.1. Rudarenje podataka i CRISP-DM

*Rudarenje* podataka[89] se prema Nikolašević (2016) definira kao 'pronalaženje zakonitosti među podacima.' *Rudarenjem* podataka dobivaju se sveobuhvatnije analize od onih korištenjem klasičnih statističkih metoda.

Za efektivnost *rudarenja* podataka potrebno je sagledati atribute koji uistinu mogu imati korelaciju s traženim ishodom i ciljevima projekta.

Šest zadataka *rudarenja* podataka koje navodi Knez (2019) prema Larose (2004) su:

- **Deskripcija**
  opisivanje međuodnosa podataka i ishoda koji iz njih proizlaze.
- **Klasifikacija**
  definiranje ciljnih varijabli (klasa) podataka.
- **Estimacija (regresija)**
  prikaz međuodnosa numeričkih skupova podataka.
- **Predikcija**
  predviđanje ishoda na temelju podataka.
- **Klasteriranje (grupiranje)**
  grupiranje numeričkih podataka u skupine (klastere).
- **Asocijacija**
  otkrivanje međuovisnosti atributa.

Mrežni sadržaj izvor je podataka za brojne operacije *rudarenja* podataka, a sami podaci se često prikupljaju u datotekama *.csv*[90] ili *.json*[91] formata koji su podložni analiziranju korištenjem programskih jezika i daljnjem procesuiranju.

Pojam koji opisuje preuzimanje mrežnog sadržaja pristupanjem izvornom kodu naziva se *struganje podataka* s mreže[92]. *Struganje podataka* s mreže možebitno nailazi na pravne prepreke s obzirom na pristupanje podacima koji potencijalno nisu poželjni za pristup od strane javnosti, ali ono se često koristi u praćenju aktualnih ponuda na platformama poput *E-baya, Amazona* i sl.

U Pythonu, metodama *struganja podataka* s mreže pristupa se javno dostupnim bibliotekama otvorenog koda poput *Beautiful Soupa, lxml-a, Seleniuma, Scrapy-a* i brojnih drugih.

---

[89] eng. 'data mining'.

[90] eng. 'comma separated values' – vrijednosti odvojene zarezima.

[91] eng. 'Javascript object notation'.

[92] eng. 'web scraping'.

```
[38]: # definiranje voznog parka u .json obliku.
vozni_park = {
    'Vozilo': [
        {
            'Tip': 'Automobil',
            'Pločica': 'ČK - 241 - DK',
            'Boja': 'Žuta',
            'Kilometraža': 112395,
        },
        {
            'Tip': 'Kombi',
            'Pločica': 'ZG - 1902 - NL',
            'Boja': 'Crvena',
            'Kilometraža': 242194,
        },
    ]
}
```

```
[39]: # popis svih vozila.
vozni_park['Vozilo']
```

```
[39]: [{'Tip': 'Automobil',
  'Pločica': 'ČK - 241 - DK',
  'Boja': 'Žuta',
  'Kilometraža': 112395},
 {'Tip': 'Kombi',
  'Pločica': 'ZG - 1902 - NL',
  'Boja': 'Crvena',
  'Kilometraža': 242194}]
```

```
[40]: # pristupanje određenom vozilu i njegovom tipu.
vozni_park['Vozilo'][0]['Tip']
```

```
[40]: 'Automobil'
```

```
[41]: # slanje vozila s 200 000 kilometara ili više na izvanredni tehnički pregled.
for vozilo in vozni_park['Vozilo']:
    reg = vozilo['Pločica']
    km = vozilo['Kilometraža']
    if km >= 200000:
        print(f'Vozilo registracije {reg} potrebno poslati na izvanredni tehnički pregled.')
    elif km < 200000:
        print(f'Vozilo registracije {reg} zadovoljava tehničke standarde.')
    else:
        break
```

```
Vozilo registracije ČK - 241 - DK zadovoljava tehničke standarde.
Vozilo registracije ZG - 1902 - NL potrebno poslati na izvanredni tehnički pregled.
```

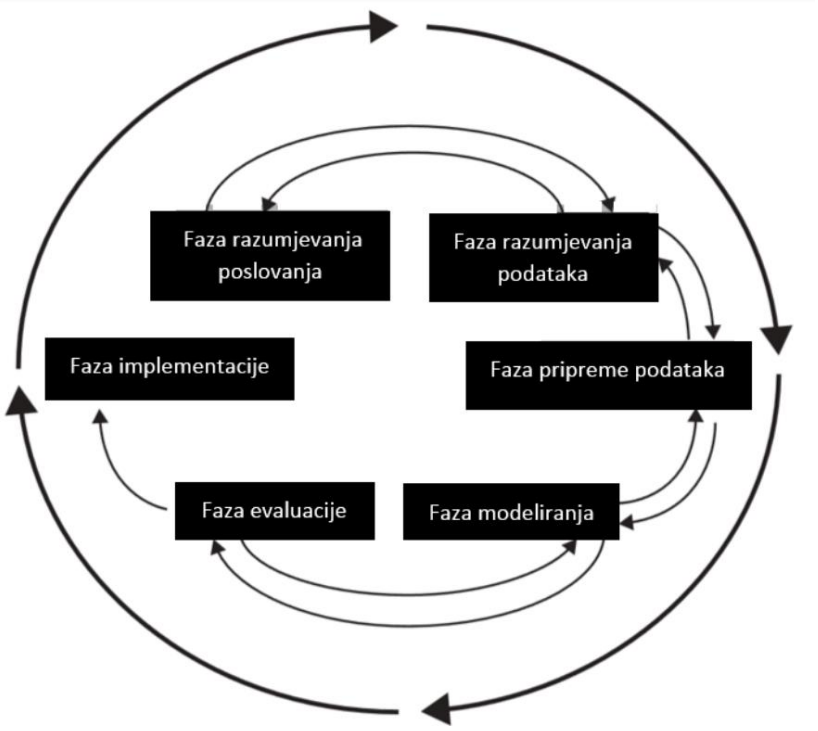
*Slika 7.2 'Primjer analize .json rječnika korištenjem Pythona',*

*Izvor: Vlastiti rad autora*

U prvom koraku prikazanom na Slici 7.2 prikazuje se definiranje podataka u *.json* obliku. U drugom se koraku ispisuje lista svih evidentiranih vozila i njihovih pripadajućih atributa, odnosno, tip vozila, registarska pločica, boja i kilometraža. Atribut kilometraže iskazan je u brojčanom iznosu (*int*), dok su ostali atributi iskazanu u tekstualnom obliku (*string*). Definiranje kilometraže u brojčanom iznosu omogućava daljnje procesuiranje i manipulaciju iznosom.

Treći korak prikazuje kombinaciju *for* i *if* petlje u ulozi ispisa registracija vozila s kilometražom većom ili jednakom 200 000 kilometara koje je potrebno poslati na izvanredni tehnički pregled, odnosno vozila s manje od 200 000 kilometara koja zadovoljavaju uvjete.

CRISP-DM[93], ili standardni proces za dubinsku analizu podataka, se prema Bašković (2018) odnosi na projekt dubinske analize podataka u šest faza prikazanih na Slici 7.3.




*Slika 7.3 'Faze CRISP-DM ciklusa',*
*Izvor: https://zir.nsk.hr/islandora/object/infri%3A313/datastream/PDF/view*

Chapman et al. (2000) prema Azevedo i Santosu (2008) faze CRISP-DM ciklusa opisuju kao:

- **Faza razumijevanja poslovanja**
  inicijalna faza razumijevanja poslovnih ciljeva i zahtjeva poslovanja.
- **Faza razumijevanja podataka**
  vrši se prikupljanje podataka, upoznavanje s podacima i njihovim nedostacima.
- **Faza pripreme podataka**
  aktivnosti kojima se kreira finalni set podataka iz sirovih podataka.
- **Faza modeliranja**
  odabire se tehnika modeliranja i vrši se ujednačenje podataka.
- **Faza evaluacije**
  provodi se detaljna evaluacija modela i ispitivanje usklađenosti s ciljevima.
- **Faza implementacije**
  održavanje modela, njegovo poboljšanje i komunikacija s klijentom.

[93] eng. 'Cross-industry standard process for data mining'.

### 7.1.2. Pearsonov i Spearmanov koeficijent korelacije između značajki

Korištenjem Pearsonovog i Spearmanovog koeficijenta korelacije u domeni znanosti o podacima utvrđuje se međuodnos dva seta značajki. Odnos (korelacija) uspoređenih značajki može biti pozitivan (obje rastu ili padaju) ili negativan (dok jedna raste, druga pada).

Bilonić (2020) navodi da 'koeficijent korelacije predstavlja mjeru asocijacije između dvije varijable, no iz njega se ne može odrediti kauzalnost', što znači da se može numerički iskazati međuodnos značajki, no ne i uzrok koji dovodi do razine utjecaja jedne značajke na drugu.

Pearsonov koeficijent korelacije se prema Bašković (2018) računa korištenjem niže navedene formule (7.1):

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (7.1)$$

r – Pearsonov koeficijent korelacije.

n – veličina uzorka.

$x_i$ , $y_i$ – individualne vrijednosti indeksa i.

$\bar{x}$ $\frac{1}{n}\sum_{i=1}^{n} x_i$ – središnja vrijednost uzorka.

Spearmanov koeficijent se prema Bilonić (2020) naziva još i Spearmanov rang.
Boslaugh (2013) prema Bilonić (2020) navodi da se Spearmanov koeficijent korelacije računa prema niže navedenoj formuli (7.2):

$$r = 1 - \frac{6*\sum d^2}{n*(n^2-1)} \quad (7.2)$$

r – Spearmanov koeficijent korelacije.

n – veličina uzorka.

d – razlika rangova.

Za donošenje zaključka o dobivenim rezultatima, uz dobivenu korelaciju, računa se i p-vrijednost kojom se opisuje širina krajnjih vrijednosti Gaussove razdiobe. P- vrijednosti također se koriste i pri analizi centriranosti i preciznosti logističkih procesa (Buntak et al., 2020).

Bašković (2018) navodi da se 'p-vrijednosti nalaze u rasponu od 0.0 do 1.0., a kako se približava 0.0, to pokazuje da je uzorak rijedak ishod u odnosu na populaciju postavljenu u hipotezi. Što je p-vrijednost bliža nuli to su veći dokazi protiv null hipoteze. Bašković (2018) također naglašava značaj postavljanja kritične p-vrijednosti koja se uzima u obzir pri usporedbi, odnosno donošenje zaključka o prihvaćanju ili odbijanju null hipoteze.

Slika 7.4 prikazuje izračun Pearsonovog i Spearmanovog koeficijenta korelacije korištenjem Pythona. U procesu izračuna korištene su Python biblioteke *scipy* (modul *stats*, funkcije *pearsonr* i *spearmanr*) te *pandas* za kreiranje testnog okvira podataka. Nakon kreiranja testnog okvira podataka, izvršavaju se funkcije:

```
scipy.stats.pearsonr(skup_1, skup_2) # definirano kao pkk.
scipy.stats.spearmanr(skup_1, skup_2) # definirano kao skk.
```

Iz ispisanih rezultata vidljivo je da se vrijednost Pearsonovog i Spearmanovog koeficijenta korelacije razlikuje u malenom iznosu (~0.02), a isto vrijedi i za izračunatu p-vrijednost koja je neznatna u oba izračunata modela.

```
[38]: # unos biblioteka
      from scipy.stats import pearsonr as pkk
      from scipy.stats import spearmanr as skk
      import pandas as pd

[39]: # kreiranje testnog okvira podataka transporta paketa. (Cijena također varira i o vrijednosti pošiljke.)
      primjerak = pd.DataFrame({
          'Masa (kg)' : [8, 3, 2, 6, 1, 1],
          'Cijena dostave (kn)' : [113, 85, 70, 98, 70, 60]
      })

[40]: # prikaz testnog okvira.
      primjerak
```

[40]:

| | Masa (kg) | Cijena dostave (kn) |
|---|---|---|
| 0 | 8 | 113 |
| 1 | 3 | 85 |
| 2 | 2 | 70 |
| 3 | 6 | 98 |
| 4 | 1 | 70 |
| 5 | 1 | 60 |

```
[41]: # definiranje varijabli za uvrštavanje u izračun.
      skup_1 = primjerak['Masa (kg)']
      skup_2 = primjerak['Cijena dostave (kn)']

[42]: # izračun Pearsonovog koeficijenta.
      pearson = pkk(skup_1, skup_2)

[43]: # izračun Spearmanovog koeficijenta korelacije.
      spearman = skk(skup_1, skup_2)

[44]: # pridodavanje vrijednosti varijablama
      P_koeficijent_korelacije, P_p_vrijednost = pearson
      S_koeficijent_korelacije, S_p_vrijednost = spearman

[45]: # ispis rezultata.
      f'Pearsonov koeficijent korelacije je {P_koeficijent_korelacije}, uz p-vrijednost od {P_p_vrijednost}.'

[45]: 'Pearsonov koeficijent korelacije je 0.9765051603362394, uz p-vrijednost od 0.000821526572493657.'

[46]: # ispis rezultata.
      f'Spearmanov koeficijent korelacije je {S_koeficijent_korelacije}, uz p-vrijednost od {S_p_vrijednost}.'

[46]: 'Spearmanov koeficijent korelacije je 0.9558823529411764, uz p-vrijednost od 0.0028766156116425898.'
```

*Slika 7.4 'Izračun Pearsonovog i Spearmanovog koeficijenta korelacije',*

*Izvor: Vlastiti rad autora*

### 7.1.3. Skaliranje značajki primjenom Tukeyeve ljestvice moći

Abdallah et al. (2017) navode da Tukey (1977) godine opisuje način drugačijeg prikazivanja varijabli koristeći se transformacijom moći, koja za cilj ima poboljšanje linearnosti između varijabli u skupu.

**Tablica 7.1 'Skala određivanja stupnja moći'**

| | | | | | | = | |
|---|---|---|---|---|---|---|---|
| Stupanj moći | -2 | -1 | -1/2 | 0 | 1/2 | 1 | 2 |
| Transformacija | $1/y^2$ | 1/y | $1/\sqrt{y}$ | log y | $\sqrt{y}$ | y | $y^2$ |

**Izvor:** Adaptirano prema https://thomaselove.github.io/2019-431-book/re-expression-tukeys-ladder-box-cox-plot.html

Tablica 7.1 prikazuje primjer skale određivanja stupnja moći vrijednosti u rangu moći koji se proteže od -2 do 2. Gornji redak prikazuje stupanj moći, a donji redak transformaciju kojom se on postiže. Stupanj moći 1 prikazan je znakom jednakosti gdje vrijedi $y = y$.

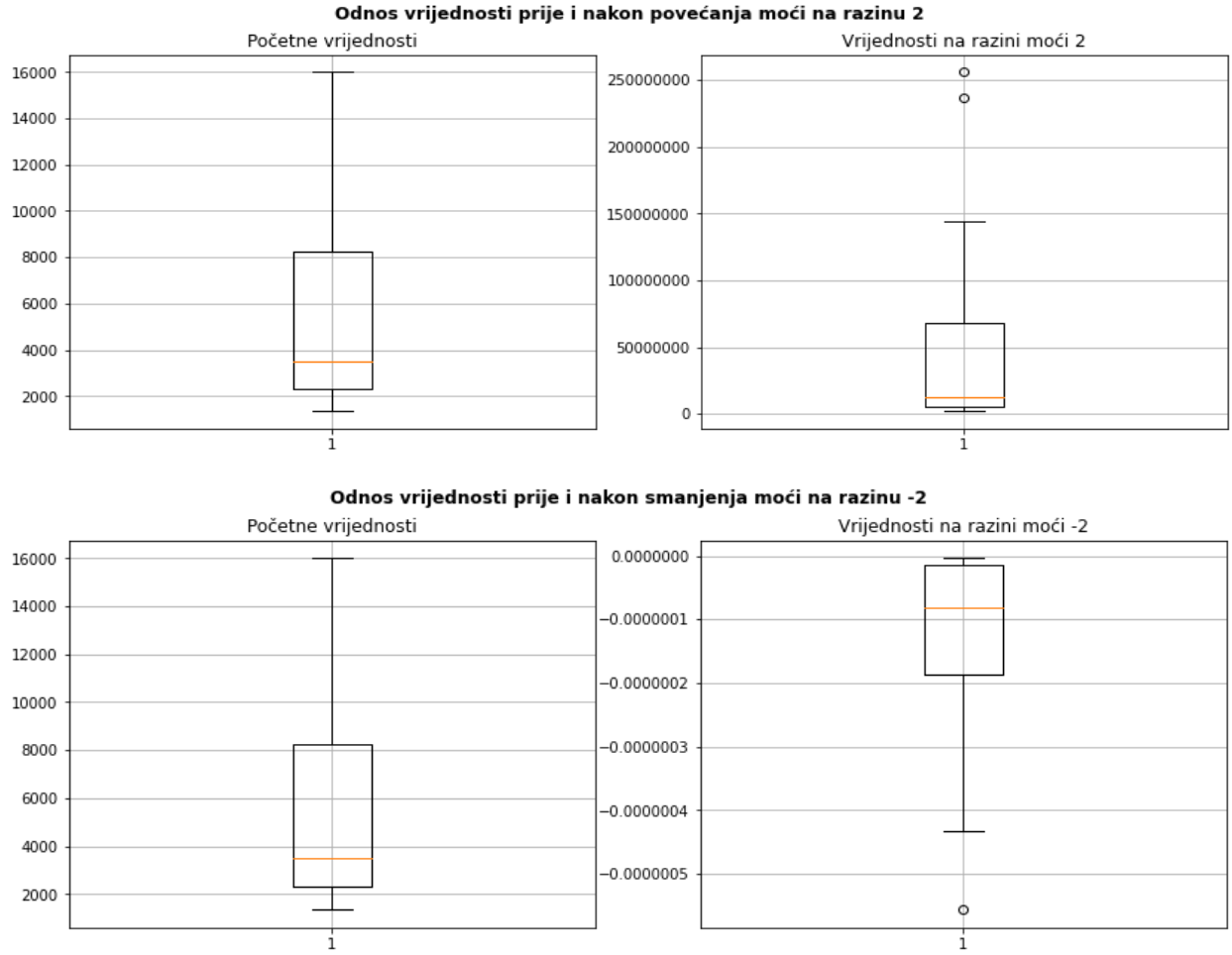
**Tablica 7.2 'Prikaz stupnja povećanja i smanjenja moći Tukeyevim ljestvama moći'**

| | Vrijednosti | Operacija (snaga 2) | Vrijednosti (snaga 2) | Operacija (snaga -2) | Vrijednosti (snaga -2) |
|---|---|---|---|---|---|
| 0 | 2020 | Vrijednosti * Vrijednosti (x * x) | 4080400 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -2.450740e-07 |
| 1 | 3210 | Vrijednosti * Vrijednosti (x * x) | 10304100 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -9.704875e-08 |
| 2 | 1340 | Vrijednosti * Vrijednosti (x * x) | 1795600 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -5.569169e-07 |
| 3 | 6320 | Vrijednosti * Vrijednosti (x * x) | 39942400 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -2.503605e-08 |
| 4 | 2310 | Vrijednosti * Vrijednosti (x * x) | 5336100 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -1.874028e-07 |
| 5 | 15400 | Vrijednosti * Vrijednosti (x * x) | 237160000 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -4.216563e-09 |
| 6 | 5630 | Vrijednosti * Vrijednosti (x * x) | 31696900 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -3.154883e-08 |
| 7 | 3520 | Vrijednosti * Vrijednosti (x * x) | 12390400 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -8.070764e-08 |
| 8 | 16000 | Vrijednosti * Vrijednosti (x * x) | 256000000 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -3.906250e-09 |
| 9 | 1520 | Vrijednosti * Vrijednosti (x * x) | 2310400 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -4.328255e-07 |
| 10 | 12000 | Vrijednosti * Vrijednosti (x * x) | 144000000 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -6.944444e-09 |
| 11 | 8250 | Vrijednosti * Vrijednosti (x * x) | 68062500 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -1.469238e-08 |
| 12 | 2410 | Vrijednosti * Vrijednosti (x * x) | 5808100 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -1.721733e-07 |

**Izvor:** Vlastiti rad autora.

Tablica 7.2 prikazuje primjer skupa vrijednosti (stupac 'Vrijednosti'), operacije kojima se postiže povećanje i smanjenje moći (stupci 'Operacija (snaga 2)' i 'Operacija (snaga -2)') te vrijednosti nakon transformacije (stupci 'Vrijednosti(snaga 2)' i 'Vrijednosti (snaga -2)').

**Grafikon 7.1 'Odnos vrijednosti prije i nakon povećanja i smanjenja moći na razine -2 i 2'**




**Izvor:** Vlastiti rad autora.

Grafikon 7.1 vizualno prikazuje raspršenost vrijednosti prikazanih u Tablici 6.2 primjenom *boxplot* dijagrama[94]. *Boxplot* dijagram sastoji se od medijalne vrijednosti koja je prikazana žutom linijom, gornje i donje granice kvadrata (kutije) koji predstavljaju 50% svih podataka, odnosno interkvartilni raspon, vanjske granice (ograde) koja se proteže u duljini 1.5 puta većoj od kvadrata i prikazuje širenje raspona vrijednosti, dok točke van skupa signaliziraju značajnije odstupanje i nazivaju se *outlier* vrijednostima (Tukey, 1977 prema *Environmental Systems Research* Institutu, 2009). Na grafikonu je prikazana usporedba raspršenosti početnih vrijednosti s lijeve strane u kontrastu s istim vrijednostima nakon izmjene stupnja moći na razinu -2 i 2. Gornji redak prikazuje podizanje stupnja moći na razinu 2, čime se većina vrijednosti približava nižoj vanjskoj granici uz pojavu dva *outliera* koji simboliziraju najviše vrijednosti, dok donji redak prikazuje snižavanje stupnja moći na razinu -2, što rezultira grupiranjem vrijednosti prema višoj vanjskoj granici i pojavu *outliera* koji simbolizira najnižu vrijednost.

[94] programski kod i detaljna razrada dostupni u Prilozima.

### 7.1.4. Zamjena nedostajućih podataka

Skupovi podataka često mogu biti nepotpuni, ali svejedno primjenjivi u području znanosti o podacima i strojnom učenju. Primjer nedostajućih podataka su prazna polja na obrascima koji ne zahtijevaju obvezan unos podataka prije slanja.

Na temelju trendova ostalih prikupljenih podataka moguće je procijeniti vrijednost značajki koje nedostaju mjerenjem korelacija između njih, odnosno, utjecaja jedne varijable na drugu. Postoji rizik od krive procjene zbog pristranosti ili stereotipnog označavanja, no moguće je s visokom preciznošću popuniti polja podacima za daljnje procesuiranje.

Neke od metoda zamjene nedostajućih podataka dostupnih u Python biblioteci *pandas*, prikazanih u nastavku poglavlja[95], su:

- *ffill* (prijenos vrijednosti retka iznad),
- *bfill* (prijenos vrijednosti retka ispod),
- interpolacija
- i srednja vrijednost.

**Tablica 7.3 'Početna tablica s nedostajućim podacima'**

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| 0 | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.1 |
| 1 | Ludbreg | 22-06-2022 | VŽ-2 | VŽ-213 | 21 | 2.1 |
| 2 | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.3 |
| 3 | Koprivnica | 23-06-2022 | VŽ-2 | KC-221 | 41 | 4.5 |
| 4 | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | NaN |
| 5 | Križevci | 23-06-2022 | VŽ-2 | KC-222 | 54 | 5.8 |
| 6 | Čakovec | 25-06-2022 | VŽ-1 | ČK-144 | 2 | 0.3 |
| 7 | Ludbreg | 25-06-2022 | VŽ-1 | VŽ-214 | 23 | 2.4 |
| 8 | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.8 |
| 9 | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | NaN |

**Izvor:** Vlastiti rad autora

Tablica 7.3 prikazuje popis isporuka logističkog poduzeća u Varaždinu s nedostajućim podacima o potrošnji goriva za lokaciju Zagreb; isporuku klijentima ZG-566 i ZG-568. Nedostajuće vrijednosti u tablici prikazane su kraticom *NaN*[96].

[95] programski kod i detaljna razrada dostupni u Prilozima.

[96] eng. 'not a number' – hrv. 'nije broj'.

**Tablica 7.4 'Izdvajanje lokacije i sortiranje vrijednosti prema udaljenosti'**

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| **0** | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.1 |
| **9** | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | NaN |
| **4** | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | NaN |
| **2** | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.3 |
| **8** | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.8 |

**Izvor:** Vlastiti rad autora

Tablica 7.4 prikazuje izdvajanje lokacije u kojoj nedostaju podaci i sortiranje od najniže udaljenosti do najveće udaljenosti. Ovim korakom povećava se preciznost daljnjih koraka zbog sličnosti u udaljenosti koja se odnosi i na potrošnju goriva.

**Tablica 7.5 'Popunjavanje nedostajućih vrijednosti metodom *ffill*'**

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| **0** | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.1 |
| **9** | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | 8.1 |
| **4** | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | 8.1 |
| **2** | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.3 |
| **8** | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.8 |

**Izvor:** Vlastiti rad autora

Tablica 7.5 prikazuje primjenu prve metode zamjene nedostajućih vrijednosti, odnosno primjenu *ffill* metode (eng. 'forward fill') kojom se nedostajuće vrijednosti mijenjaju vrijednošću koja im prethodi u istom stupcu.

U navedenom slučaju potrošnja goriva nedostajućih vrijednosti poprima vrijednost od 8.1 litara jer se ono definira u tom iznosu retka iznad.

Problem spomenute metode je mogući nedostatak prve vrijednosti u skupu iz koje se informacija preuzima u drugu vrijednost. Nakon definiranja okvira podataka[97] i početnog usklađenja redoslijeda vrijednosti, metodi popunjavanja vrijednosti prethodnom pristupa se korištenjem metode Python *pandas* biblioteke:

```
podaci.fillna(method = 'ffill')
```

[97] za potrebe demonstracije funkcije objekt koji sadrži skup podataka nazvan je 'podaci'.

**Tablica 7.6 'Popunjavanje nedostajućih vrijednosti metodom *bfill*'**

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| **0** | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.1 |
| **9** | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | 9.3 |
| **4** | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | 9.3 |
| **2** | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.3 |
| **8** | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.8 |

**Izvor:** Vlastiti rad autora

U Tablici 7.6. prikazano je popunjavanje podataka metodom *bfill* (eng. 'backward fill') sadržane u Python biblioteci *pandas*. Ovom se metodom popunjavaju prazna polja postojećim vrijednostima iza nedostajućih vrijednosti.

Metoda *bfill* nailazi na poteškoće pri popunjavanju predzadnje nedostajuće vrijednosti uz postojanje nedostajuće zadnje vrijednosti u skupu. Nakon definiranja okvira podataka[98] i sortiranja vrijednosti, metodi se pristupa korištenjem metode sadržane u Python *pandas* biblioteci:

```
podaci.fillna(method = 'bfill')
```

**Tablica 7.7 'Popunjavanje nedostajućih vrijednosti metodom interpolacije'**

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| **0** | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.1 |
| **9** | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | 8.5 |
| **4** | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | 8.9 |
| **2** | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.3 |
| **8** | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.8 |

**Izvor:** Vlastiti rad autora

Tablica 7.7 prikazuje primjenu metode interpolacije u popunjavanju nedostajućih podataka, odnosno procjenu vrijednosti u odnosu na trendove kretanja postojećih podataka. Nakon definiranja okvira podataka[99] i sortiranja vrijednosti, interpolacija se nad podacima primjenjuje korištenjem Python *pandas* biblioteke i metode:

```
podaci.interpolate()
```

[98] za potrebe demonstracije funkcije objekt koji sadrži skup podataka nazvan je 'podaci'.

[99] isto.

**Tablica 7.8 'Popunjavanje nedostajućih vrijednosti metodom srednje vrijednosti'**

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| **0** | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.100000 |
| **9** | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | 9.066667 |
| **4** | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | 9.066667 |
| **2** | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.300000 |
| **8** | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.800000 |

**Izvor:** Vlastiti rad autora

Posljednja prikazana metoda popunjavanja nedostajućih vrijednosti odnosi se na primjenu srednje vrijednosti. Ovom se metodom računa srednja vrijednost skupa i automatski se dodjeljuje svim nedostajućim vrijednostima.

Glavna prednost ove metode je neovisnost o vrijednosti iznad i ispod one u promatranom retku. Metoda srednje vrijednosti primjenjiva je uz uvjet prihvatljive standardne devijacije skupa i manjih oscilacija vrijednosti, a njezina se učinkovitost očituje u veličini skupa postojećih podataka zbog veće reprezentativnosti. Po definiranju okvira podataka[100], srednja vrijednost se izračunava i primjenjuje korištenjem metode sadržane u Python *pandas* biblioteci:

```
podaci.fillna(podaci.mean(numeric_only = True))
```

Zaključno, odabir najbolje metode ovisi o strukturi i učestalosti nedostajućih podataka te specifičnim potrebama pojedinog projekta.

### 7.1.5. Vizualizacija podataka u prostoru primjenom linearne algebre

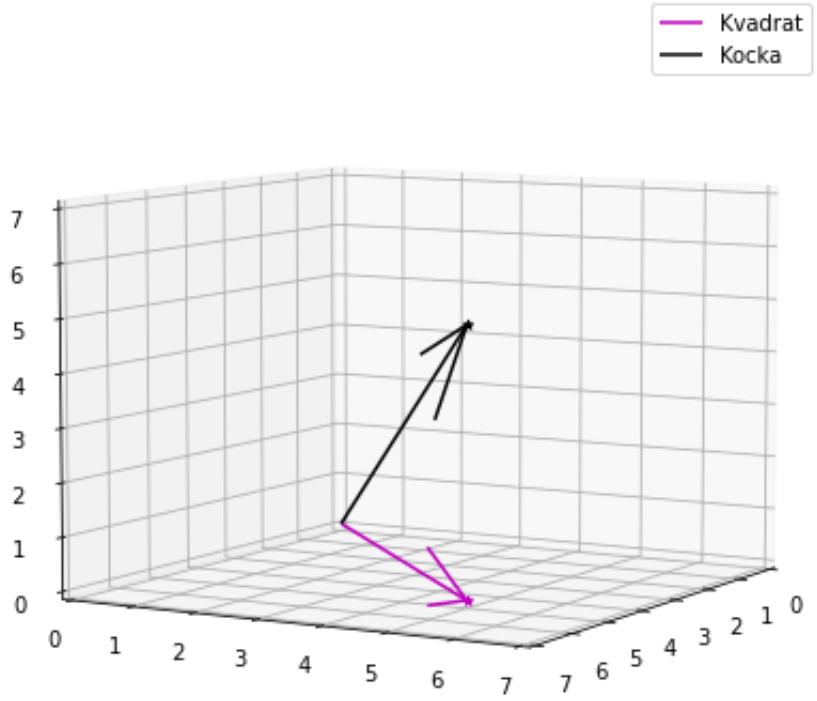
Salton et al. (1975) predlažu model ucrtavanja podataka u prostoru pomoću vektora. Salton et al. (1975) također navode da višedimenzionalnost prostora u kojem se ucrtavaju vektori nadilaze čovjeku poznati svijet u tri dimenzije i da se broj dimenzija vektora mijenja paralelno s količinom postojećih skalara (indeksa) u njemu koji ga opisuju u određenom prostoru. Ucrtavanjem podataka u prostoru pomoću vektora omogućava se indeksiranje računalnim programima, odnosno povezivanje traženog pojma s postojećim pojmom. Ovom se metodom također vizualno prikazuje bliskost vektora u vektorskom prostoru. Gras et al. (2013) uz vizualnu reprezentaciju podataka vektorima spominju i prikaz pomoću površine, odnosno ravnine u prostoru.

[100] za potrebe demonstracije funkcije objekt koji sadrži skup podataka nazvan je 'podaci'.

Pri pohrani i indeksiranju podataka koriste se osnovni elementi linearne algebre poput skalara, vektora (nizova), matrica i tenzora. Skalari su osnovne vrijednosti i sastavni dio ostalih elemenata, vektori (nizovi) prikazuju skup skalara, dok matrice opisuju vektore u više dimenzija, a tenzori matrice u više dimenzija.

Broj dimenzija objekta ima važnu ulogu pri ucrtavanju vektora u vektorskom prostoru, odnosno, ako se radi o podatkovnom paru poput lokacije koja sadrži koordinate osi *x* i osi *y*, koordinata *z* primjenjiva je eventualno u nadzemnom prostoru koji označava i visinu na kojoj se odvija promet. Spomenuta treća dimenzija lokacije, visina, posebno je značajna u upravljanju operacijama zrakoplova prema navođenju kontrole leta u zračnim lukama.

**Grafikon 7.2 'Prikazivanje podataka u vektorskom prostoru'**




**Izvor:** Vlastiti rad autora

Grafikon 7.2 prikazuje ucrtavanje dva objekta[101], kvadrata i kocke, u vektorskom prostoru pri čemu su oni u vektorskom obliku zapisani kao:

$$Kvadrat = \begin{bmatrix} 5 \\ 5 \\ 0 \end{bmatrix}, \qquad Kocka = \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix}$$

Prvi redak kvadrata i kocke prikazuje vrijednost na osi *x*, drugi redak prikazuje vrijednost na osi *y*, a treći redak vrijednost na osi *z*. Na primjeru kvadrata, vrijednost na osi z iznosi 0 zbog dvodimenzionalnosti objekta, a isto vrijedi i za sve naredne dimenzije.

[101] programski kod i detaljna razrada dostupni u Prilozima.

## 7.2. Strojno i duboko učenje

Kelleher i Tierney (2018) strojno učenje opisuju kao 'područje istraživanja koje razvija algoritme koje računala slijede kako bi u podacima pronašla i iz njih izdvojila korisne uzorke'. Spomenuti autori također dijele strojno učenje na nadzirano i nenadzirano, iako Boning (n.d.) uz spomenutu podjelu navodi i djelomično nadzirano učenje.

Strojno učenje i duboko učenje usko su povezani pojmovi, a kod nekih algoritama poput neuronskih mreža, duboko učenje se povezuje s brojem skrivenih slojeva na temelju kojih donose odluke, iako točan broj slojeva u kojem strojno učenje postaje duboko učenje nije strogo definiran, nego podložan subjektivnoj procjeni. (Kelleher i Tierney, 2018 prema Kundih et al., 2022)

### 7.2.1. Nadzirano učenje

Kelleher i Tierney (2018) navode da je nadzirano učenje karakteristično po postojanju ciljnog atributa koji se želi postići. Ono se može opisati kao proces uspoređivanja niza ulaznih vrijednosti s poznatim rješenjima, pri čemu prevladava procjena s najvišom sličnosti s poznatim rješenjem. Primjer nadziranog učenja je treniranje algoritma detekcije objekata s poznatim ciljnim klasifikacijama. Algoritam se može posebno programirati na uočavanje objekata nad kojima je izvršen proces učenja, u kontrastu s nenadziranim učenjem koje može samostalno procijeniti postojanje objekata, no ne ih nužno i klasificirati. Prema Bojaniću (2019) nadziranim učenjem se nastoji pronaći preslikavanje ulaznih vrijednosti koje za rezultat imaju ciljnu vrijednost, matematički izraženom formulom:

$$f(x) = y \quad (7.3)$$

f – funkcija.
x – ulazne vrijednosti.
y – ciljna vrijednost.

Za proces izrade algoritma nadziranog strojnog učenja potrebno je prikupiti mnoštvo različitih verzija iste klase kako bi se izbjegla pristranost pri odlučivanju i šturo poznavanje njezinih odrednica. Raznolikost verzija klase najlakše je opisati kod primjene algoritama namijenjenih za detekciju objekata s obzirom na to da pojedina fotografija objekta može biti fotografirana u različitim vremenskim uvjetima i ambijentu, pod različitim kutovima i osvjetljenju te u različitim bojama i uzorcima. Klasifikacija može biti kategorička ili binarna, odnosno, za rezultat imati odluku da/ne (binarna) ili ime klase (kategorička), što je navedeno u dokumentaciji Python biblioteke *keras* (Kundih et al., 2022).

### 7.2.2. Nenadzirano učenje

Prema Kelleheru i Tierneyu (2018) nenadzirano učenje definira se kao 'oblik strojnog učenja čiji je cilj utvrditi uzorke podataka, uključujući klastere sličnih podatkovnih slogova ili odnose između atributa pri čemu nije definiran ciljni atribut skupa podataka'.

Algoritmima nenadziranog učenja mogu se identificirati oku nevidljive povezanosti između podataka iz čega se donosi zaključak. S obzirom da ciljni atribut skupa podataka nije definiran, algoritmi sami prepoznaju trendove i međuodnos značajki podataka te predlažu rješenje. Rizici koji se vezuju uz djelovanje nenadziranog učenja povezani su s nejasnim ciljem ishoda projekta umjetne inteligencije jer algoritam samostalno pokušava donijeti zaključak na temelju unesenih podataka bez usmjerenja na to što uistinu mora tražiti.

Ghahramani (2003) navodi da algoritmi nenadziranog učenja za cilj imaju reprezentaciju ulaznih vrijednosti na način da se izlazna vrijednost, koja proizlazi iz djelovanja algoritma, može koristiti za donošenje odluka, predviđanje budućih ulaznih vrijednosti i traganje za korelacijama unutar loše strukturiranih podataka.

### 7.2.3. Podjela seta podataka

Prema Tan et al. (2021) gotovo svi, ako ne i svi, suvremeni algoritmi strojnog učenja počivaju na podjeli podataka na uzorak koji se koristi pri treniranju algoritama i uzorak koji se koristi pri testiranju, odnosno validaciji uspješnosti njegova djelovanja.

Treniranje algoritma vrši se nad određenim postotkom dostupnih podataka u kojima se rezultat funkcija algoritma procjenjuje s pravom vrijednosti kroz niz iteracija i postepeno poboljšanje rezultata sa svakom epohom do određene granice gdje ostvaruje svoj maksimum.

Testiranjem algoritma strojnog učenja i validacijom ispravnosti donošenja odluka procjenjuje se njegova efektivnost u postotku koji se izračunava stavljanjem u odnos broja ispravnih procjena i veličine uzorka nad kojim je testiranje izvršeno.

Brojnim istraživanjima se pokušava dokazati postojanje optimalnog omjera podataka koji se koristi pri treniranju, odnosno testiranju učinkovitosti djelovanja algoritama. Problem podjele podataka u dva seta posebno je izražen kod postojanja inicijalnog seta s manjom količinom podataka. Podaci koji se koriste u procesu treniranja i testiranja podataka moraju biti različiti kako bi se izbjegla pristranost. Validacija uspješnosti algoritma vrši se nad neviđenim podacima, odnosno podacima koji nisu korišteni u procesu treniranja algoritma. Primjeri omjera podjele podataka u dva seta navode se u istraživanju Polat et al. (2008) i prikazani su kao: 50-50%, 70-30% u korist podataka za treniranje i 80-20% u korist podataka za treniranje.

**Tablica 7.9 'Prikaz tablice s podacima o korisnicima javnog prijevoza'**

| | Mod | Spol | Dob | Udaljenost do posla (m) | Vozačka dozvola | Automobili u kućanstvu |
|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 32 | 1500 | 1 | 2 |
| **1** | 1 | 0 | 26 | 3500 | 1 | 1 |
| **2** | 1 | 1 | 29 | 18500 | 1 | 2 |
| **3** | 0 | 0 | 35 | 100 | 1 | 1 |
| **4** | 1 | 1 | 44 | 5000 | 0 | 1 |
| **5** | 1 | 1 | 43 | 500 | 1 | 1 |
| **6** | 0 | 0 | 30 | 8500 | 0 | 1 |
| **7** | 0 | 0 | 32 | 7500 | 1 | 2 |
| **8** | 0 | 1 | 50 | 9500 | 1 | 4 |
| **9** | 1 | 1 | 38 | 2500 | 1 | 1 |
| **10** | 0 | 1 | 28 | 500 | 1 | 0 |
| **11** | 0 | 0 | 56 | 1500 | 0 | 1 |
| **12** | 1 | 0 | 44 | 1500 | 0 | 0 |
| **13** | 0 | 1 | 59 | 100 | 1 | 2 |
| **14** | 1 | 1 | 54 | 2500 | 1 | 1 |
| **15** | 0 | 1 | 31 | 1500 | 1 | 1 |
| **16** | 1 | 1 | 34 | 11500 | 1 | 0 |
| **17** | 1 | 0 | 65 | 1500 | 1 | 1 |
| **18** | 0 | 0 | 26 | 2500 | 1 | 0 |
| **19** | 1 | 0 | 24 | 3000 | 0 | 0 |
| **20** | 1 | 0 | 56 | 1500 | 1 | 1 |

**Izvor:** Vlastiti rad autora

Tablica 7.9 prikazuje primjer podataka o korisnicima usluge javnog prijevoza[102] pri čemu su vrijednosti iskazane u numeričkom obliku za potrebe treniranja i testiranja, s obzirom na to da se predviđanje izlazne vrijednosti vrši korištenjem funkcija. Kao i ostale značajke sadržane u datoteci, korisnici usluga javnog prijevoza iskazani su numerički (0 = nisu korisnici usluge, 1 = korisnici su usluge) što je izvršeno funkcijom *replace* Python *pandas* biblioteke[103]:

```
podaci['Mod'].replace('Ne', 0, inplace = True) # nisu korisnici usluge.
podaci['Mod'].replace('Da, 1, inplace = True) # jesu korisnici usluge.
```

[102] programski kod i detaljna razrada dostupni u Prilozima.

[103] za potrebe demonstracije funkcije objekt koji sadrži skup podataka nazvan je 'podaci'.

Podjela podataka na setove namijenjene testiranju i treniranju izvršena je metodom *train_test_split* Python modula *model_selection*, *sklearn* biblioteke, a ona nasumično odabire elemente skupova i sortira ih prema željenom omjeru nad skupovima *x* i *y*, pri čemu skup *x* ne uključuje traženu vrijednost (ciljnu klasu), a skup *y* se sastoji samo od ciljne klase:

```
x_treniranje, x_testiranje, y_treniranje, y_testiranje =
sklearn.model_selection.train_test_split(x, y, test_size = 0.2)
```

x_treniranje

| | Spol | Dob | Udaljenost do posla (m) | Vozačka dozvola | Automobili u kućanstvu |
|---|---|---|---|---|---|
| **0** | 1 | 32 | 1500 | 1 | 2 |
| **14** | 1 | 54 | 2500 | 1 | 1 |
| **13** | 1 | 59 | 100 | 1 | 2 |
| **2** | 1 | 29 | 18500 | 1 | 2 |
| **4** | 1 | 44 | 5000 | 0 | 1 |
| **16** | 1 | 34 | 11500 | 1 | 0 |
| **8** | 1 | 50 | 9500 | 1 | 4 |
| **3** | 0 | 35 | 100 | 1 | 1 |
| **19** | 0 | 24 | 3000 | 0 | 0 |
| **5** | 1 | 43 | 500 | 1 | 1 |
| **17** | 0 | 65 | 1500 | 1 | 1 |
| **1** | 0 | 26 | 3500 | 1 | 1 |
| **10** | 1 | 28 | 500 | 1 | 0 |
| **20** | 0 | 56 | 1500 | 1 | 1 |
| **7** | 0 | 32 | 7500 | 1 | 2 |
| **18** | 0 | 26 | 2500 | 1 | 0 |

y_treniranje

| | Mod |
|---|---|
| **0** | 0 |
| **14** | 1 |
| **13** | 0 |
| **2** | 1 |
| **4** | 1 |
| **16** | 1 |
| **8** | 0 |
| **3** | 0 |
| **19** | 1 |
| **5** | 1 |
| **17** | 1 |
| **1** | 1 |
| **10** | 0 |
| **20** | 1 |
| **7** | 0 |
| **18** | 0 |

*Slika 7.5 'Prikaz skupa podataka korištenog pri treniranju algoritma',*
*Izvor: Vlastiti rad autora*

Slika 7.5 prikazuje set podataka koji se koriste u fazi treniranja algoritma strojnog učenja nakon primijenjene metode nad inicijalnim setom podataka i podjelom u dva dijela prema 80-20 omjeru. Tablica *x_treniranje* prikazuje set podataka iz kojih se pronalazi njihova međuovisnost. Tablica *y_treniranje* prikazuje stvarni prometni mod koji korisnici koriste u putovanju do radnog mjesta. Preciznost procjene modela vrši se iterativnim procesom testiranja rezultata funkcija testnog modela u usporedbi sa stvarnim podacima. Uspješnost modela ovisi o postavljenim parametrima poput veličine uzorka i korisnosti odabranih značajki, ali i o samom setu podataka.

Slika 7.6 prikazuje primjer treniranja i testiranja u 300 epoha, koristeći pritom osam nasumično odabranih elemenata u setu. Iz slike je vidljivo da preciznost algoritma strojnog učenja svakom novom epohom raste, no potrebno je odrediti optimalan broj epoha kako bi se izbjegla pristranost algoritma strojnog učenja pri odlučivanju, posebice nad neviđenim podacima.

```
Epoch 38/300
8/8 [==============================] - 0s 2ms/step - loss: 0.2526 - accuracy: 0.8988
Epoch 39/300
8/8 [==============================] - 0s 2ms/step - loss: 0.2426 - accuracy: 0.8947
Epoch 40/300
8/8 [==============================] - 0s 4ms/step - loss: 0.2413 - accuracy: 0.9028
Epoch 41/300
8/8 [==============================] - 0s 2ms/step - loss: 0.2278 - accuracy: 0.9069
Epoch 42/300
8/8 [==============================] - 0s 2ms/step - loss: 0.2290 - accuracy: 0.9109
```

Prijevod: epoch – epoha, step – korak, loss – gubitak, accuracy – preciznost.

*Slika 7.6 'Prikaz postupka treniranja i testiranja algoritma',*
*Izvor: Vlastiti rad autora – snimak zaslona procesa tensorflow.keras.Sequential modela*

Drugi dio seta podataka izdvojenih funkcijom koristi se za validaciju uspješnosti algoritma strojnog učenja nad neviđenim podacima. Korištenje podataka koji ni na koji način nisu upotrebljavani u procesu treniranja i testiranja nužno je zbog pripreme algoritma strojnog učenja na djelovanje u realnom svijetu i adaptaciju procesa odlučivanja u neizvjesnim situacijama.

Set podataka za validaciju prikazan je na slici 7.7, pri čemu tablica *x_testiranje* prikazuje primjer značajki neke osobe, dok tablica *y_testiranje* prikazuje realno stanje korištenja moda prijevoza do radnog mjesta. U setu za validaciju neophodno je korištenje raznolikih podataka zbog analiziranja fleksibilnosti algoritma strojnog učenja i moguće pristranosti. Pristranost algoritma strojnog učenja može se identificirati i u slučajevima velike neproporcionalnosti u količini podataka između ciljnih klasa koje se koriste u svim fazama procesa osposobljavanja algoritma strojnog učenja na donošenje odluka, od treniranja i testiranja do same validacije.

x_testiranje

| | Spol | Dob | Udaljenost do posla (m) | Vozačka dozvola | Automobili u kućanstvu |
|---|---|---|---|---|---|
| **11** | 0 | 56 | 1500 | 0 | 1 |
| **6** | 0 | 30 | 8500 | 0 | 1 |
| **15** | 1 | 31 | 1500 | 1 | 1 |
| **12** | 0 | 44 | 1500 | 0 | 0 |
| **9** | 1 | 38 | 2500 | 1 | 1 |

y_testiranje

| | Mod |
|---|---|
| **11** | 0 |
| **6** | 0 |
| **15** | 0 |
| **12** | 1 |
| **9** | 1 |

*Slika 7.7 'Prikaz skupa podataka korištenog pri validaciji uspješnosti algoritma',*
*Izvor: Vlastiti rad autora*

## 7.3. Primjena algoritama u prometnim i logističkim sustavima

U ovom potpoglavlju opisuje se primjena algoritama te algoritama strojnog učenja u optimizaciji prometnih i logističkih procesa. Prikazane su metode grupiranja vrijednosti, pronalaska optimalne putanje, simuliranja događaja i procjene vrijednosti u odnosu na parametre.

### 7.3.1. Grupiranje lokacija primjenom algoritma K-srednjih vrijednosti

Poduzeće L bavi se logističkim djelatnostima. Tablica 7.10 prikazuje popis prvih 10 od 300 lokacija na koje je potrebno dostaviti robu.

**Tablica 7.10 'Prikaz koordinata za dostavu'**

| | Geo_širina | Geo_dužina |
|---|---|---|
| **0** | 15.918550 | 45.771235 |
| **1** | 16.009798 | 45.808749 |
| **2** | 16.050008 | 45.809756 |
| **3** | 16.089429 | 45.815235 |
| **4** | 16.042004 | 45.757380 |
| **5** | 15.999879 | 45.853556 |
| **6** | 16.057970 | 45.768824 |
| **7** | 15.981983 | 45.827724 |
| **8** | 16.022059 | 45.835960 |
| **9** | 15.891357 | 45.804350 |

**Izvor:** Vlastiti rad autora

Metodologija provedbe grupiranja bazirana je na *Jupyter Notebook* predlošku *GitHub* korisnika *dhavalsays (2019).* Za potrebe daljnjeg procesuiranja podataka[104] koristi se vlastita biblioteka izrađena za potrebe diplomskog rada i opće namjene, imena *kundih.* Biblioteka omogućava prikaz koda na hrvatskom jeziku, a pritom koristi elemente ostalih biblioteka i modula:

- pandas,
- numpy,
- sklearn,
- i matplotlib.pyplot.

[104] programski kod i detaljna razrada dostupni u Prilozima.

Za optimalno grupiranje podataka koristi se algoritam K-srednjih vrijednosti[105] koji grupira lokacije oko predloženih točaka.

Gabrovec (2020) algoritam K-srednjih vrijednosti opisuje kao 'model strojnog učenja bez nadzora, koji se prvenstveno koristi pri klasifikaciji i regresiji danih podataka', a također navodi da se u praksi najčešće koristi Lloydov algoritam koji razmješta *m* točaka u *k* skupina.

Nakon unosa datoteke i postavljanja okvira podataka[106] unesena je biblioteka u radno sučelje i pokrenuta inicijalizacija objekta nad varijablom KSV s primjerom podjele podataka na 5 klastera, a ono se vrši programskim kodom:

```
import kundih
KSV = kundih.KSrednjeVrijednosti(datoteka = podaci, broj_klastera = 5,
koordinate = ['Geo_širina', 'Geo_dužina'])
```

Bradley i Fayyad (1998) navode da je nužno uskladiti (eng. 'fitting') podatke kako bi se odgovorilo na njihovo moguće iskrivljenje.

**Tablica 7.11 'Prikaz koordinata za dostavu nakon usklađivanja'**

| | Geo_širina | Geo_dužina |
|---|---|---|
| **0** | 0.262590 | 0.194583 |
| **1** | 0.581372 | 0.538464 |
| **2** | 0.721845 | 0.547695 |
| **3** | 0.859564 | 0.597920 |
| **4** | 0.693885 | 0.067576 |
| **5** | 0.546718 | 0.949210 |
| **6** | 0.749663 | 0.172480 |
| **7** | 0.484198 | 0.712413 |
| **8** | 0.624204 | 0.787905 |
| **9** | 0.167589 | 0.498145 |

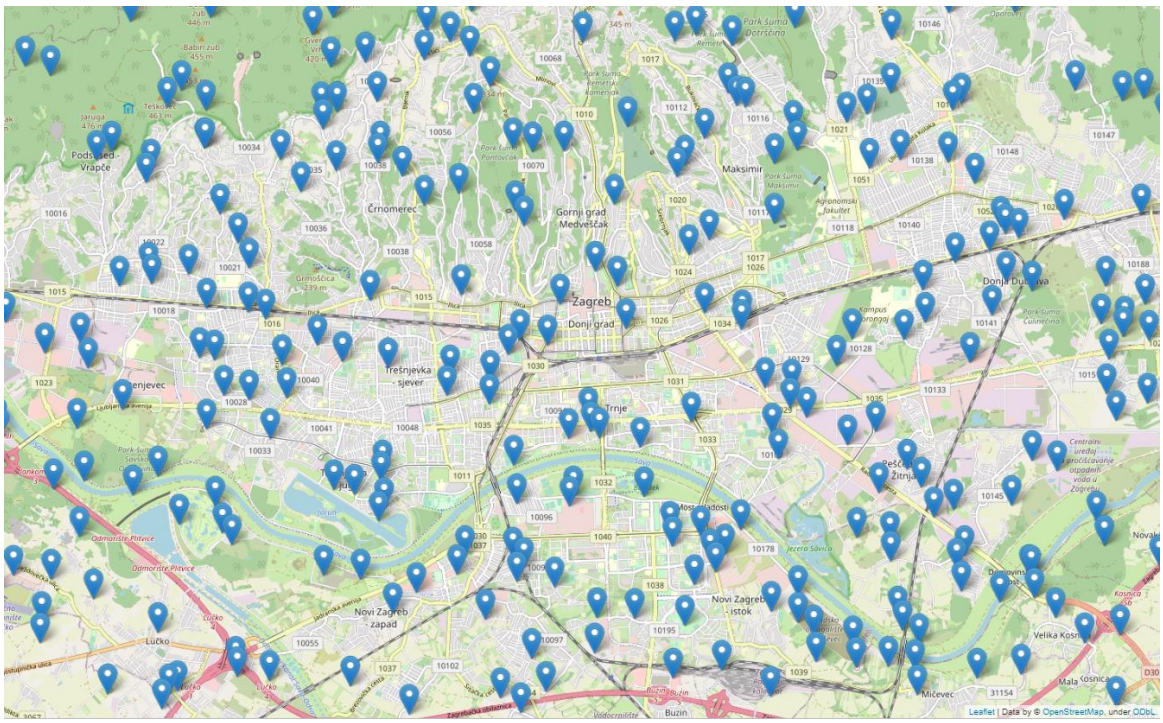
**Izvor:** Vlastiti rad autora

Tablica 7.11 prikazuje popis prvih 10 vrijednosti nakon primjene metode objekta koja redefinira inicijalne vrijednosti uz eliminaciju iskrivljenja:

```
KSV.podesi_omjer()
```

[105] eng. 'K-means clustering.'

[106] za potrebe demonstracije funkcije objekt koji sadrži skup podataka nazvan je 'podaci'.




*Slika 7.8 'Primjer prikaza lokacija za dostavu',*
*Izvor: Vlastiti rad autora na predlošku OpenStreetMap uz primjenu biblioteka folium i leaflet*

Slika 7.8 prikazuje unos svih postojećih lokacija dostave Tablice 7.10 na kartu korištenjem Python biblioteke *folium*, uz podršku *leafleta* i *OpenStreetMapa.*

Postupku grupiranja lokacija prema inicijalno zadanom broju klastera nadalje se pristupa korištenjem metode koja izvodi matematičke operacije algoritma:

```
KSV.kreiraj_klastere()
```

Nakon kreiranja klastera, centroidnim vrijednostima pristupa se korištenjem metode:

```
KSV.prikaz_centroida()
```

Centroidi algoritma K-srednjih vrijednosti prikazuju točke kojima gravitiraju okolne točke iz definiranog skupa vrijednosti, odnosno u kontekstu logističkog poduzeća L, lokacije na koje se vrši dostava paketa.

```
array([[0.2717221 , 0.78835594],
       [0.81420603, 0.2015424 ],
       [0.4844006 , 0.26480131],
       [0.75159483, 0.72179618],
       [0.15539349, 0.29367079]])
```

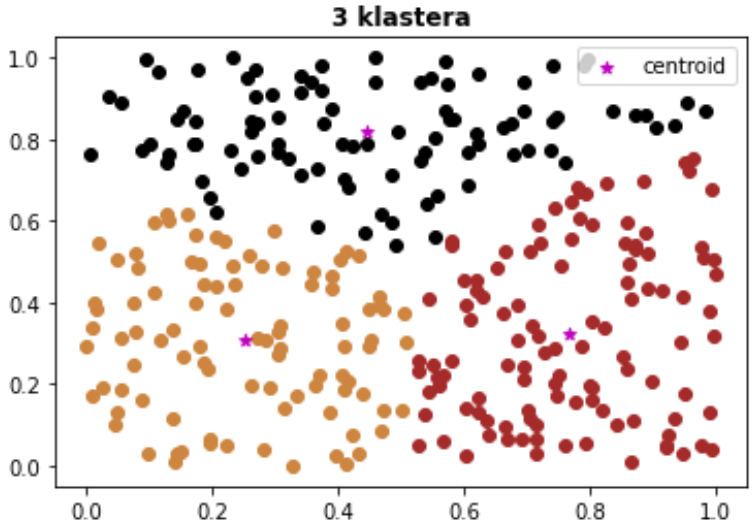
Prijevod: array – niz.

*Slika 7.9 'Ispis lokacija centroidnih točaka nad usklađenim podacima u 5 klastera',*
*Izvor: Vlastiti rad autora – snimak zaslona funkcije KSV.prikaz_centroida()*

Prikazu grafa podjele vrijednosti na n-broj klastera pristupa se metodom objekta:

```
KSV.prikaz_grafa()
```

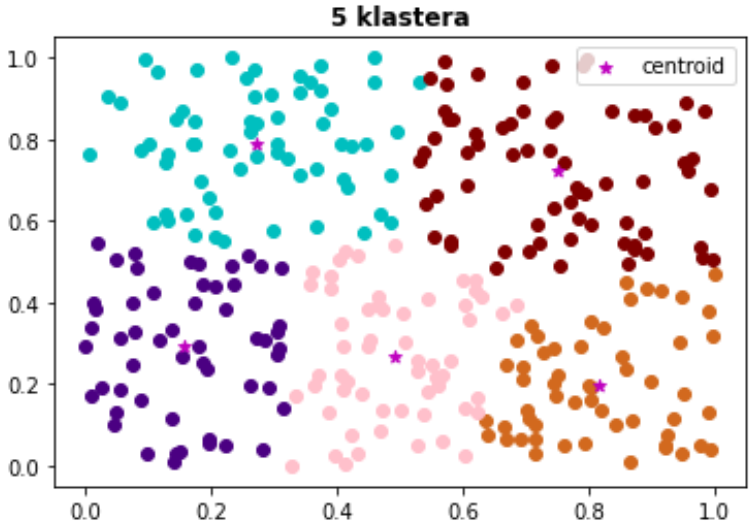
**Grafikon 7.3 'Podjela lokacija dostave na 3 klastera'**




**Izvor:** Vlastiti rad autora

Grafikon 7.3 prikazuje podjelu lokacija na 3 dijela, a centroidne točke (n = 3) kojima gravitiraju prikazane su kao zvjezdice. U kontekstu logističkog poduzeća L, ovom se podjelom radni opseg dijeli na 3 dostavljača (dostavna vozila).

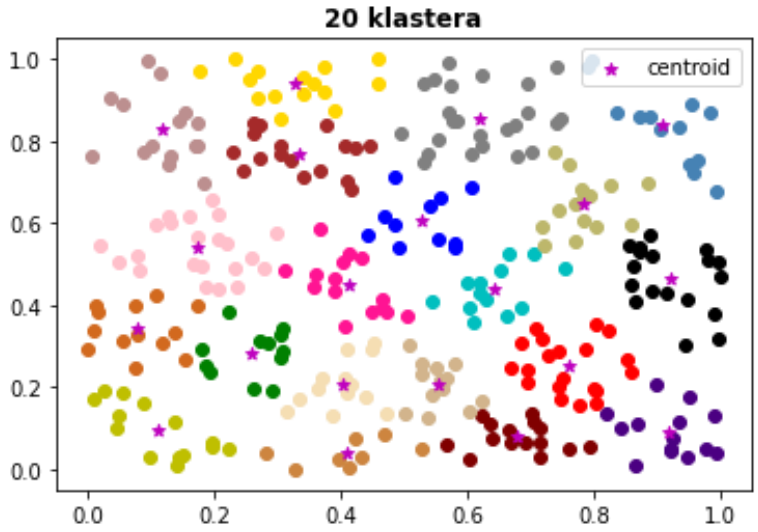
**Grafikon 7.4 'Podjela lokacija dostave na 5 klastera'**




**Izvor:** Vlastiti rad autora

Grafikon 7.4 prikazuje podjelu lokacija na 5 dijelova koji gravitiraju centroidnim točkama (n=5). Poduzeće L na ovaj način dijeli radni opseg na 5 dostavljača (dostavnih vozila).

**Grafikon 7.5 'Podjela lokacija dostave na 20 klastera'**




**Izvor:** Vlastiti rad autora

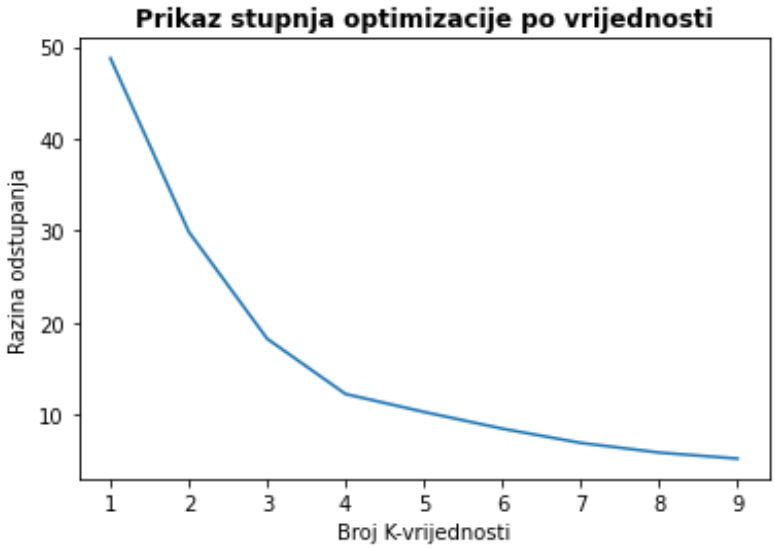
Grafikon 7.5 pokazuje podjelu lokacija na 20 klastera koji reprezentiraju radni opseg 20 dostavljača (dostavnih vozila) poduzeća L.

Prikazu stupnja optimizacije prema broju klastera pristupa se metodom objekta:

```
KSV.prikaz_grafa()
```

Grafikon 7.6 se prema Patelu (2019) naziva i *grafom lakta*, a prikazuje zbroj kvadriranih odstupanja u odnosu na dodavanje novih centroidnih točaka. Iz prikaza je vidljivo da razina optimizacije udaljenosti svakom novom točkom doprinosi manje nego li onom prethodnom.

**Grafikon 7.6 'Prikaz stupnja optimizacije skupa vrijednosti dodavanjem točaka'**

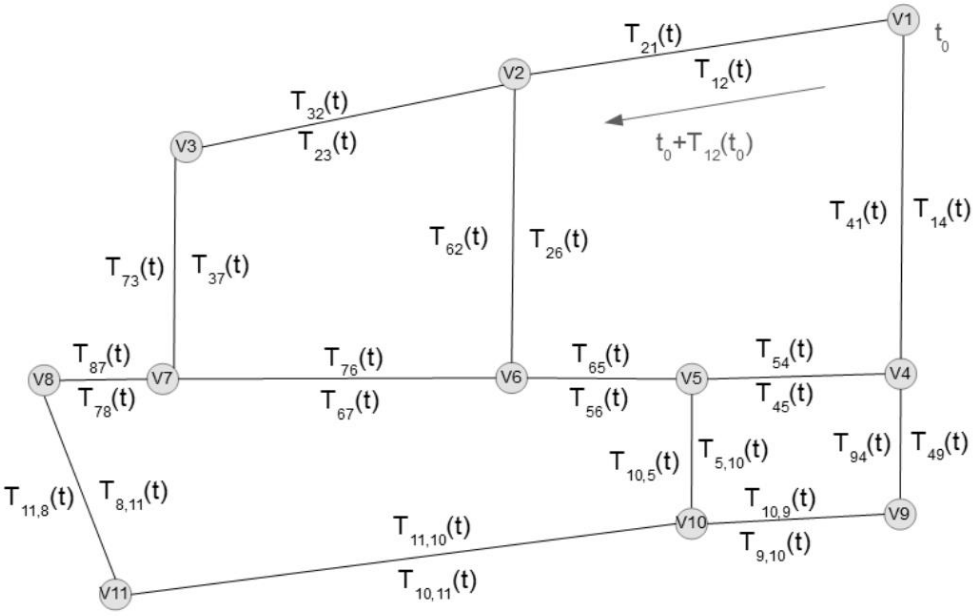



**Izvor:** Vlastiti rad autora

### 7.3.2. Definiranje optimalne putanje Dijkstra algoritmom

Dijkstra algoritam je klasični algoritam pronalaska najkraće putanje, uvrštvan u mnogim područjima znanosti, a općenito se koristi za usporedbu udaljenosti početnog čvora do svih ostalih čvorova u okolini. Princip rada algoritma prvi je predložio E.W. Dijkstra u 1959. godini.

(Dijkstra, 1957 prema Rosyidi et al., 2014)




*Slika 7.10 'Dijkstra algoritam u usporedbi linija prometa',*
*Izvor: Rosyidi, L., Pradityo, H.P., Gunawan, D. i Sari, R.F., (2014) : 'Timebase dynamic weight for Dijkstra Algorithm implementation in route planning software', International conference on intelligent green building and smart grid (IGBSG) (pp. 1-4). IEEE.*

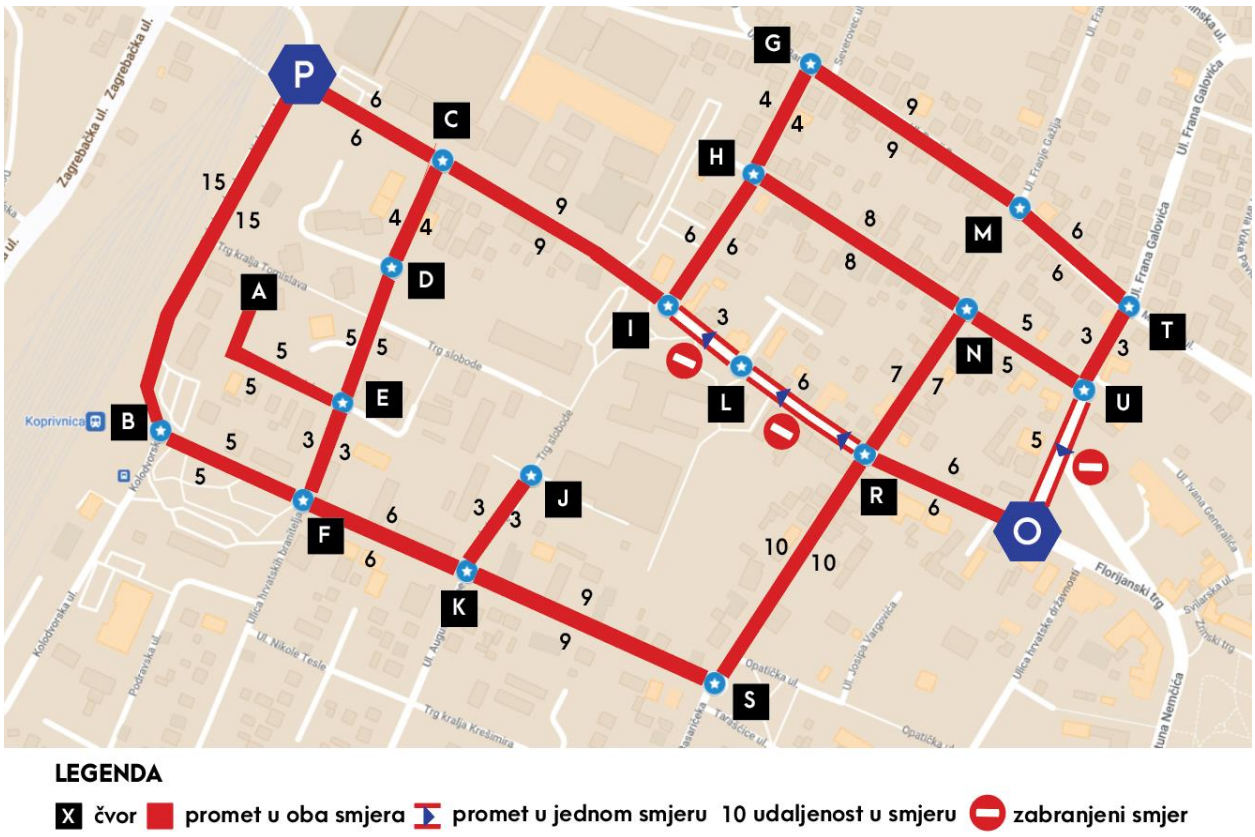
Slika 7.10 prikazuje pregled čvorova i pripadajućih putanja koje ih povezuju. Putanje koje ih povezuju nose određenu težinu, a ona može biti izražena u stvarnoj udaljenosti ili kompleksnosti prolaska. Iako određena prometna putanja može biti fizički najkraća, potrebno je sagledati i stupanj njezine zagušenosti ostalim subjektima na istoj putanji koji mogu ometati protok ili putovanje.

Dijkstra algoritam jedan je od brojnih algoritama koji se koriste u pronalasku optimalne putanje, a uz njega postoje još i Floyd-Warshall algoritam, Bellman-Ford algoritam i genetski algoritam (Magzhan i Jani, 2013).

Candra et al. (2021) navode da na temelju Dijkstra algoritma nastaje i A*[107] algoritam.

Dijkstra algoritam koristi se u pronalasku optimalne putanje u putničkom prometu, ali i telekomunikacijama pronalaskom optimalnog odašiljača signala. (Benaicha i Taibi, 2013)

[107] naziva se i engleskim nazivom 'A-star', odnosno hrv. 'A-zvijezda.'




*Slika 7.11 'Prikaz čvorova i udaljenosti između lokacije na primjeru kvarta grada Koprivnice', Izvor: Vlastiti rad autora nad kartom Google My Maps*

Slika 7.11 prikazuje putanje i čvorove kvarta u gradu Koprivnici. Zadatak poduzeća B je prijevoz putnika od čvora P (polazište) do čvora O (odredište) uvažavajući pritom prometne propise, a čvorovi i pripadajuće putanje za potrebe daljnje obrade definiraju se Python rječnikom:

```
čvorovi = {
      'p':{'c': 6, 'b': 15},
      'b':{'f': 5, 'p': 15},
      'c':{'p': 6, 'd': 4, 'i': 9},
      'd':{'c': 4, 'e': 5},
      'e':{'d': 5, 'a': 5, 'f': 3},
      'f':{'e': 3, 'b': 5, 'k': 6},
      'g':{'h': 4, 'm': 9},
      'h':{'g': 4, 'n': 8, 'i': 6},
      'i':{'c': 9, 'l': np.inf, 'h': 6},
      'j':{'k': 3},
      'k':{'j': 3, 'f': 6, 's': 9},
      'l':{'i': 3, 'r': np.inf},
      'm':{'g': 9, 't': 6},
      'n':{'h': 8, 'u': 5, 'r': 7},
      'o':{'r': 6, 'u': np.inf},
      'a':{'e': 5},
      'r':{'o': 6, 's': 10, 'l': 6, 'n': 7},
      's':{'k': 9, 'r': 10},
      't':{'u': 3, 'm': 6},
       'u':{'o': 5, 'n': 5, 't': 3},
       }
```

Za unos beskonačnih vrijednosti u putanje rječnika korištena je biblioteka *numpy* (unesena kao *np*) i varijabla *inf*[108], a za pokretanje Dijkstra algoritma korištena je vlastita biblioteka *kundih* primjenom programskog koda[109]:

```
import kundih
import numpy as np
```

Unos beskonačne varijable koristi se zbog postojanja prometa u jednom smjeru, što podrazumijeva zabranu prometa u suprotnom smjeru. Dijkstra algoritam koristi udaljenosti za potrebe definiranja optimalne putanje i njezine udaljenosti, a na ovaj se način eliminira mogućnost odabira putanje koja se suprotstavlja prometnim propisima.

Programski kod Dijkstra algoritma u vlastitoj biblioteci *kundih* djelomično je inspiriran metodologijom i logikom algoritma koju navodi Dey (2019).

Nakon unosa potrebnih biblioteka, kreira se varijabla nad kojom se inicijalizira objekt Dijkstra biblioteke *kundih* uz potrebne parametre:

```
Dijkstra = kundih.Dijkstra(čvor = čvorovi, polazište = 'p', odredište
= 'o')
```

Argumentu objekta *čvor* pridodaje se rječnik s definiranim čvorovima i putanjama. Argument *polazište* određuje polaznu točku putovanja, dok argument *odredište* predstavlja ciljnu točku do koje se putovanje provodi.

Nakon inicijalizacije objekta s pripadajućim argumetnima, za izračun optimalne rute (putanje) koristi se ugrađena metoda objekta:

```
Dijkstra.optimalna_ruta()
```

Metodom *optimalna_ruta()* pokreće se Dijkstra algoritam i ispisuje se tablični prikaz (Tablica 7.12), pružajući informacije o optimalnoj putanji (ako ona postoji i ako je moguće ostvariti putovanje između čvorova), te o minimalnoj udaljenosti između polazišnog i odredišnog čvora.

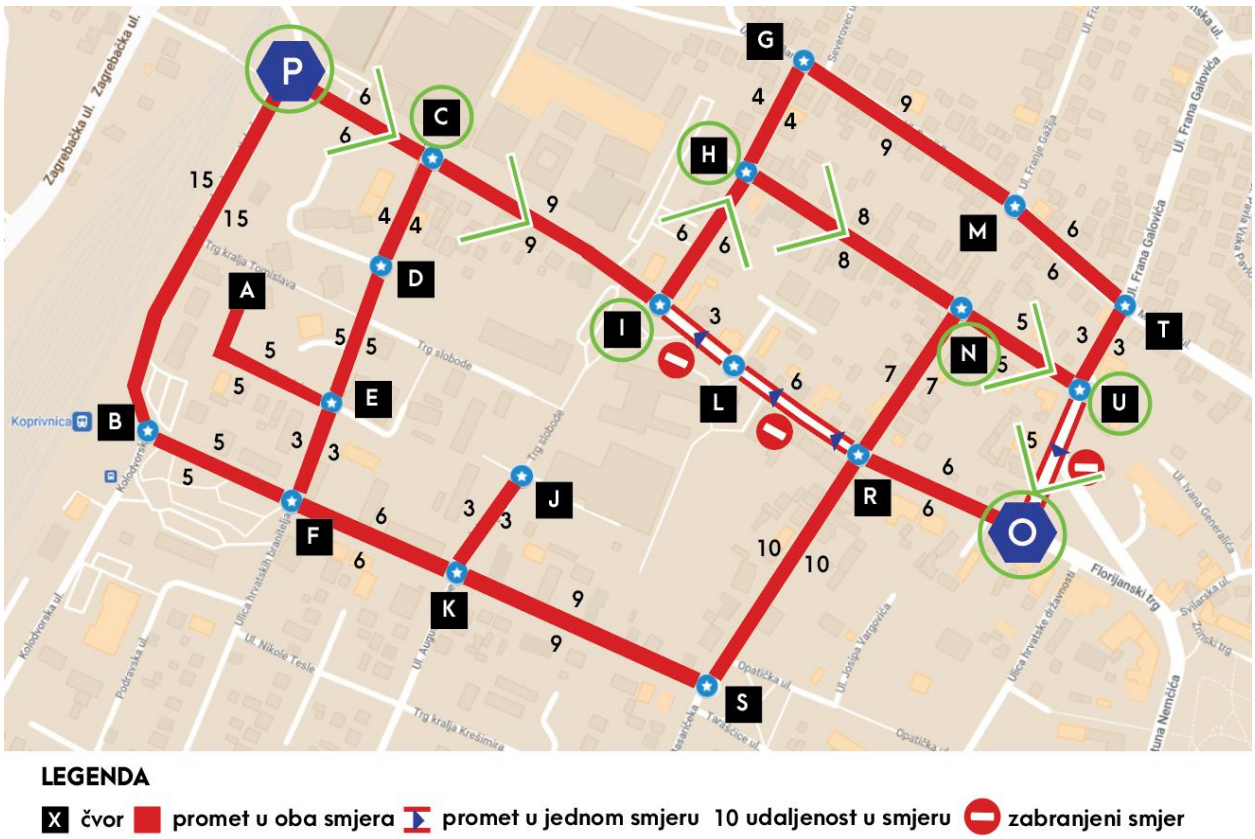
**Tablica 7.12 'Prikaz optimalne putanje izračunate Dijkstra algoritmom'**

| | Izračun putanje |
|---|---|
| **Optimalna putanja** | [p, c, i, h, n, u, o] |
| **Minimalna udaljenost** | 39 |

**Izvor:** Vlastiti rad autora – snimak zaslona metode Dijkstra.optimalna_ruta()

[108] eng. 'infinity' – hrv. beskonačnost.

[109] programski kod i detaljna razrada dostupni u Prilozima.




*Slika 7.12 'Prikaz optimalne putanje izračunate Dijkstra algoritmom nad kartom', Izvor: Vlastiti rad autora nad kartom Google My Maps*

Slika 7.12 prikazuje primjenu izračunate optimalne rute (putanje) nad inicijalno prikazanom kartom poduzeća B koje prevozi putnike od čvora polazište (P) do čvora odredište (O).

### 7.3.3. Izračun cijene transporta robe cestovnim putem u odnosu na udaljenost primjenom algoritma linearne regresije

Algoritam linearne regresije može se koristiti u slučajevima povezanog odnosa podataka koji se lociraju u koordinatnom sustavu pomoću osi *x* i osi *y*. Za podatke među kojima vlada nelinearan odnos koriste se složeniji algoritmi poput umjetnih neuronskih mreža.

Kelleher i Tierney (2018) regresijsku funkciju opisuju kao 'stroj koji ulazne vrijednosti atributa pretvara u izlaznu vrijednost, a same parametre kao postavke koje kontroliraju ponašanje stroja'. Postupak funkcioniranja algoritma linearne regresije opisuje se kao inicijalno pogađanje vrijednosti uz primjenu opetovanog ažuriranja vrijednosti čime se nastoji postići optimalnu funkciju koja opisuje međuodnose između podataka (Kelleher i Tierney, 2018).
Primjena algoritma linearne regresije prepoznaje se u kalkulatorima koji nastoje predložiti izlaznu vrijednost na temelju ulazne vrijednosti, a preduvjet za to leži u što većoj količini podataka.

Na primjeru logističkog poduzeća L, algoritam procjenjuje ukupnu cijenu transporta robe korištenjem cestovne infrastrukture prema broju unesenih kilometara, analizom cijena konkurentskih poduzeća koje obavljaju istu djelatnost.

Razrađeni simulacijski model procjene u obzir uzima prosječnu cijenu od 1.57€ (11.79 HRK) po kilometru transporta[110], koju navodi Mahnken (2021).

Za potrebe izrade modela korišten je objekt *LinearnaRegresija* iz vlastite biblioteke *kundih* s pripadajućim metodama, a za kreiranje okvira podataka korištena je biblioteka *pandas*[111].

Varijabla kojoj je pridodan okvir podataka nosi naziv *podaci*, podaci o postojećim udaljenostima pohranjeni su u stupcu naziva *Udaljenost (km)*, a cijene transporta konkurentskih poduzeća prikazane su stupcem naziva *Cijena transporta (HRK)*.

Programski kod algoritma linearne regresije u vlastitoj biblioteci *kundih* djelomično je inspiriran metodologijom i logikom algoritma koju navodi Fowers (2019).

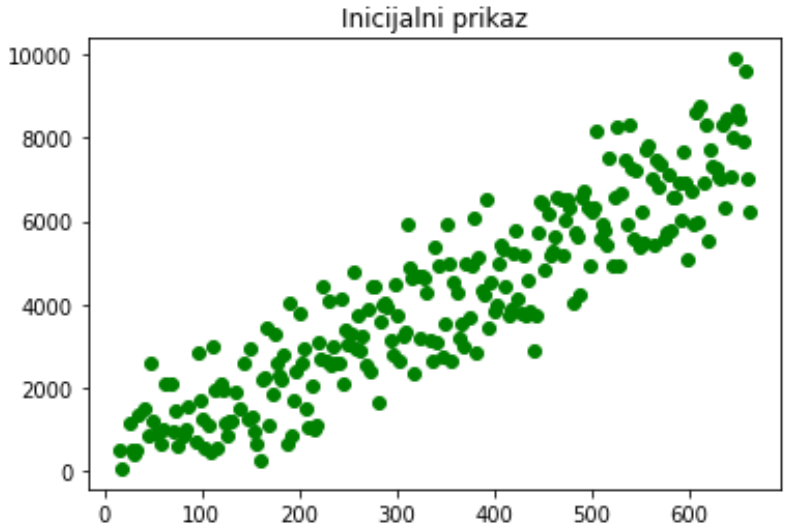
Unos i inicijalizacija objekta nad varijablom LR izvršen je programskim kodom:

```
LR = kundih.LinearnaRegresija(podaci, 'Udaljenost (km)', 'Cijena
transporta (HRK)')
```

Nakon inicijalizacije objekta nad varijablom LR, pristupa se izradi prikaza odnosa vrijednosti (Grafikon 7.7) korištenjem metode:

```
LR.prikaz_grafa()
```

**Grafikon 7.7 'Prikaz inicijalnog odnosa vrijednosti'**




**Izvor:** Vlastiti rad autora

[110] prema tečaju 1 EUR = 7.51 HRK.

[111] programski kod i detaljna razrada dostupni u Prilozima.

Podjela seta na onaj namijenjen za treniranje te onaj namijenjen na testiranje vrši se funkcijom:

```
LR.podjela_seta(omjer_testa = 0.20, slučajno_stanje = 40)
```

Argument *omjer_testa* označava postotak vrijednosti koje se izdvajaju u set za testiranje, dok se ostatak vrijednosti koristi za treniranje algoritma. Argument *slučajno_stanje* omogućava zamrzavanje točaka na istoj razini, s obzirom da postupak treniranja i testiranja može polučiti različite rezultate pri ponovnom pokretanju metode.[112]

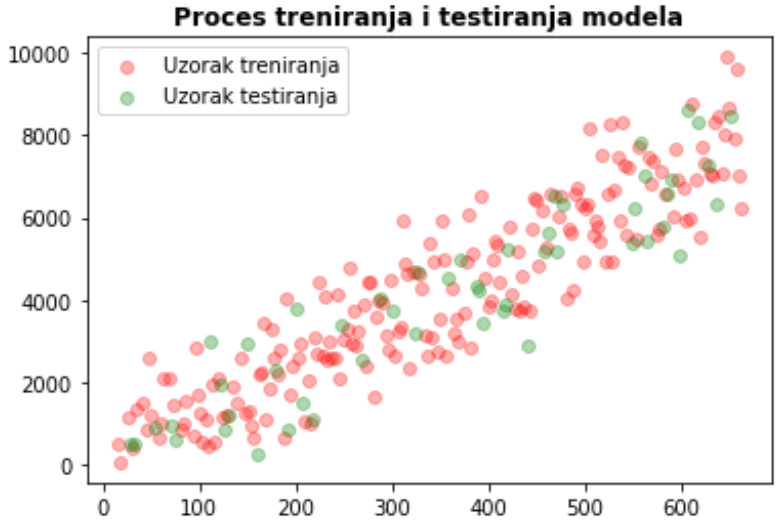
Grafički prikaz elemenata koji se koriste u procesu treniranja i testiranja algoritma dobiva se korištenjem metode:

```
LR.prikaz_procesa()
```

Postupak usklađivanja podataka (eng 'fitting') ugrađen je u funkciju te se pokreće u pozadini prije izvršavanja same funkcije.

Rezultat izvršavanja navedene metode prikazan je na Grafikonu 7.8. Ucrtane crvene točke prikazuju uzorak koji se uzima pri treniranju modela (80%) i uzorak koji se uzima za testiranje modela (20%). Ovim korakom omogućava se usporedba predloženog heurističkog modela procjene funkcije linearne regresije sa stvarnim vrijednostima skupa podataka nakon čega se mjeri njegova uspješnost i vrši automatsko usklađivanje. Kelleher i Tierney (2018) navode da se 'pogreška funkcije za svaki pojedinačni podatak računa kao razlika procijenjene vrijednosti atributa cilja i njezine stvarne vrijednosti.'

**Grafikon 7.8 'Proces treniranja i testiranja modela'**




**Izvor:** Vlastiti rad autora

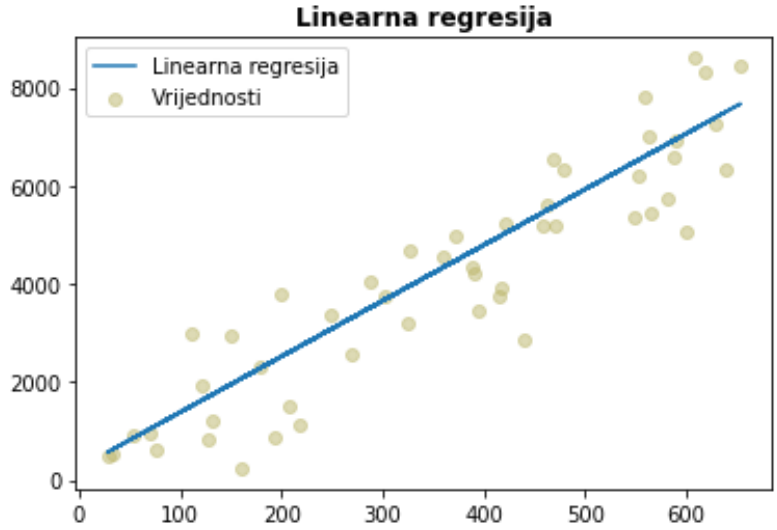
[112] detaljna razrada podjele seta podataka nalazi se u poglavlju 7.2.3

Grafičkom prikazu izračunate linearne regresije pristupa se metodom:

```
LR.prikaz_linearne_regresije()
```

Grafikon 7.9 prikazuje izlaznu vrijednost spomenute metode, pri čemu je linearna regresija prikazana pravcem plave boje, a vrijednosti na temelju kojih je uspostavljen trend linije prikazane su crvenom bojom.

**Grafikon 7.9 'Prikaz linearne regresije'**




**Izvor:** Vlastiti rad autora

U kontekstu logističkog poduzeća L, algoritam može procijeniti cijenu transporta izraženog u HRK s obzirom na unos udaljenosti (argumenta *vrijednost* metode). Procjena se vrši na temelju podataka korištenih u procesu treniranja i testiranja algoritma, a postiže se metodom:

```
LR.procjena_varijable(vrijednost = 500)
```

```
 >>> Odgovarajuća vrijednost za 500 je 5932.8011315484855
```

Preciznost modela ovisi o pozitivnim i negativnim korelacijama između podataka definiranih na osi *x* i osi *y*, a u danom slučaju preciznost modela iznosi visokih 83.5%:

```
LR.preciznost_modela()
```

```
 >>> Uspješnost modela iznosi 0.8345932878948212
```

Formula kojom se izračunava odnos unesene vrijednosti (u prikazanom slučaju *500* na osi *x*) i pripadajuće vrijednosti na osi *y* izračunava se metodom:

```
LR.formula()
```

```
 >>> Formula modela je y = 500 * 11.367176124296732 + 249.21306940012028
```

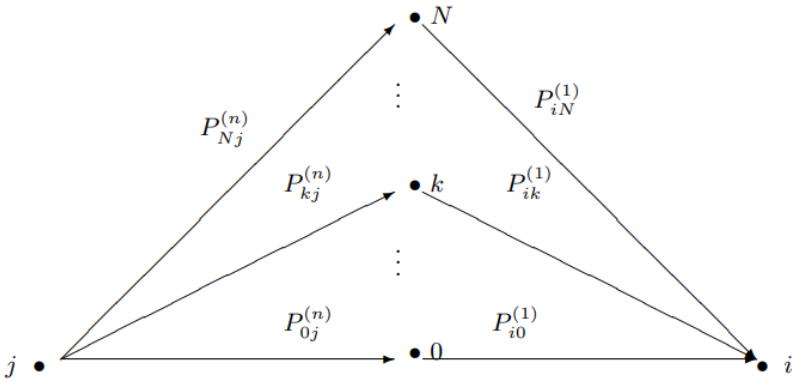
### 7.3.4. Procjena oscilacije opterećenosti prometnice u budućem razdoblju i evaluacija rizika primjenom Monte Carlo simulacije

Monte Carlo simulacije skup su različitih statističkih metoda procjene kretanja vrijednosti. Koriste se pri simuliranju kretanja vrijednosti u određenom vremenskom periodu prema izračunatim oscilacijama početnog skupa podataka.

Mažuranić (2017) navodi da je Monte Carlo simulaciju 1946. godine osmislio Stanislaw Ulam dok je 'radio na razvoju nuklearnog oružja u *Los Alamos National Laboratoryu*, a ime je dobila po kasinima Monte Carla gdje je ujak Stanislawa Ulama često kockao.'

Krpan (2021) prema Kundih (2021) Monte Carlo simulaciju u financijskom vrednovanju rizika projekta opisuje kao skup ponavljanja slučajnih ekstrakcijskih skupova zadanih vrijednosti za kritične varijable koje su uzete unutar određenih intervala uz pretpostavku izračuna indeksa financijskih pokazatelja projekta poput interne stope profitabilnosti te neto sadašnje vrijednosti iz svake ekstrahirane vrijednosti skupa.

Monte Carlo simulacije se uz procjenu rizika i simuliranje događaja često koriste i u kombinaciji s Markovljevim lancima (Slika 7.13), čime daju prikaz stohastičkih odnosa događaja poput predviđanja vremenskih uvjeta, kretanja vrijednosti na tržištu kapitala i slično.




*Slika 7.13 'Shematski prikaz Markovljevih lanaca',*
*Izvor: Wai-Ki Ching i Michael K. Ng, (2006): 'Markov Chains: Models, Algorithms and Applications', International Series in Operations Research and Management Science*

Nad provedenom Monte Carlo simulacijom može se primijeniti Empirijsko pravilo, koje prikazuje obuhvat mogućih rješenja u tri kategorije:

- ~68% rezultata unutar jedne standardne devijacije od srednje vrijednosti,
- ~95% rezultata unutar 1.96 standardne devijacije od srednje vrijednosti
- i ~99.7% rezultata unutar 3 standardne devijacije od srednje vrijednosti.

(Grimson et al., 2016 prema Kundih, 2021)

Prometna uprava P ima zadatak odrediti granice mogućeg generičkog povećanja ili smanjenja broja dnevnih vozila na promatranoj prometnici kroz 365 nadolazećih dana na temelju povijesnih podataka prikupljenih radarom u godini dana. Početna vrijednost (PGDP[113]) koja se uzima u obzir pri analizi označava prosječni broj vozila na prometnici u danu na temelju podataka iz cijele godine, a iznosi 14 000 vozila dnevno.

Izradi simulacije pristupa se unosom vlastite biblioteke *kundih i* kreiranjem simulacijskog okvira podataka[114]:

```
import kundih
```

Programski kod za izradu Monte Carlo simulacije u vlastitoj biblioteci *kundih* djelomično je inspiriran metodologijom i logikom funkcioniranja simulacije koju navodi Dupuis (2017).

Nakon unosa biblioteke, nad varijablom MC inicijalizira se objekt *MonteCarlo* s pripadajućim argumentima, a to su *lista_vrijednosti*, *period* i *broj_simulacija*:

```
MC = kundih.MonteCarlo(lista_vrijednosti = podaci, period = 365,
broj_simulacija = 300)
```

Definirana varijabla pohranjuje potrebne podatke za daljnje korake izvršavanja simulacije, a za konkretan slučaj definiran je period od 365 dana uz 300 simulacija. Nakon definiranja varijable, koristi se metoda kojom se pokreće simulacija nad njom i prikazuju vrijednosti (Tablica 7.13):

```
MC.izračunaj()
```

**Tablica 7.13 'Vrijednosti Monte Carlo simulacije'**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 14111.025930 | 14148.242796 | 14105.975996 | 14082.254465 | 14069.894468 | 14052.587768 | 14059.024449 | 14118.159501 | 14128 |
| **1** | 14022.537638 | 14027.194855 | 14189.147863 | 14099.634196 | 14057.400302 | 14071.255518 | 14034.664746 | 14180.401472 | 14136 |
| **2** | 14044.502877 | 14006.657020 | 14184.032338 | 13958.600798 | 14077.748769 | 14061.047633 | 13955.049865 | 14172.519898 | 14057 |
| **3** | 14041.152700 | 14073.563083 | 14135.492196 | 13932.112079 | 14049.757860 | 14087.809995 | 13972.828290 | 14175.287053 | 14081 |
| **4** | 14121.276910 | 14008.263392 | 14016.546657 | 13808.058202 | 14000.012751 | 14051.213165 | 13907.602255 | 14128.473886 | 14142 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **361** | 16167.737000 | 14838.059716 | 13789.598187 | 13833.747263 | 13568.623048 | 14565.334773 | 12885.904467 | 12143.441462 | 13307 |
| **362** | 16287.736957 | 14814.749400 | 13738.708664 | 13846.209559 | 13579.095741 | 14520.325827 | 13022.066929 | 12189.690486 | 13366 |
| **363** | 16420.136075 | 14736.732105 | 13655.424817 | 13797.128038 | 13539.603639 | 14555.759656 | 13120.744526 | 12171.866290 | 13322 |
| **364** | 16474.937281 | 14606.539335 | 13649.071010 | 13717.749516 | 13575.683707 | 14495.033605 | 13189.639140 | 12096.311522 | 13392 |
| **365** | 16326.290132 | 14672.871647 | 13620.800252 | 13671.212131 | 13570.162405 | 14494.612910 | 13227.859684 | 12063.357535 | 13483 |

**Izvor:** Vlastiti rad autora

[113] prosječni godišnji dnevni promet.

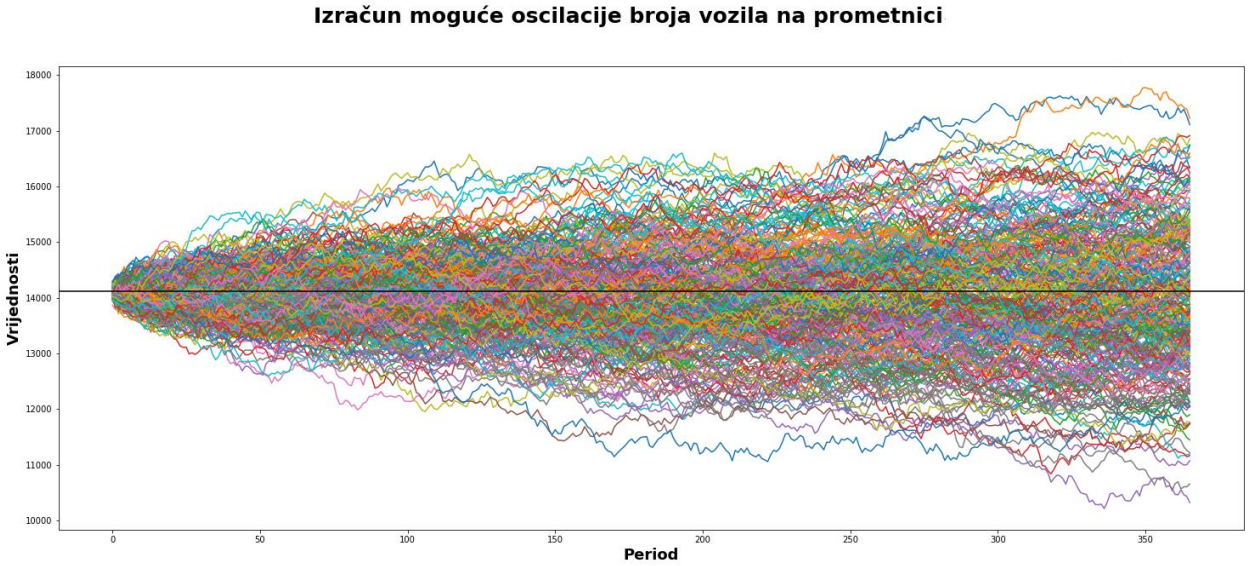
[114] programski kod i detaljna razrada dostupni u Prilozima.

Nakon tabličnog pohranjivanja podataka, postoje preduvjeti za prikaz simulacije na grafu, koristeći se ugrađenom metodom s pripadajućim argumentima za uređivanje prikaza ovisno o području primjene simulacije (*naslov*, *x_naslov* i *y_naslov*):

```
MC.prikaži_graf(naslov = 'Izračun moguće oscilacije broja vozila na
prometnici.', x_naslov = 'Period', y_naslov = 'Vrijednosti')
```

Grafikon 7.10 prikazuje Monte Carlo simulaciju broja vozila na prometnici u periodu od 365 dana kroz 300 simulacija. Očekivano je grupiranje većine vrijednosti oko sredine uz postojanje manje čestih ekstremnih varijacija.

**Grafikon 7.10 'Izračun moguće oscilacije broja vozila na prometnici'**




**Izvor:** Vlastiti rad autora

Grafički prikaz mogućeg kretanja broja vozila u nadolazećem periodu od 365 dana indikativno prikazuje najviši i najniži mogući broj vozila na prometnici prema simulaciji, no detaljnim informacijama pristupa se korištenjem metode:

```
MC.prikaži_statistiku()
```

**Tablica 7.14 'Statistički rezultati Monte Carlo simulacije'**

| | Statistika |
|---|---|
| **Srednja vrijednost** | 13981.88 |
| **Standardna devijacija** | 1262.15 |
| **Maksimalna vrijednost** | 17221.65 |
| **Minimalna vrijednost** | 10318.40 |

**Izvor:** Vlastiti rad autora

Tablica 7.14 prikazuje statističke podatke provedene Monte Carlo simulacije. Vidljivo je da je potencijalan maksimalan broj vozila na prometnici u danu definiran kao 17222, a najniži mogući broj 10318. Prema prethodno prikazanom Empirijskom pravilu, najizvjesnije je da će broj vozila na prometnici za 365 dana biti prikazan u rasponu srednje vrijednosti i jedne standardne devijacije u pozitivnom i negativnom izrazu, odnosno, 13982 +/- 1262 vozila na dan. Povijesni podaci daju informacije o standardnoj devijaciji na temelju kojih se izračunava postotna dnevna promjena.

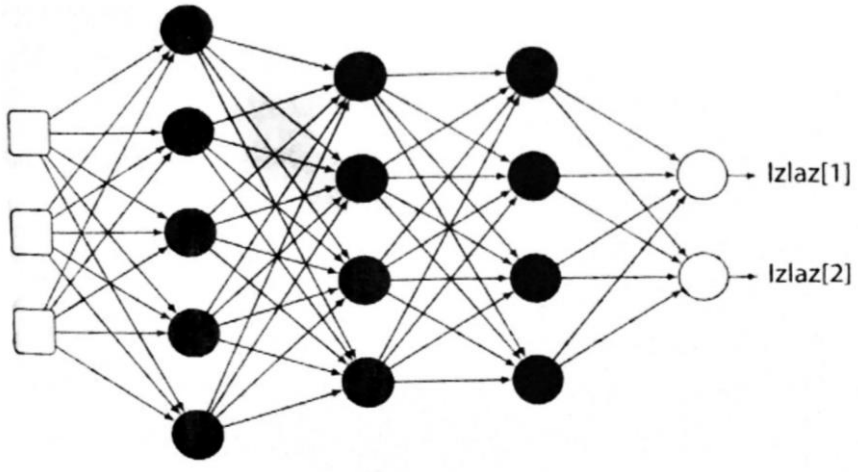
Spomenuta simulacija izračunata je na temelju povijesnih podataka koji možebitno ne obuhvaćaju velike promjene u padu zagušenja prometnice poput drastičnog pada vozila na prometnicama u vrijeme zatvaranja zbog pandemije SARSa-COV-2 i slabije turističke sezone.

U kontrastu s mogućim padom broja vozila na prometnici, postoje i brojni vanjski faktori koji utječu na možebitno povećanje zagušenja prometnice poput zatvaranja okolnih pravaca zbog radova i traganja stanovništva za alternativnim pravcima putovanja.

Ključ uspješnosti simulacije leži u količini prikupljenih podataka i reprezentativnosti realnih odnosa u prometu numeričkim načinom.

### 7.3.5. Umjetne neuronske mreže

Umjetne neuronske mreže jedno su od najvećih postignuća i rezultata istraživanja umjetne inteligencije te strojnog učenja. Umjetne neuronske mreže, za razliku od primjerice linearne regresije, osposobljene su za pronalazak međuodnosa između podataka u nelinearnim odnosima, što ih čini preciznima u simuliranju ljudskog ponašanja i projekciji kompleksnih ljudskih aktivnosti na strojeve, robote i programska rješenja. Slika 7.14 prikazuje shemu neuronske mreže s dvije ciljne klase.




*Slika 7.14 'Prikaz sheme umjetne neuronske mreže'*

*Izvor: Kelleher J.D. i Tierney, B. (2018): 'Data science', Massachusetts Institute of Technology, Mate d.o.o. (2021), Zagreb, Hrvatska*

Slika 7.14 uz dvije ciljne klase prikazuje i ulazni sloj umjetne neuronske mreže prikazan kvadratnim oblicima koji simboliziraju značajke ulaznih vrijednosti, te tri skrivena sloja umjetne neuronske mreže označenih crnim točkama s pripadajućim težinskim vrijednostima (linijama).

'Umjetne neuronske mreže sastoje se od ulaznog sloja, jednog ili više skrivenih slojeva i izlaznog sloja. Ulazni sloj sastoji se od značajki seta podataka koje se prosljeđuju naprijed prema skrivenim slojevima uz pridodavanje težina. Jedan ili više slojeva koriste se za ekstrakciju znanja iz podataka i detekciju korelacija kroz težinske vrijednosti uz analizu pristranosti… U izlaznom se sloju transferira prikupljeno znanje iz neurona u skrivenom sloju, te se vrši klasifikacija u umjetnim neuronskim mrežama nadziranog učenja, odnosno, općenito prikazana izlazna vrijednost u slučajevima umjetnih neuronskih mreža nenadziranog učenja.'

Kundih et al. (n.d.)

Kelleher i Tierney (2018) navode da se umjetne neuronske mreže sastoje od neurona koji izvode vrlo jednostavne operacije:

- množenje svake ulazne vrijednosti težinskim faktorom,
- zbrajanje rezultata množenja
- i propuštanje rezultata zbrajanja kroz aktivacijsku funkciju.

Neuroni izvršavaju matematičke operacije i oblikuju logiku zaključivanja neuronske mreže.

Postoje razne verzije umjetnih neuronskih mreža, a često se spominju rekurentne neuronske mreže koje implementiraju petlje između neurona te konvolucijske neuronske mreže koje se često koriste u algoritmima detekcije objekata (Kelleher i Tierney, 2018).

Sastavni dijelovi umjetne neuronske mreže navedeni u dokumentaciji Python biblioteke *keras* su funkcije gubitka i aktivacijska funkcija.

Funkcije gubitka koje se koriste u modelu umjetne neuronske mreže ovise o željenom ishodu koji se postavlja pred algoritam strojnog učenja, pri čemu se ističu kategorička krosentropija i binarna krosentropija (Kundih et al., n. d.). Uloga kategoričke krosentropije u modelu je optimizacija procesa učenja algoritma s ciljnim kategorijama, dok binarna krosentropija za rezultat ima optimizirati proces učenja algoritma s izlaznim vrijednostima 0 i 1 ili dvjema kategorijama.

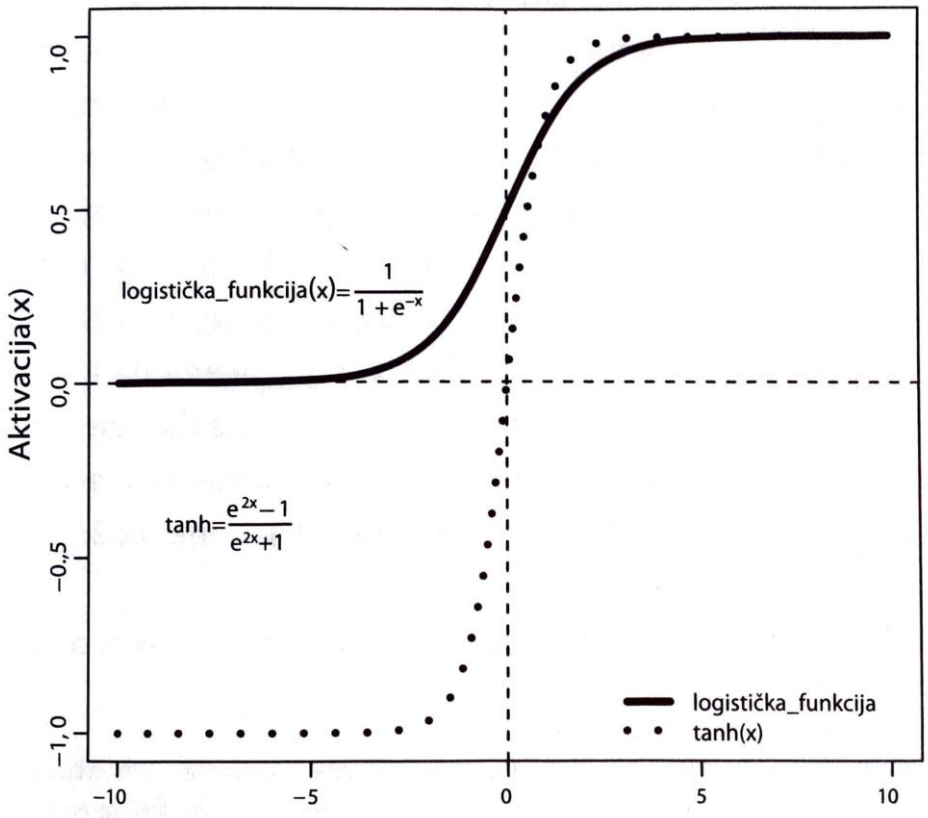
'Kelleher i Tierney (2018) navode da su funkcija logističke regresije i funkcija *tanh*[115] korištene najčešće, ali upotreba aktivacijske funkcije ovisi o okolnostima pa se često upotrebljavaju i složeniji oblici poput *ReLu*[116], *hardswish* i *softmax* funkcije.'

(Kundih et al., n. d.)

[115] eng. 'hyperbolic tangent' – hrv. hiperbolička tangenta.

[116] eng. 'rectified linear unit' – hrv. uzdignuta linearna jedinica.

**Grafikon 7.11 'Aktivacijske funkcije tanh i logistička regresija'**




**Izvor:** Kelleher J.D. i Tierney, B. (2018): 'Data science', Massachusetts Institute of Technology, Mate d.o.o. (2021), Zagreb, Hrvatska

Grafikon 7.11 prikazuje usporedbu aktivacijske funkcije *tanh* i logističke regresije.

Prema spomenutom prikazu vidljivo je da vrijednosti raspona *tanh* funkcije teže -1 ili 1, dok su vrijednosti funkcije logističke regresije isključivo u pozitivnom rasponu u kojem teže 0 ili 1.

Aktivacijska funkcija osposobljava umjetnu neuronsku mrežu na samostalno odlučivanje primjenom prikazanih formula na Grafikonu.

Prometna uprava P ima zadatak kreirati prediktivni model definiranja raspona broja vozila u određenom satu na temelju prikupljenih podataka na raskrižjima. Raspon broja vozila na raskrižju prikazan je kroz 9 ciljnih kategorija po 25 vozila u svakom rasponu.

Metodologija izrade programskog rješenja djelomično se bazira na modelu koji predlaže Galli (2020), uz preinake u odabiru korištenih aktivacijskih funkcija.

Nakon kreiranja okvira podataka i filtriranja, pristupa se njegovoj podjeli na dva dijela, odnosno, na testne podatke i podatke za treniranje.

U programsko sučelje unosi se objekt *Sequential*, biblioteke *tensorflow* (unesene kao *tf*) uz dodatak biblioteke *keras*, te se vrši njegova inicijalizacija nad varijablom model:

```
model = tf.keras.Sequential()
```

Nakon inicijalizacije objekta nad varijablom, modelu se pridodaju neuroni i njihov nasumičan odabir aktivacije. Skriveni slojevi neurona koriste aktivacijsku funkciju *ReLu,* dok je u posljednjem sloju neurona korištena aktivacijska funkcija *softmax* koja procjenjuje vjerojatnost svake klase i predlaže onu s najvišim brojčanim rješenjem. Funkcija gubitka koja se koristi u procesu je kategorička krosentropija, s obzirom na primjenu umjetnih neuronskih mreža na klasifikacijskom problemu, što se kroz programski kod prikazuje kao:

```
model.add(tf.keras.layers.Dense(64,input_shape = x_treniranje.shape,
activation = 'relu'))
model.add(Dropout(0.2))
model.add(tf.keras.layers.Dense(32, activation = 'relu'))
model.add(Dropout(0.2))
model.add(tf.keras.layers.Dense(32, activation = 'relu'))
model.add(Dropout(0.2))
model.add(tf.keras.layers.Dense(32, activation = 'relu'))
model.add(Dropout(0.2))
model.add(tf.keras.layers.Dense(16, activation = 'relu'))
model.add(Dropout(0.2))
model.add(tf.keras.layers.Dense(activation = 'softmax', units = 9))

model.compile(optimizer = 'adam', loss = 'categorical_crossentropy',
metrics = ['accuracy'])
```

Nakon definiranja modela i parametara koji se koriste u optimizaciji, pristupa se procesu treniranja i testiranja uspješnosti modela. Za spomenuti model korišten je proces od 10 iteracija po 32 serije, što je prikazano programskim kodom:

```
model.fit(x_treniranje, y_treniranje, epochs = 10, batch_size = 32)
```

Nakon provedenih 10 epoha testiranja i treniranja modela, korištena je metoda kojom se procjenjuje uspješnost njegova predviđanja, što je postignuto korištenjem programskog koda:

```
model.evaluate(x_testiranje, y_testiranje)
```

```
>>> 301/301 [==============================] - 1s 1ms/step - loss:
0.6273 - accuracy: 0.7761
```

Navedena izlazna vrijednost prikazuje uspješnost predviđanja u 77.6% slučajeva nad testnim podacima. Ključnu ulogu u uspješnosti predviđanja ima postavljena aktivacijska funkcija i granica prihvatljivosti uvjerenosti modela u ispravnu klasifikaciju. Klasifikaciji raspona pojedinog retka u testnom skupu za validaciju (u primjeru redak *600*) pristupa se programskim kodom:

```
y_testiranje[600]
```

```
>>> array([0., 1., 0., 0., 0., 0., 0., 0., 0.], dtype = float32)
```

Prikazan je niz koji sadrži istinitu vrijednost prikazanu kao *1*, a ono predstavlja drugi po redu raspon broja vozila na promatranom raskrižju, odnosno onaj između 25 i 50.

Testiranje modela s uvjerenosti u ispravnu klasifikaciju od 80% ili više prikazano je nad testnim primjerom ulazne vrijednosti u 600-tom retku, pri čemu je vrijednost okarakterizirana kao istina označena izrazom *True* (raspon broja vozila 25-50), dok su sve ostale prikazane izrazom *False* (rasponi 0-25, 50-75, 75-100, 100-125, 125-150, 150-175, 175-200, 200+):

```
granica = 0.8
predviđanje = model.predict(x_testiranje)
ispitaj = predviđanje >= granica
ispitaj[600]
```

```
>>> array([False,  True, False, False, False, False, False, False, False])
```

Spuštanjem granice na uvjerenost algoritma u ispravnu klasifikaciju od 10% ili više pristupa se jednakim programskim kodom uz preinaku vrijednosti granice tolerancije, pri čemu su vrijednosti okarakterizirane kao istina označene izrazom *True* (rasponi 25-50 i 50-75), dok su sve ostale vrijednosti prikazane izrazom *False* (rasponi 0-25, 75-100, 100-125, 125-150, 150-175, 175-200, 200+):

```
granica = 0.1
predviđanje = model.predict(x_testiranje)
ispitaj = predviđanje >= granica
ispitaj[600]
```

```
>>> array([False,  True,  True, False, False, False, False, False, False])
```

Iz navedenog prikaza vidljivo je da u spomenutom scenariju postoje dvije istinite procjene, čemu je razlog korištenja *softmax* aktivacijske funkcije koja provjerava postotak uvjerenosti modela u ispravnu odluku, što znači da postoji određeni postotak uvjerenosti modela i u krivu klasifikaciju. Postavljanje previsoke granice uvjerenosti modela u ispravnu klasifikaciju može rezultirati propustom u detekciji istinitih klasifikacija, dok se postavljanjem preniske granice mogu obuhvatiti i neistinite klasifikacije.

Prometna uprava P na temelju unesenih značajki (ulaznih vrijednosti) može procijeniti prometno zagušenje na pojedinom raskrižju u danom vremenu kroz izlaznu vrijednost koju predlaže izrađeni model.

Značajke koje se uzimaju u obzir pri treniranju i testiranu algoritma imaju ključnu ulogu u definiranju smjera odlučivanja modela. Pojedine značajke mogu biti irelevantne za donošenje odluke, stoga je nužno definirati ciljeve projekta prije izrade samog modela. Definiranje ciljeva modela direktno je povezano sa strateškim i operativnim ciljevima faze prikupljanja podataka na raskrižju na primjeru Prometne uprave P. Ispitivanje pozitivnog i negativnog odnosa između značajki ulaznih vrijednosti i izlazne vrijednosti može se iskazati numerički, korištenjem prethodno spomenutog Spearmanovog te Pearsonovog koeficijenta korelacije.

# 8. Digitalna transformacija prometnih i logističkih procesa

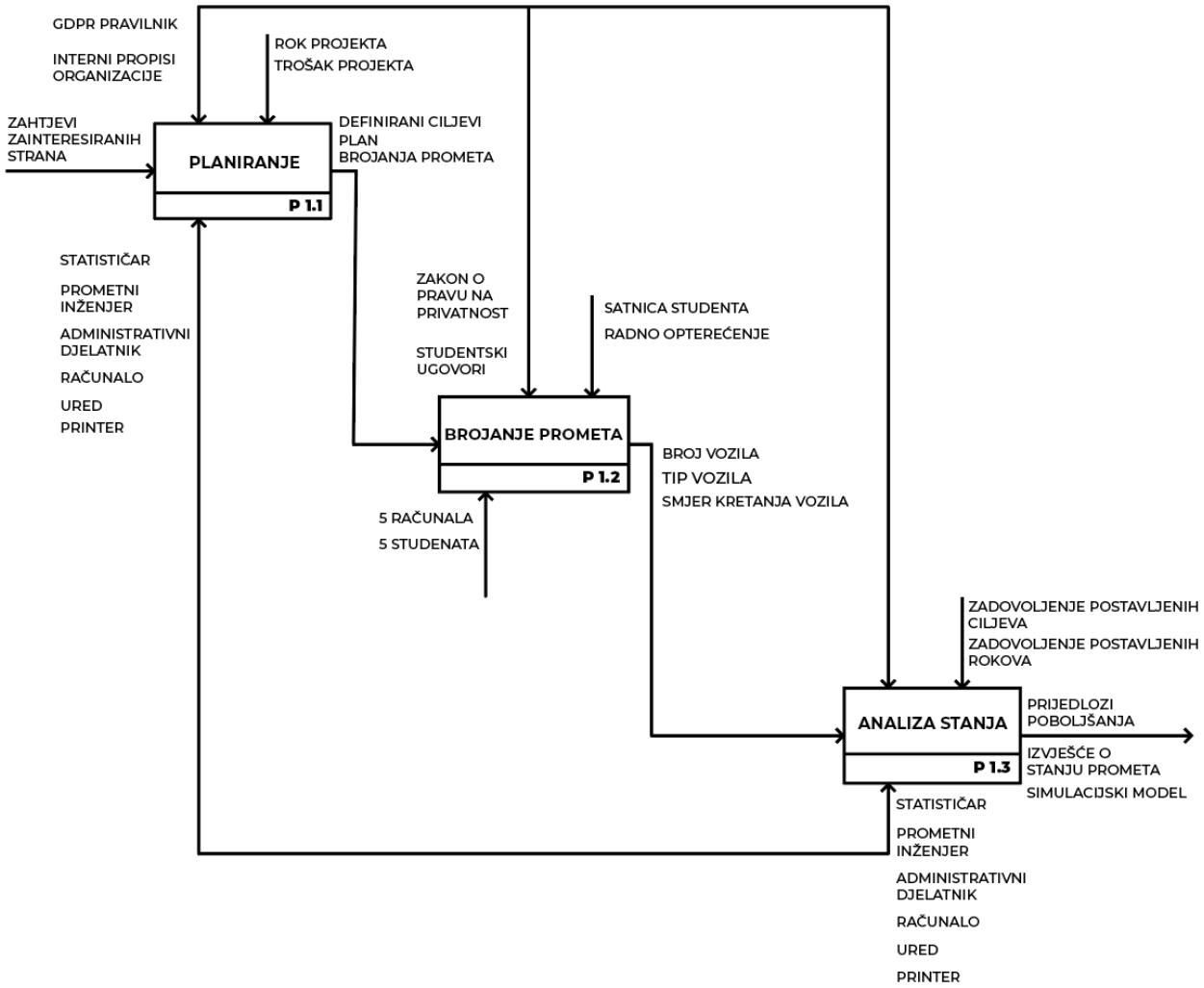
U ovom su poglavlju objedinjene metode koje se spominju u radu, te se predlaže rješenje digitalne transformacije prometno-logističkog procesa. Promatrani proces nad kojim se vrši digitalna transformacija je *utvrđivanje prometnog stanja na raskrižju*.

Proces je prikazan primjenom IDEF0 metodologije.[117]

## 8.1. Početni prikaz procesa

Početni prikaz dekompozicije procesa (Slika 8.1) definira potrebne ulaze, izlaze, kontrole, mehanizme i pravila svih potprocesa koji čine proces utvrđivanja prometnog stanja na raskrižju.

Spomenuti proces prikazan je kroz ulogu čovjeka pri brojanju prometa, što bitno umanjuje mogućnost procesuiranja i zapisivanja informacija s raskrižja u realnom vremenu.

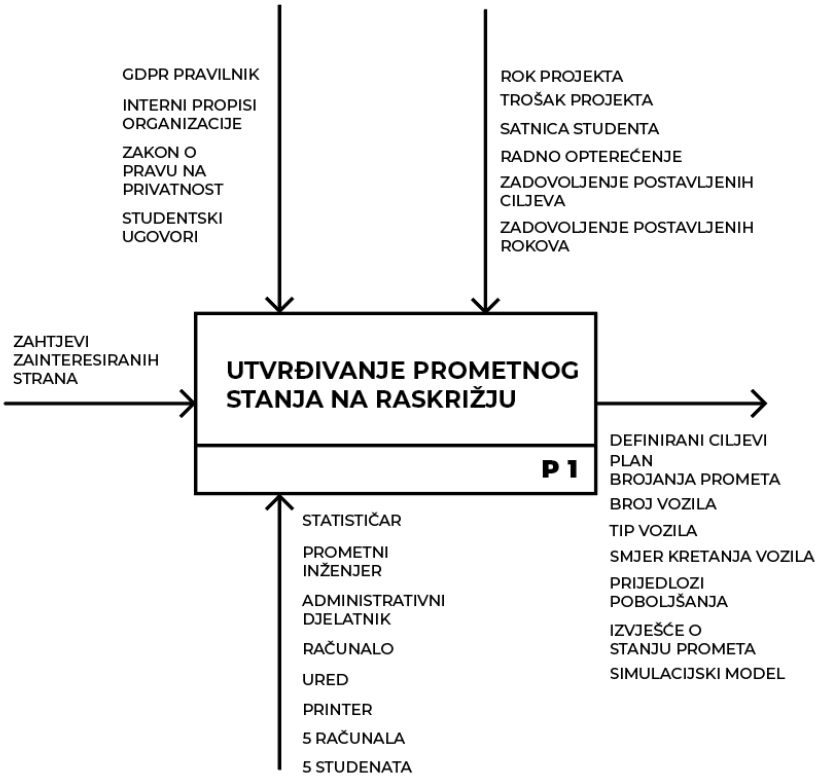



*Slika 8.1 'Prikaz početnog definiranja procesa utvrđivanja prometnog stanja na raskrižju', Izvor: Vlastiti rad autora*

[117] više o IDEF0 metodologiji u poglavlju 2.5.2.

Dekompozicija procesa izvršena je u tri koraka; planiranje, brojanje prometa i analizu stanja. Definirani potprocesi (*n*) su označeni simbolima P 1.*n*.

Nakon prikazanog primjera dekompozicije procesa, izrađen je dijagram konteksta s pripadajućim vrijednostima, a prikazan je na Slici 8.2.




*Slika 8.2 'Dijagram konteksta početnog procesa utvrđivanja prometnog stanja na raskrižju', Izvor: Vlastiti rad autora*

Proces utvrđivanja prometnog stanja na raskrižju prikazan je simbolom P 1.

Prikazani proces zadovoljava inicijalno postavljene zahtjeve projekta kroz prikupljanje informacija o broju vozila, tipu vozila i smjeru kretanja vozila na raskrižju, što pruža podlogu za izradu prijedloga poboljšanja, izvješća o stanju prometa i simulacijskog modela.

Pravna pravila koja reguliraju odvijanje procesa vezana su uz zakon prava na privatnost (s obzirom na prikupljanje podataka bilo koje vrste) i GDPR pravilnik koji opisuje operativne prepreke prikupljanja podataka usklađenih sa zahtjevima koje postavlja Europska unija.

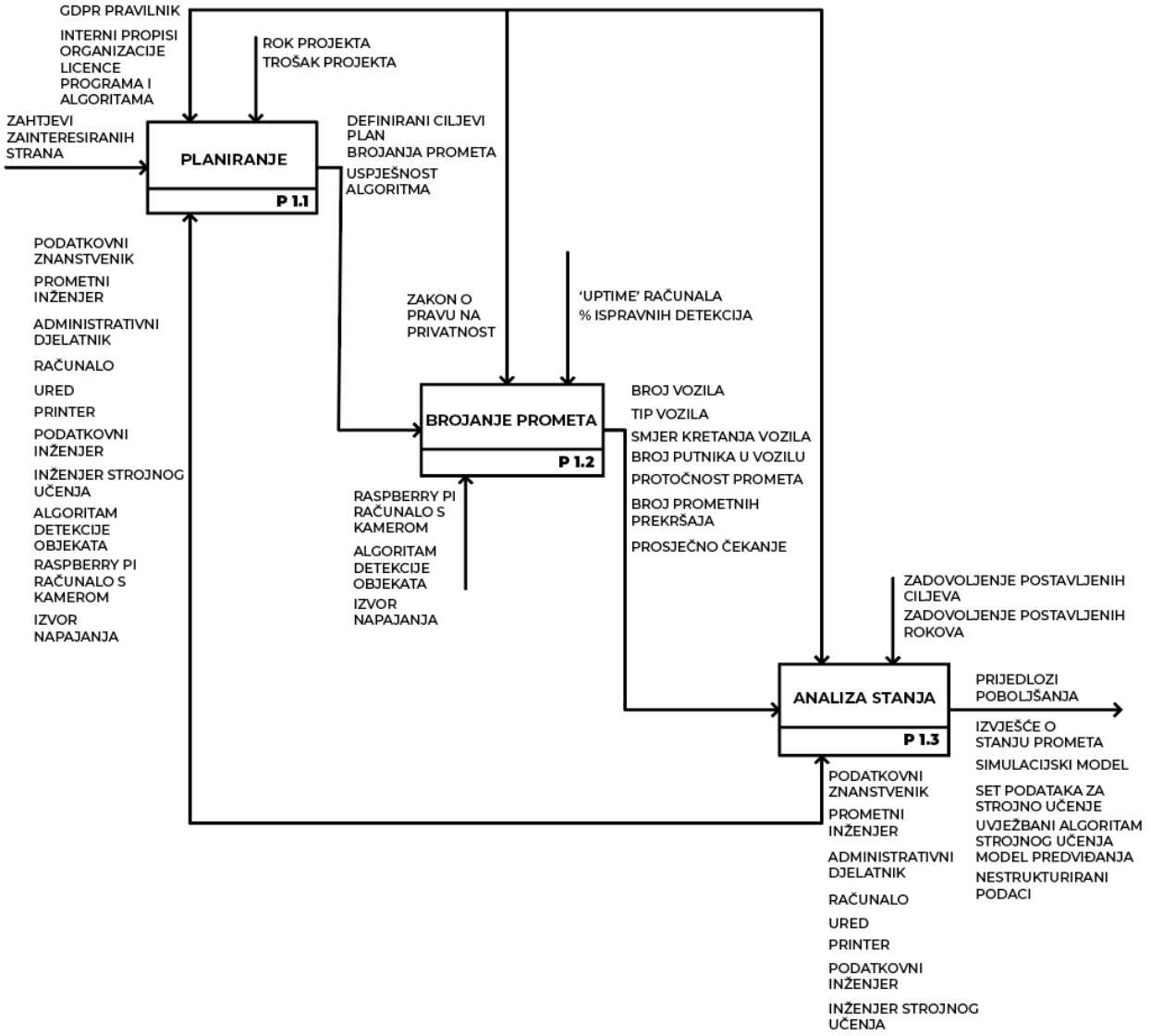
Interni propisi organizacije kreiraju smjernice obavljanja djelatnosti i predlažu ujednačenje ciljeva projekta s ciljevima organizacije.

Studentski ugovori definiraju satnicu, prava i obveze studenata koji vode evidenciju o zadanim parametrima istraživanja na raskrižju.

Ostali mehanizmi koji se koriste u procesu vezani su uz potrebne ljudske resurse, tehnologiju i infrastrukturu.

## 8.2. Digitalno transformirani proces

Primjer digitalne transformacije procesa prikazan je na identičnom procesu korištenom u prethodnom potpoglavlju, uz primjenu digitalnih tehnologija s ciljem digitalne transformacije procesa i stvaranja dodane vrijednosti iz procesa.

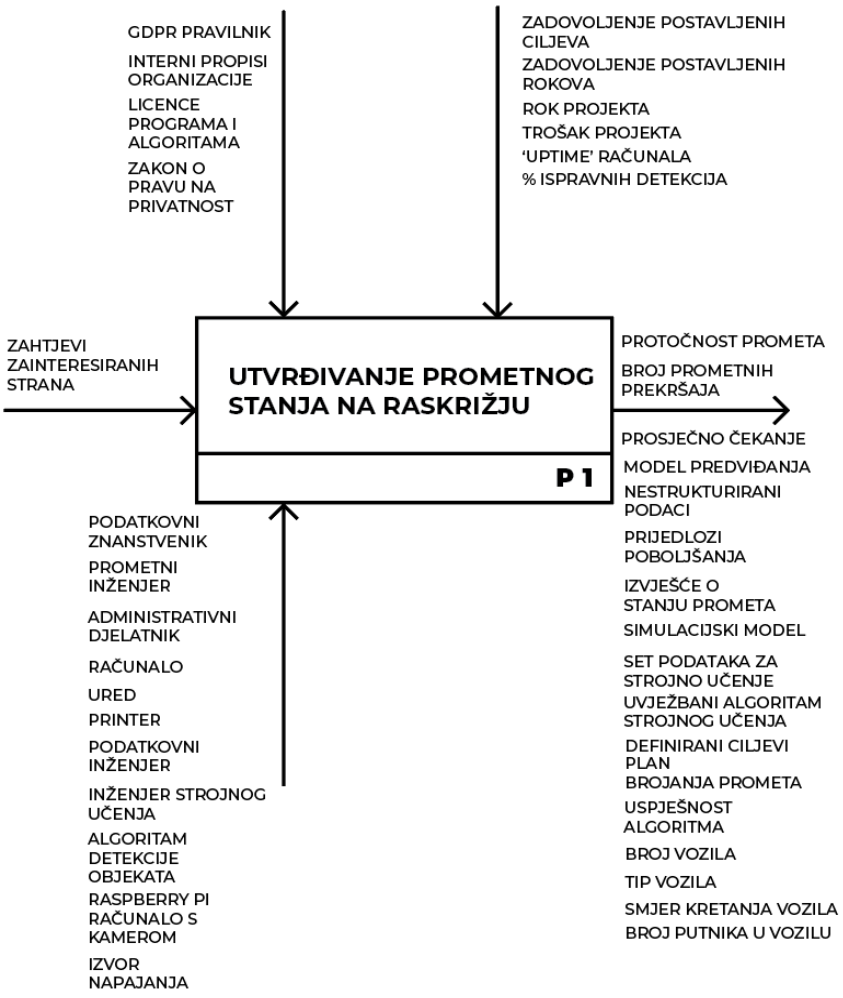



*Slika 8.3 'Prikaz definiranja digitalno transformiranog procesa utvrđivanja prometnog stanja na raskrižju',*

*Izvor: Vlastiti rad autora*

Slika 8.3 prikazuje dekompoziciju procesa utvrđivanja prometnog stanja na raskrižju pod učinkom digitalne transformacije. Korištenjem digitalnih tehnologija u procesu stvorena je dodana vrijednost, što se posebno očituje u količini izlaznih vrijednosti svakog od potprocesa. Izlazne vrijednosti prikazuju širok spektar različitih elemenata koji se mogu koristiti u svrhe koje nadilaze i one inicijalno predviđene projektom.

Nakon dekompozicije procesa utvrđivanja prometnog stanja na raskrižju, kreiran je prikaz dijagrama konteksta koji objedinjuje stavke svih potprocesa (Slika 8.4).




*Slika 8.4 'Dijagram konteksta digitalno transformiranog procesa utvrđivanja prometnog stanja na raskrižju',*

*Izvor: Vlastiti rad autora*

U odnosu na proces prethodnog potpoglavlja, digitalno transformirani proces provodi brojanje prometa korištenjem RaspBerry Pi uređaja s pripadajućom kamerom i algoritmom strojnog učenja koji vrši detekciju objekata i brojanje prometa u realnom vremenu. Sposobnosti algoritma strojnog učenja u nekim segmentima nadilaze ljudsko djelovanje, odnosno, mogu izvršiti brojanje prometa svih prometnih pravaca istovremeno, a uz broj vozila mogu dati i informacije o broju putnika u vozilu, tipovima vozila, broju prometnih prekršaja, prosječnom čekanju i slično. Prezentiranim benefitima implementacije algoritma strojnog učenja u brojanju prometa potrebno je odgovoriti i adekvatno osposobljenom radnom snagom sa specijaliziranim znanjima iz prikupljanja podataka i osposobljavanja algoritama za detekciju objekata. Prikupljeni nestrukturirani podaci mogu predstavljati temelje za buduće projekte i prediktivne modele podatkovnih znanstvenika. Ako se u procesu koriste tuđa programska rješenja potrebno je uskladiti njihovo djelovanje s licencom.

# 9. Zaključak

Održiva mobilnost i razvoj temeljen na tranzitu nužni su za podizanje kvalitete života u gradovima kroz smanjenje eksternih troškova prometa i harmonizaciju usluge javnog prijevoza namijenjene potrebama stanovništva. Segment integriranog prijevoza putnika od visokog je značaja povezivanja ruralnih područja s regionalnim središtima korištenjem različitih modova prijevoza, pri čemu željeznica ima presudnu ulogu s obzirom na kapacitet i minimalan ekološki utjecaj. Korištenjem digitalnih tehnologija teži se poboljšanju inteligentnih transportnih sustava kroz pravovremeno pružanje informacija i optimizaciju prometnih tokova.

Elementi digitalne transformacije bitno doprinose optimizaciji prometnih i logističkih procesa. Algoritmi strojnog učenja u mogućnosti su prepoznati međuodnose između podataka, čime u svojoj sposobnosti često nadilaze i ljudsku intuiciju. Korištenjem simulacijskih modela i algoritama strojnog učenja moguće je predvidjeti kretanje vrijednosti u budućem razdoblju, kao i detektirati međuodnos između unesenih značajki, pružajući pritom nužne informacije za odlučivanje u poslovnim i ostalim procesima.

Opravdanost uvođenja naprednih digitalnih tehnologija leži u prirodnom razvoju čovječanstva temeljenog na inovacijama koje doprinose kvaliteti ljudskog života. Kroz povijest se javljaju brojne inovacije koje radikalno mijenjaju dotadašnji način rada, počevši od parnih lokomotiva koje su konkurirale transportu kočijama, pa sve do izuma električne energije koja nadilazi mogućnosti parnih strojeva i stoga se i od umjetne inteligencije očekuje iskorak prema novim spoznajama i mogućnostima tehnologije. Razlika utjecaja umjetne inteligencije u odnosu na prethodno spomenute inovacije je sposobnost autonomnog razmišljanja strojeva i robota što može rezultirati dosad neviđenim promjenama svih sfera društva.

Usporedbom prometnih i logističkih procesa prije i nakon njihove digitalne transformacije primjećuje se kreiranje dodane vrijednosti digitalno transformiranih procesa s obzirom na prikupljanje podataka i njihovu upotrebu u svrhu razvoja digitalnih proizvoda te optimizaciju samih početnih procesa kroz automatizaciju i autonomno odlučivanje algoritama strojnog učenja.

Potvrđuje se mogućnost primjene algoritama s ciljem ekstrakcije znanja iz podataka i kreiranja modela predviđanja, pri čemu programski jezik Python s pripadajućim bibliotekama omogućava realizaciju ideja u stvarnost.

**H1: Digitalnom transformacijom prometnog ili logističkog procesa generira se širi niz izlaznih vrijednosti u odnosu na klasični proces bez primijenjene digitalne transformacije.**

Potvrđena je hipoteza rada H1. Usporedbom procesa utvrđivanja prometnog stanja na raskrižju prije i nakon njegove digitalne transformacije primjećuje se veći obujam i raznolikost prikupljenih podataka digitalno transformiranog procesa u odnosu na inicijalni proces.

Sveučilište
Sjever

SVEUČILIŠTE
SJEVER

**IZJAVA O AUTORSTVU**
**I**
**SUGLASNOST ZA JAVNU OBJAVU**

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, DAVID KUNDIH *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/~~ica~~ ~~-završnog~~/diplomskog (*obrisati nepotrebno*) rada pod naslovom DIGITALNA TRANSFORMACIJA PROMETNIH I LOGISTIČKIH SUSTAVA *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/~~ica~~:
*(upisati ime i prezime)*

(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, DAVID KUNDIH *(ime i prezime)* neopozivo izjavljujem da sam suglasan/na s javnom objavom ~~-završnog~~/diplomskog (*obrisati nepotrebno*) rada pod naslovom DIGITALNA TRANSFORMACIJA PROMETNIH I LOGISTIČKIH SUSTAVA *(upisati naslov)* čiji sam autor/~~ica~~.

Student/ica:
*(upisati ime i prezime)*

(vlastoručni potpis)

# Literatura


[1] Abadi, Martin et al., (2016): ‘Tensorflow: A system for large-scale machine learning’, In 12th $USENIX$ Symposium on Operating Systems Design and Implementation ($OSDI$ 16). pp. 265–283.

[2] Abdallah, Z.S., Du, L. i Webb, G.I., (2017): ‘Data Preparation.’, (dostupno na: https://academia.edu, pristupljeno: 28.06.2022.)

[3] AI Data & Analytics Network (2021): ‘2021 AI, Data & Analytics spend and Trends annual report’, (dostupno na: https://www.aidataanalytics.network/data-science-ai/whitepapers/2021-ai-data-analytics-spend-trends-annual-report, pristupljeno: 20.05.2022.)

[4] Awasthi, P. i George, J. J. (2020): ‘A Case for Data Democratization,’ in Proceedings of the 26th Americas Conference on Information Systems (AMCIS) (Vol. 23), Virtual conference, August 10.

[5] Azevedo, A. I. R. L. i Santos, M. F. (2008): ‘KDD, SEMMA and CRISP-DM: a parallel overview. IADS-DM’, (dostupno na: http://recipp.ipp.pt/bitstream/10400.22/136/3/KDDCRISP-SEMMA.pdf, pristupljeno: 22.06.2022.)

[6] Bašić, T. (2020): 'Primjena metoda analize i rudarenja podataka u edukaciji', Diplomski rad, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:217:081259, pristupljeno: 21.06.2022.)

[7] Bašković, A. (2018): 'Znanost o podacima: Postupci testiranja hipoteze', Diplomski rad, Sveučilište u Rijeci, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:195:452897, pristupljeno: 21.06.2022.)

[8] Benaicha, R. i Taibi, M., (2013): ‘Dijkstra algorithm implementation on fpga card for telecom calculations’, International Journal of Engineering Sciences & Emerging Technologies, 4(2). (dostupno na: https://www.ijeset.com/media/0001/14N8-IJESET0402831.pdf, pristupljeno: 05.07.2022.)

[9] Bilonić, B. (2020): 'Primjena programskog jezika F# u analizi i prikazu podataka', Završni rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:200:372569, pristupljeno: 24.06.2022.)

[10] Bitar, B. N. (2018): 'Internet stvari - IoT', Diplomski rad, Sveučilište Sjever, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:122:602841, pristupljeno: 31.05.2022.)

[11] Blagović, L. (2016): 'Vizualizacija podataka u sustavu poslovne inteligencije', Diplomski rad, Sveučilište Jurja Dobrile u Puli, Pula – Pola, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:137:821985, pristupljeno: 24.05.2022.)

[12] Bogati, J. (2011): 'Norme informacijske sigurnosti ISO/IEC 27k', Praktični menadžment, 2(2), str. 112-117, (dostupno na: https://hrcak.srce.hr/76462, pristupljeno: 23.05.2022.)

[13] Bojanić, D. (2019): 'Strojno učenje putem regresije i SVM', Diplomski rad, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:217:060657, pristupljeno: 30.06.2022.)

[14] Boning, D. S. (n.d.): 'AI and Machine Learning for Everyone', Module 2: 'Overview and Taxonomy', MITxPRO module, Emeritus – studijska grupa veljača, 2022.

[15] Boslaugh, S. (2013): 'Statistics in a Nutshell', Second Edition, O'Reilly Media, Inc., SAD.

[16] Bradley, P.S. i Fayyad, U.M., (1998): 'Refining initial points for k-means clustering', In ICML (Vol. 98, pp. 91-99)., (dostupno na: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.50.8528&rep=rep1&type=pdf, pristupljeno: 04.07.2022.)

[17] Braniša J. (2019): 'Intralogistika', [Završni rad]. Koprivnica: Sveučilište Sjever; 2019, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:122:182854, pristupljeno: 17.03.2022.)

[18] Bruun, E.P. i Duka, A., (2018): 'Artificial intelligence, jobs and the future of work: Racing with the machines', Basic Income Studies, 13(2)., (dostupno na: https://www.researchgate.net/profile/Edvard-Bruun/publication/329025467_Artificial_Intelligence_Jobs_and_the_Future_of_Work_Racing_with_the_Machines/links/5bf4b0da4585150b2bc63384/Artificial-Intelligence-Jobs-and-the-Future-of-Work-Racing-with-the-Machines.pdf, pristupljeno: 14.06.2022.)

[19] Buble, M. et al. (2005): 'Strateški menadžment', Sinergija nakladništvo d.o.o., Zagreb

[20] Buckingham, R. (2011). Customer Once, Client Forever. Washington: Kiplinger Books.

[21] Buntak, K. (2016): 'Organizacijska kompetentnost' - prezentacija, Hrvatska gospodarska komora, (dostupno na: https://www.hgk.hr/documents/kresimir-buntak-iso-forum-hgk16122016585a530356f57.pdf, pristupljeno: 01.06.2022.)

[22] Buntak, K., Kovačić, M. i Kondić, Ž. (2020): 'Upravljanje kvalitetom 2', Sveučilište Sjever. Varaždin.

[23] Buntak, K., Kovačić, M. i Premužić, B. (2020): 'Upravljanje poslovnim procesima - praktikum', Sveučilište Sjever (Sveučilišni centar Koprivnica)

[24] Buntak, K; Sesar,V; Kovačić,M (2016): 'Kontroling poslovnih procesa', Quality system condition for successful business and competitiveness. Kopaonik, Srbija

[25] Candra, A., Budiman, M.A. i Pohan, R.I., (2021): ‘Application of A-Star Algorithm on Pathfinding Game’, Journal of Physics: Conference Series (Vol. 1898, No. 1, p. 012047). IOP Publishing. (dostupno na: https://iopscience.iop.org/article/10.1088/1742-6596/1898/1/012047/pdf, pristupljeno: 05.07.2022.)

[26] Čavrak,V. (2003): 'Makroekonomski management i strategija prometa Hrvatske', Zagreb

[27] CERT CARNet – arhivirani članak, (2010): ‘Cloud Computing’ - https://www.cert.hr/wp-content/uploads/2010/03/NCERT-PUBDOC-2010-03-293.pdf, (pristupljeno: 02.06.2022.)

[28] Cervero, R. (1998): 'The Transit Metropolis: A Global Inquiry', Island Press, Washington, DC, SAD.

[29] Chapman, P. et al., (2000): ‘CRISP-DM 1.0 - Step-by-step data mining guide’, (dostupno na: http://www.crisp-dm.org/CRISPWP-0800.pdf, pristupljeno: 22.06.2022.)

[30] Chen, Y., Paxson, V., Katz i Randy H. (2010): ‘What’s New About Cloud Computing Security?‘, Electrical Engineering and Computer Sciences University of California at Berkeley, (dostupno na: http://www.utdallas.edu/~muratk/courses/cloud13s_files/what-is-new-in-cloud-security.pdf, pristupljeno: 02.06.2022.)

[31] Chollet, F. et al., (2015): ‘Keras’. (dostupno na: https://github.com/fchollet/keras, pristupljeno: 11.07.2022.)

[32] Cole, D., (2020): ‘The Chinese Room Argument’, The Stanford Encyclopedia of Philosophy (Winter 2020 Edition), Edward N. Zalta (ed.), (dostupno na: https://plato.stanford.edu/archives/win2020/entries/chinese-room/, pristupljeno: 18.06.2022.)

[33] Coombs, W. T., (2007): ‘Crisis management and communications’, Institute for Public Relations

[34] Creemers, R. (2018): ‘China's Social Credit System: an evolving practice of control’, Available at SSRN 3175792., (dostupno na: http://mx.nthu.edu.tw/~cshwang/data-economics/course-infoecon/INFE14-Meaning/Creemers=China%20Social%20Credit%20System-2020.pdf, pristupljeno: 16.06.2022.)

[35] Datacamp – web stranica (Jalswal, S.) – članak (2017), ‘Python Data Structures Tutorial’, (dostupno na: https://www.datacamp.com/tutorial/data-structures-python, pristupljeno: 23.06.2022.)

[36] Debug – web stranica (Vrbanus, S.) – članak, (2021) - https://www.debug.hr/gideon-brothers-jedan-je-od-vodecih-inovatora-u-podrucju-industrijske-robotike/, (pristupljeno: 01.06.2022.)

[37] Deloitte – web stranica (Burke, R., Laaper, S., Hartigan, M., Sniderman, B.) – članak (2017): https://www2.deloitte.com/us/en/insights/focus/industry-4-0/smart-factory-connected-manufacturing.html/#endnote-sup-6, (pristupljeno 01.06.2022.)

[38] Dey, A. (2019): 'Dijkstra's algorithm in Python explained', GitHub repozitorij (amitabhadey) (dostupno na: https://github.com/amitabhadey, pristupljeno: 05.07.2022.)

[39] Dijkstra E.W. (1959): 'A note on two problems in connection with graphs', Numerische Mathematik 1, pp.269–271.

[40] Dourado, E. i Brito, J. (2014): 'The New Palgrave Dictionary of Economics', Online Edition., (dostupno na: http://jerrybrito.com/pdf/cryptocurrency-newpalgrave.pdf, pristupljeno: 08.06.2022.)

[41] Dukić, B. i Gale, V., (2015): 'Upravljanje odnosima s potrošačima u funkciji zadržavanja potrošača', Ekonomski vjesnik/Econviews-Review of Contemporary Business, Entrepreneurship and Economic Issues, 28(2), pp.583-598.

[42] Dupuis, F. (2017): 'Simple Monte Carlo simulation of Stock Prices in Python', Python datoteka.

[43] Environmental Systems Research Institute – web stranica, neautorizirano (2009): 'Box plot graphs', verzija 9.3 (dostupno na: https://webhelp.esri.com/arcgisdesktop/9.3/body.cfm?tocVisable=1&ID=478&TopicName=Box%20plot%20graphs#:~:text=The%20lower%20and%20upper%20edges,smaller%20than%20the%20third%20quartile., pristupljeno: 29.06. 2022.)

[44] EUR-Lex (2016): 'Uredba (EU) 2016/679 Europskog parlamenta i Vijeća od 27. travnja 2016. o zaštiti pojedinaca u vezi s obradom osobnih podataka i o slobodnom kretanju takvih podataka te o stavljanju izvan snage Direktive 95/46/EZ (Opća uredba o zaštiti podataka)', (dostupno na: https://eur-lex.europa.eu/legal-content/HR/TXT/?uri=CELEX:32016R0679, pristupljeno: 23.05.2022.)

[45] Evans, B. (2020): 'The Zoom revolution: 10 eye-popping stats from tech's new superstar', Cloud Wars, (dostupno na: https://cloudwars.co/covid-19/zoom-quarter-10-eyepopping-stats-from-techs-new-superstar/, pristupljeno: 03.06.2022.)

[46] Flowers, J.C., (2019): 'Strong and Weak AI: Deweyan Considerations', In AAAI Spring Symposium: Towards Conscious AI Systems (Vol. 2287, No. 7)., (dostupno na: http://ceur-ws.org/Vol-2287/paper34.pdf, pristupljeno: 18.06.2022.)

[47] Fowers, R. (2019): 'Linear Regression Python Sklearn [FROM SCRATCH]', Jupter Notebook datoteka.

[48] Franceschet, M., Colavizza, G. i Smith, T., (2020): 'Crypto art: A decentralized view', Leonardo pp. 1–8 (2020)

[49] Gabrovec, M. (2020): 'Statističko i računalno modeliranje širenja pandemije COVID-19', Završni rad, Sveučilište Jurja Dobrile u Puli, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:137:781096, pristupljeno: 04.07.2022.)

[50] Galli, K. (2020): 'Introduction to Neural Networks in Python (what you need to know) | Tensorflow/Keras', Python datoteka

[51] Ghahramani, Z., (2003): 'Unsupervised learning', In Summer school on machine learning (pp. 72-112). Springer, Berlin, Heidelberg. (dostupno na: http://datajobstest.com/data-science-repo/Unsupervised-Learning-Guide-[Zoubin-Ghahramani].pdf, pristupljeno: 01.07.2022.)

[52] Gras, C., Smith, N., Sengmanivong, L., Gandini, M., Kubelka, C.F. i Herbeuval, J.P., (2013): 'TRAIL protein localization in human primary T cells by 3D microscopy using 3D interactive surface plot: a new method to visualize plasma membrane', Journal of immunological methods, 387(1-2), pp.147-156., (dostupno na: https://dl.acm.org/doi/pdf/10.1145/361219.361220, pristupljeno: 30.06.2022.)

[53] Grimson E., Guttag, J., i Bell, A., (2016): 'Introduction to Computational Thinking and Data Science'. Massachusetts Institute of Technology (MIT), (dostupno na: https://ocw.mit.edu/courses/6-0002-introduction-to-computational-thinking-and-data-science-fall-2016/, pristupljeno: 07.07.2022.)

[54] Handbook on the external costs of transport (2019), Bruxelles: European Commission, Directorate-General for Mobility and Transport

[55] Harris, C.R. et al., (2020): 'Array programming with NumPy', Nature, 585, pp.357–362.

[56] Hax, A., (n.d): n.d., MIT Sloan, MITxPRO module, Emeritus – studijska grupa veljača, 2022.

[57] Himanshu, V. (n.d.): 'AI & ML for Everyone', Office Hours with Himanshu Vaidya, Emeritus, studijska grupa veljača, 2022.

[58] Hrvatska udruga koncesionara za autoceste s naplatom cestarine (2018): 'Ključne brojke', (dostupno na: http://www.huka.hr/files/docs/HUKA_KF_2018_KB_final-Web.pdf, pristupljeno: 12.07.2022)

[59] Hunter, J.D., (2007): 'Matplotlib: A 2D graphics environment', Computing in science & engineering, 9(3), pp.90–95.

[60] Igrec, A. (2018): 'Digitalna transformacija', Diplomski rad, Sveučilište u Zagrebu, Fakultet organizacije i informatike, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:211:961042, pristupljeno: 19.05.2022.)

[61] Inmon, W.H., Strauss, D. i Neushloss, G. (2008): 'DW 2.0: the architecture for the next generation of data warehousing', Morgan Kaufmann

[62] Investopedia – web stranica (Frankenfield, J.) – članak (2022): ‘Hash’ - https://www.investopedia.com/terms/h/hash.asp#:~:text=A%20hash%20is%20a%20function,produce%20the%20same%20hashed%20value. (pristupljeno: 06.06.2022.)

[63] Ivaković Č., Stanković R., Šafran M. (2010): 'Špedicija i logistički procesi', Fakultet prometnih znanosti, Zagreb

[64] Jakopić, K. (2021): 'Efekt biča u lancima opskrbe', [Diplomski rad]. Zagreb: Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje; 2021, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:235:352944, pristupljeno: 01.04.2022.)

[65] Jašarević, A. (2019): 'Upravljanje podacima - nekad i danas', Završni rad, Sveučilište u Rijeci, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:195:809682, pristupljeno: 27.05.2022.)

[66] Jonathan Hartley & Arnon Yaari (2013-2020): ‘Colorama’, BSD 3-Clause license

[67] Jugo, D. (2017): ’Menadžment kriznog komuniciranja’, Sveučilište u Dubrovniku, naklada Školska knjiga, Zagreb

[68] Kaisler, S.H., Espinosa, J.A., Armour, F. i Money, W.H. (2014): ‘Advanced Analytics--Issues and Challenges in a Global Environment’, In 2014 47th Hawaii international conference on system sciences (pp. 729-738). IEEE. (dostupno na: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6758694, pristupljeno: 25.05.2022.)

[69] Kangas, O., Jauhiainen, S., Simanainen, M. i Ylikännö, M., (2019): ‘The basic income experiment 2017–2018 in Finland: Preliminary results’, (dostupno na: https://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/161361/Report_The%20Basic%20Income%20Experiment%2020172018%20in%20Finland.pdf, pristupljeno: 14.06.2022.)

[70] Kaplan, A.D., Cruit, J., Endsley, M., Beers, S.M., Sawyer, B.D. i Hancock, P.A., (2021): ‘The effects of virtual reality, augmented reality, and mixed reality as training enhancement methods: A meta-analysis’, Human factors, 63(4), pp.706-726.

[71] Karl, K.A., Peluchette, J.V. i Aghakhani, N. (2022): ‘Virtual work meetings during the COVID-19 pandemic: The good, bad, and ugly’, Small Group Research, 53(3), pp.343-365., (dostupno na: https://journals.sagepub.com/doi/pdf/10.1177/10464964211015286, pristupljeno: 03.06.2022.)

[72] Kelleher J.D. i Tierney, B. (2018): 'Data science', Massachusetts Institute of Technology, Mate d.o.o. (2021), Zagreb, Hrvatska

[73] Khine, P.P. i Wang, Z.S. (2018): ‘Data Lake: a new ideology in big data era’, In ITM web of conferences (Vol. 17, p. 03025). EDP Sciences. (dostupno na: https://www.itm-conferences.org/articles/itmconf/pdf/2018/02/itmconf_wcsn2018_03025.pdf, pristupljeno: 23.05.2022.)

[74] Kirschen, P. i Burnell, E. (2021): 'Hyperloop system optimization'. arXiv preprint arXiv:2104.03907., (dostupno na: https://arxiv.org/pdf/2104.03907.pdf, pristupljeno: 16.05.2022.)

[75] Klečina, A. (2020): 'Mjere održive mobilnosti u provedbenim dokumentima prostornog uređenja - primjer općine Trnovec Bartolovečki', Diplomski rad, Sveučilište Sjever, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:122:913648, pristupljeno: 13.05.2022.)

[76] Kluyver, T. et al., (2016): 'Jupyter Notebooks – a publishing format for reproducible computational workflows', In F. Loizides & B. Schmidt, eds. Positioning and Power in Academic Publishing: Players, Agents and Agendas. pp. 87–90.

[77] Knight, A. i Creemers, R., (2021): 'Going viral: The social credit system and COVID-19', Available at SSRN 3770208., (dostupno na: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3770208, pristupljeno: 16.06.2022.)

[78] Knowles, R.D. (2012): 'Transit oriented development in Copenhagen, Denmark: from the finger plan to Ørestad', Journal of transport geography, 22, pp.251-261. (dostupno na: https://www.researchgate.net/profile/Richard-Knowles-3/publication/254609662_Transit_Oriented_Development_in_Copenhagen_Denmark_From_the_Finger_Plan_to_Orestad/links/6140c40bea4aa800110456a4/Transit-Oriented-Development-in-Copenhagen-Denmark-From-the-Finger-Plan-to-Orestad.pdf, pristupljeno: 16.05.2022.)

[79] Kolinger, D. (2013): 'Primjena automatski vođenih vozila'. Završni rad, Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje, (dostupno na: http://repozitorij.fsb.hr/2135/1/21_02_2013_Zavrsni_rad_AGV_konacna_verzija.pdf, pristupljeno: 14.03.2022.)

[80] Koprek, D. (2021): 'Skladištenje podataka i poslovna inteligencija na platformi Microsoft Azure', Diplomski rad, Sveučilište u Zagrebu, Fakultet organizacije i informatike, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:211:085386, pristupljeno: 20.05.2022.)

[81] Kovačić, M. (2019): 'Kontroling poslovnih procesa', Diplomski rad, Sveučilište Sjever, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:122:586143, pristupljeno: 19.04.2022.)

[82] Kramberger, T., Duk, S. i Kovačević, R. (2018): 'Baze podataka', Tehničko veleučilište u Zagrebu, Zagreb

[83] Krelja Kurelović, E., Tomljanović, J. i Bronić, K., (2014): 'Uporaba aplikacija u oblaku kod studenata', Zbornik Veleučilišta u Rijeci, 2(1), pp.13-26.

[84] Krpan, Lj. (2017): 'Modeliranje upravljačkog sustava u cestovnom prometu urbanih područja', Sveučilište Sjever, Sveučilišni centar Koprivnica

[85] Krpan, Lj. (2021). 'Upravljanje i vrednovanje projekata' Sveučilište Sjever, Koprivnica.

[86] Kumar, V. i Reinartz, W. (2018): '*Customer relationship management*', Springer-Verlag GmbH Germany, part of Springer Nature 2006, 2012, 2018.

[87] Kundih, D. (2020): 'Upravljanje rizicima u okolnostima krizne situacije', Završni rad, Sveučilište Sjever, Sveučilišni centar Koprivnica, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:122:239125, pristupljeno: 23.05.2022.)

[88] Kundih, D. (2021) - Python biblioteke: duality, vandal, logistics i kundih (dostupno na: https://pypi.org/user/dkundih/ i https://github.com/dkundih, pristupljeno: 11.07.2022.)

[89] Kundih, D. (2021): 'Procjena rizika u financijskom menadžmentu primjenom Monte Carlo simulacija', seminar iz kolegija Financije i računovodstvo, Sveučilište Sjever, Koprivnica

[90] Kundih, D. (2022): 'Umjetna inteligencija u logistici', Suvremena trgovina 3(47)-2022; tema broja: logistika i sajmovi (str. 32-33), lipanj 2022., Zagreb (dostupno na: https://issuu.com/st-1-2015-online/docs/suvremena-trgovina-3-2022, pristupljeno: 29.06.2022.)

[91] Kundih, D., Biškup, N. i Buntak, K. (n. d.): 'Digital transformation of sustainable mobility systems using Artificial neural networks', – neobjavljeno, u pripremi

[92] Larose, D. (2004): 'Discovering Knowledge in Data: An Introduction to Data Mining', WileyInterscience, 2004.

[93] Lefebvre, H., Legner, C. i Fadler, M. (2021): 'Data democratization: toward a deeper understanding', (dostupno na: https://www.researchgate.net/profile/Christine-Legner-2/publication/354906721_Data_democratization_toward_a_deeper_understanding/links/61539051fd7b3d1215599a07/Data-democratization-toward-a-deeper-understanding.pdf, pristupljeno: 23.05.2022.)

[94] Liu, B., (2021): 'Weak AI" is Likely to Never Become" Strong AI', So What is its Greatest Value for us?, arXiv preprint arXiv:2103.15294., (dostupno na: https://arxiv.org/pdf/2103.15294.pdf, pristupljeno: 18.06.2022.)

[95] Love, T. E. (2019): 'Data science for biological, medical and health research: Notes for PQHS 431', (dostupno na: https://thomaselove.github.io/2019-431-book/re-expression-tukeys-ladder-box-cox-plot.html, pristupljeno: 27.06.2022.)

[96] Lovrinović, I. (2018): 'Digitalna transformacija nije informatizacija', Završni rad, Sveučilište u Rijeci, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:195:323999, pristupljeno: 19.05.2022.)

[97] Lozina, D. (1994): 'Teorija sustava kao instrument društvene analize', Društvena istraživanja-Časopis za opća društvena pitanja, 3(14), pp.671-684.

[98] Lutz, M., (2013): 'Learning python: Powerful object-oriented programming', O'Reilly Media, Inc.

[99] M. Gupta (2017): ‘Blockchain for Dummies’, IBM, John Wiley & Sons, Inc., Hoboken, New York

[100] Magzhan, K. i Jani, H.M., (2013): ‘A review and evaluations of shortest path algorithms’, International journal of scientific & technology research, 2(6), pp.99-104. (dostupno na: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.307.5792&rep=rep1&type=pdf, pristupljeno: 05.07.2022.)

[101] Mahnken, D. (2021): ‘Prices in European road transport increase in Q1 2021’, (dostupno na: https://www.saloodo.com/blog/prices-in-european-road-transport-increase-in-q1-2021/, pristupljeno: 06.07.2022.)

[102] Malin F. (2017): ‘Dense automatic speed enffectively reduces speeding’, VTT Technical Research Centre of Finland, Oulu, Finska

[103] Mažuranić, J. (2017). 'Primjena Monte Carlo simulacija u procjeni rizika', Završni rad, Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje (dostupno na: https://urn.nsk.hr/urn:nbn:hr:235:862354, pristupljeno 07.07.2022.)

[104] McKinney, W. et al., (2010): ‘Data structures for statistical computing in python’, In Proceedings of the 9th Python in Science Conference. pp. 51–56.

[105] Miloslavskaya, N. i Tolstoy, A. (2016): ‘Big data, fast data and data lake concepts’, Procedia Computer Science, 88, pp.300-305., (dostupno na: https://www.sciencedirect.com/science/article/pii/S1877050916316957, pristupljeno: 20.05.2022.)

[106] Mislav, K. (2019). 'Rudarenje podataka u poduzetništvu', Završni rad, Sveučilište u Zagrebu, Fakultet organizacije i informatike, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:211:071761, pristupljeno: 21.06.2022.)

[107] Mrkonja, D. (2020): 'Mogućnost primjene koncepta Industrije 4.0 u Hrvatskoj', Završni rad, Sveučilište Sjever, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:122:223386, pristupljeno: 01.06.2022.)

[108] Mutavdžija, M. (2019): 'Metodološki pristup određivanja konteksta organizacije', Diplomski rad, Sveučilište Sjever, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:122:486423, pristupljeno: 14.04.2021.)

[109] Mysar, A.B. (2021): ‘Lung cancer’, Kaggle dataset, (dostupno na: https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer?resource=download, pristupljeno 25.05.2022.)

[110] Nanaj, A. (2020): ‘Huawei blacklist: The effect on the USA technology sector’, (dostupno na: https://repository.ihu.edu.gr/xmlui/handle/11544/29599, pristupljeno: 26.05.2022.)

[111] Narodne novine (2014): ‘Nacionalni program za razvoj i uvođenje inteligentnih transportnih sustava u cestovnom prometu za razdoblje od 2014. do 2018. Godine’, Zagreb, Narodne novine d.d., broj 82. (dostupno na: http://www.poslovni-savjetnik.com/propisi/nacionalni-program-za-razvoj-iuvodenje-inteligentnih-transportnih-sustava-u-cestovnom, pristupljeno 17.05.2022.)

[112] Nelson, M.J., (2012): ‘Soviet and American precursors to the gamification of work’, In: Proceedings of the 16th International Academic MindTrek Conference. Presented at MindTrek'12. ACM, pp. 23–26.

[113] Nikolašević, S. (2016): 'Metode i programi za rudarenje po podacima', Završni rad, Sveučilište Jurja Dobrile u Puli, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:137:527068, pristupljeno: 21.06.2022.)

[114] O’Neill, C. i Rachel S. (2013): Doing data science: Straight talk from the frontline. O’Reilly Media, Inc.

[115] Oulovsky, N. (2018): 'Utjecaj koncepta Industrija 4.0 na razvoj distribucijskih sustava', Diplomski rad, Sveučilište u Zagrebu, Fakultet prometnih znanosti, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:119:689861, pristupljeno: 01.06.2022.)

[116] Patel, D. (2019): ‘Clustering with K means – Python Tutorial’, GitHub repozitorij (dhavalsays), (dostupno na: https://github.com/codebasics/py/blob/master/ML/13_kmeans/13_kmeans_tutorial.ipynb, pristupljeno: 04.07.2022.)

[117] Pavličević, N. (2019): 'Primjena inteligentnih transportnih sustava (ITS) na primjeru grada Varaždina', Diplomski rad, Sveučilište Sjever, Sveučilišni centar Koprivnica, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:122:686137, pristupljeno: 17.05.2022.)

[118] Pedregosa, F. et al., (2011): ‘Scikit-learn: Machine learning in Python’, Journal of machine learning research, 12(Oct), pp.2825–2830.

[119] Pehar, J. (2021). 'Utjecaj 5G mreže na razvoj usluga inteligentnih transportnih sustava', Završni rad, Sveučilište u Zagrebu, Fakultet prometnih znanosti, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:119:765637, pristupljeno: 17.05.2022.)

[120] Pejčić, L. (2020): 'Sustav praćenja opskrbnog lanca u automobilskoj industriji zasnovan na tehnologiji lanca blokova', Diplomski rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:200:916397, pristupljeno: 06.06.2022.)

[121] Peričić, D. (2014): n.d – pogovor knjige ‘1984’ Georgea Orwella, Varaždin, 2015., Šareni dućan, Koprivnica

[122] Perish, T. T (2020): 'Infantry wins battles, logistics wins wars – Roseburg VA Supply chain vital during COVID-19 fight', DVIDS, (dostupno na: https://www.dvidshub.net/news/369929/infantry-wins-battles-logistics-wins-wars-roseburg-va-supply-chain-vital-during-covid-19-fight#:~:text=%E2%80%9CInfantry%20wins%20battles%2C%20logistics%20wins,the%20Western%20Front%20during%20WWI., pristupljeno: 14.03.2022.)

[123] Perko, M. (2019): 'Implementacija pametnog ogledala na Raspberry Pi', Završni rad, Sveučilište Jurja Dobrile u Puli, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:137:310549, pristupljeno: 01.06.2022.)

[124] Petar, S. i Matajčić, M. (2021): 'Ekonomična količina nabave', prezentacija kolegija Upravljanje lancima opskrbe, Sveučilište Sjever, Koprivnica

[125] Pickering, J.E., Podsiadly, M. i Burnham, K.J., (2019): 'A model-to-decision approach for the autonomous vehicle (av) ethical dilemma: Av collision with a barrier/pedestrian (s)', IFAC-PapersOnLine, 52(8), pp.257-264. (dostupno na: https://www.researchgate.net/profile/James-Pickering-2/publication/333720358_A_Model-to-Decision_Approach_for_the_Autonomous_Vehicle_AV_Ethical_Dilemma_AV_Collision_with_a_BarrierPedestrians, pristupljeno: 15.06.2022.)

[126] Plan održive urbane mobilnosti grada Koprivnice – SUMP (2015), Grad Koprivnica, (dostupno na: https://koprivnica.hr/wp-content/uploads/2015/08/Plan-odr--ive-urbane-mobilnosti-Grada-Koprivnice-SUMP.pdf, pristupljeno: 18.05.2022.)

[127] Polat, K., Akdemir, B. i Güneş, S. (2008): 'Computer aided diagnosis of ECG data on the least square support vector machine', Digital Signal Processing, 18(1), pp.25-32. (dostupno na: https://www.sciencedirect.com/science/article/abs/pii/S1051200407000929, pristupljeno: 01.07.2022.)

[128] Prister, V. (2019): 'Umjetna inteligencija', Media, culture and public relations, 10(1), str. 67-72.

[129] Puzek, Ž. (2018): 'Etika umjetne inteligencije', Diplomski rad, Sveučilište u Zagrebu, Filozofski fakultet,(dostupno na: https://urn.nsk.hr/urn:nbn:hr:131:795477, pristupljeno 14.06.2022.)

[130] python-visualization (2020): 'Folium' (dostupno na: https://python-visualization.github.io/folium/, pristupljeno: 11.07.2022.)

[131] Redžić, L. (2018): 'Računalo Raspberry Pi', Završni rad, Sveučilište u Zagrebu, Fakultet organizacije i informatike, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:211:196136, pristupljeno: 01.06.2022.)

[132] Rendulić, K. (2022): 'Pametni bicikli i pametni grad - pespektive razvoja', Diplomski rad, Sveučilište u Zagrebu, Ekonomski fakultet (dostupno na: https://urn.nsk.hr/urn:nbn:hr:148:911147, pristupljeno: 17.05.2022.)

[133] Rosyidi, L., Pradityo, H.P., Gunawan, D. i Sari, R.F., (2014) : 'Timebase dynamic weight for Dijkstra Algorithm implementation in route planning software', Iinternational conference on intelligent green building and smart grid (IGBSG) (pp. 1-4). IEEE. (dostupno na: https://www.researchgate.net/profile/Lukman-Rosyidi/publication/269300785_Timebase_dynamic_weight_for_Dijkstra_Algorithm_implementation_in_route_planning_software/links/5959e7f10f7e9ba95e126d30/Timebase-dynamic-weight-for-Dijkstra-Algorithm-implementation-in-route-planning-software.pdf, pristupljeno: 05.07.2022.)

[134] Salton, G., Wong, A. i Yang, C.S., (1975): 'A vector space model for automatic indexing', Communications of the ACM, 18(11), pp.613-620. (dostupno na: https://dl.acm.org/doi/pdf/10.1145/361219.361220, pristupljeno: 30.06.2022.)

[135] Šantić, T. (2017): 'Programski jezik Python', Završni rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:200:374473, pristupljeno: 22.06.2022.)

[136] Šantić, T. (2017): 'Programski jezik Python', Završni rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:200:374473, pristupljeno: 09.07.2022.)

[137] Schneid, F. (2005): 'Napoleon's Conquest of Europe: The War of the Third Coalition', dostupno na: https://archive.org/details/napoleonsconques00fred, pristupljeno: 14.03.2022.)

[138] Seaborn, K. i Fels, D.I., (2015): 'Gamification in theory and action: A survey', International Journal of human-computer studies, 74, pp.14-31.

[139] Searle. J., (1980): 'Minds, brains, and programs', Behavioral and Brain Sciences, vol. 3, no. 3, pp. 417–457, 1980.

[140] Shen, C.F., (2019): 'Social credit system in China', City University of Hong Kong. (dostupno na: https://www.kas.de/documents/288143/4843367/panorama_digital_asia_v3a_Shen.pdf, pristupljeno: 16.06.2022.)

[141] Šimunjak, I. (2021): 'Implementacija blockchain sustava u Pythonu', Završni rad, Sveučilište u Zagrebu, Fakultet organizacije i informatike, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:211:871908, pristupljeno: 06.06.2022.)

[142] Skobelev, P.O. i Borovik, S.Y., (2017): ‘On the way from Industry 4.0 to Industry 5.0: From digital manufacturing to digital society’, Industry 4.0, 2(6), pp.307-311.

[143] Studija razvoja održivog prometa i mobilnosti grada Ludbrega, 2021.

[144] Subirana, B. (n.d.): ‘AI and Machine Learning for Everyone’, Module 1: ‘Introduction to the Artificial Design Process’, MITxPRO module, Emeritus – studijska grupa veljača, 2022.

[145] Tan, J., Yang, J., Wu, S., Chen, G. i Zhao, J., (2021): ‘A critical look at the current train/test split in machine learning’, arXiv preprint arXiv:2106.04525. (dostupno na: https://arxiv.org/pdf/2106.04525.pdf, pristupljeno: 01.07.2022.)

[146] Tipurić, D., Garača, Ž. i Krajnović, A. (2020): 'Univerzalni temeljni dohodak: utopija ili buduća zbilja', Ekonomski pregled, 71(6), str. 632-656, (dostupno na: https://doi.org/10.32910/ep.71.6.4, pristupljeno: 14.06.2022.)

[147] Tomić, K. (2021): 'RFID prepoznavanje pozicije za igre na ploči', Diplomski rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:200:824469, pristupljeno: 06.06.2022.)

[148] Tukey, J. W. (1977): ‘Exploratory data analysis’, Addison-Wesley, Reading, MA, (2-3).

[149] Turudić, D.A., Milić, J. i Štulina, K. (2017): 'Korištenje kriptovaluta u međunarodnom poslovanju', Zbornik sveučilišta Libertas, 1-2(1-2), str. 191-210., (dostupno na: https://hrcak.srce.hr/191294, pristupljeno: 08.06.2022.)

[150] Valerjev, P., (2006): ‘Povijest i perspektiva razvoja umjetne inteligencije u istraživanju uma’

[151] Van Rossum, G. & Drake Jr, F.L., (1995): ‘Python reference manual’, Centrum voor Wiskunde en Informatica Amsterdam.

[152] Varga, M. (2021): ‘Upravljanje podacima‘, Mladen Varga – vlastita naklada, Udžbenik Sveučilišta u Zagrebu, Zagreb

[153] Visa, S., Ramsay, B., Ralescu, A.L. i Van Der Knaap, E. (2011): ‘Confusion matrix-based feature selection’, MAICS, 710(1), pp.120-127.

[154] Vrbanić, F. (2017): 'Inteligentni transportni sustavi u funkciji poštivanja prometnih propisa', Završni rad, Sveučilište u Zagrebu, Fakultet prometnih znanosti, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:119:363098, pristupljeno: 31.05.2022.)

[155] Vukovski, D. (2019): 'Tercijarnologistički sustavi', Završni rad, Sveučilište Sjever, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:122:903609, pristupljeno: 11.04.2022.)

[156] Wai-Ki Ching i Michael K. Ng, (2006): 'Markov Chains: Models, Algorithms and Applications', International Series in Operations Research and Management Science.

[157] Wang, Q., Li, R., Wang, Q. i Chen, S., (2021): ‘Non-fungible token (NFT): Overview, evaluation, opportunities and challenges’, arXiv preprint arXiv:2105.07447., (dostupno na: https://arxiv.org/pdf/2105.07447.pdf?ref=https://githubhelp.com, pristupljeno: 08.06.2022.)

[158] Zekić, Z. (2018): 'Menadžment opskrbnog lanca–suvremeni koncept razvoja kooperativne konkurentnosti', Oeconomica Jadertina, (dostupno na: https://hrcak.srce.hr/clanak/318068, pristupljeno: 01.04.2022.)

[159] Zelenika, R. (2005): ‘Logistički sustavi’, Sveučilište u Rijeci, Ekonomski fakultet Sveučilišta u Rijeci

[160] Zelenika, R. i Pupavac, D., (2008): ‘Menadžment logističkih sustava’, Sveučilište u Rijeci, Ekonomski fakultet Sveučilišta u Rijeci

[161] Zelenika, R., (2000): ‘Metodologija i tehnologija izrade znanstvenog i stručnog djela’, Ekonomski fakultet Sveučilišta u Rijeci, Ekonomska fakulteta Univerze u Ljubljani, Rijeka

[162] Zovko, I. (2017): ‘Sigurnost i računarstvo u oblaku’, Sveučilište u Zagrebu, Filozofski fakultet u Zagrebu, (dostupno na: http://darhiv.ffzg.unizg.hr/id/eprint/8968/, pristupljeno: 02.06.2022.)

[163] Zrnić, M. (2020): 'Bežični NFC čitač', Diplomski rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:200:044953, pristupljeno: 06.06.2022.)

[164] ZSE (2022) - povijesni pregled kretanja CROBEX indeksa Zagrebačke burze, Zagreb, (dostupno na: https://zse.hr/hr/indeks/365?isin=HRZB00ICBEX6&tab=index_history, pristupljeno: 24.05.2022.)

[165] Zuboff, S. (2019): ‘The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power’

[166] Živković, N. (2017): 'Raspberry Pi', Završni rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:126:616288, pristupljeno: 01.06.2022.)

[167] Živković, S. (2018): 'Blockchain tehnologija', Završni rad, Sveučilište u Rijeci, (dostupno na: https://urn.nsk.hr/urn:nbn:hr:195:472651. Pristupljeno: 06.06.2022.)

# Popis slika

# Popis tablica

# Popis grafikona

# Prilozi

**Prilog 1 – Programski kod aplikacije za izračun EOQ pokazatelja *argparse* metodom.**

- izvorno pripada vlastito izrađenim bibliotekama *vandal, duality* i *logistics.*

```
# coloring.
from colorama import Fore, init

init()

# type hints and annotations.
from logistics.plugins.metaclass import Meta

# imports all data types.
from logistics.plugins.types import *

# package.
class EOQ:

    '''

    (OBJECT INFO)
    -------------

    eg. EOQ = vandal.EOQ()

    vandal.EOQ - main class that stores the data required for an EOQ
simulation.
        * takes 3 additional arguments.
            annual_demand_quantity - integer of demanded quantity in
a year.
            order_fixed_cost - integer or float presenting the fixed
cost of the order.
            annual_holding_cost - cost of holding the goods in a
year.

    (OBJECT FUNCTIONS)
    ------------------

    eg. vandal.EOQ.function()

        .execute() - executes the calculation of EOQ over the defined
parameters.
            * takes 1 additional argument.
                filtered (defualt: filtered = True) - filters the
print option and leaves just the return object.

    EOQapp (EXECUTABLE CLI MODULE)
    ------------------------

    vandal.EOQapp is an executable function that runs the Command
Line Interface of the vandal EOQ module.
        print(help(vandal.EOQapp)) in order to see available
features.

    '''
```

```
    # metadata of the used library.
    from vandal.misc._meta import (
        __author__,
        __copyright__,
        __credits__,
        __license__,
        __version__,
        __documentation__,
        __contact__,
        __donate__,
        __APPversion__,
    )

    def __init__(
        self,
        annual_demand_quantity : NumberType = None,
        order_fixed_cost : NumberType = None,
        annual_holding_cost : NumberType = None,
    ) -> ReturnType:

        '''
        * initial launch.
        '''

        self.annual_demand_quantity = annual_demand_quantity
        self.order_fixed_cost = order_fixed_cost
        self.annual_holding_cost = annual_holding_cost

        return

    def __str__(
        self,
        ) -> StringType:

        '''
        * class information.
        '''

        return f'EOQ defining object that contains annual demand
quantity of {self.annual_demand_quantity} pieces, fixed cost of the
order of {self.order_fixed_cost} and with the annual holding cost of
{self.annual_holding_cost}.'

    def __repr__(
        self,
        ) -> StringType:

        '''
        * class information.
        '''

        return f'EOQ defining object that contains annual demand
quantity of {self.annual_demand_quantity} pieces, fixed cost of the
order of {self.order_fixed_cost} and with the annual holding cost of
{self.annual_holding_cost}.'

    def execute(
```

```
        self,
        filtered = True,
        ) -> NumberType:

        '''
        * executes the calculation of EOQ over the defined
parameters.

        - filtered (True/False) - filters the data of EOQ initial
parameters printed.
        # DEFAULT: EOQ.execute(filtered : BooleanType = True.)
        '''

        if filtered == False:
            print(Fore.GREEN + 'EOQ je postavljen za godišnju
količinu nabave od ' + Fore.RESET + f'{self.annual_demand_quantity}'
+ Fore.GREEN + ' komada, uz fiksni trošak narudžbe po jedinici od ' +
Fore.RESET + f'{self.order_fixed_cost}' + Fore.GREEN + ' i troškom
skladištenja od ' + Fore.RESET + f'{self.annual_holding_cost}' +
Fore.GREEN + ' po jedinici.' + Fore.RESET)

        import math
        eoq = math.sqrt((((2 * self.annual_demand_quantity) *
self.order_fixed_cost) / self.annual_holding_cost))
        eoq = round(eoq, 2)

        if filtered == False:
            print(Fore.GREEN + 'Ekonomična količina narudžbe je:',
Fore.RESET, eoq)

        return eoq

# CLI application.
def EOQapp():

    '''
    runs as:

        * IDE: vandal.EOQapp()
        * TERMINAL: python -m vandal -e eoq / python -m vandal --
entry eoq
    '''

    print(Fore.YELLOW + 'EOQ app is initializing...', Fore.RESET)

    # relevant imports.
    import os
    from vandal.misc._meta import (
        __version__,
        __APPversion__,
    )
    os.system('cls')

    # greeting.
    print(Fore.YELLOW + '\n - vandal Command Line Interface
Aplikacija © David Kundih -', __APPversion__)
    print(Fore.YELLOW + ' - vandal verzija paketa (HR verzija) -',
'v',__version__, Fore.RESET, '\n')
```

```
    adq = float(input('Unesi godišnju količinu nabave: '))
    ofc = float(input('Unesi fiksni trošak narudžbe za jedinicu: '))
    ahc = float(input('Unesi godišnji trošak skladištenja po
jedinici: '))

    EOQObj = EOQ(annual_demand_quantity = adq, order_fixed_cost =
ofc, annual_holding_cost = ahc)
    print('')
    res = EOQObj.execute(filtered = False)
    print(Fore.GREEN + 'Optimalan broj narudžbi godišnje: ',
Fore.RESET, adq / res)


# runs module as an app.
if __name__ == '__main__':
    EOQapp()
```

**Prilog 2 - Programski kod aplikacije za izračun EOQ pokazatelja *duality* metodom.**

- izvorno pripada vlastito izrađenim bibliotekama *duality*, *vandal* i *logistics*.

```
# coloring.
from colorama import Fore, init

init()

import duality

app = duality.DualityApp()

# type hints and annotations.
from logistics.plugins.metaclass import Meta

# imports all data types.
from logistics.plugins.types import *

# imports all data types.
from logistics.plugins.coloring import *

# package.
class EOQ:

    '''

    (OBJECT INFO)
    -------------

    eg. EOQ = vandal.EOQ()

    vandal.EOQ - main class that stores the data required for an EOQ
simulation.
        * takes 3 additional arguments.
            annual_demand_quantity - integer of demanded quantity in
a year.
            order_fixed_cost - integer or float presenting the fixed
cost of the order.
            annual_holding_cost - cost of holding the goods in a
year.

    (OBJECT FUNCTIONS)
    ------------------

    eg. vandal.EOQ.function()

        .execute() - executes the calculation of EOQ over the defined
parameters.
            * takes 1 additional argument.
                filtered (defualt: filtered = True) - filters the
print option and leaves just the return object.
```

```
    EOQapp (EXECUTABLE CLI MODULE)
    ------------------------

    vandal.EOQapp is an executable function that runs the Command
Line Interface of the vandal EOQ module.
        print(help(vandal.EOQapp)) in order to see available
features.

    '''

    # metadata of the used library.
    from vandal.misc._meta import (
        __author__,
        __copyright__,
        __credits__,
        __license__,
        __version__,
        __documentation__,
        __contact__,
        __donate__,
        __APPversion__,
    )

    @app.entry(option_name = 'unos parametara', option_description =
'definiranje parametara')
    def __init__(
        self,
        godišnja_količina_nabave : NumberType = app.store(variable =
'godišnja_količina_nabave', type = 'float', overwrite = 'GODIŠNJU
KOLIČINU NABAVE'),
        fiksni_trošak_po_jedinici : NumberType = app.store(variable =
'fiksni_trošak_po_jedinici', type = 'float', overwrite = 'FIKSNI
TROŠAK PO JEDINICI'),
        trošak_skladištenja_po_jedinici : NumberType =
app.store(variable = 'trošak_skladištenja_po_jedinici', type =
'float', overwrite = 'TROŠAK SKLADIŠTENJA PO JEDINICI'),
    ) -> ReturnType:

        '''
        * initial launch.
        '''

        self.annual_demand_quantity = godišnja_količina_nabave
        self.order_fixed_cost = fiksni_trošak_po_jedinici
        self.annual_holding_cost = trošak_skladištenja_po_jedinici

        return

    def __str__(
        self,
        ) -> StringType:

        '''
        * class information.
        '''

        return f'EOQ defining object that contains annual demand
quantity of {self.annual_demand_quantity} pieces, fixed cost of the
```

```
order of {self.order_fixed_cost} and with the annual holding cost of
{self.annual_holding_cost}.'

    def __repr__(
        self,
        ) -> StringType:

        '''
        * class information.
        '''

        return f'EOQ defining object that contains annual demand
quantity of {self.annual_demand_quantity} pieces, fixed cost of the
order of {self.order_fixed_cost} and with the annual holding cost of
{self.annual_holding_cost}.'

    @app.entry(option_name = 'izračunaj', option_description =
'pokretanje izračuna.', print_val = True)
    def execute(
        self,
        filtered = True,
        ) -> NumberType:

        '''
        * executes the calculation of EOQ over the defined
parameters.

        - filtered (True/False) - filters the data of EOQ initial
parameters printed.
        # DEFAULT: EOQ.execute(filtered : BooleanType = True.)
        '''

        if filtered == False:
            print(Fore.GREEN + 'EOQ je postavljen za godišnju
količinu nabave od ' + Fore.RESET + f'{self.annual_demand_quantity}'
+ Fore.GREEN + ' komada, uz fiksni trošak narudžbe po jedinici od ' +
Fore.RESET + f'{self.order_fixed_cost}' + Fore.GREEN + ' i troškom
skladištenja od ' + Fore.RESET + f'{self.annual_holding_cost}' +
Fore.GREEN + ' po jedinici.' + Fore.RESET)

        import math
        eoq = math.sqrt((((2 * self.annual_demand_quantity) *
self.order_fixed_cost) / self.annual_holding_cost))
        eoq = round(eoq, 2)

        if filtered == False:
            print(Fore.GREEN + 'Ekonomična količina narudžbe je:',
Fore.RESET, eoq)

        self.eoq = eoq
        return eoq

    @app.entry(option_name = 'optimalna narudžba', option_description
= 'prikazuje optimalnu godišnju narudžbu.')
    def optimal_order(self):
        opt_ord = self.annual_demand_quantity / self.eoq
        return print('Optimalan broj narudžbi godišnje:', opt_ord)
```

```
# runs module as an app.

if __name__ == '__main__':
    print('\n')
    app.wheel(type = 'dynamic', display_headline = 'RASPOLOŽIVE
OPCIJE', display_message = 'UNESI OPCIJU: ', output_message =
'ODABRAO SI: ', exit_message = 'GASIM...', enter_message = 'UNESI ',
break_key = 'izlaz')
```

**Prilog 3 – Programski kod duality.DualityApp objekta**

- izvorno pripada vlastito izrađenoj biblioteci *duality*.

```
# makes multiple instances of the object available.
from logistics.plugins.metaclass import Meta

# for CLI functionality.
import os
from colorama import (
    Fore,
    Back,
    Style,
    init,
)

init()

# imports all data types.
from logistics.plugins.types import *

# imports coloring.
from logistics.plugins.coloring import *

# package.
class DualityApp(metaclass = Meta):

    '''
    * stores menu options over functions and class methods for
listing.

    COLORSET (custom_color_mode template)
    ________

    colorset = {
        'credit' : 'Fy', - credit pop-up color.
        'display_headline' : 'Fy', - headline of the menu color.
        'display_message' : 'Fc', - enter the option color.
        'output_message' : 'Fy', - confirmation of choice color
(DEPRECATED).
        'enter_message' : 'Fc', - input variables color.
        'tmp_name_list' : 'Fy', - headline of chosen option color.
        'tmp_func' : 'Fc', - output of the chosen option color.
        'warning' : 'Fr', - general warning and error handling color.
        'exit_message' : 'Fy', - exit out of the application color.
    }
    '''

    def __init__(
        self,
        credit : StringType = 'duality © David Kundih, 2021 -',
        include : ListType = ['menu', 'cls', 'exit'],
        ) -> ReturnType:

        '''
        * initializes the object and function it is decorating.
```

```
        credit - headline credit of the application.
        include - includes pre-built options for menu, clear screen
and exit.

        # DEFAULT: DualityApp(credit : StringType = 'duality © David
Kundih, 2021 -', include : ListType = ['menu', 'cls', 'exit'])
        '''

        self.credit : StringType = credit
        self.include : ListType = include
        self.basic_menu : ListType = []
        self.descriptive_menu : ListType = []
        self.dictionary_menu : DictionaryType = {}
        self.individual_dict : DictionaryType = {}
        self.reset_dict : DictionaryType = {}
        self.poolsize : IntegerType = 0
        self.stored_keys : ListType = []
        self.print_val_dict : ListType = {}
        self.option_names : ListType = []
        self.overwrite_variable : DictionaryType = {}
        self.overwrite_types : DictionaryType = {}
        self.verbose_display_message_list : ListType = []
        self.verbose_set : ReturnType = None # prevents duplicate
verbose option list creation.

        self.hidden_basic_menu : ListType = []
        self.hidden_descriptive_menu : ListType = []
        self.hidden_dictionary_menu : DictionaryType = {}
        self.contains_autoinit : BooleanType = False

    def entry(
        self,
        option_name : StringType = '',
        option_description : StringType = '',
        autoinit: BooleanType = False,
        print_val: BooleanType = False,
        ) -> StringDictionary:

        '''
        * creates and entry that is stored in a basic menu,
descriptive menu and a dictionary menu.

        - option_name - stores the name for the display of functions.
        - option_description - stores the description of the function
for display functions.
        - autoinit (True/False) - automatically initializes the
function without storing into the dictionary menu.
        - print_val (True/False) - enables the print of function
output.
        # DEFAULT: DualityApp.entry(option_name : StringType = '',
option_description : StringType = '', autoinit : BooleanType = False,
print_val : BooleanType = False).
        '''

        self.option_name = option_name
        self.option_names += [self.option_name]
        self.option_description = option_description
        self.dict_name = self.option_name
```

```
        self.verbose_display_message_list += [self.option_name]
        self.print_val = print_val
        self.individual_dict[self.dict_name] = {}
        self.reset_dict[self.dict_name] = {}
        self.overwrite_variable[self.dict_name] = {} # store
redefine.
        self.overwrite_types[self.dict_name] = {} # store redefine.
        self.print_val_dict[self.option_name] = self.print_val

        def _record_function(func):

            if autoinit == False:
                self.dictionary_menu[self.option_name] = func
                self.basic_menu += [self.option_name]
                self.descriptive_function = str(self.option_name) +
str(paint_text(' -> ', color = 'Fy', print_trigger = False)) +
str(self.option_description)
                self.descriptive_menu += [self.descriptive_function]

            elif autoinit == True:
                self.hidden_dictionary_menu[self.option_name] = func
                self.hidden_basic_menu += [self.option_name]
                self.descriptive_function = str(self.option_name) +
str(paint_text(' -> ', color = 'Fy', print_trigger = False)) +
str(self.option_description)
                self.hidden_descriptive_menu +=
[self.descriptive_function]
                self.contains_autoinit = True

            def _wrap(*args, **kwargs):
                return func(*args, **kwargs)

            return _wrap

        return _record_function

    def display(
        self,
        style : StringType = 'decorator',
        method : StringType = 'dictionary',
        return_option : StringType = 'logs',
        ) -> StringDictionary:

        '''
        * outputs saved menu as a function or decorator.

        - style ('decorator' - appends to a function.)
        - style ('function' - executes as a standalone function.)
        - method ('basic' - shows just DualityApp.entry stored
names.)
        - method ('descriptive' - shows DualityApp.entry stored names
and DualityApp.entry.option_description as a help menu.)
        - method ('dictionary' - creates a dictionary of
DualityApp.entry names and the function they are appended on.)
        - return_option ('logs' - executes the function, but returns
logs.)
        - return_option ('function' - shows logs, but returns the
function.)
```

```
        # DEFAULT: DualityApp.display(style : StringType =
'decorator', method : StringType = 'dictionary', return_option :
StringType = 'logs'.)
        '''

        if style == 'decorator':
            if return_option == 'logs':
                if method == 'basic':
                    def _wrapper(func):
                        def _decorator(self,*args, **kwargs):
                            func(*args, **kwargs)
                            return self.basic_menu
                        return _decorator

                    return _wrapper

                elif method == 'descriptive':
                    def _wrapper(func):
                        def _decorator(self,*args, **kwargs):
                            func(*args, **kwargs)
                            return self.descriptive_menu
                        return _decorator

                    return _wrapper

                elif method == 'dictionary':
                    def _wrapper(func):
                        def _decorator(self, *args, **kwargs):
                            func(*args, **kwargs)
                            return self.dictionary_menu
                        return _decorator

                    return _wrapper

            elif return_option == 'function':
                if method == 'basic':
                    def _wrapper(func):
                        def _decorator(self, *args, **kwargs):
                            print(self.basic_menu)
                            return func(*args, **kwargs)
                        return _decorator

                    return _wrapper

                elif method == 'descriptive':
                    def _wrapper(func):
                        def _decorator(self, *args, **kwargs):
                            print(self.descriptive_menu)
                            return func(*args, **kwargs)
                        return _decorator

                    return _wrapper

                elif method == 'dictionary':
                    def _wrapper(func):
                        def _decorator(self, *args, **kwargs):
                            print(self.dictionary_menu)
                            return func(*args, **kwargs)
```

```
                        return _decorator

                    return _wrapper

        elif style == 'function':
            if method == 'basic':
                return self.basic_menu

            elif method == 'descriptive':
                return self.descriptive_menu

            elif method == 'dictionary':
                return self.dictionary_menu

    def script(
        self,
        type : StringType = 'dynamic',
        display_headline : StringType ='AVAILABLE OPTIONS',
        verbose_display_message : BooleanType = True,
        display_message : StringType = 'ENTER THE OPTION ',
        output_message : StringType = 'YOU HAVE CHOSEN: ',
        break_key : StringType = 'exit',
        exit_message : StringType = 'Exiting...',
        enter_message : StringType = 'ENTER THE ',
        method : StringType = 'descriptive',
        alignment : StringType = 'newline',
        queue: StringType = 'y',
        show_dtypes: BooleanType = True,
        show_confirmation : BooleanType = False,
        color_mode : StringType = 'dark',
        custom_color_mode : DictionaryType = None,
        clear_screen : BooleanType = True,
        ) -> SpecialType:

        '''
        * creates an executable menu from defined entries on top of
functions.

        - type ('static' - adapts to the execution of static non-self
methods and functions.)
        - type ('dynamic' - adapts to the execution of an object
class self methods and functions as dynamic.)
        - display_headline - displays the desired headline.
        - verbose_display_message (True/False) - displays all menu
options next to the input option request.
        - display_message - displays input value message.
        - output_message - confirmation of the chosen value.
        - break_key - key that breaks the loop while queue = 'y'.
        - exit_message - message while exiting.
        - enter_message - enter choice message.
        - method ('descriptive' - shows stored option_name and it's
description.)
        - method ('basic' - shows only the stored option_name.)
        - alignment ('basic' - shows all stored option_name and
option_description values in a row.)
        - alignment ('newline' -shows all stored option_name and
option_description values in a new line.)
```

```
        - queue (DO NOT CHANGE!) - enables stacking of functions and
executing them in a chain.
        - show_dtypes (True/False) - shows the dtype of the input
value.
        - show_confirmation (True/False) - confirmation of the chosen
option.
        - color_mode ('dark' - for dark terminal.)
        - color_mode ('light' - for light terminal.)
        - custom_color_mode - custom dictionary set of colors.
        - clear_screen (True/False) - clears the screen before
starting the app

        # DEFAULT: DualityApp.script(
        type : StringType = 'dynamic',
        display_headline : StringType ='AVAILABLE OPTIONS',
        verbose_display_message : BooleanType = True,
        display_message : StringType = 'ENTER THE OPTION ',
        output_message : StringType = 'YOU HAVE CHOSEN: ',
        break_key : StringType = 'exit',
        exit_message : StringType = 'Exiting...',
        enter_message : StringType = 'ENTER THE ',
        method : StringType = 'descriptive',
        alignment : StringType = 'newline',
        queue: StringType = 'y',
        show_dtypes: BooleanType = True,
        show_confirmation : BooleanType = False,
        color_mode : StringType = 'dark',
        custom_color_mode : DictionaryType = None,
        clear_screen : BooleanType = True
        ) -> SpecialType:
        '''

        self.clear_screen = clear_screen

        if self.clear_screen == True:
            os.system('cls')

        self.color_mode = color_mode # dark or ligth appearance.
        self.custom_color_mode = custom_color_mode # option to
introduce own color dictionary.

        # set up coloring in the CLI.
        if self.color_mode == 'dark':
            self.colorset = {
                'credit' : 'Fy',
                'display_headline' : 'Fy',
                'display_message' : 'Fc',
                'output_message' : 'Fy',
                'enter_message' : 'Fc',
                'tmp_name_list' : 'Fy',
                'tmp_func' : 'Fc',
                'warning' : 'Fr',
                'exit_message' : 'Fy',
            }

        elif self.color_mode == 'light':
            self.colorset = {
                'credit' : 'Fg',
```

```
                'display_headline' : 'Fg',
                'display_message' : 'Fb',
                'output_message' : 'Fg',
                'enter_message' : 'Fb',
                'tmp_name_list' : 'Fg',
                'tmp_func' : 'Fb',
                'warning' : 'Fr',
                'exit_message' : 'Fg',
            }

        elif self.color_mode == 'custom':
            self.colorset = self.custom_color_mode

        self.break_key = break_key

        self.display_headline = display_headline
        self.verbose_display_message = verbose_display_message #
displays all menu options next to the input option request.

        if 'menu' in self.include:
            if 'menu' not in self.option_names:
                @self.entry('menu', 'shows the menu.')
                def _show_menu(self):
                    paint_text(self.credit, self.colorset['credit'],
print_trigger = True)
                    print('')
                    show_menu = self.display(style = 'function',
method = 'descriptive')
                    paint_text(self.display_headline,
self.colorset['display_headline'])
                    print('-----------------')
                    for line in show_menu:
                        print(line)
                    if 'exit' in self.include:
                        print(str(self.break_key) + str(paint_text('
-> ', color = 'Fr', print_trigger = False)) + 'exit.')

        if 'cls' in self.include:
            if 'cls' not in self.option_names:
                @self.entry('cls', 'clears the screen.')
                def _clear_screen(self):
                    os.system('cls')
                    paint_text(self.credit, self.colorset['credit'],
print_trigger = True)
                    print('')
                    show_menu = self.display(style = 'function',
method = 'descriptive')
                    paint_text(self.display_headline,
self.colorset['display_headline'])
                    print('-----------------')
                    for line in show_menu:
                        print(line)
                    if 'exit' in self.include:
                        print(str(self.break_key) + str(paint_text('
-> ', color = 'Fr', print_trigger = False)) + 'exit.')

        if self.break_key not in self.verbose_display_message_list:
            self.verbose_display_message_list.append(self.break_key)
```

```
        if self.verbose_set != True:
            if self.verbose_display_message == True:
                self.verbose_set = True
                self.display_message = paint_text(display_message +
str(self.verbose_display_message_list) + ': ',
self.colorset['display_message'], print_trigger = False)
            elif self.verbose_display_message == False:
                self.display_message = paint_text(display_message,
self.colorset['display_message'], print_trigger = False)
        else:
            self.display_message = paint_text(display_message,
self.colorset['display_message'], print_trigger = False)

        self.output_message = paint_text(output_message,
self.colorset['output_message'], print_trigger = False)
        self.enter_message = paint_text(enter_message,
self.colorset['enter_message'], print_trigger = False)
        self.queue = queue
        self.show_dtypes = show_dtypes
        self.yield_name = 0 # list item counter that enables
iterating through the list.
        self.show_confirmation = show_confirmation # confirmation of
chosen option.
        self.exit_message = exit_message # message while exiting.

        # assert deprecated warning.
        if self.queue == 'n':
            print('WARNING: Option is deprecated.')
            print('Write type = \'script\' or type = \'wheel\' to
change how this impacts the behaviour of executed functions in the
menu.')
            print('')

        if alignment == 'basic':

            if method == 'basic':
                paint_text(self.credit, self.colorset['credit'],
print_trigger = True)
                print('')
                show_menu = self.display(style = 'function', method =
'basic')
                paint_text(self.display_headline,
self.colorset['display_headline'])
                print('-----------------')
                print(show_menu)
                if 'exit' in self.include:
                    print(str(self.break_key) + str(paint_text(' ->
', color = 'Fr', print_trigger = False)) + 'exit.')
                print('')

            elif method == 'descriptive':
                paint_text(self.credit, self.colorset['credit'],
print_trigger = True)
                print('')
                show_menu = self.display(style = 'function', method =
'descriptive')
```

```
                paint_text(self.display_headline,
self.colorset['display_headline'])
                print('-----------------')
                print(show_menu)
                if 'exit' in self.include:
                    print(str(self.break_key) + str(paint_text(' ->
', color = 'Fr', print_trigger = False)) + 'exit.')
                print('')

            elif method == 'none':
                pass

        elif alignment == 'newline':

            if method == 'basic':
                paint_text(self.credit, self.colorset['credit'],
print_trigger = True)
                print('')
                show_menu = self.display(style = 'function', method =
'basic')
                paint_text(self.display_headline,
self.colorset['display_headline'])
                print('-----------------')
                for line in show_menu:
                    print(line)
                if 'exit' in self.include:
                    print(str(self.break_key) + str(paint_text(' ->
', color = 'Fr', print_trigger = False)) + 'exit.')
                print('')

            elif method == 'descriptive':
                paint_text(self.credit, self.colorset['credit'],
print_trigger = True)
                print('')
                show_menu = self.display(style = 'function', method =
'descriptive')
                paint_text(self.display_headline,
self.colorset['display_headline'])
                print('-----------------')
                for line in show_menu:
                    print(line)
                if 'exit' in self.include:
                    print(str(self.break_key) + str(paint_text(' ->
', color = 'Fr', print_trigger = False)) + 'exit.')
                print('')

            elif method == 'none':
                pass

        if queue == 'n':

            self.option = input(self.display_message)
            self.print_option = self.print_val_dict[self.option]

            if self.output_message == True:
                print(self.output_message, self.option)

            print('')
```

```
            # executes autoinit function.
            if self.contains_autoinit == True:

                try:
                    for i in self.hidden_dictionary_menu:
                        self.hidden_dictionary_menu[i](self)
                except:
                    for i in self.hidden_dictionary_menu:
                        self.hidden_dictionary_menu[i]()

            # disables a loop of functions.

            self._queue_break()

            if type == 'static':

                for tmp_func in self.tmp_list:
                    self.clone_dict =
self.tmp_name_list[self.yield_name]
                    self.print_option =
self.tmp_print_list[self.yield_name]
                    print(self.tmp_name_list[self.yield_name])

                    self._redefine()

                    try:
                        if self.print_option == False:

tmp_func(**self.individual_dict[self.clone_dict])
                            print('')
                            self.yield_name += 1
                        else:

print(tmp_func(**self.individual_dict[self.clone_dict]))
                            print('')
                            self.yield_name += 1
                    except:
                        if self.print_option == False:
                            tmp_func(self,
**self.individual_dict[self.clone_dict])
                            print('')
                            self.yield_name += 1
                        else:
                            print(tmp_func(self,
**self.individual_dict[self.clone_dict]))
                            print('')
                            self.yield_name += 1

            elif type == 'dynamic':

                    for tmp_func in self.tmp_list:
                        self.clone_dict =
self.tmp_name_list[self.yield_name]
                        self.print_option =
self.tmp_print_list[self.yield_name]
                        print(self.tmp_name_list[self.yield_name])
```

```
                        self._redefine()

                        try:
                            if self.print_option == False:
                                tmp_func(self,
**self.individual_dict[self.clone_dict])
                                print('')
                                self.yield_name += 1
                            else:
                                print(tmp_func(self,
**self.individual_dict[self.clone_dict]))
                                print('')
                                self.yield_name += 1
                        except:
                            if self.print_option == False:

tmp_func(**self.individual_dict[self.clone_dict])
                                print('')
                                self.yield_name += 1
                            else:

print(tmp_func(**self.individual_dict[self.clone_dict]))
                                print('')
                                self.yield_name += 1

        # THIS OPTION IS ONLY FOR INTERNAL USE, DO NOT ATTEMPT TO USE
IT OUTSIDE OF THE SCRIPT OR WHEEL METHODS.
        if queue == 'y':
            # executes autoinit functions.
            if self.contains_autoinit == True:
                try:
                    for i in self.hidden_dictionary_menu:
                        self.hidden_dictionary_menu[i](self)
                except:
                    for i in self.hidden_dictionary_menu:
                        self.hidden_dictionary_menu[i]()

            # enables a loop to execute functions in a chain.
            self._queue_handler()

            if type == 'static':

                for tmp_func in self.tmp_list:
                    self.clone_dict =
self.tmp_name_list[self.yield_name]
                    self.print_option =
self.tmp_print_list[self.yield_name]
                    paint_text(self.tmp_name_list[self.yield_name],
self.colorset['tmp_name_list'])

                    self._redefine()

                    try:
                        if self.print_option == False:
                            tmp_func(self,
**self.individual_dict[self.clone_dict])
                            print('')
                            self.yield_name += 1
```

```
                        else:
                            paint_text(tmp_func(self,
**self.individual_dict[self.clone_dict]), self.colorset['tmp_func'])
                            print('')
                            self.yield_name += 1
                    except:
                        if self.print_option == False:

tmp_func(**self.individual_dict[self.clone_dict])
                            print('')
                            self.yield_name += 1
                        else:

paint_text(tmp_func(**self.individual_dict[self.clone_dict]),
self.colorset['tmp_func'])
                            print('')
                            self.yield_name += 1

                paint_text(self.exit_message,
self.colorset['exit_message'], print_trigger = True)

            elif type == 'dynamic':

                for tmp_func in self.tmp_list:
                    self.clone_dict =
self.tmp_name_list[self.yield_name]
                    self.print_option =
self.tmp_print_list[self.yield_name]
                    paint_text(self.tmp_name_list[self.yield_name],
self.colorset['tmp_name_list'])

                    self._redefine()

                    try:
                        if self.print_option == False:

tmp_func(**self.individual_dict[self.clone_dict])
                            print('')
                            self.yield_name += 1
                        else:

paint_text(tmp_func(**self.individual_dict[self.clone_dict]),
self.colorset['tmp_func'])
                            print('')
                            self.yield_name += 1
                    except:
                        if self.print_option == False:
                            tmp_func(self,
**self.individual_dict[self.clone_dict])
                            print('')
                            self.yield_name += 1
                        else:
                            paint_text(tmp_func(self,
**self.individual_dict[self.clone_dict]), self.colorset['tmp_func'])
                            print('')
                            self.yield_name += 1
```

```
                paint_text(self.exit_message,
self.colorset['exit_message'], print_trigger = True)

        if queue == 'wheel' or queue == 'w':

            # executes autoinit functions.
            if self.contains_autoinit == True:
                try:
                    for i in self.hidden_dictionary_menu:
                        self.hidden_dictionary_menu[i](self)
                except:
                    for i in self.hidden_dictionary_menu:
                        self.hidden_dictionary_menu[i]()

            # enables a loop to execute functions in a wheel loop.
            self._wheel_queue_handler()

            if type == 'dynamic':

                for tmp_func in self.tmp_list:
                    self.clone_dict =
self.tmp_name_list[self.yield_name]
                    self.print_option =
self.tmp_print_list[self.yield_name]
                    paint_text(self.tmp_name_list[self.yield_name],
self.colorset['tmp_name_list'])

                    self._redefine()

                    try:
                        if self.print_option == False:
                            tmp_func(self,
**self.individual_dict[self.clone_dict])
                            print('')
                            self.yield_name += 1
                        else:
                            paint_text(tmp_func(self,
**self.individual_dict[self.clone_dict]), self.colorset['tmp_func'])
                            print('')
                            self.yield_name += 1
                    except:
                        if self.print_option == False:

tmp_func(**self.individual_dict[self.clone_dict])
                            print('')
                            self.yield_name += 1
                        else:

paint_text(tmp_func(**self.individual_dict[self.clone_dict]),
self.colorset['tmp_func'])
                            print('')
                            self.yield_name += 1

            if type == 'static':

                for tmp_func in self.tmp_list:
                    self.clone_dict =
self.tmp_name_list[self.yield_name]
```

```
                    self.print_option =
self.tmp_print_list[self.yield_name]
                    paint_text(self.tmp_name_list[self.yield_name],
self.colorset['tmp_name_list'])

                    self._redefine()

                    try:
                        if self.print_option == False:
                            tmp_func(self,
**self.individual_dict[self.clone_dict])
                            print('')
                            self.yield_name += 1
                        else:
                            paint_text(tmp_func(self,
**self.individual_dict[self.clone_dict]), self.colorset['tmp_func'])
                            print('')
                            self.yield_name += 1
                    except:
                        if self.print_option == False:

tmp_func(**self.individual_dict[self.clone_dict])
                            print('')
                            self.yield_name += 1
                        else:

paint_text(tmp_func(**self.individual_dict[self.clone_dict]),
self.colorset['tmp_func'])
                            print('')
                            self.yield_name += 1

        return

    # wheel application.
    def wheel(
        self,
        type = 'dynamic',
        display_headline : StringType ='AVAILABLE OPTIONS',
        verbose_display_message : BooleanType = True,
        display_message : StringType = 'ENTER THE OPTION ',
        output_message : StringType = 'YOU HAVE CHOSEN: ',
        enter_message : StringType = 'ENTER THE ',
        color_mode : StringType = 'dark',
        custom_color_mode : DictionaryType = None,
        clear_screen = True,
        break_key = 'exit',
        exit_message = 'Exiting...'
        ) -> SpecialType:

        '''
        Limited changing capabilities, for more information about
method variables examine script method.

        # DEFAULT: DualityApp.wheel(
        type = 'dynamic',
        display_headline : StringType ='AVAILABLE OPTIONS',
        verbose_display_message : BooleanType = True,
        display_message : StringType = 'ENTER THE OPTION ',
```

```
        output_message : StringType = 'YOU HAVE CHOSEN: ',
        enter_message : StringType = 'ENTER THE ',
        color_mode : StringType = 'dark',
        custom_color_mode : DictionaryType = None,
        clear_screen = True,
        break_key = 'exit',
        exit_message = 'Exiting...'
        ) -> SpecialType:
        '''

        self.clear_screen = clear_screen
        self.display_headline = display_headline
        self.type = type

        if self.clear_screen == True:
            os.system('cls')

        self.color_mode = color_mode # dark or ligth appearance.
        self.custom_color_mode = custom_color_mode # option to
introduce own color dictionary.

        # set up coloring in the CLI.
        if self.color_mode == 'dark':
            self.colorset = {
                'credit' : 'Fy',
                'display_headline' : 'Fy',
                'display_message' : 'Fc',
                'output_message' : 'Fy',
                'enter_message' : 'Fc',
                'tmp_name_list' : 'Fy',
                'tmp_func' : 'Fc',
                'warning' : 'Fr',
                'exit_message' : 'Fy',
            }

        elif self.color_mode == 'light':
            self.colorset = {
                'credit' : 'Fg',
                'display_headline' : 'Fg',
                'display_message' : 'Fb',
                'output_message' : 'Fg',
                'enter_message' : 'Fb',
                'tmp_name_list' : 'Fg',
                'tmp_func' : 'Fb',
                'warning' : 'Fr',
                'exit_message' : 'Fg',
            }

        elif self.color_mode == 'custom':
            self.colorset = self.custom_color_mode

        self.display_message = display_message # basic display
message.
        self.verbose_display_message = verbose_display_message #
displays all menu options next to the input option request.

        self.output_message = output_message
        self.enter_message = enter_message
```

```
        self.break_key = break_key
        self.exit_message = exit_message

        paint_text(self.credit, self.colorset['credit'],
print_trigger = True)
        print('')
        show_menu = self.display(style = 'function', method =
'descriptive')
        paint_text(self.display_headline,
self.colorset['display_headline'])
        print('-----------------')
        for line in show_menu:
            print(line)
        if 'menu' in self.include:
            print('menu' + str(paint_text(' -> ', color = 'Fy',
print_trigger = False)) + 'shows the menu.')
        if 'cls' in self.include:
            print('cls' + str(paint_text(' -> ', color = 'Fy',
print_trigger = False)) + 'clears the screen.')
        if 'exit' in self.include:
            print(str(self.break_key) + str(paint_text(' -> ', color
= 'Fr', print_trigger = False)) + 'exit.')
        print('')

        while True:
            if self.type == 'dynamic':
                self.script(type = 'dynamic', queue = 'wheel', method
= 'none', clear_screen = False, verbose_display_message =
self.verbose_display_message, display_message = self.display_message,
output_message = self.output_message, color_mode = self.color_mode,
custom_color_mode = self.custom_color_mode, break_key =
self.break_key, enter_message = self.enter_message, exit_message =
self.exit_message)
            if self.type == 'static':
                self.script(type = 'static', queue = 'wheel', method
= 'none', clear_screen = False, verbose_display_message =
self.verbose_display_message, display_message = self.display_message,
output_message = self.output_message, color_mode = self.color_mode,
custom_color_mode = self.custom_color_mode, break_key =
self.break_key, enter_message = self.enter_message, exit_message =
self.exit_message)

    def store(
        self,
        variable : StringType = '',
        type : StringType = 'str',
        overwrite : StringType = None,
        ) -> DictionaryType:

        '''
        * function that stores the input value in a dictionary.

        - variable - name of the function argument input is being
passed as.
        - type - type of the data being passed ('int', 'float', 'str'
and 'list' supported.)
        - overwrite - custom text that overwrites the variable name
when shown in the CLI.
```

```
        # DEFAULT: DualityApp.store(variable : StringType = '', type
: StringType = 'str', overwrite : StringType = None.)
        '''

        if overwrite == None:
            overwrite = variable

        self.type = type
        self.variable = variable
        self.overwrite = overwrite
        self.individual_dict[self.dict_name][self.variable] =
self.type
        self.reset_dict[self.dict_name][self.variable] = self.type
        self.overwrite_variable[self.dict_name][self.variable] =
self.overwrite
        self.overwrite_types[self.dict_name][self.overwrite] =
self.type

        return self.dict_name

    # this is a help function, do not call it when using a package.
    def _redefine(
        self,
        ) -> DictionaryType:

        '''
        * function that casts an input of a certain data type and
formats it before sending as a function argument.
        '''

        if self.show_dtypes == True:
            if self.overwrite_types[self.clone_dict]:
                print(self.overwrite_types[self.clone_dict])

            elif self.reset_dict[self.clone_dict]:
                print(self.reset_dict[self.clone_dict])

        for i in self.individual_dict[self.clone_dict]:
            self.tmp_msg =
self.overwrite_variable[self.clone_dict][i]
            self.format = self.reset_dict[self.clone_dict][i]
            if self.format != 'list':
                self.new_i = input(self.enter_message +
paint_text(f'{self.tmp_msg}: ', color =
self.colorset['enter_message'], print_trigger = False))
            self.dtypes = {
            'int': self._set_int,
            'float': self._set_float,
            'str': self._set_str,
            'list' : self._set_list,
        }
            self.new_i = self.dtypes[self.format]()
            self.individual_dict[self.clone_dict][i] = self.new_i

        return self.individual_dict[self.clone_dict]


    # this is a help function, do not call it when using a package.
```

```
    def _set_int(
        self,
        ) -> ReturnType:

        '''
        * converts input to integer.
        '''

        self.new_i = int(self.new_i)
        return self.new_i

    # this is a help function, do not call it when using a package.
    def _set_float(
        self,
        ) -> ReturnType:

        '''
        * converts input to float.
        '''

        self.new_i = float(self.new_i)
        return self.new_i

    # this is a help function, do not call it when using a package.
    def _set_str(
        self,
        ) -> ReturnType:

        '''
        * converts input to string.
        '''

        self.new_i = str(self.new_i)
        return self.new_i

    # this is a help function, do not call it when using a package.
    def _set_list(
        self,
        ) -> ReturnType:

        '''
        * converts input to list.
        '''

        self.range = int(input('Number of list values: '))
        self.new_i = []

        for i in range(0, self.range):
            tmp_list_element = int(input(f'Enter the value: '))
            self.new_i.append(tmp_list_element)

        return self.new_i

    # this is a help function, do not call it when using a package.
(SCRIPT)
    def _queue_handler(
        self,
        iterate: BooleanType = True,
```

```
        ) -> ListType:

        '''
        * enables a loop for the queue.

        - iterate (True/False) - enables the functionality of queue,
do not change!
        '''

        self.print_val_list = []
        self.tmp_list = []
        self.iterate = iterate
        self.tmp_name_list = []
        self.tmp_print_list = []
        self.script_error_input = []

        while self.iterate == True:

            self.option = input(self.display_message)

            if 'exit' in self.include:
                if self.option == self.break_key:
                    print('')
                    break

            while not self.option in self.option_names:

                self.option = input(paint_text('INVALID OPTION, ENTER
THE OPTION: ', self.colorset['warning'], print_trigger = False))

                if self.option == self.break_key:
                    self.script_error_input += [self.option]
                    break

            if self.option not in self.script_error_input:
                self.tmp_name_list += [self.option]
                self.print_val_list += self.option

                if self.show_confirmation == True:
                    print(self.output_message, self.option + '\n')

                self.tmp_list += [self.dictionary_menu[self.option]]
                self.tmp_print_list +=
[self.print_val_dict[self.option]]

        return self.tmp_list

    # this is a help function, do not call it when using a package.
(WHEEL)
    def _wheel_queue_handler(
        self,
        iterate: BooleanType = True,
        ) -> ListType:

        '''
        * enables a loop for the queue.
```

```
        - iterate (True/False) - enables the functionality of queue,
do not change!
        '''

        self.print_val_list = []
        self.tmp_list = []
        self.iterate = iterate
        self.tmp_name_list = []
        self.tmp_print_list = []
        self.wheel_error_input = []

        self.option = input(self.display_message)

        if 'exit' not in self.include:
            self.break_key = None

        if self.option == self.break_key:
            print('')
            paint_text(self.exit_message,
self.colorset['exit_message'], print_trigger = True)
            exit()

        while not self.option in self.option_names:

            if self.option == self.break_key:
                self.wheel_error_input += [self.option]
                break

            self.option = input(paint_text('INVALID OPTION, ENTER THE
OPTION: ', self.colorset['warning'], print_trigger = False))

        if self.option not in self.wheel_error_input:
            print('')
            self.tmp_name_list += [self.option]
            self.print_val_list += self.option

            if self.show_confirmation == True:
                print(self.output_message, self.option + '\n')

            self.tmp_list += [self.dictionary_menu[self.option]]
            self.tmp_print_list += [self.print_val_dict[self.option]]

        return self.tmp_list

    # this is a help function, do not call it when using a package.
(NO QUEUE)
    def _queue_break(
        self,
        ) -> ListType:

        '''
        * breaks the loop for the queue.
        '''

        self.print_val_list = []
        self.tmp_list = []
        self.tmp_name_list = []
        self.tmp_print_list = []
```

```
self.tmp_name_list += [self.option]
self.print_val_list += self.option

if self.show_confirmation == True:
    print(self.output_message, self.option + '\n')

self.tmp_list += [self.dictionary_menu[self.option]]
self.tmp_print_list += [self.print_val_dict[self.option]]

return self.tmp_list
```

**Prilog 4 – Biblioteka izvedenih tipova podataka**

- izvorno pripada vlastito izrađenoj biblioteci *logistics*.

```
# dependencies import.
from typing import Dict, List, Union

# prototype.
VandalType = 'Specific Input / Output vandal type of supported data.'

# generic types.
IntegerType: VandalType = int # integer.
FloatType: VandalType = float # float.
NumberType: VandalType = Union[IntegerType, FloatType] # both floats
and integers supported.
ReturnType: VandalType = object # basic return object or empty
return.
PrintType: VandalType = str # print function as an output.
GraphType: VandalType = object # graph return.
StringType: VandalType = str # string.
ListType: VandalType = list # list.
TupleType: VandalType = tuple # tuple.
DictionaryType: VandalType = dict # dictionary.
BooleanType: VandalType = bool # bool.
FilePathType: VandalType = 'File path.' # file path.
SpecialType: VandalType = 'Special data type that does not fit in any
of the categories.' # data type that does not fit in any of the
categories.

# structured types.
NumberVector: VandalType = List[float] # one-dimensional vector of
integers or floats.
StringVector: VandalType = List[str] # one-dimensional vector of
strings.
StringDictionary: VandalType = Dict[str, str] # one-dimensional 'str'
: 'str' dictionary vector.
DictionaryVector: VandalType = Dict[str, NumberVector] # one-
dimensional 'str' : 'NumberVector' dictionary.

# complex types.
NumberVectorAlike: VandalType = Union[NumberVector, DictionaryVector]
# only number-supported list/vector of values.
NumberArrayAlike: VandalType = Union[List[NumberVector],
List[DictionaryVector]] # array of numerical values only.
AnyArrayAlike: VandalType = Union[List[NumberVector],
List[StringVector], List[StringDictionary], List[DictionaryVector]] #
any =>2 dimensional type.
AnyVectorAlike: VandalType = Union[NumberVector, StringVector,
StringDictionary, DictionaryVector] # any one-dimensional type.
AnyType: VandalType = Union[IntegerType, FloatType, NumberType,
StringType, ListType, TupleType, DictionaryType, NumberVector,
StringVector, StringDictionary, DictionaryVector, NumberVectorAlike,
NumberArrayAlike, AnyVectorAlike, AnyArrayAlike] # any type.
```

```
# all available types.
VandalTypes = [
    'VandalType',
    'IntegerType',
    'FloatType',
    'NumberType',
    'ReturnType',
    'PrintType',
    'GraphType',
    'StringType',
    'ListType',
    'TupleType',
    'DictionaryType',
    'BooleanType',
    'FilePathType',
    'SpecialType',
    'NumberVector',
    'StringVector',
    'StringDictionary',
    'DictionaryVector',
    'NumberVectorAlike',
    'NumberArrayAlike',
    'AnyArrayAlike',
    'AnyVectorAlike',
    'AnyType',
]
```

**Prilog 5 – Funkcija za bojanje teksta.**

- izvorno pripada vlastito izrađenoj biblioteci *logistics*.

```
# dependencies.
from logistics.plugins.types import *
from colorama import (
    Fore,
    Back,
    Style,
    init,
)

# initializes coloring.
init()

# paints the text with a desired color ('fr', 'fg', 'fb', 'fk', 'fc',
'fm', 'fy', 'br', 'bg', 'bb', 'bk', 'bc', 'bm', 'by').
def paint_text(
        text : StringType,
        color : StringType,
        print_trigger : BooleanType = True
        ) -> StringType:

        '''
        * coloring of CLI.

        - text - desired text to print.
        - color - desired color to print in.
        - print_trigger (True/False) - modify return type.
        '''

        # Fore coloring.
        colors = {
            'Fr' : Fore.RED,
            'Fg' : Fore.GREEN,
            'Fb' : Fore.BLUE,
            'Fk' : Fore.BLACK,
            'Fm' : Fore.MAGENTA,
            'Fy' : Fore.YELLOW,
            'Fc' : Fore.CYAN,

        # Back coloring.
            'Br' : Back.RED,
            'Bg' : Back.GREEN,
            'Bb' : Back.BLUE,
            'Bk' : Back.BLACK,
            'Bm' : Back.MAGENTA,
            'By' : Back.YELLOW,
            'Bc' : Back.CYAN,
            }

        if print_trigger == True:
            return print(colors[color] + str(text) + Style.RESET_ALL)

        elif print_trigger == False:
            return colors[color] + str(text) + Style.RESET_ALL
```

## Prilog 6 – Boxplot dijagram

[1]:
```
# unos biblioteka.
import pandas as pd
import math
import matplotlib.pyplot as plt
```

[2]:
```
# kreiranje okvira podataka
df = pd.DataFrame({
    'Vrijednosti' : [2020, 3210, 1340, 6320, 2310, 15400, 5630, 3520, 16000, 1520, 12000, 8250, 2410,]
})
```

[3]:
```
# prikaz okvira podataka.
df
```

[3]:

| | Vrijednosti |
|---|---|
| 0 | 2020 |
| 1 | 3210 |
| 2 | 1340 |
| 3 | 6320 |
| 4 | 2310 |
| 5 | 15400 |
| 6 | 5630 |
| 7 | 3520 |
| 8 | 16000 |
| 9 | 1520 |
| 10 | 12000 |
| 11 | 8250 |
| 12 | 2410 |

[4]:
```
# kreiranje operacija
df['Operacija (snaga 2)'] = 'Vrijednosti * Vrijednosti (x * x)'
df['Vrijednosti (snaga 2)'] = df['Vrijednosti'] * df['Vrijednosti']
df['Operacija (snaga -2)'] = '1 / Vrijednosti * Vrijednosti (1/ x * x)'
df['Vrijednosti (snaga -2)'] = 1 / -(df['Vrijednosti (snaga 2)'] )
```

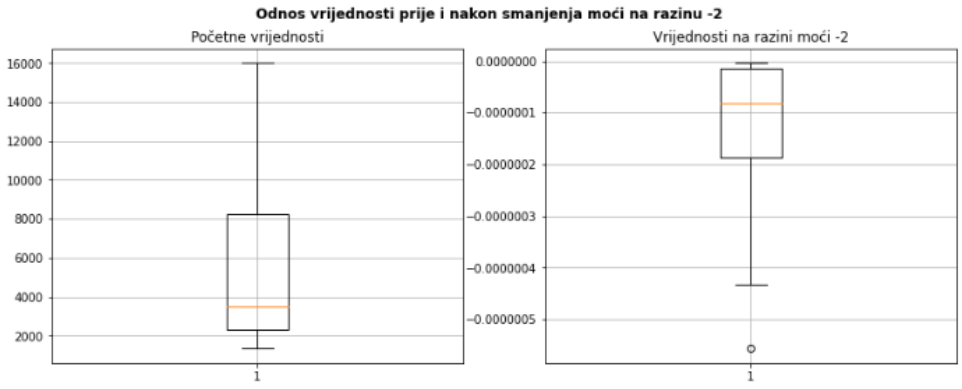
[5]:
```
# prikaz novog okvira podataka.
df
```

[5]:

| | Vrijednosti | Operacija (snaga 2) | Vrijednosti (snaga 2) | Operacija (snaga -2) | Vrijednosti (snaga -2) |
|---|---|---|---|---|---|
| 0 | 2020 | Vrijednosti * Vrijednosti (x * x) | 4080400 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -2.450740e-07 |
| 1 | 3210 | Vrijednosti * Vrijednosti (x * x) | 10304100 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -9.704875e-08 |
| 2 | 1340 | Vrijednosti * Vrijednosti (x * x) | 1795600 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -5.569169e-07 |
| 3 | 6320 | Vrijednosti * Vrijednosti (x * x) | 39942400 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -2.503605e-08 |
| 4 | 2310 | Vrijednosti * Vrijednosti (x * x) | 5336100 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -1.874028e-07 |
| 5 | 15400 | Vrijednosti * Vrijednosti (x * x) | 237160000 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -4.216563e-09 |
| 6 | 5630 | Vrijednosti * Vrijednosti (x * x) | 31696900 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -3.154883e-08 |
| 7 | 3520 | Vrijednosti * Vrijednosti (x * x) | 12390400 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -8.070764e-08 |
| 8 | 16000 | Vrijednosti * Vrijednosti (x * x) | 256000000 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -3.906250e-09 |
| 9 | 1520 | Vrijednosti * Vrijednosti (x * x) | 2310400 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -4.328255e-07 |
| 10 | 12000 | Vrijednosti * Vrijednosti (x * x) | 144000000 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -6.944444e-09 |
| 11 | 8250 | Vrijednosti * Vrijednosti (x * x) | 68062500 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -1.469238e-08 |
| 12 | 2410 | Vrijednosti * Vrijednosti (x * x) | 5808100 | 1 / Vrijednosti * Vrijednosti (1/ x * x) | -1.721733e-07 |

[6]:
```
# prikaz boxplot dijagrama.
fig, (ax1, ax2) = plt.subplots(1,2)
fig.set_figheight(5)
fig.set_figwidth(14)
fig.suptitle('Odnos vrijednosti prije i nakon povećanja moći na razinu 2\n', fontweight='bold')
ax1.boxplot(df['Vrijednosti'])
ax1.set_title('Početne vrijednosti')
ax1.grid()
ax1.ticklabel_format(style = 'plain', axis = 'y')
ax2.set_title('Vrijednosti na razini moći 2')
ax2.grid()
ax2.boxplot(df['Vrijednosti (snaga 2)'])
ax2.ticklabel_format(style = 'plain', axis = 'y')
```


[7]:
```
# prikaz boxplot dijagrama.
fig, (ax3, ax4) = plt.subplots(1,2)
fig.set_figheight(5)
fig.set_figwidth(14)
fig.suptitle('Odnos vrijednosti prije i nakon smanjenja moći na razinu -2\n', fontweight = 'bold')
ax3.boxplot(df['Vrijednosti'])
ax3.grid()
ax3.set_title('Početne vrijednosti')
ax3.ticklabel_format(style = 'plain', axis = 'y')
ax4.set_title('Vrijednosti na razini moći -2')
ax4.grid()
ax4.boxplot(df['Vrijednosti (snaga -2)'])
ax4.ticklabel_format(style = 'plain', axis = 'y')
```



## Prilog 7 – Zamjena nedostajućih podataka

[3]:
```
# unos biblioteka.
import pandas as pd
import numpy as np
```

[4]:
```
# definiranje okvira podataka logističkog poduzeća s uslugom dostave (Varaždin).
df = pd.DataFrame({
    'Lokacija' : ['Zagreb', 'Ludbreg', 'Zagreb', 'Koprivnica', 'Zagreb',
                  'Križevci', 'Čakovec', 'Ludbreg', 'Zagreb', 'Zagreb'],
    'Datum isporuke' : ['22-06-2022', '22-06-2022', '23-06-2022', '23-06-2022', '23-06-2022',
                        '23-06-2022', '25-06-2022', '25-06-2022', '26-06-2022', '27-06-2022'],
    'Šifra vozila' : ['VŽ-1', 'VŽ-2', 'VŽ-3', 'VŽ-2', 'VŽ-1',
                      'VŽ-2', 'VŽ-1', 'VŽ-1', 'VŽ-2', 'VŽ-3'],
    'Šifra klijenta' : ['ZG-564', 'VŽ-213', 'ZG-565', 'KC-221', 'ZG-566',
                        'KC-222', 'ČK-144', 'VŽ-214', 'ZG-567', 'ZG-568'],
    'Udaljenost (km)' : [72, 21, 85, 41, 80, 54, 2, 23, 92, 74],
    'Potrošnja goriva (l)' : [8.1, 2.1, 9.3, 4.5, np.nan, 5.8, 0.3, 2.4, 9.8, np.nan],
})
```

[266]:
```
df
```

[266]:

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| 0 | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.1 |
| 1 | Ludbreg | 22-06-2022 | VŽ-2 | VŽ-213 | 21 | 2.1 |
| 2 | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.3 |
| 3 | Koprivnica | 23-06-2022 | VŽ-2 | KC-221 | 41 | 4.5 |
| 4 | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | NaN |
| 5 | Križevci | 23-06-2022 | VŽ-2 | KC-222 | 54 | 5.8 |
| 6 | Čakovec | 25-06-2022 | VŽ-1 | ČK-144 | 2 | 0.3 |
| 7 | Ludbreg | 25-06-2022 | VŽ-1 | VŽ-214 | 23 | 2.4 |
| 8 | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.8 |
| 9 | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | NaN |

[267]:
```
# izdvajanje lokacije 'Zagreb' s nedostajećim vrijednostima.
df = df[df['Lokacija'] == 'Zagreb']
```

[268]:
```
# sortiranje vrijednosti prema pređenoj udaljenosti.
df = df.sort_values('Udaljenost (km)')
```

[269]:
```
# prikaz sortiranih udaljenosti.
df
```

[269]:

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| 0 | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.1 |
| 9 | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | NaN |
| 4 | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | NaN |
| 2 | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.3 |
| 8 | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.8 |

[270]:
```
# primjena interpolacije.
df.interpolate()
```

[270]:

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| 0 | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.1 |
| 9 | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | 8.5 |
| 4 | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | 8.9 |
| 2 | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.3 |
| 8 | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.8 |

[271]:
```
# primjena metode slanja vrijednosti prethodnog retka.
df.fillna(method = 'ffill')
```

[271]:

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| **0** | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.1 |
| **9** | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | 8.1 |
| **4** | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | 8.1 |
| **2** | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.3 |
| **8** | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.8 |

[272]:
```
# primjena metode popunjavanja vrijednosti definirane u narednom retku.
df.fillna(method = 'bfill')
```

[272]:

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| **0** | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.1 |
| **9** | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | 9.3 |
| **4** | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | 9.3 |
| **2** | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.3 |
| **8** | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.8 |

[273]:
```
# primjena metode upisa srednje vrijednosti.
df.fillna(df.mean(numeric_only = True))
```

[273]:

| | Lokacija | Datum isporuke | Šifra vozila | Šifra klijenta | Udaljenost (km) | Potrošnja goriva (l) |
|---|---|---|---|---|---|---|
| **0** | Zagreb | 22-06-2022 | VŽ-1 | ZG-564 | 72 | 8.100000 |
| **9** | Zagreb | 27-06-2022 | VŽ-3 | ZG-568 | 74 | 9.066667 |
| **4** | Zagreb | 23-06-2022 | VŽ-1 | ZG-566 | 80 | 9.066667 |
| **2** | Zagreb | 23-06-2022 | VŽ-3 | ZG-565 | 85 | 9.300000 |
| **8** | Zagreb | 26-06-2022 | VŽ-2 | ZG-567 | 92 | 9.800000 |

## Prilog 8 – Linearna algebra

[2]:
```
# unos biblioteka.
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd
```

[3]:
```
# definiranje kvadrata i kocke.
objekti = pd.DataFrame({
    'Kvadrat' : [5, 5, 0],
    'Kocka' : [5, 5, 5],
})
```

[4]:
```
# transponiranje matrice.
objekti = objekti.transpose()
```

[5]:
```
# preimenovanje stupaca.
objekti.rename({0 : 'x', 1 : 'y', 2 : 'z'}, axis = 'columns')
```

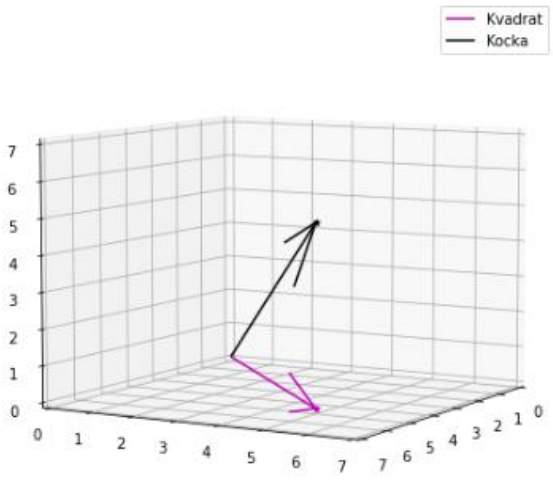
[5]:

| | x | y | z |
|---|---|---|---|
| **Kvadrat** | 5 | 5 | 0 |
| **Kocka** | 5 | 5 | 5 |

[6]:
```
# dohvaćanje koordinata kvadrata.
objekti.loc['Kvadrat']
```

[6]:
```
0    5
1    5
2    0
Name: Kvadrat, dtype: int64
```

[19]:
```
# crtanje objekata (vektora) u prostoru.
ax = plt.axes(projection = '3d')
ax.set_xlim([0, 7])
ax.set_ylim([0, 7])
ax.set_zlim([0, 7])

ax.scatter3D(objekti.loc['Kvadrat'][0],objekti.loc['Kvadrat'][1],
             objekti.loc['Kvadrat'][2], marker = '*', color = 'm')
ax.scatter3D(objekti.loc['Kocka'][0],objekti.loc['Kocka'][1],
             objekti.loc['Kocka'][2], marker = '*', color = 'k')
ax.quiver(0, 0, 0,objekti.loc['Kvadrat'][0],objekti.loc['Kvadrat'][1],
          objekti.loc['Kvadrat'][2], color = 'm',label = 'Kvadrat')
ax.quiver(0, 0, 0,objekti.loc['Kocka'][0],objekti.loc['Kocka'][1],
          objekti.loc['Kocka'][2], color = 'k', label = 'Kocka')
ax.legend()
ax.figure.set_size_inches(12,7)
ax.view_init(7, 30)
```



## Prilog 9 – Podjela seta podataka

[78]:
```
# unos biblioteka.
import pandas as pd
import sklearn
```

[79]:
```
# unos datoteke.
df = pd.read_excel(r'import_data/filtered_data.xlsx')
```

[80]:
```
# prikaz datoteke.
df
```

[80]:

| | Mod | Spol | Dob | Udaljenost do posla (m) | Vozačka dozvola | Automobili u kućanstvu |
|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 32 | 1500 | 1 | 2 |
| **1** | 1 | 0 | 26 | 3500 | 1 | 1 |
| **2** | 1 | 1 | 29 | 18500 | 1 | 2 |
| **3** | 0 | 0 | 35 | 100 | 1 | 1 |
| **4** | 1 | 1 | 44 | 5000 | 0 | 1 |
| **5** | 1 | 1 | 43 | 500 | 1 | 1 |
| **6** | 0 | 0 | 30 | 8500 | 0 | 1 |
| **7** | 0 | 0 | 32 | 7500 | 1 | 2 |
| **8** | 0 | 1 | 50 | 9500 | 1 | 4 |
| **9** | 1 | 1 | 38 | 2500 | 1 | 1 |
| **10** | 0 | 1 | 28 | 500 | 1 | 0 |
| **11** | 0 | 0 | 56 | 1500 | 0 | 1 |
| **12** | 1 | 0 | 44 | 1500 | 0 | 0 |
| **13** | 0 | 1 | 59 | 100 | 1 | 2 |
| **14** | 1 | 1 | 54 | 2500 | 1 | 1 |
| **15** | 0 | 1 | 31 | 1500 | 1 | 1 |
| **16** | 1 | 1 | 34 | 11500 | 1 | 0 |
| **17** | 1 | 0 | 65 | 1500 | 1 | 1 |
| **18** | 0 | 0 | 26 | 2500 | 1 | 0 |
| **19** | 1 | 0 | 24 | 3000 | 0 | 0 |
| **20** | 1 | 0 | 56 | 1500 | 1 | 1 |

[89]:
```
# podjela x i y.
x = df.drop(columns = ['Mod'])
y = df['Mod']
```

[90]:
```
#dodjeljivanje podataka varijablama.
x_treniranje, x_testiranje, y_treniranje, y_testiranje = sklearn.model_selection.train_test_split(x, y, test_size = 0.2)
```

[91]:
```
# prikaz podataka.
x_testiranje
```

[91]:

| | Spol | Dob | Udaljenost do posla (m) | Vozačka dozvola | Automobili u kućanstvu |
|---|---|---|---|---|---|
| **11** | 0 | 56 | 1500 | 0 | 1 |
| **6** | 0 | 30 | 8500 | 0 | 1 |
| **15** | 1 | 31 | 1500 | 1 | 1 |
| **12** | 0 | 44 | 1500 | 0 | 0 |
| **9** | 1 | 38 | 2500 | 1 | 1 |

[92]:
```
# podaci x testa u okvir podataka.
y_testiranje = pd.DataFrame({
    'Mod' : y_testiranje,
})
```

[93]:
```
# prikaz podataka.
y_testiranje
```

[93]:

| | Mod |
|---|---|
| **11** | 0 |
| **6** | 0 |
| **15** | 0 |
| **12** | 1 |
| **9** | 1 |

[94]:
```
#prikaz podataka.
x_treniranje
```

[94]:

| | Spol | Dob | Udaljenost do posla (m) | Vozačka dozvola | Automobili u kućanstvu |
|---|---|---|---|---|---|
| **0** | 1 | 32 | 1500 | 1 | 2 |
| **14** | 1 | 54 | 2500 | 1 | 1 |
| **13** | 1 | 59 | 100 | 1 | 2 |
| **2** | 1 | 29 | 18500 | 1 | 2 |
| **4** | 1 | 44 | 5000 | 0 | 1 |
| **16** | 1 | 34 | 11500 | 1 | 0 |
| **8** | 1 | 50 | 9500 | 1 | 4 |
| **3** | 0 | 35 | 100 | 1 | 1 |
| **19** | 0 | 24 | 3000 | 0 | 0 |
| **5** | 1 | 43 | 500 | 1 | 1 |
| **17** | 0 | 65 | 1500 | 1 | 1 |
| **1** | 0 | 26 | 3500 | 1 | 1 |
| **10** | 1 | 28 | 500 | 1 | 0 |
| **20** | 0 | 56 | 1500 | 1 | 1 |
| **7** | 0 | 32 | 7500 | 1 | 2 |
| **18** | 0 | 26 | 2500 | 1 | 0 |

[95]:
```
# podaci x traina u okvir podataka.
y_treniranje = pd.DataFrame({
    'Mod' : y_treniranje,
})
```

[96]:
```
# prikaz podataka.
y_treniranje
```

[96]:
```
# prikaz podataka.
y_treniranje
```

[96]:

| | Mod |
|---|---|
| **0** | 0 |
| **14** | 1 |
| **13** | 0 |
| **2** | 1 |
| **4** | 1 |
| **16** | 1 |
| **8** | 0 |
| **3** | 0 |
| **19** | 1 |
| **5** | 1 |
| **17** | 1 |
| **1** | 1 |
| **10** | 0 |
| **20** | 1 |
| **7** | 0 |
| **18** | 0 |

## Prilog 10 – K-srednje vrijednosti

- Metodologija provedbe grupiranja djelomično je bazirana na *Jupyter Notebook* predlošku *GitHub* korisnika *dhavalsays (2019)*

[1]:
```
# Pomoćna datoteka nasumičnog ucrtavanja točaka.
```

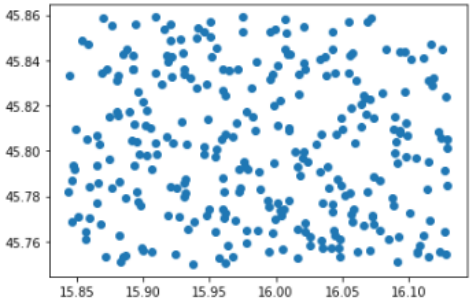
[2]:
```
import random
import pandas as pd
```

[3]:
```
x = [random.uniform(15.84, 16.13) for i in range(300)]
y = [random.uniform(45.75, 45.86) for i in range(300)]
```

[4]:
```
test_lok = pd.DataFrame({
    'Geo_širina' : x,
    'Geo_dužina' : y,
})
```

[36]:
```
import matplotlib.pyplot as plt
plt.scatter(x= test_lok['Geo_širina'], y = test_lok['Geo_dužina'])
```

[36]:
```
<matplotlib.collections.PathCollection at 0x1c6e1365550>
```




[38]:
```
test_lok.to_excel('test_lok.xlsx')
```

[1]:
```
# unos biblioteka.
import kundih
import pandas as pd
import folium
```

[2]:
```
# unos lokacija i filtriranje.
lokacije = pd.read_excel(r'import_data/test_lok.xlsx')
lokacije.drop(columns = 'Unnamed: 0', inplace = True)
```

[3]:
```
# prikaz prvih deset lokacija.
lokacije.head(10)
```

[3]:

| | Geo_širina | Geo_dužina |
|---|---|---|
| 0 | 15.918550 | 45.771235 |
| 1 | 16.009798 | 45.808749 |
| 2 | 16.050008 | 45.809756 |
| 3 | 16.089429 | 45.815235 |
| 4 | 16.042004 | 45.757380 |
| 5 | 15.999879 | 45.853556 |
| 6 | 16.057970 | 45.768824 |
| 7 | 15.981983 | 45.827724 |
| 8 | 16.022059 | 45.835960 |
| 9 | 15.891357 | 45.804350 |

[6]:
```
# inicijalni prikaz karte.
karta = folium.Map(location = [45.80,15.96], zoom_start=13)
karta
```

[5]:
```
# dodavanje markera na kartu.
for i in range(len(lokacije['Geo_širina'])):
    folium.Marker(location=[lokacije['Geo_dužina'][i], lokacije['Geo_širina'][i]]).add_to(karta)
karta
```

[5]:



[150]:
```
# inicijalizacija objekta.
KSV = kundih.KSrednjeVrijednosti(datoteka = lokacije, broj_klastera = 3, koordinate = ['Geo_širina', 'Geo_dužina'])
```

[152]:
```
# podešavanje omjera (fitting).
KSV.podesi_omjer().head(10)
```

[152]:

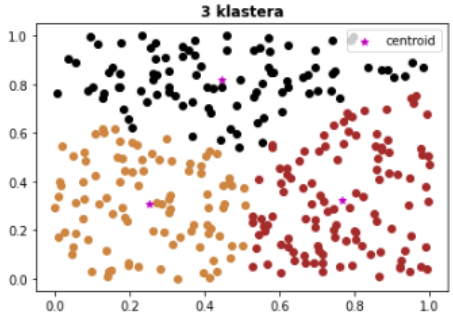
| | Geo_širina | Geo_dužina | Klaster |
|---|---|---|---|
| 0 | 0.262590 | 0.194583 | 4 |
| 1 | 0.581372 | 0.538464 | 3 |
| 2 | 0.721845 | 0.547695 | 3 |
| 3 | 0.859564 | 0.597920 | 3 |
| 4 | 0.693885 | 0.067576 | 1 |
| 5 | 0.546718 | 0.949210 | 3 |
| 6 | 0.749663 | 0.172480 | 1 |
| 7 | 0.484198 | 0.712413 | 0 |
| 8 | 0.624204 | 0.787905 | 3 |
| 9 | 0.167589 | 0.498145 | 4 |

[153]:
```
# metoda koja kreira klastere nad podacima.
KSV.kreiraj_klastere()
```

[154]:
```
# metoda koja prikazuje lokacije centroida.
KSV.prikaz_centroida()
```

[154]:
```
array([[0.44741087, 0.82071491],
       [0.2511301 , 0.30809406],
       [0.76813427, 0.32479262]])
```

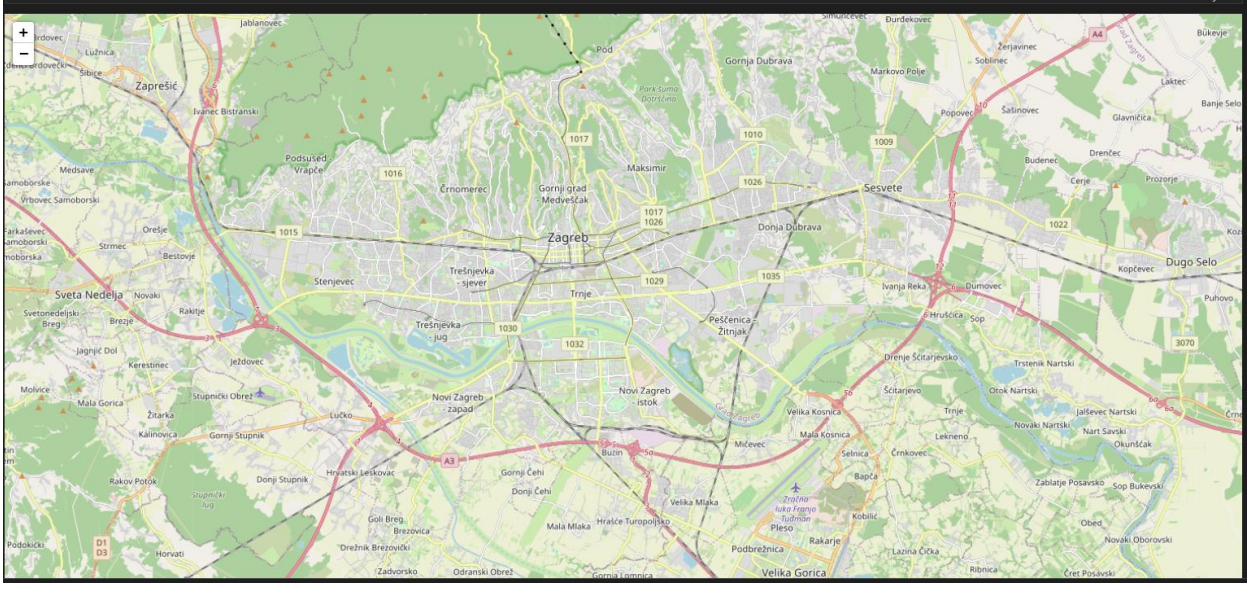
[172]:
```
# prikazivanje grafa s 3 klastera.
f = KSV.prikaz_grafa()
```



```
[158]: # prikaz grafa lakta.
       KSV.prikaz_promjene()
```

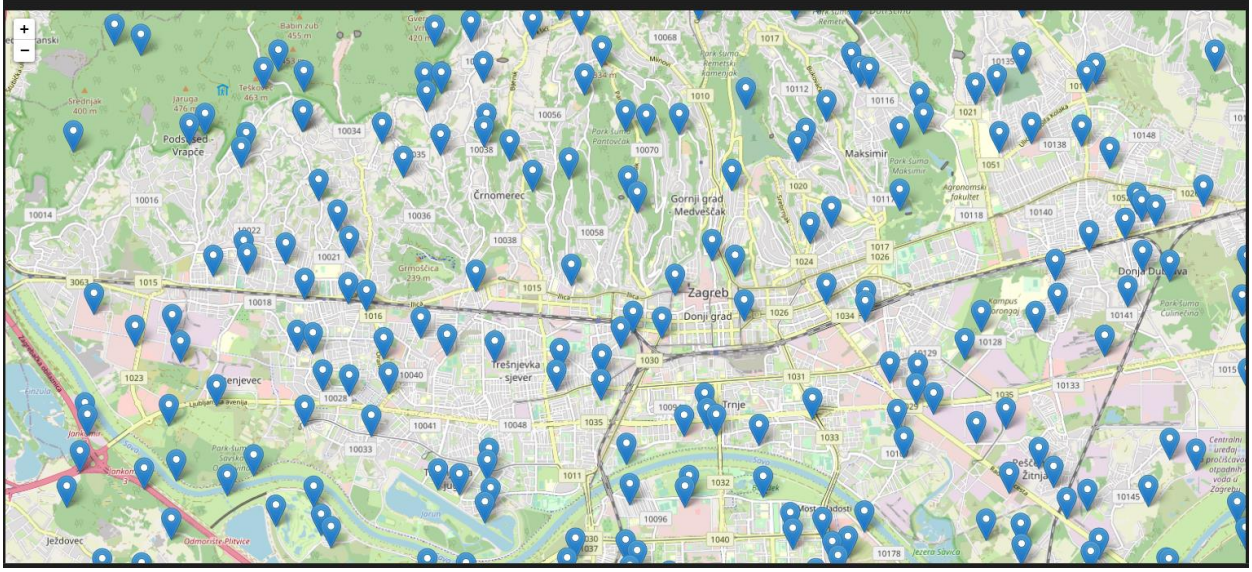

```
# inicijalni prikaz karte.
karta = folium.Map(location = [45.80,15.96], zoom_start=13)
karta
```

Python




```
# dodavanje markera na kartu.
for i in range(len(lokacije['Geo_širina'])):
    folium.Marker(location=[lokacije['Geo_dužina'][i], lokacije['Geo_širina'][i]]).add_to(karta)
karta
```

Python



## Prilog 11 – Dijkstra algoritam

- Programski kod Dijkstra algoritma u vlastitoj biblioteci *kundih* djelomično je inspiriran metodologijom i logikom algoritma koju navodi Dey (2019).

[11]:
```
# unos biblioteka.
import kundih
import numpy as np
```

[12]:
```
# definiranje čvorova.
čvorovi = {
'p':{'c': 6, 'b': 15},
'b':{'f': 5, 'p': 15},
'c':{'p': 6, 'd': 4, 'i': 9},
'd':{'c': 4, 'e': 5},
'e':{'d': 5, 'a': 5, 'f': 3},
'f':{'e': 3, 'b': 5, 'k': 6},
'g':{'h': 4, 'm': 9},
'h':{'g': 4, 'n': 8, 'i': 6},
'i':{'c': 9, 'l': np.inf, 'h': 6},
'j':{'k': 3},
'k':{'j': 3, 'f': 6, 's': 9},
'l':{'i': 3, 'r': np.inf},
'm':{'g': 9, 't': 6},
'n':{'h': 8, 'u': 5, 'r': 7},
'o':{'r': 6, 'u': np.inf},
'a':{'e': 5},
'r':{'o': 6, 's': 10, 'l': 6, 'n': 7},
's':{'k': 9, 'r': 10},
't':{'u': 3, 'm': 6},
'u':{'o': 5, 'n': 5, 't': 3},
}
```

[13]:
```
# inicijalizacija objekta.
Dijkstra = kundih.Dijkstra(čvor = čvorovi, polazište = 'p', odredište = 'o')
```

[14]:
```
# izračun optimalne putanje (rute).
Dijkstra.optimalna_ruta()
```

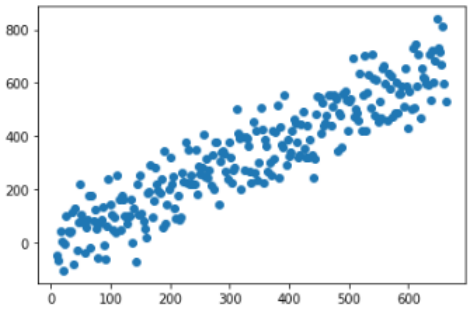
[14]:

| | Izračun putanje |
|---|---|
| **Optimalna putanja** | [p, c, i, h, n, u, o] |
| **Minimalna udaljenost** | 39 |

## Prilog 12 – Linearna regresija

- Programski kod algoritma linearne regresije u vlastitoj biblioteci *kundih* djelomično je inspiriran metodologijom i logikom algoritma koju navodi Fowers (2019).

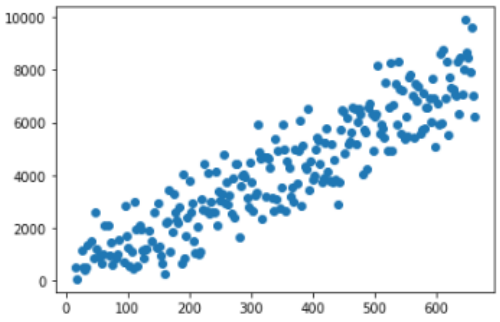
```
[442]: # Pomoćna datoteka nasumičnog definiranja varijabli.

[443]: import random
       import numpy as np
       import matplotlib.pyplot as plt
       import pandas as pd

[444]: x = np.arange(10.3, 663.2, 2.53)

[445]: x

[445]: array([ 10.3 ,  12.83,  15.36,  17.89,  20.42,  22.95,  25.48,  28.01,
               30.54,  33.07,  35.6 ,  38.13,  40.66,  43.19,  45.72,  48.25,
               50.78,  53.31,  55.84,  58.37,  60.9 ,  63.43,  65.96,  68.49,
               71.02,  73.55,  76.08,  78.61,  81.14,  83.67,  86.2 ,  88.73,
               91.26,  93.79,  96.32,  98.85, 101.38, 103.91, 106.44, 108.97,
              111.5 , 114.03, 116.56, 119.09, 121.62, 124.15, 126.68, 129.21,
              131.74, 134.27, 136.8 , 139.33, 141.86, 144.39, 146.92, 149.45,
              151.98, 154.51, 157.04, 159.57, 162.1 , 164.63, 167.16, 169.69,
              172.22, 174.75, 177.28, 179.81, 182.34, 184.87, 187.4 , 189.93,
              192.46, 194.99, 197.52, 200.05, 202.58, 205.11, 207.64, 210.17,
              212.7 , 215.23, 217.76, 220.29, 222.82, 225.35, 227.88, 230.41,
              232.94, 235.47, 238.  , 240.53, 243.06, 245.59, 248.12, 250.65,
              253.18, 255.71, 258.24, 260.77, 263.3 , 265.83, 268.36, 270.89,
              273.42, 275.95, 278.48, 281.01, 283.54, 286.07, 288.6 , 291.13,
              293.66, 296.19, 298.72, 301.25, 303.78, 306.31, 308.84, 311.37,
              313.9 , 316.43, 318.96, 321.49, 324.02, 326.55, 329.08, 331.61,
              334.14, 336.67, 339.2 , 341.73, 344.26, 346.79, 349.32, 351.85,
              354.38, 356.91, 359.44, 361.97, 364.5 , 367.03, 369.56, 372.09,
              374.62, 377.15, 379.68, 382.21, 384.74, 387.27, 389.8 , 392.33,
              394.86, 397.39, 399.92, 402.45, 404.98, 407.51, 410.04, 412.57,
              415.1 , 417.63, 420.16, 422.69, 425.22, 427.75, 430.28, 432.81,
              435.34, 437.87, 440.4 , 442.93, 445.46, 447.99, 450.52, 453.05,
              455.58, 458.11, 460.64, 463.17, 465.7 , 468.23, 470.76, 473.29,
              475.82, 478.35, 480.88, 483.41, 485.94, 488.47, 491.  , 493.53,
              496.06, 498.59, 501.12, 503.65, 506.18, 508.71, 511.24, 513.77,
              516.3 , 518.83, 521.36, 523.89, 526.42, 528.95, 531.48, 534.01,
              536.54, 539.07, 541.6 , 544.13, 546.66, 549.19, 551.72, 554.25,
              556.78, 559.31, 561.84, 564.37, 566.9 , 569.43, 571.96, 574.49,
              577.02, 579.55, 582.08, 584.61, 587.14, 589.67, 592.2 , 594.73,
              597.26, 599.79, 602.32, 604.85, 607.38, 609.91, 612.44, 614.97,
              617.5 , 620.03, 622.56, 625.09, 627.62, 630.15, 632.68, 635.21,
              637.74, 640.27, 642.8 , 645.33, 647.86, 650.39, 652.92, 655.45,
              657.98, 660.51, 663.04])

[446]: y = np.random.normal(x, 80)

[447]: plt.scatter(x, y)

[447]: <matplotlib.collections.PathCollection at 0x1b9d75d2eb0>
```




```
[452]: df = pd.DataFrame({
           'Masa (kg)' : x,
           'Cijena transporta (HRK)' : y
       })

[453]: df = df.loc[(df['Cijena transporta (HRK)'] > 5) & (df['Cijena transporta (HRK)'] < 1000)]

[454]: df['Cijena transporta (HRK)'] = df['Cijena transporta (HRK)'] * 11.779
```

[456]:
```
plt.scatter(df['Masa (kg)'], df['Cijena transporta (HRK)'])
```

[456]:
```
<matplotlib.collections.PathCollection at 0x1b9d76fbb50>
```




[457]:
```
df.to_excel('podaci.xlsx')
```

[45]:
```
# unos biblioteka.
import kundih
import pandas as pd
```

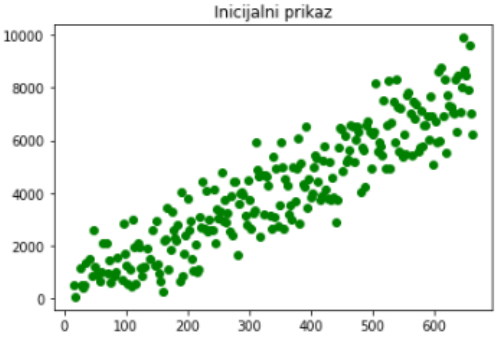
[46]:
```
# unos podataka.
podaci = pd.read_excel(r'import_data/podaci.xlsx')
podaci.drop(columns = {'Unnamed: 0'}, inplace = True)
```

[47]:
```
podaci.tail()
```

[47]:

| | Udaljenost (km) | Cijena transporta (HRK) |
|---|---|---|
| **241** | 652.92 | 8447.181748 |
| **242** | 655.45 | 7892.732988 |
| **243** | 657.98 | 9589.149679 |
| **244** | 660.51 | 7039.932166 |
| **245** | 663.04 | 6247.777417 |

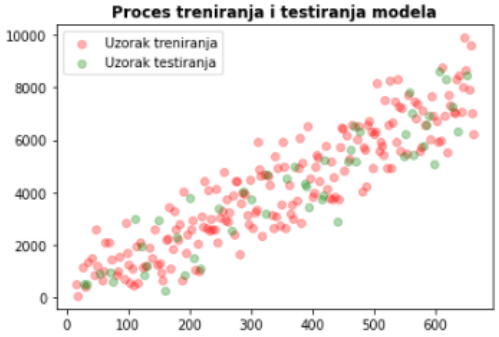
[48]:
```
# inicijalizacija objekta.
LR = kundih.LinearnaRegresija(podaci, 'Udaljenost (km)', 'Cijena transporta (HRK)')
LR.prikaz_grafa()
```



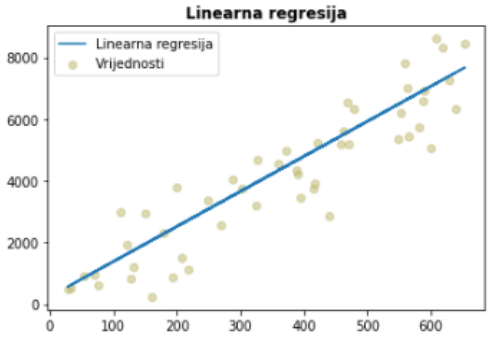

[49]:
```
# podjela seta
LR.podjela_seta(omjer_testa = 0.20, slučajno_stanje = 40)
```

[50]:
```
# prikaz procesa treniranja i testiranja.
LR.prikaz_procesa()
```

[50]:
```
# prikaz procesa treniranja i testiranja.
LR.prikaz_procesa()
```




[51]:
```
# prikaz linearne regresije.
LR.prikaz_linearne_regresije()
```




[52]:
```
# procjena vrijednosti zavisne varijable u odnosu na nezavisnu.
LR.procjena_varijable(vrijednost = 500)
```

```
Odgovarajuća vrijednost za 500 je 5932.8011315484855.
```

[53]:
```
# prikaz preciznosti i formule modela.
LR.preciznost_modela()
LR.formula()
```

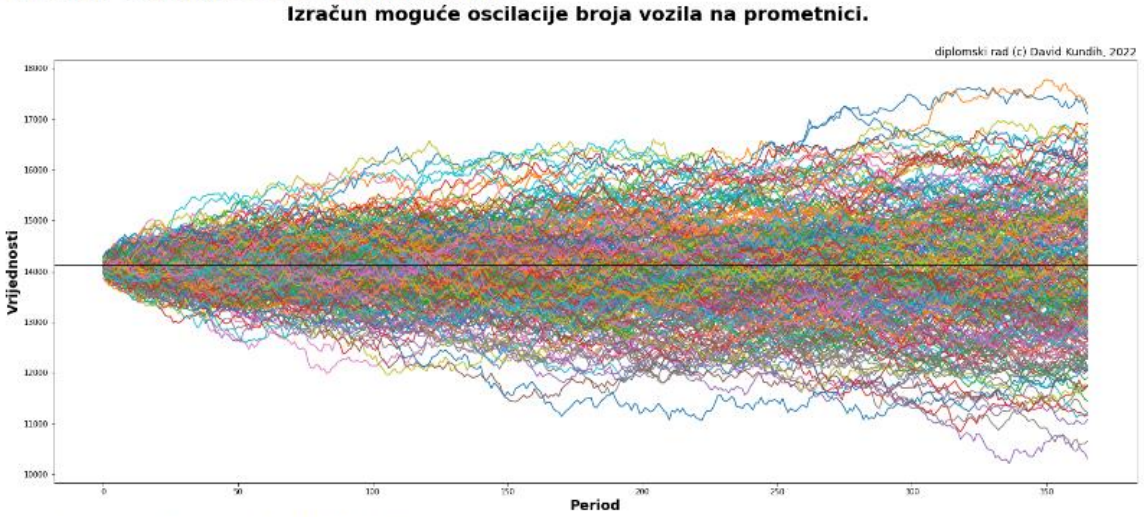
```
Uspješnost modela iznosi 0.8345932878948212.
Formula modela je y = 500 * 11.367176124296732 + 249.21306940012028
```

## Prilog 13 – Monte Carlo simulacija

- Programski kod za izradu Monte Carlo simulacije u vlastitoj biblioteci *kundih* djelomično je inspiriran metodologijom i logikom funkcioniranja simulacije koju navodi Dupuis (2017).

```
[106]: # unos biblioteka.
       import kundih
       import numpy as np

[107]: # kreiranje simulacijskog seta podataka.
       data = [np.random.normal(14000, 50) for i in range(10000)]

[108]: # inicijalizacija objekta nad varijablom.
       MC = kundih.MonteCarlo(data, 365, 300)

[ ]:   # pokretanje Monte Carlo simulacije.
       MC.izračunaj()

*[114]: # grafički prikaz Monte Carlo simulacije.
        MC.prikaži_graf(naslov = 'Izračun moguće oscilacije broja vozila na prometnici.',
                        x_naslov = 'Period', y_naslov = 'Vrijednosti')
```

```
Započelo je iscrtavanje MonteCarlo simulacije na grafu.
```




```
Iscrtavanje Monte Carlo simulacije izvršeno.
```

```
[115]: # statistički podaci simulacije.
       MC.prikaži_statistiku()
```

[115]:

| | Statistika |
|---|---|
| **Srednja vrijednost** | 13981.88 |
| **Standardna devijacija** | 1262.15 |
| **Maximalna vrijednost** | 17221.65 |
| **Minimalna vrijednost** | 10318.40 |

**Prilog 14 – Monte Carlo simulacija – objekt vandal.MonteCarlo**

- Programski kod djelomično je inspiriran metodologijom i logikom funkcioniranja simulacije koju navodi Dupuis (2017), dok ostatak pripada vlastitoj biblioteci *vandal* s preinakama na hrvatskom jeziku prikazanom kroz biblioteku *kundih*.

```
# coloring.
from colorama import (
    Fore,
    Back,
    Style,
    init,
)

init()

# type hints and annotations.
from logistics.plugins.metaclass import Meta

# imports all data types.
from logistics.plugins.types import *

# package.
class MonteCarlo:

    '''
    (OBJECT INFO)
    -------------

    eg. MonteCarlo = vandal.MonteCarlo()

    vandal.MonteCarlo - main class that stores the simulation data.

        * takes 5 additional arguments.
            list_of_values - pandas dataframe of values.
            time_seq - desired time sequence.
            num_sims - desired number of simulation iterations.
            ref_col (default: ref_col = 0) - column index of the
dictionary values.
            ref_row (default: ref_row = 0) - row index on which the
starting point of the simulation is created.
        * Requirements:
            data stored as dictionary of key, value pair | list |
numpy array | pandas DataFrame.

    (OBJECT FUNCTIONS)
    ------------------

    eg. vandal.MonteCarlo.function()

        .execute() - executes a Monte Carlo simulation on a defined
data set.
            * takes 1 additional argument.
                filtered (defualt: filtered = True) - filters the
print option and leaves just the return object.
```

```
        .graph() - plots the Monte Carlo simulation on a graph.
            * takes 5 optional customization arguments. (default:
graph_title = 'Monte Carlo simulation', x_title = 'X axis', y_title =
'Y axis', plot_size = (25,10), perform_block = True).
                graph_title - title of the graph.
                x_title - title of the X axis.
                y_title - title on the Y axis.
                plot_size - desired size of the graph. eg. -
(x_lenght_num, y_lenght_num). - NOTE: values must be inside the
parentheses and divided by a comma.
                perform_block (default: perform_block = True) -
False/True may be used depending on the IDE requirements.

        .get_risk() - calculates the risk of value decrease over
time.
            * takes 1 additional argument.
                risk_sims (default: risk_sims = 5000) - number of
simulations to perform the risk percantage.

        .get_stats() - shows the statistics of the Monte Carlo
simulation.
            * takes no additional arguments.

        .get_change() - shows the percentage of Monte Carlo
simulation value change for every iteration.
            * takes no additional arguments.

        .hist() - plots the histogram of Monte Carlo simulation.
            * takes 7 optional customization arguments. (default:
graph_title = 'Monte Carlo simulation', x_title = 'X axis', y_title =
'Y axis', plot_size = (25,10), set_bins = None, perform_block = True,
method = 'b').
            If method = 'e' is chosen, no customization arguments
apply.
                graph_title - title of the graph.
                x_title - title of the X axis.
                y_title - title on the Y axis.
                plot_size - desired size of the graph. eg. -
(x_lenght_num, y_lenght_num). - NOTE: values must be inside the
parentheses and divided by a comma.
                perform_block (default: perform_block = True) -
False/True may be used depending on the IDE requirements.
                method - default method is Basic histogram and it's
performed by automation. In order to plot Empirical rule histogram
add method = 'e' as the last argument. - NOTE: method of a histogram
must be placed within quotation marks.
                set_bins - sets the amount of bins of the histogram.
(default: set_bins = self.time_seq)
            * automatically executes the .get_stats() function in
order to get standard deviation for the Empirical rule plotting.

    MCapp (EXECUTABLE CLI MODULE)
    -------------------------

    vandal.MCapp is an executable function that runs the Command Line
Interface of the vandal MonteCarlo module.
        print(help(vandal.MCapp)) in order to see available features.
    '''
```

```
    # metadata of the used library.
    from vandal.misc._meta import (
        __author__,
        __copyright__,
        __credits__,
        __license__,
        __version__,
        __documentation__,
        __contact__,
        __donate__,
        __APPversion__,
    )

    def __init__(
        self,
        list_of_values : NumberVectorAlike = None,
        time_seq : IntegerType = None,
        num_sims : IntegerType = None,
        ref_col : IntegerType = 0,
        ref_row : IntegerType = 0,
        ) -> ReturnType:

        '''
        * initial launch.
        '''

        self.list_of_values = list_of_values
        self.time_seq = time_seq
        self.num_sims = num_sims
        self.ref_col = ref_col
        self.ref_row = ref_row

        return

    def __str__(
        self,
        ) -> StringType:

        '''
        * class information.
        '''

        return f'Monte Carlo defining object that stores the
configuration data for creating {self.num_sims} simulations in a
period of {self.time_seq} time measurement units.'

    def __repr__(
        self,
        ) -> StringType:

        '''
        * class information.
        '''

        return f'Monte Carlo defining object that stores the
configuration data for creating {self.num_sims} simulations in a
period of {self.time_seq} time measurement units.'
```

```
    def execute(
        self,
        filtered : BooleanType = True,
        ) -> NumberArrayAlike:

        '''
        * executes a Monte Carlo simulation on a defined data set.

        - filtered (True/False) - filters the data setup print and
warnings.
        # DEFAULT: MonteCarlo.execute(filtered : BooleanType = True.)
        '''

        if filtered == False:
            print(Fore.GREEN + f'Monte Carlo has been set up for
{self.num_sims} simulations in a period of {self.time_seq} time
measurement units and executed.' + Fore.RESET)
            print(Fore.RED + 'NOTE: Use data with reasonable standard
deviation in order to prevent exponential growth of the function that
cannot be plotted properly, recognize such abnormal values by a +
sign anywhere in the data executed below.' + Fore.RESET)

        import numpy as np
        import pandas as pd

        # this removes pandas warning of highly fragmented DataFrame
for newer pandas versions.
        from warnings import simplefilter
        simplefilter(action = 'ignore', category =
pd.errors.PerformanceWarning)
        # end of pandas warning removal block.

        try:
            self.list_of_values = pd.DataFrame(self.list_of_values)
            self.list_of_values = self.list_of_values.iloc[:,
self.ref_col]
        except:
            raise KeyError('Impossible to reach a defined key, value
pair. Data types supported: dictionary, list, numpy array, pandas
DataFrame. Make sure to set index row_col on an existing field. Must
be of type: int.')

        today_value = self.list_of_values.iloc[self.ref_col]
        data = pd.DataFrame()
        loading = 0

        for num_sim in range(self.num_sims):
            rand_change = self.list_of_values.pct_change().std()
            count = 0
            index_array = []
            index_array += [today_value + (today_value *
np.random.normal(0, rand_change))]

            for num_day in range(self.time_seq):
                rand_change = self.list_of_values.pct_change().std()
                if count == self.time_seq:
                    break
```

```
                index_array += [index_array[count] + (today_value *
np.random.normal(0, rand_change))]
                count += 1

            loading += 1
            print(end = '\r')
            print(loading, 'iterations out of', self.num_sims,
'executed so far', end = '')

            data[num_sim] = index_array
        print(end = '\r')
        print(Fore.GREEN + 'Monte Carlo simulation set up and ready
to plot.' + Fore.RESET)
        self.results = data

        return data

    def get_change(
        self,
        ) -> NumberArrayAlike:

        '''
        * shows the percentage of Monte Carlo simulation value change
for every iteration.
        '''

        return self.results.pct_change()

    def get_risk(
        self,
        risk_sims : IntegerType = 5000,
        ) -> NumberType:

        '''
        * calculates the risk of negative values occuring.

        - risk_sims - number of simulations to run risk evaluation
on.
        # DEFAULT: MonteCarlo.get_risk(risk_sims : IntegerType =
5000.)
        '''

        import random
        import pandas as pd

        # this removes pandas warning of highly fragmented DataFrame
for newer pandas versions.
        from warnings import simplefilter
        simplefilter(action = 'ignore', category =
pd.errors.PerformanceWarning)
        # end of pandas warning removal block.

        today_value = self.list_of_values.iloc[self.ref_row]
        percent_change = self.list_of_values.pct_change()
        data = pd.DataFrame()
        smaller = []

        for num_sim in range(risk_sims):
```

```
            random_change = random.choice(percent_change)
            index_array = []
            index_array += [today_value * (1 + (random_change))]
            data[num_sim] = index_array

            for sim in data[num_sim]:
                if sim < today_value:
                    smaller += [sim]
        NRisk = len(smaller) / num_sim * 100
        assert (NRisk < 100), '\nTime sequence and/or number of
iterations are too low for the proper risk calculation.'

        return str(round(NRisk, 2)) + '%'

    def graph(
        self,
        graph_title : StringType = 'Monte Carlo simulation',
        x_title : StringType = 'X axis',
        y_title : StringType = 'Y axis',
        plot_size : TupleType = (25,10),
        perform_block : BooleanType = True,
        ) -> GraphType:

        '''
        * plots the Monte Carlo simulation on a graph.

        - graph title - sets a graph title.
        - x_title - sets an x axis title.
        - y_title - sets and y axis title.
        - plot_size - sets a size of graph in a tuple (eg. (x,y).)
        - perform_block (True/False) - customizable wheel to disable
block in some IDEs.
        # DEFAULT: MonteCarlo.graph(graph_title : StringType = 'Monte
Carlo simulation', x_title : StringType = 'X axis', y_title :
StringType = 'Y axis', plot_size : TupleType = (25,10), perform_block
: BooleanType = True.)
        '''

        print(Fore.GREEN + '\nMonteCarlo() plotting initialized.' +
Fore.RESET)
        import matplotlib.pyplot as plt
        plt.figure(figsize = plot_size)
        plt.title('vandal (c) David Kundih, 2021-2022', fontsize =
14, weight = 'regular', loc = 'right')
        plt.suptitle(graph_title, fontsize = 25, weight = 'bold')
        plt.plot(self.results)
        plt.axhline(y = self.results[0][0], color = 'k', linestyle =
'solid')
        plt.xlabel(x_title, fontsize = 18, weight = 'semibold')
        plt.ylabel(y_title, fontsize = 18, weight = 'semibold')
        plt.show(block = perform_block)
        print(Fore.GREEN + 'MonteCarlo() plotting finished.' +
Fore.RESET)

        return

    def get_stats(
        self,
```

```
        ) -> AnyArrayAlike:

        '''
        * shows the statistics of the Monte Carlo simulation.
        '''

        import numpy as np
        import pandas as pd

        mean_value = np.mean(self.results.loc[self.time_seq])
        mean_value = round((mean_value),2)
        standard_deviation =
round(np.std(self.results.loc[self.time_seq]),2)
        standard_deviation = round((standard_deviation),2)
        maximum_value = np.max(self.results.loc[self.time_seq])
        maximum_value = round((maximum_value),2)
        minimum_value = np.min(self.results.loc[self.time_seq])
        minimum_value = round((minimum_value),2)
        self.standard_deviation = standard_deviation
        self.mean_value = mean_value

        stats = {
            'Mean value' : mean_value,
            'Standard deviation' : standard_deviation,
            'Maximum value ' : maximum_value,
            'Minimum value' : minimum_value,
            }

        stats = pd.DataFrame(stats, index = ['Statistics'])
        stats = stats.transpose()

        return stats

    def hist(
        self,
        graph_title : StringType = 'Histogram of value frequencies',
        x_title : StringType = 'X axis',
        y_title : StringType = 'Y axis',
        plot_size : TupleType = (25,10),
        perform_block : BooleanType = True,
        set_bins : IntegerType = None,
        **method : StringType,
        ) -> GraphType:

        '''
        * plots the histogram of Monte Carlo simulation.

        - graph title - sets a graph title.
        - x_title - sets an x axis title.
        - y_title - sets and y axis title.
        - plot_size - sets a size of graph in a tuple (eg. (x,y).)
        - perform_block (True/False) - customizable wheel to disable
block in some IDEs.
        - set_bins - sets amount of histogram bins.
        - **method ('e' - empirical, 'b' - basic)- method of a
histogram.
        # DEFAULT: MonteCarlo.hist(graph_title : StringType =
'Histogram of value frequencies', x_title : StringType = 'X axis',
```

```
y_title : StringType = 'Y axis', plot_size : TupleType = (25,10),
perform_block : BooleanType = True,set_bins : IntegerType = None,
**method : StringType.)
        '''

        self.get_stats()
        std_plus = self.mean_value + self.standard_deviation
        std_minus = self.mean_value - self.standard_deviation
        std_plus2 = self.mean_value + (self.standard_deviation * 2)
        std_minus2 = self.mean_value - (self.standard_deviation * 2)
        std_plus3 = self.mean_value + (self.standard_deviation * 3)
        std_minus3 = self.mean_value - (self.standard_deviation * 3)

        if self.time_seq > 50:
            print(Fore.RED + 'NOTE: Time sequence defined greatly
impacts the lenght of histogram plotting.\n' + Fore.RESET)

        print(Fore.GREEN + 'Histogram plotting initiated...' +
Fore.RESET)
        import matplotlib.pyplot as plt
        plt.figure(figsize = plot_size)
        plt.title('vandal (c) David Kundih, 2021-2022', fontsize =
14, weight = 'regular', loc = 'right')

        if method.get("method") != "e":
        # DEPRECATED, THIS METHOD REQUIRES A REDESIGN AND SHOULD NOT
BE A PART OF HISTOGRAM PLOTTING!!!
            print(Fore.GREEN + 'CHOSEN METHOD: Basic histogram
model.' + Fore.RESET)
            plt.suptitle(graph_title, fontsize = 25, weight = 'bold')

        if method.get("method") == "e":
            print(Fore.GREEN + 'CHOSEN METHOD: Empirical rule.' +
Fore.RESET)
            plt.suptitle('Value division based on the Empirical
rule', fontsize = 25, weight = 'bold')
            plt.axvline(x = std_plus, color = 'g', linestyle =
'dashed')
            plt.axvline(x = std_minus, color = 'r', linestyle =
'dashed')
            plt.axvline(x = self.mean_value, color = 'k', linestyle =
'solid')
            plt.axvline(x = std_plus2, color = 'g', linestyle =
'dashed')
            plt.axvline(x = std_minus2, color = 'r', linestyle =
'dashed')
            plt.axvline(x = std_plus3, color = 'g', linestyle =
'dashed')
            plt.axvline(x = std_minus3, color = 'r', linestyle =
'dashed')

        if set_bins == None:
            set_bins = self.time_seq

        plt.hist(self.results, bins = set_bins, ec = 'm')
        plt.xlabel(x_title, weight = 'semibold')
        plt.ylabel(y_title, weight= 'semibold')
        plt.show(block = perform_block)
```

```
        print(Fore.GREEN + 'Histogram plotting finished.',
Fore.RESET)

        return


# CLI application.
def MCapp():

    '''
    runs as:

        * IDE: vandal.MCapp()
        * TERMINAL: python -m vandal -e montecarlo / python -m vandal
--entry montecarlo
    '''

    print(Fore.YELLOW + 'MonteCarlo app is initializing...',
Fore.RESET)

    # relevant imports.
    import os
    import pandas as pd
    from vandal.misc._meta import (
        __version__,
        __APPversion__,
    )
    from vandal.objects.montecarlo import MonteCarlo
    from vandal.hub.toolkit import (
        save_to,
        file_handler,
    )
    os.system('cls')

    # greeting.
    print(Fore.YELLOW + '\n - vandal Command Line Interface
Application © David Kundih -', __APPversion__)
    print(Fore.YELLOW + ' - vandal package version -',
'v',__version__, Fore.RESET, '\n')
    print(Fore.YELLOW + 'DATA INPUT OPTIONS', Fore.RESET)
    print('0 | Manual input')
    print('1 | File input\n')

    inputchoice = input('Enter the option number: ')

    # manual input.
    if inputchoice == '0':
        iter = True
        print(Fore.RED + '\nWrite any non-number value to stop.',
Fore.RESET)
        data = []

        while iter:
            try:
                listinput = input('Enter a value: ')
                listinput = listinput.replace(",", ".")
                listinput = float(listinput)
                data.append(listinput)
```

```
            except:
                print('')
                iter = False

        data = pd.DataFrame(data)
        data = data[0]

    # file input.
    elif inputchoice == '1':
        inputfile = input('Enter the file destination: ')

        data = file_handler(file = inputfile)

    # error on input.
    else:
        raise KeyError('Invalid input, please choose one of the
stated options.')

    # simulation parameters and execution.
    simulations = int(input('Enter number of simulations: ') or 100)
    period = int(input('Enter desired period: ') or 50)

    MC = MonteCarlo(list_of_values = data, num_sims = simulations,
time_seq = period)
    print('')
    executed = MC.execute(filtered = False)

    # options after defining the parameters.
    while True:
        action = input('\n>>> ACTIONS: graph, change, values, stats,
risk, hist, help, home: ')

        if action == 'graph':
            print('')
            title = input('Title: ')
            x_axis = input('X axis title:')
            y_axis = input('Y axis title:')
            MC.graph(graph_title = title, x_title = x_axis, y_title =
y_axis)

        if action == 'change':
            print(Fore.YELLOW + '\nSAVE OPTIONS', Fore.RESET)
            print('0 | csv')
            print('1 | xlsx')
            print('2 | json')

            file_type = input('\nEnter the number or name of file
type: ')
            output = MC.get_change()

            try:
                save_to(file = output, prefix = 'vandal.MonteCarlo -
' , func_name = 'change', choice = file_type)
            except:
                raise Exception('=== UNABLE TO SAVE, PLEASE SELECT
ONE OF THE OPTIONS AND/OR RUN THE TERMINAL AS AN ADMINISTRATOR. ===')

        if action == 'values':
```

```
            print(Fore.YELLOW + '\nSAVE OPTIONS', Fore.RESET)
            print('0 | csv')
            print('1 | xlsx')
            print('2 | json')

            file_type = input('\nEnter the number or name of file
type: ')

            try:
                save_to(file = executed, prefix = 'vandal.MonteCarlo
- ', func_name = 'values', choice = file_type)
            except:
                raise Exception('=== UNABLE TO SAVE, PLEASE RUN THE
TERMINAL AS AN ADMINISTRATOR. ===')

        if action == 'stats' or action == 'statistics':
            print('\n', MC.get_stats())

        if action == 'risk':
            sample = int(input('Number of iterations to measure risk
on: ') or 5000)
            executed_risk = MC.get_risk(risk_sims = sample)
            print(Fore.YELLOW + '\nRisk for this option is' +
Fore.RESET, executed_risk[:-1], Fore.YELLOW + '%.' + Fore.RESET)

        if action == 'hist' or action == 'histogram':
            print('')
            x_axis = input('X axis title:')
            y_axis = input('Y axis title:')
            print(Fore.YELLOW + '\nHISTOGRAM METHODS', Fore.RESET)
            print('0 | Basic Histogram')
            print('1 | Empirical Rule Histogram')

            method = input('\nEnter the histogram method number: ')

            if method == '0':
                MC.hist(x_title = x_axis, y_title = y_axis)
            elif method == '1':
                MC.hist(x_title = x_axis, y_title = y_axis, method =
'e')
            else:
                print(Fore.RED + '=== INVALID METHOD. ===\n',
Fore.RESET)

        if action == 'home':
            print(Fore.YELLOW + 'Exiting...', Fore.RESET)

            break

        if action == 'help':
            print(Fore.YELLOW +
'\nhttps://github.com/dkundih/vandal', Fore.RESET)

# runs module as an app.
if __name__ == '__main__':
    MCapp()
```