# FENOMEN VELIKIH PODATAKA, POSTUPCI PODATKOVNE ANALIZE I OBRADE U PYTHON PROGRAMSKOM JEZIKU

**Jerončić Grba, Karlo**

**Undergraduate thesis / Završni rad**

**2019**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Split, Faculty of economics Split / Sveučilište u Splitu, Ekonomski fakultet**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:124:671757

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-03-29**

*Repository / Repozitorij:*

REFST - Repository of Economics faculty in Split

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

**SVEUČILIŠTE U SPLITU**

**EKONOMSKI**

**FAKULTET**




**ZAVRŠNI RAD**

# FENOMEN VELIKIH PODATAKA, POSTUPCI PODATKOVNE ANALIZE I OBRADE U PYTHON PROGRAMSKOM JEZIKU

**Mentor:**

**Izv.prof.dr.sc Hell Marko**

**Student :**

**Karlo Jerončić Grba**

**Broj indeksa : 1175610**

**Split, kolovoz 2019.**

# SADRŽAJ


**1.UVOD ..... 4**

**1.1. Definiranje problema istraživanja ..... 4**

**1.2. Ciljevi rada ..... 4**

**1.3. Metode rada ..... 4**

**1.4. Struktura rada ..... 5**

**2. FENOMEN VELIKIH PODATAKA ..... 6**

**2.1 Definicija velikih podataka ..... 6**

**2.2 Razlika između "velikih" i "malih" podataka ..... 8**

**2.3 Implementacijske poteškoće u sustavima Velikih Podataka ..... 11**

**3. VELIKI PODATCI I PYTHON PROGRAMSKI JEZIK ..... 14**

**3.1. Python kao programski jezik za obradu podataka ..... 14**

**3.2. Razvojna okruženja za obradu podataka u Pythonu ..... 17**

3.2.1 Python interpreter ..... 17

3.2.2 Jupyter Project i JupyterLab razvojno okruženje ..... 17

3.2.3 Sypder razvojno okruženje ..... 19

**3.3. Popularne programske biblioteke za obradu podataka u Pythonu ..... 20**

3.3.1 Pandas i Matplotlib programske biblioteke ..... 20

3.3.2 Scikit-learn i SciPy programske biblioteke ..... 21

**4. PRAKTIČNI PRIMJERI OBRADE I ANALIZE PODATAKA U PYTHON PROGRAMSKOM JEZIKU ..... 23**

**4.1. Deskriptivna statistika u Python programskom jeziku ..... 23**

**4.2. Linearna korelacija i regresija u Python programskom jeziku ..... 28**

4.2.1 Koeficijent linearne korelacije i postupak njegovog izračuna ..... 28

4.2.2 Jednostruka linearna regresija i postupak njezinog izračuna ..... 30

4.2.3 Višestruka linearna regresija i postupak njezinog izračuna ..... 34

**4.3. Vizualizacija podataka u Python programskom jeziku ..... 36**

**4.4. Manipulacija podataka u Python programskom jeziku ..... 39**

4.4.1 Jednostavni postupci manipulacije podataka ..... 39

4.4.2 Napredni postupci manipulacije podataka ..... 41

**5. ZAKLJUČAK ..... 45**

**LITERATURA ..... 46**

**PRILOZI ..... 48**

**SAŽETAK** ..... **50**
**SUMMARY** ..... **51**

# 1.UVOD

## 1.1. Definiranje problema istraživanja

Ekspanzija interneta i razvoj tehnologije u 21. stoljeću prouzrokovao je pravu revoluciju u privatnom i poslovnom okruženju. Jedna od posljedica takvih promjena je masovna količina podataka koju takvi integrirani sustavi i uređaji generiraju. Zbog navedenih promjena javni i privatni subjekti bit će prinuđeni prilagoditi se promjenama tako da u svoje poslovanje implementiranju procedure i sustave koji će im pomoći da obrade i analiziraju velike skupove generiranih podataka. Konačni cilj implementacije specijaliziranih procedura i sustava za analizu podataka je kvalitetnije i dugoročno uspješnije donošenje strateških i operativnih odluka u modernom poslovnom okruženju, te stjecanje kompetitivne prednosti na tržištu.

## 1.2. Ciljevi rada

Što je fenomen velikih podataka i na koji će način primjena koncepta velikih podataka utjecati na gospodarske i institucionalne subjekte. Nadalje, cilj rada je navesti i pojasniti konkretne programske biblioteke i alate koji se koriste u praksi implementirajući ih kroz programski jezik Python, te u konačnici na konkretnim primjerima pokazati fleksibilnost i moć koju dotični programski jezik pruža u manipulaciji, obradi i analizi podataka, pritom koristeći alate obrađene u teoretskom dijelu

## 1.3 Metode rada

U izradi teorijskog dijela ovog rada korištene su: metoda analize, metoda deskripcije i metoda dedukcije. Izvori su isključivo sekundarne prirode – znanstvene knjige, članci i stručni radovi. U izradi praktičnog dijela korištene su: metoda analize, metoda deskripcije, metoda sinteze i metoda modeliranja. Također se obilato služim službenim web stranicama pojedinih programskih biblioteka i podatkovnih izvora, budući da se na njima nalazi službena dokumentacija koja se konstantno nadopunjuje.

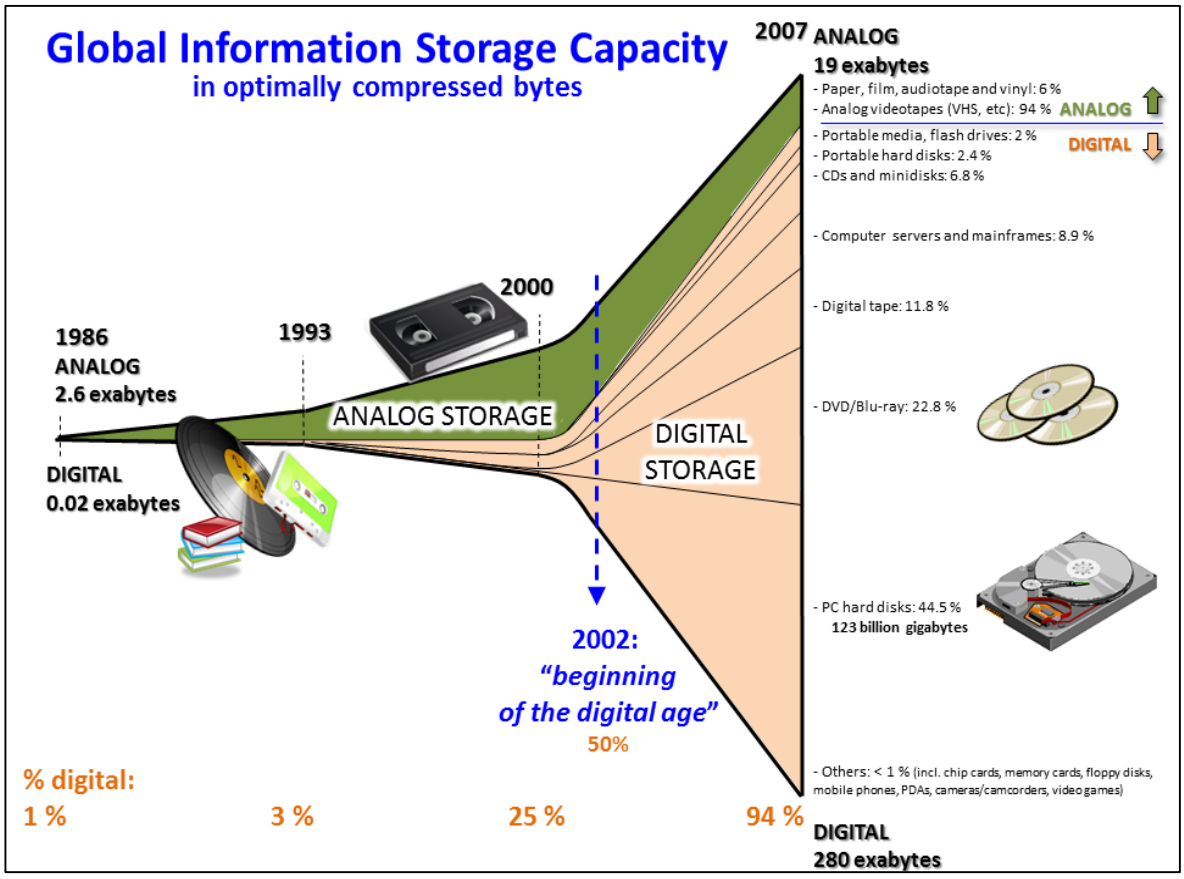
## 1.4. Struktura rada

Struktura završnog rada se sastoji od pet međusobno povezanih cjelina, odnosno poglavlja. Svako poglavlje sadrži potpoglavlja. U uvodnom dijelu rada razrađen je problem istraživanja, ciljevi rada, metode rada i struktura rada. U drugom dijelu rada definirani su osnovni pojmovi o fenomenu velikih podataka. Nadalje, u trećem i četvrtom djelu se analiziraju alati iz prakse te njihova primjena na konkretnom primjeru. U završnom dijelu rada slijede zaključak, korištena literatura te popis slika, grafikona i tablica.

# 2. FENOMEN VELIKIH PODATAKA

## 2.1 Definicija velikih podataka

Svakog dana milijarde umreženih uređaja proizvode enormne količine podataka. Procjenjuje se da ukupna količina pohranjenih podataka na računalima iznosi cca. 300 egzabajta odnosno 300 milijardi gigabajta. Zahtjevi za podatkovnim skladištenjem povećavaju se po stopi od 28% godišnje. Podatci koji se skladište su zanemarivi u odnosu na one koji se transmitiraju bez skladištenja. Godišnja transmisija podataka iznosi 1.9 zetabajta odnosno 1900 milijardi gigabajta.[1] Iz ovako postavljenih odnosa, rodio se novi fenomen; fenomen velikih podataka popularno poznat samo kao „Big Data“ .



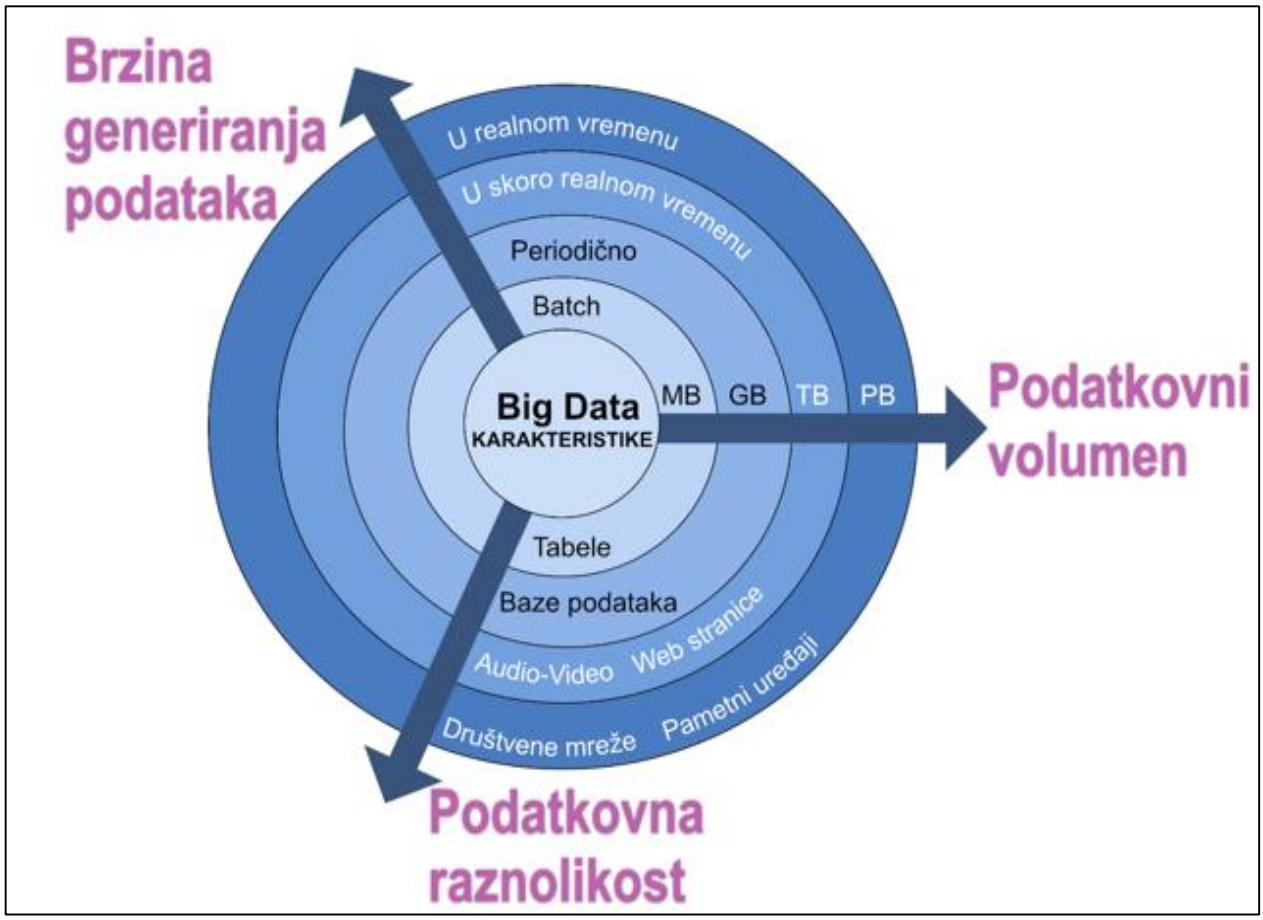

**Slika 1: Količine generiranih podataka i kapacitet uređaja kroz godine**

Izvor: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. Science, 332(6025), 60 –65. doi:10.1126/science.1200970

[1] Martin Hilbert M, Lopez P. (2011). The world's technological capacity to store, communicate, and compute information. Science, str. 332:60–5.

Što je točno fenomen velikih podataka ili „Big Data“? Fenomen velikih podataka karakteriziraju 3V odnosno[2]:

- Volumen ( velike količine podataka )
- Varijacija ( drugačiji tipovi i forme podataka, što uključuje baze podataka, dokumente, slike, kartoteke…)
- Brzina ( konstanta akumulacija novih podataka, te digitalizacija starih podataka )




**Slika 2: Karakteristike velikih podataka**
Izvor: autor

[2] Schmidt S. (2012). Data is exploding: the 3V's of Big Data. Business Computing World.

U svojoj srži, fenomen velikih podataka ima skoro pa samo jednu ulogu, a to je pravovremena i točna informacija kao podloga za kvalitetne procjene u poslovanju. Iako se fenomen velikih podataka često stavlja u kontekst umjetne inteligencije. Takva karakterizacija je često zbunjujuća; fenomen velikih podataka ne vrti se oko toga da se neki uređaj odnosno sustav primjenom metoda strojnog učenja nauči misliti ili se ponašati kao čovjek, već da se primjenom matematičkih, statističkih i računalnih metoda na velike skupove podataka dođe što točnijih vjerojatnosti u procesu odlučivanja kao na primjer: vjerojatnosti da neki program sadrži maliciozni kod ili da na osnovu kupovne povijesti potrošača kreiramo što bolju predikciju za njegovo buduće ponašanje.

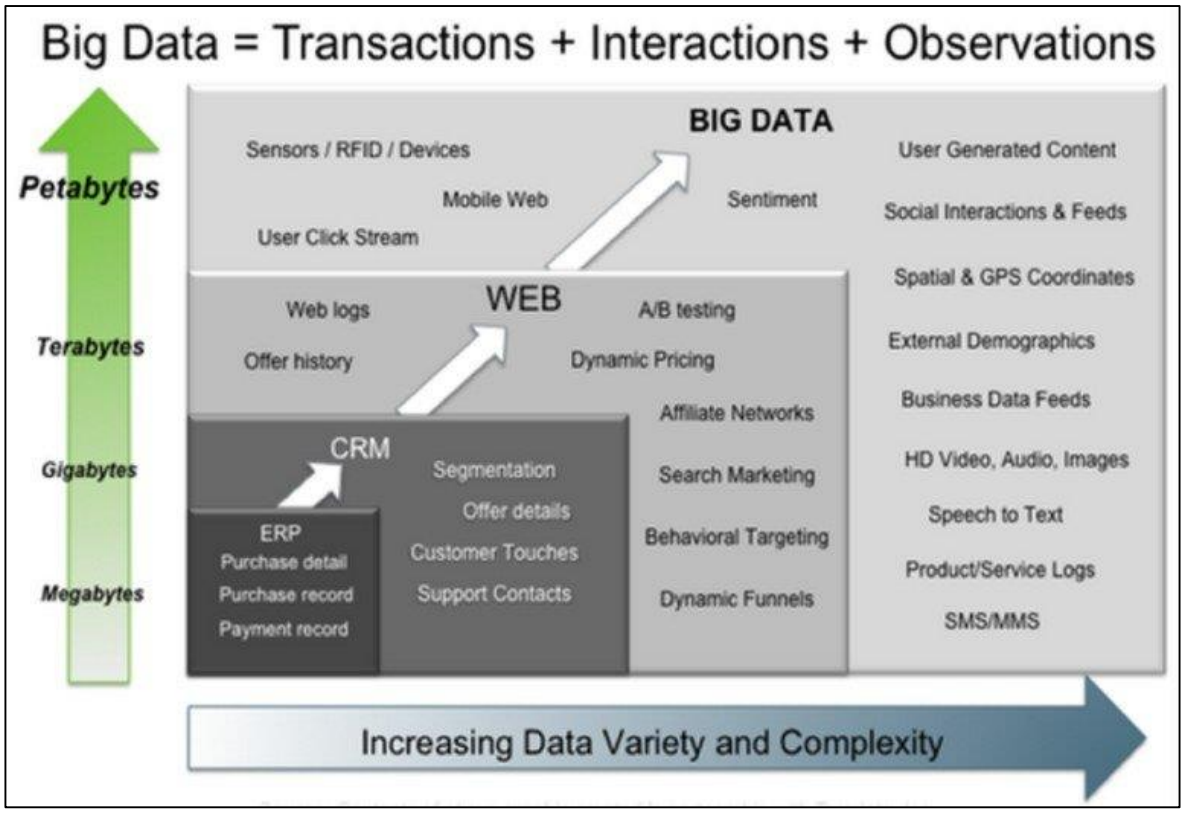
U budućnosti, znatno prije nego što prosječan čovjek projicira, mnogi aspekti svijeta u kojem živimo i u kojem ljudski faktor pri donošenju važnih odluka igra odlučujuću odluku bit će potpomognuti ili zamijenjeni računalnim sustavima. Ovdje ne govorimo samo o aplikacijama za prijevoz ili o autonomnim sigurnosnim sustavima u automobilu, već i o znatno intelektualno intenzivnijim zadaćama. Ako se malo osvrnemo na današnji situaciju u visoko digitaliziranom svijetu u kojem najveće tehnološke korporacije poput Facebook-a i Google-a prikupljaju masovnu količinu podataka o nama; pitanje je vremena kada će se ista tehnologija početi masovno primjenjivati i u drugim industrijskim granama kao npr. u medicini da bi se pravovremeno dijagnosticirala i prevenirala neka bolest ili u sigurnosnom sektoru gdje će kriminalistički sustav prepoznati kazneno djelo prije njegovog počinjenja. Baš kao što je i pojava Interneta iz temelja promijenila način na koji današnji svijet funkcionira, tako će i fenomen Velikih Podataka iz temelja izmijeniti naše živote davajući matematičku dimenziju svakom naizgled nebitnom podatku prikupljenom o nama.[3]

## 2.2. Razlika između „velikih“ i „malih“ podataka

Velikim podatcima ne možemo smatrati male količine podataka koje su s vremenom mutirale u smislu podatkovne veličine tako da se ne mogu otvoriti u klasičnim programima za obradu podataka kao što je na primjer Excel ili u nekoj bazi podataka koja nadilazi resurse servera na kojem je skladištena. Takva uglavnom pogrešna percepcija česta je kod stručnjaka koji smatraju da se primjenom postojećih znanja i metoda mogu analizirati kompleksniji i veći podatkovni skupovi, što kasnije može

[3]Viktor Mayer-Schönberger , Kenneth Cukier (2014). Big Data: A Revolution That Will Transform How We Live, Work, and Think, str. 6-14

dovesti do iznenadnih financijskih troškova i loših ishoda zbog krivo donešenih odluka. U tom kontekstu u daljnjem tekstu valja razmotriti fundamentalnu razliku između tzv. velikih i malih podataka, razlika se najbolje očituje na slici br.3 koja nam na Y osi prikazuje količinu podataka koju generiraju manji sustava, a na X osi podatkovnu raznolikost i kompleksnost koju postižu sustavi u većim kategorijama.




**Slika 3: Kategorizacija i razlike između podatkovnih sustava**
Izvor: Moniruzzaman, A B M & Hossain, Syed. (2013). NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. Int J Database Theor Appl. 6.

Osnovne razlike između velikih i malih podataka očituju se u[4]:

- kontekstu **cilja** koji se želi postići:

  mali podatci obično su osmišljeni kako bi odgovorili na točno određena poslovna pitanja, s druge strane veliki podatci osmišljeni su s ciljem dugoročne upotrebe i vremenske fleksibilnosti. Na plastičnom primjeru to bi značilo da u kontekstu velikih podataka prikupljamo i obrađujemo podatke iz npr. : pravosuđa, policije, državnog odvjetništva i odvjetničke komore; kako bismo mogli provesti više

[4]Jules J. Berman (2018). Principles and Practice of Big Data: Preparing, Sharing, and Analyzing Complex Information 2nd Edition, str. 16-21

studija o utjecaju i implementaciji potencijalnih novih zakonskih rješenja te praćenju postojećih.

- kontekstu **lokacije**:

  mali podatci obično se nalaze unutar jedne institucije, odnosno na jednom serveru, a ponekad se nalaze i u samo jednoj datoteci. S druge strane veliki podatci disperzirani su na više internetskih servera i u više institucija.

- kontekstu **strukture i sadržaja:**

  mali podatci obično su značajno strukturirani, odnosno dolaze iz jedne discipline ili sektora, što znači da je njihova struktura obično uniformirana i jednolična. Veliki podatci s druge strane nisu strukturirani, odnosno podatci dolazi iz više izvora u raznim podatkovnim strukturama i formatima.

- kontekstu **pripreme podataka:**

  kada je riječ o malim podatcima, u većini slučajeva korisnik priprema samostalno svoje podatke koje obično koristi za osobne potrebe. Kod velikih podataka podatci dolaze iz više izvore i od strane više korisnika, odnosno sama osoba koja ih obrađuje rijetko kad sudjeluje u njihovoj pripremi i generiranju.

- kontekstu **životnog vijeka podataka:**

  životni vijek malih podataka obično se svodi na životni vijek samog projekta, odnosno nakon što se ispunila svrha oni se često permanentno brišu. Kod velikih podataka skladištenje podataka mora biti dugoročno, jer je njihova upotreba višenamjenska i za buduća istraživanja potrebno je imati prethodne podatke.

- kontekstu **financijskih ulaganja:**

  kod malih podataka financijski troškovi su ograničeni, te se subjekti korištenja mogu relativno brzo oporaviti od gubitka podataka. Kod primijene koncepta velikih podatka, gubitak podataka ili nepredviđena katastrofa mogu dovesti do bankrota subjekta i značajnih financijskih gubitaka.

- kontekstu **analize:**

  mali podatci se relativno lako i brzo analiziraju, obično u jednom koraku i upotrebom jednog postupka. Kod velikih podataka, uz par iznimaka, analiza podataka se obično odvija u više koraka. Podatke je potrebno: skladištiti, pripremiti za obradu, provjeriti, analizirati te interpretirati

## 2.3. Implementacijske poteškoće u sustavima Velikih Podataka

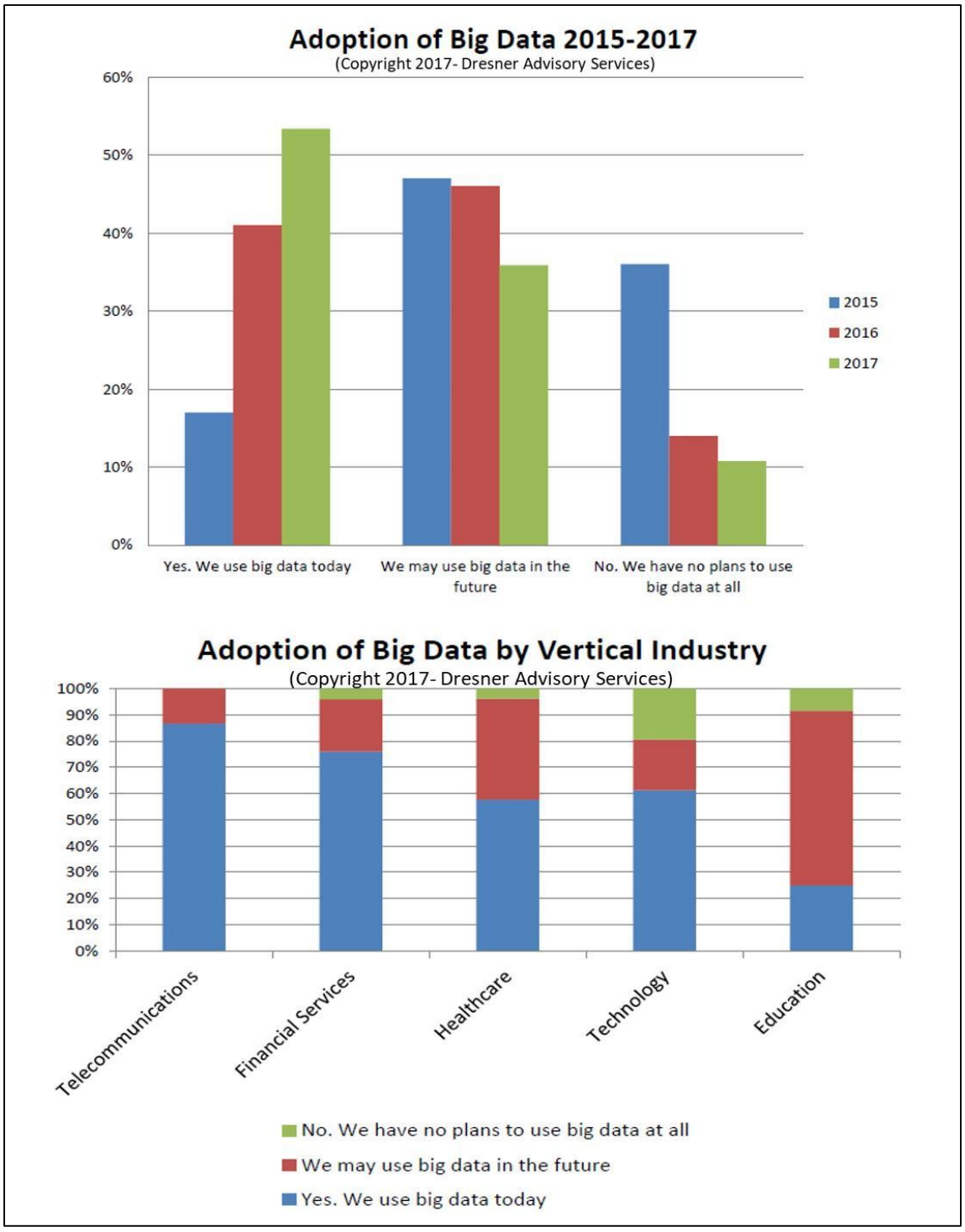
Sustavi Velikih Podataka iznimno su kompleksni; potrebno je mnogo vremena za njihovo izgrađivanje i održavanje, ako dođe do poteškoća ili grešaka u sustava Velikih podataka rješenja često nisu jeftina i jednostavna za otkloniti.
Većina poteškoća u sustavima velikih podataka ne dolazi od slučajnih nezgoda, već od neadekvatnih i nepotpuno implementiranih rješenja prilikom izrade sustava, odnosno sam sustav ne zadovoljava potrebne kriterije da bi ušao u operativnu uporabu. Do poteškoća najčešće dolazi zbog: loše selekcije ljudskih resursa, premalo ili previše financijskih ulaganja, poteškoća s intelektualnim vlasništvom, loše kvalitete ulaznih podatka i slabih sigurnosnih politika u sustavu.[5]

Čest problem u adekvatnoj izradi sustava velikih podataka jeste taj što se radi o relativno novom konceptu, pa stoga na tržištu skoro uvijek manjka kadra s adekvatnim vještinama i dugogodišnjim stručnim iskustvom, također procjenjuje se da će u budućnosti zbog rasta potražnje, stručnjaci za obradu i analizu podataka postati iznimno deficitarna radna snaga.[6]

Ako se neko poduzeće i odluči na implementaciju sustava velikih podataka, u svome financijskom planu mora osigurati značajna financijska sredstva, što predstavlja značajni izazov i za velike korporativne entitete, a kamo li za start-up poduzeća koja imaju velikih problema s manjkom likvidnosti.

Nestručna implementacija sustava Velikih podataka često može biti pogubna za poduzeće, ne samo u slučajevima iznenadnih nesreća i tehničkih poteškoća, već i ukoliko analitički dio podatkovnog tima zbog krivo postavljenog sustava ne dobiva dovoljno kvalitetne i uporabljive informacije, dolazi do krive interpretacije u završnoj obradi i u analitičkim izvješćima, što za posljedicu može imati krivo donošenje strateških i operativnih odluka u drugim segmentima poslovanja i rezultirati gubitkom tržišne pozicije ili financijskim krahom u konačnici.

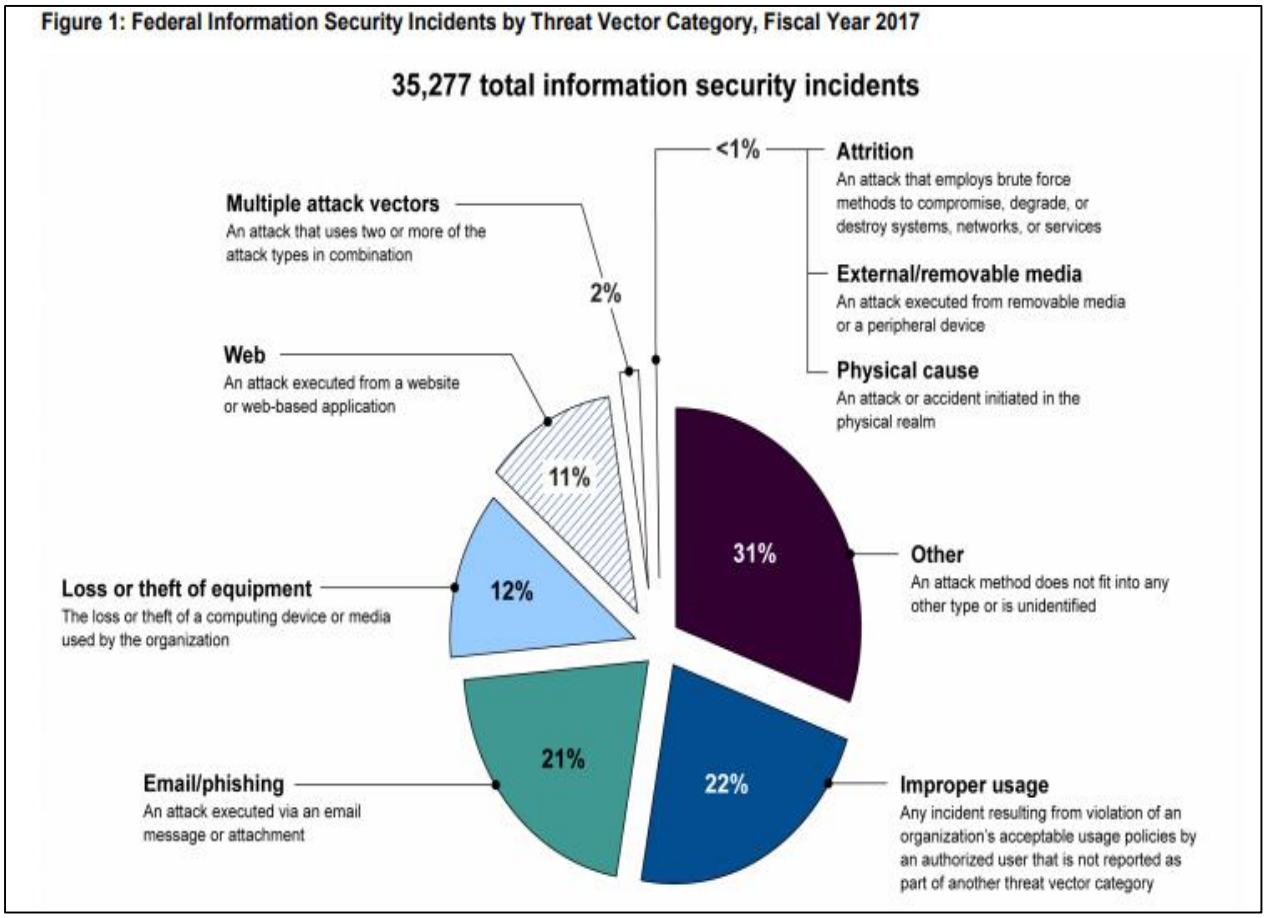
[5] Jules J. Berman (2018). Principles and Practice of Big Data: Preparing, Sharing, and Analyzing Complex Information 2nd Edition, str. 326-329
[6] Team, E. (2018). Infographic: The Data Scientist Shortage - insideBIGDATA. [online] insideBIGDATA. Dostupno na: https://insidebigdata.com/2018/08/19/infographic-data-scientist-shortage/ [Pristupljeno 18 Lipanj. 2019].




**Grafikon 1: Spremnost na implementaciju sustava velikih podataka i njihova integriranost po industrijskim granama**

Izvor: Dresner Advisory Services. (2017). 2017 Big Data Analytics Market Study. part of the Wisdom of Crowds® series of research. The 3rd annual report.

Grafikon br. 1 prikazuje nam spremnost poduzeća na implementaciju sustava velikih podataka, vidljivo je da se najveći korisnici takvih sustava uglavnom nalaze u telekomunikacijskom i financijskom sektoru, dok je najlošija situaciju u obrazovnim institucijama koje često nemaju adekvatno financiranje i potreban stručni kadar.




**Grafikon 2: Broj prijava i vrsta IT nezgoda od strane Američkih federalnih agencija**
Izvor: United States Government Accountability Office, Report to Congressional Committees. (2017). INFORMATION SECURITY Agencies Need to Improve Implementation of Federal Approach to Securing Systems and Protecting against Intrusions.

Značajnu prijetnju sustavima Velikih podataka predstavljaju i sve učestaliji kibernetički napadi, čija uspješnost osim velike materijalne štete i pauziranja poslovanja dok se sustav ne normalizira može rezultirati i velikim posljedicama po ljudsko zdravlje, pa čak i smrću. Naime, napadom kriptovirusa WannaCry na britansko ministarstvo zdravstva stručnjaci za kibernetičku sigurnost procjenjuju da je prilikom blokade sustava došlo do otkazivanja čak dvadeset tisuća pregleda, dok profesor računalnih znanosti s fakulteta u Swanea-u Harold

Thimbleby procjenjuje da godišnje čak devetsto ljudi umre zbog neadekvatno implementiranih računalnih sustava i grešaka koje rezultiraju od posljedica po osnovi toga.[7]

Grafikon br.2 prikazuje nam količinu prijavljenih IT nezgoda od strane američkih federalnih agencija i njihovu kategorizaciju po vrstama, vidimo da unatoč uvriježenom mišljenju kako većina IT nezgoda dolazi zbog eksternih prijetnji to ovdje nije slučaj, skoro je jednak broj IT nezgoda uzrokovanih nestručnim rukovanjem i nepažnjom s onima koji su uzrokovani eksternim utjecajem.

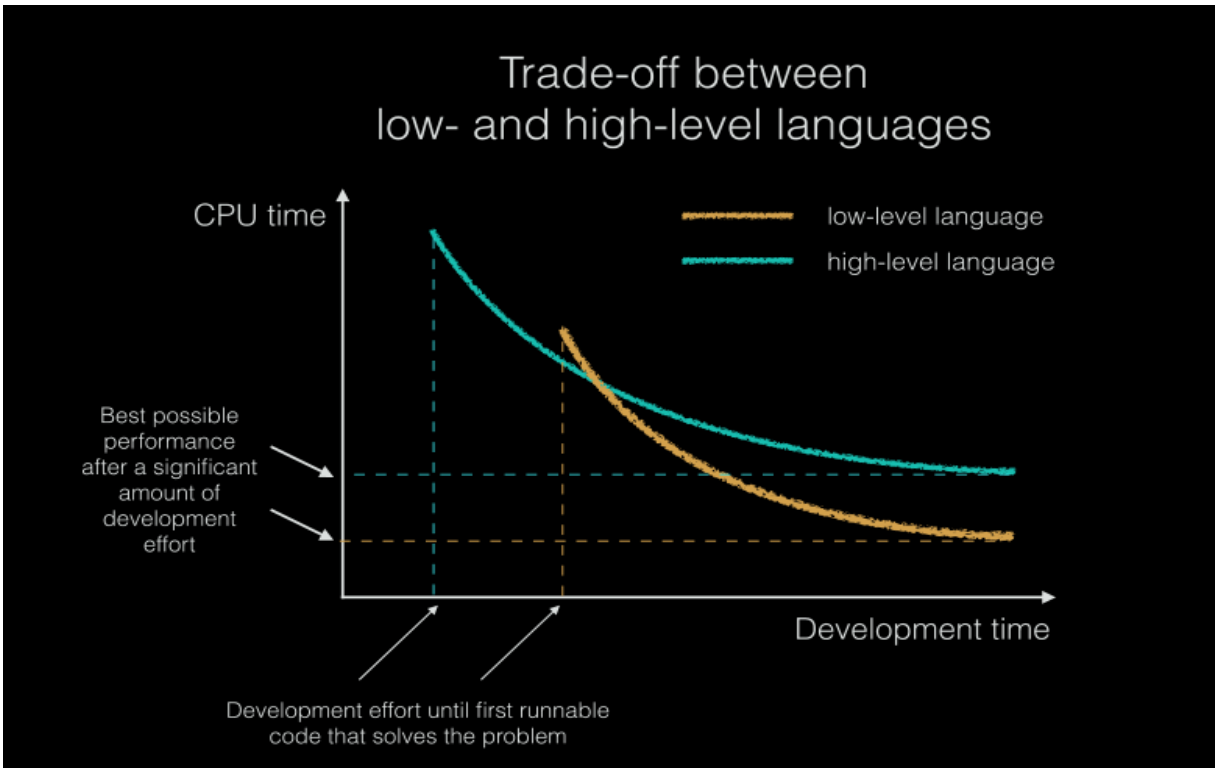
# 3. VELIKI PODATCI I PYTHON PROGRAMSKI JEZIK

## 3.1. Python kao programski jezik za obradu podataka

U računalnom inženjerstvu jedna od važnijih stvari je sposobnost da računalni algoritmi brzo i efikasno rješavaju zadane probleme, ali i da je njihova implementacija što jednostavnija i jeftinija. Prirodno je težiti optimiziranom računalnom kodu odnosno programskom jeziku koji najmanje opterećuje računalne resurse i pruža velik stupanj pouzdanosti. U određenim situacijama korištenje takvih programskih jezika kao što su npr. : C, Fortran ili ADA neizbježno je da bi se dobila maksimalna efikasnost i efektivnost korištenih resursa u kombinaciji s visokom pouzdanošću sustava u kritičnim trenutcima kao što je to npr.: slučaj u zrakoplovstvu. No nije uvijek mudro da se takvi programski jezici koristi i za velik broj ostalih zadaća koju mogu odraditi drugi programski jezici s puno manje linija koda, što u modernim poduzećima nije uvijek slučaj. Prije svega potrebno je razmotriti vrijeme razvoja koje je potrebno da bi se određeni problem riješio u nekom programskom jeziku, korištenje tzv. low level programskih jezika u rješavanju problema koji ne trebaju ekstremno visoku pouzdanost i efikasnost zahtjeva povećanje vremena provedenog u razvoju i moguće dodatne

[7] Laura Donnelly (2018). More than 900 NHS deaths yearly may be caused by IT failings . [online] The Telegraph. Available at: https://www.telegraph.co.uk/news/2018/02/06/900-nhs-deaths-yearly-may-caused-failings/ [Pristupljeno 19 Lipanj. 2019].

troškove. Tu na scenu stupaju tzv. High-level programski jezici koji jednostavnijom sintaksom, fleksibilnošću i lakšim sustavom održavanja značajno skraćuju vrijeme provedeno u razvoju odnosno rješavanju određenoga problema.[8]



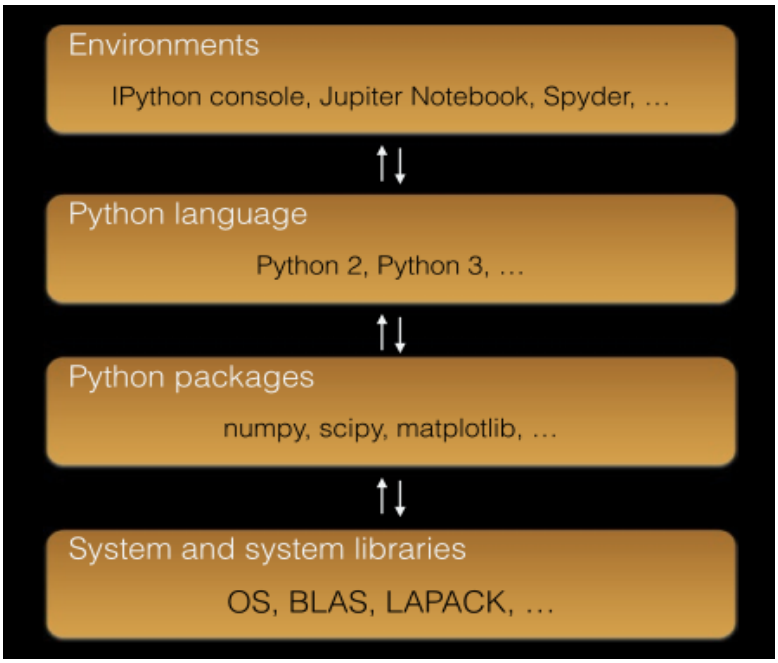

**Grafikon 3: „Trade-off“ u računalnim resursima i vremenu razvoja između programskih jezika više i niže razine**
Izvor: Robert Johansson, Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib 2nd ed. Edition

Grafikon br. 3 nam prikazuje razliku, odnosno kompromis između programskih jezika više i niže razine. U posljednje vrijeme programski jezici više razine postali su iznimno popularni čak i u područjima gdje se prije zahtijevala značajna ušteda u računalnim resursima, razlog tomu je prvenstveno to što je procesorska snaga koja se mjeri u MIPS/$ (Millions of Instructions Per Second) postala sve jeftinija zbog kompetitivnijeg tržišta procesorskih proizvođača, s druge strane ljudsko znanje, odnosno ljudski kapital postao je sve skuplji po satu rada za računalne znanosti zbog eksponencijalnog rasta Interneta.[9]

[8] Robert Johansson (2018). Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib 2nd ed. Edition, str. 1-2

[9] AI Impacts. (2015). Trends in the cost of computing. [online] Dostupno na: https://aiimpacts.org/trends-in-the-cost-of-computing/ [Pristupljeno 19 Lipanj. 2019].

U velikom broju slučajeva rješenje se nazire u korištenju tzv. višejezičnih programskih modela gdje se za određene zadaće koje zahtijevaju visoku razinu pouzdanosti koriste programski jezici niže razine, a za ostale dijelove programski jezici više razine. U takvoj vrsti integracije najbolje prolaze programski jezici poput Python-a koji je postao sinonim za tzv. programski jezik koji služi kao ljepilo koje spaja vremenski kritične i kompjutacijski intenzivne zadaće s jednostavnim korisničkim interfejsom. Iz ove pretpostavke razvio se moderni Python programski jezik, koji je postao puno više od navedene uloge spajalice, naime, posljednjih godina razvio se i zaživio cijeli ekosustav popratnih razvojnih okruženja i programskih biblioteka. Velik broj ovih biblioteka „ispod površine“ pisane su u nekom jeziku niže razine da bi imali najbolje performanse, dok je njihova kontrola maksimalno pojednostavljena korištenjem jednostavne i pregledne sintakse Python-a. Upravo ovakav ekosustav i navedene činjenice razlog su uspjeha Python-a koji je postao jedan od najpopularnijih programskih jezika današnjice.[10]




**Slika 4: Python-ov ekosustav**
Izvor: Robert Johansson, Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib 2nd ed. Edition

Slika br. 4 najbolje ilustrira Python-ov ekosustav, gdje se na vrhu lanca nalaze razvojna okruženja u kojima se piše izrazito jednostavnom i preglednom sintaksom Python

[10] Robert Johansson (2018). Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib 2nd ed. Edition, str. 2-4

programskom jezika, a na dnu lanca nalaze se sistemske biblioteke pisane u nižim programskim jezicima koje vrše direktnu interakciju s operativnim sustavom.

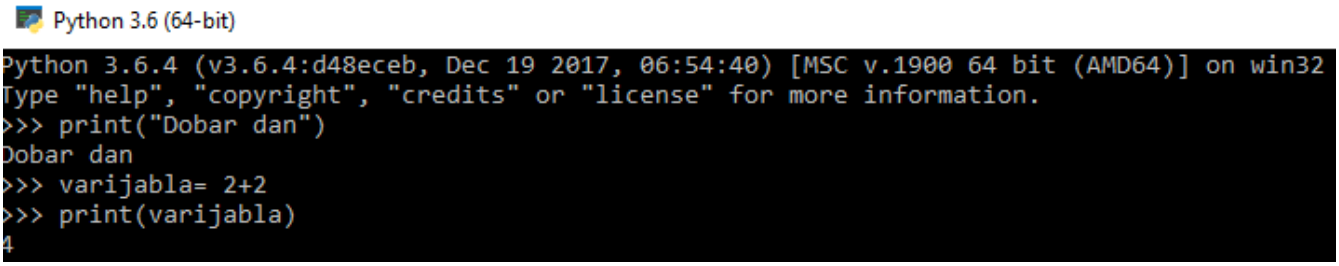
Valja napomenuti i važnu činjenicu da je većina razvojnih okruženja i programskih biblioteka otvorenog koda odnosno dolaze s besplatnim licencama za intelektualno vlasništvo, što za onoga koji razvija aplikaciju predstavlja značajne uštede u tom segmentu razvoja.

## 3.2. Razvojna okruženja za obradu podataka u Python-u

Programski jezik Python ima bezbroj razvojnih okruženja. Mi ćemo se ovdje fokusirati na samo ona koja dolaze s besplatnom licencom i otvorenog su koda.

### 3.2.1 Python interpreter

Prvo i najprimitivnije razvojno okruženje je tzv. Python interpreter koji dolazi standardno s instalacijom programskog paketa skinutog sa službene stranice.




**Slika 5: Python interpreter**
Izvor: autor

Kao što je prikazano na slici br. 5, Python interpreter dolazi u tzv. konzolnom okruženju gdje se upisuju programski kod i već u idućem retku vrši njegova egzekucija. Kao što možete i pretpostaviti skoro nitko ne piše programe koristeći ovakvo razvojno okruženje, već se ono uglavnom koristi kao uvod u Python programski jezik. [11]

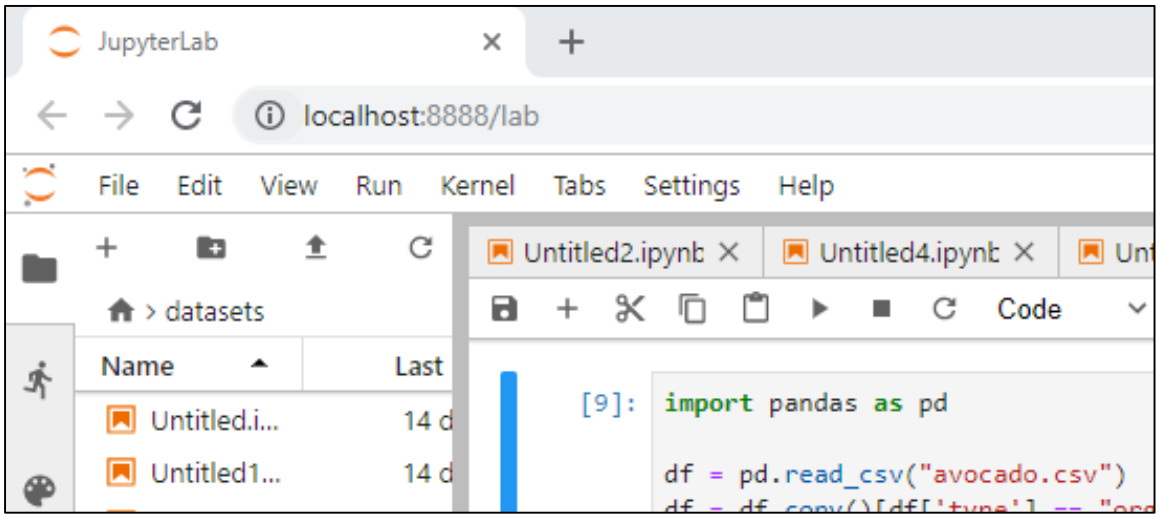
### 3.2.2 Jupyter Project i JupyterLab razvojno okruženje

Jupyter project je neprofitna inicijativa kreirana od strane cjelokupne programerske zajednice, ali prvenstveno se može smatrati kao nastavak IPython projekta, unutar sebe sadrži nekoliko razvojnih okruženje kao što su: JupyterLab, Jupyter Notebook i Qt Console. Ova razvojna okruženja prvenstveno su bila namijenjena akademicima i znanstvenicima, ali se danas praktički koriste u svakoj industrijskoj grani za podatkovnu analizu, obradu i vizualizaciju.

[11] Docs.python.org. (2019). Using the Python Interpreter — Python 3.7.4rc1 documentation. [online] Dostupno na: https://docs.python.org/3/tutorial/interpreter.html [Pristupljeno 15 Lipanj. 2019].

Prednost razvojnih okruženja koja sadrži Jupyter Project prvenstveno leži u tome što svako razvojno okruženje sadrži na stotine integriranih biblioteka, što značajno olakšava posao programerima. Također možda i najveća prednost ovoga projekta leži u tome što u njemu sudjeluju programeri iz cijelog svijeta te se on konstantno nadograđuje novim zakrpama i poboljšanjima. [12]

U praktičnom dijelu ovoga radu koristit ćemo JupyterLab okruženje, koje pogoni IPython jezgra, odnosno možemo reći da je IPython jezgra tzv. „backend“ dio Jupytera, koji nam omogućava brzu i efikasnu interakciju između biblioteka i operativnog sustava u pozadini.




**Slika 6: JupyterLab**
Izvor: autor

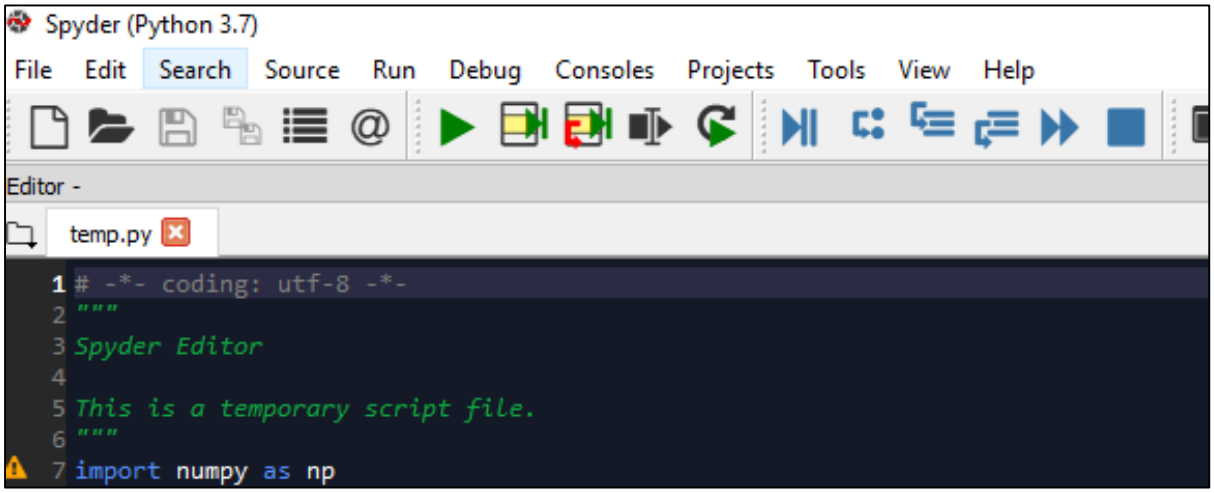
Kao što možemo vidjeti na slici br. 6, sam JupyterLab pogoni se kao razvojno okruženje na web sučelje, odnosno jezgra prilikom paljenja razvojnog okruženja pokreće server na lokalnoj adresi, na kojeg se korisnik spaja u web pregledniku i pristupa razvojnom okruženju. JupyterLab ima tzv. ćelije u koje se unosi programski kod i vrši momentalna egzekucija upisanog; valja napomenuti da su ćelije međusobno memorijski povezane, odnosno ono što se deklariralo kao npr. neka varijabla u prvoj ćeliji ostaje u memoriji cijelo vrijeme rada na tom radnom listu.

[12] Jupyter.org. (2019). Project Jupyter. [online] Dostupno na: https://jupyter.org/about [Pristupljeno 15 Lipanj.

### 3.2.3 Sypder razvojno okruženje

Spyder razvojno okruženje također spada u jedno od popularnijih razvojnih okruženja otvorenog koda. Ono također kao i JupyterLab koristi IPython kao svoju jezgru, međutim između JupyterLab-a i Spydera-a postoji nekoliko razlika koje treba razmotriti[13]:

- Spyderovo razvojno okruženje ima puno kvalitetniju i bolju „AutoComplete“ funkciju koja nekad zna značajno olakšati i ubrzati programiranje
- Spyder posjeduje kvalitetniju „debugging“ funkciju koja omogućava korisniku da vrši egzekuciju koda liniju po liniju, te puno kvalitetnije uočava možebitne pogreške i nedostatke
- Spyder posjeduje „object-inspector“ koji izlistava dokumentaciju za svaki objekt određene programske biblioteke
- kao što je vidljivo na slici br. 7 Spyder dolazi kao zaseban program, što ga čini manje mobilnim od JupyterLab-a, naime kod JupyterLab-a korisnik može ostaviti server upaljen i na njega se spojiti s udaljene lokacije, što kod Spyder-a nije slučaj.




**Slika 7: Spyder IDE**
Izvor: autor

[13] PyPI. (2019). spyder. [online] Dostupno na: https://pypi.org/project/spyder/ [Pristupljeno 20 Lipanj. 2019].

## 3.3. Popularne programske biblioteke za obradu podataka u Pythonu

U ovom poglavlju bit će samo načelo teoretski obrađene programske biblioteke koje ćemo koristiti u praktičnom dijelu.

### 3.3.1 Pandas i Matplotlib programske biblioteke

Pandas programska biblioteka jedna je od najpoznatijih biblioteka otvorenog koda za podatkovnu obradu i manipulaciju. Ova biblioteka pruža procesiranje podatkovnih struktura koje na brz i fleksibilan način omogućuju rad s raznim vrstama podataka i vremenskim serijama podataka.

Pandas biblioteka dizajnirana je za sljedeće vrste podataka[14]:

- Tabelarni podatci kao Excel i SQL strukturirane tablice
- Sortirani i nesortirane vremenske serije podataka
- Heterogeni i homogeni matrični podatci iz raznih podatkovnih struktura

Pandas biblioteka pogodna je za sljedeće vrste operacija:

- Jednostavno procesiranja podataka koji nedostaju u određenom podatkovnom skupu.
- Operacije podatkovne manipulacije: stupci se mogu umetati i brisati iz podatkovnog okvira.
- Jednostavno pretvaranja podataka iz jedne podatkovne strukture u drugu
- Pametno rezanje i indeksiranje podataka
- Spajanje i pripajanje podatkovnih skupova
- Specijalne funkcije za vremenske serije podataka omogućuju jednostavnu podatkovnu analizu i obradu na vremenskim serijama

Podatke obrađene pomoć u Pandas biblioteke najčešće se grafički prikazuje Matplotlib bibliotekom, koja je i ujedno najpopularnija biblioteka za grafičko prikazivanje. Najveća prednost ove biblioteke leži u njezinoj raznovrsnosti, univerzalnosti i jednostavnosti korištenja.

[14] PyPI. (2019). pandas. [online] Dostupno na: https://pypi.org/project/pandas/ [Pristupljeno 20 Lipanj. 2019].

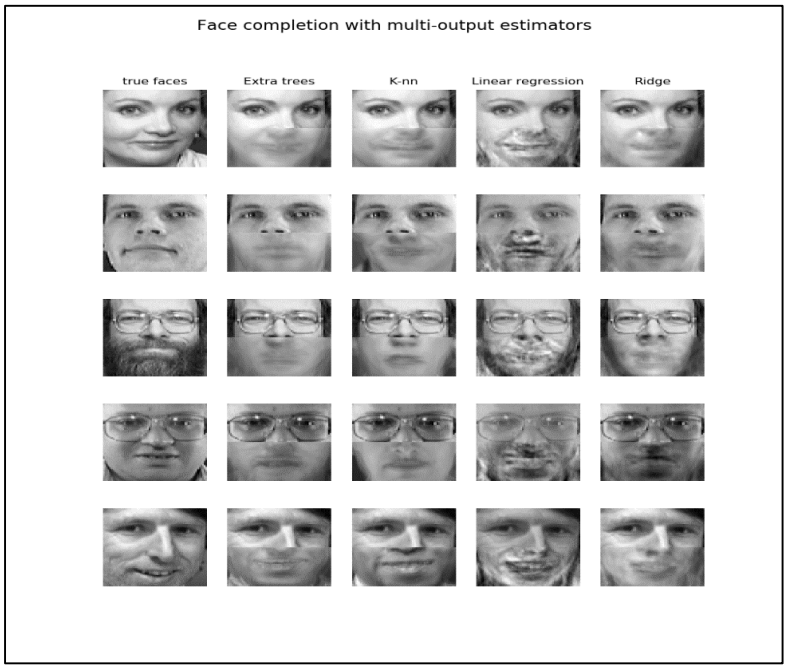
### 3.3.2 Scikit-learn i SciPy programske biblioteke

Scikit-learn najpopularnija je biblioteka otvorenog koda za strojno učenje, no mi ćemo je u praktičnom dijelu uglavnom koristiti za linearnu regresiju.

Scikit-learn pogodna je za sljedeće vrste operacija[15]:

- Klasifikaciju - identifikacija objekata po kategoriji (detekcija spam-a, prepoznavanje slika)
- Regresija -linearna jednostruka i višestruka regresija (strojno učenje)
- Clustering - automatsko grupiranje sličnih objekata u setove (segmentacija kupaca, grupiranje ishoda eksperimenata)
- Redukcija dimenzionalnosti – smanjenje broja nasumičnih varijabli prilikom odabira (uglavnom kod strojnog učenja – PCA redukcija)
- Selekcija modela – poboljšanje preciznosti kroz podešavanje parametara
- Pred procesiranje – transformacija i obrada ulaznih podataka za strojno učenje

Na slici br.8 vidljivo je korištenje Scikit-learn biblioteke koja pomoću algoritama strojnog učenja kompletira slike ljudskog lica.




**Slika 8: Primjer korištenja scikit-learn biblioteke**
Izvor: https://scikit-learn.org/stable/auto_examples/plot_multioutput_face_completion.html

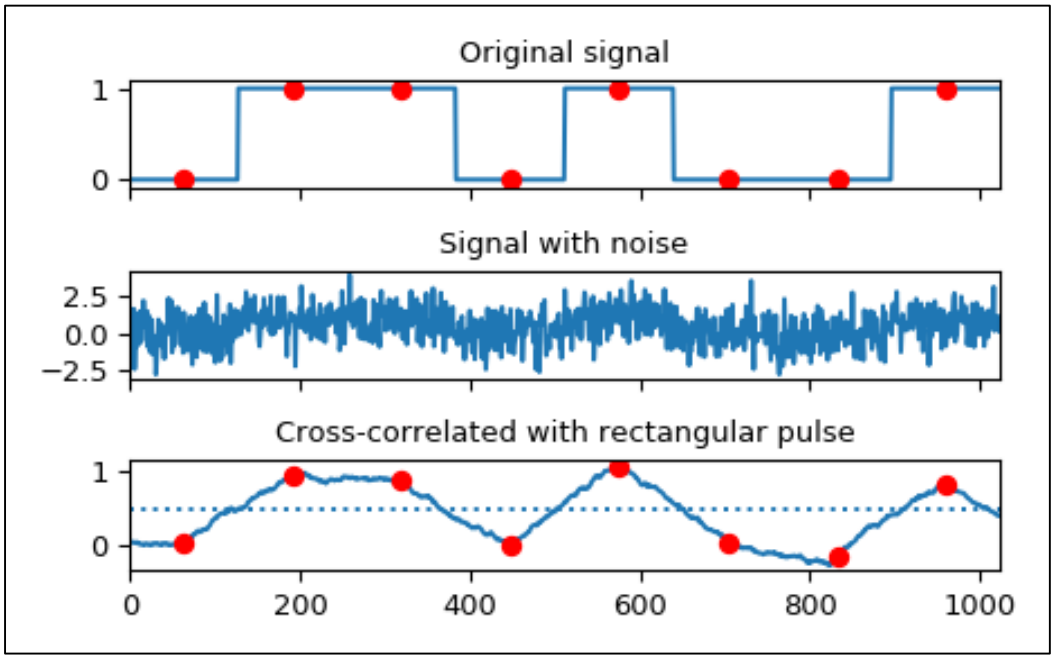
[15] Scikit-learn.org. (2019). scikit-learn: machine learning in Python — scikit-learn 0.21.2 documentation. [online] Dostupno na: https://scikit-learn.org/stable/ [Pristupljeno 21 Lipanj. 2019].

SciPy ili Scientific Python biblioteku koristit ćemo za određene funkcije deskriptivne statistike i korelaciju.

SciPy biblioteka pogodna je za sljedeće vrste operacija[16]:

- Linearnu algebru
- Procesiranje signala
- Fourierove transformacije
- Matematičku interpolaciju
- Statistiku
- Multidimenzionalno procesiranje slika

Na slici br.9 možemo vidjeti jedan od primjera korištenja biblioteke SciPy prilikom obrade signala..




**Slika 9: Primjer korištenja SciPy biblioteke**
Izvor: https://docs.scipy.org/doc/scipy-1.2.1/reference/generated/scipy.signal.correlate.htmll

[16] Docs.scipy.org. (2019). SciPy — SciPy v1.3.0 Reference Guide. [online] Dostupno na: https://docs.scipy.org/doc/scipy/reference/ [Pristupljeno 21 Lipanj. 2019].

# 4. PRAKTIČNI PRIMJERI OBRADE I ANALIZE PODATAKA U PYTHON PROGRAMSKOM JEZIKU

Podatke koji se koriste u sljedećim primjerima preuzimat ćemo s popularne stranice Kaggle[17] koja služi analitičarima i studentima kao izvor organskih ili umjetno generiranih podataka koji se koriste za obrazovanje i usavršavanje vještina iz podatkovne analitike i analize. Također podatke ćemo skidati i s državnog zavoda za statistiku Republike Hrvatske i otvorenog portala podataka vlade RH[18]. Podatci se mogu skinuti u više formata, ovisno radili se o bazama podataka ili tablicama. Cjelokupna analiza vršit će se u integriranom razvojnom okruženju JupyterLab-u koji dolazi s Anaconda[19] integriranim programskim paketom otvorenog koda koji osim JupterLab-a ima integrirano više razvojnih okruženja i na stotine programskih biblioteka.

## 4.1 Deskriptivna statistika u Python programskom jeziku

Deskriptivna statistika je dio matematičke statistike koji se koristi za opisivanje i bolje razumijevanje izmjerenog (ili zadanog) skupa podataka, odnosno deskriptivna statistika opisuje podatke kroz numeričku sumarizaciju, tablice i grafove. Osnovni pojmovi deskriptivne statistike su[20]:

- Suma podataka
- Najveći i najmanji podatak
- Raspon je razlika najvećeg i najmanjeg podatka.
- Aritmetička sredina (prosjek)
- Medijan je srednji podatak. Pola podataka nalazi se iznad, a pola ispod medijana.
- Prvi kvartil (donji kvartil) je broj od kojeg je manje ili jednako 25% podataka.

[16] Kaggle.com. (2019). Kaggle: Your Home for Data Science. [online] Dostupno na: https://www.kaggle.com/ [Pristupljeno 18 May 2019].
[17] Dzs.hr. (2019). DRŽAVNI ZAVOD ZA STATISTIKU - REPUBLIKA HRVATSKA. [online] Dostupno na: https://www.dzs.hr/ [Pristupljeno 18 May 2019].
[18] Anaconda. (2019). Anaconda Python/R Distribution - Anaconda. [online] Dostupno na: https://www.anaconda.com/distribution/ [Pristupljeno 20 Lipanj. 2019].
[19] Matematika.fkit.hr. (2019). [online] Dostupno na: http://matematika.fkit.hr/novo/statistika_i_vjerojatnost/vjezbe/cjeline/1Deskriptivnastatistika.pdf [Pristupljeno 18 Svibnja 2019].

- Drugi kvartil je broj od kojeg je manje ili jednako 50% podataka. Drugi kvartil je isto što i medijan.
- Treći kvartil (gornji kvartil) je broj od kojeg je manje ili jednako 75% podataka.
- Mod uzorka je podatak koji se u uzorku pojavljuje najviše puta.
- Raspon Varijacije obilježja
- Varijanca i standardna devijacija
- Koeficijent varijacije

Prethodno navedeni pojmovi mogu se također podijeliti na tzv. srednje vrijednosti i mjere disperzije. Srednje vrijednosti nazivaju se još i mjerama centralne tendencije. Srednja vrijednost izračunava se uvijek kao prosječna vrijednost obilježja elemenata iz kojih se izračunava, odnosno nalazi se između najmanje i najveće vrijednosti obilježja. S druge strane mjere disperzije služe za ocjenjivanje reprezentativnosti srednje vrijednosti obilježja.[21]

Za statističku obradu koristit ćemo Python biblioteke SciPy, Numpy i Pandas. Podatke koje ćemo koristiti u analizi skinuti ćemo s portala otvorenih podataka RH, konkretno skinuti ćemo podatke o financiranju udruga od strane županija za 2016. godinu.[22]

```
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

import scipy
from scipy import stats
```

```
udruge = pd.read_excel("udruge.xlsx")

udruge.columns = ['davatelj', 'organizacija', 'OIB', 'projekt', 'voditelj', 'Županija', 'godina', 'iznos']

udruge.head()
```

| | davatelj | organizacija | OIB | projekt | voditelj | Županija | godina | iznos |
|---|---|---|---|---|---|---|---|---|
| 0 | Bjelovarsko-bilogorska županija | Vatrogasna zajednica Bjelovarsko-bilogorske žu... | 98050856762 | Financiranje vatrogasne zajednice Bjelovarsko-... | NaN | Bjelovarsko-bilogorska | 2016 | 512384.89 |
| 1 | Bjelovarsko-bilogorska županija | Športska zajednica BBŽ | 31756452763 | Financiranje Programa javnih potreba u športu,... | NaN | Bjelovarsko-bilogorska | 2016 | 450000.00 |
| 2 | Bjelovarsko-bilogorska županija | ZAJEDNICA KULTURNO-UMJETNIČKIH UDRUGA BBŽ | 92090899205 | Podrška aktivnostima organizacije | NaN | Bjelovarsko-bilogorska | 2016 | 200000.00 |
| 3 | Bjelovarsko-bilogorska županija | Društvo crvenog križa BBŽ | 59459161715 | Redovna dotacija temeljem zakona | NaN | Bjelovarsko-bilogorska | 2016 | 168603.79 |

**Slika 10: Deskriptivna statistika u Python programskom jeziku 1. dio**

Izvor: autor

[21] Ante Rozga (2009). Statistika za Ekonomiste, 5 izdanje, str. 39-60

[22] Data.gov.hr. (2019). Podaci o financiranju udruga - Skupovi podataka | data.gov.hr. [online] Dostupno na: https://data.gov.hr/dataset/podaci-o-financiranju-udruga/resource/57368e79-fa17-4fb8-99e3-eee47fe28768 [Pristupljeno 24 Svibanj. 2019].

Kao što je vidljivo na slici br. 10, prvi korak je davanje do znanja prevoditelju koje biblioteke želimo koristiti funkcijom „import“. Zatim određenim bibliotekama dajemo tzv. alias za lakše korištenje, to činimo funkcijom „as“ odnosno „kao“ . Također kod nekih biblioteka koje raspolažu s ogromnim brojem klasa, potrebno je specificirati koje ćemo konkretno klase koristiti funkcijom „from“.  U drugom koraku kreiramo varijablu udruge koje će poslužiti kao podatkovni okvir za pohranu podataka iz uvezene tablične datoteke. Tabličnu datoteku uvozimo u podatkovni okvir funkcijom „pd.read_excel()“. Funkcijom „columns“ podatkovnom okviru dajemo do znanja da želimo kreirati 7 stupaca, ovu naredbu moguće je izostaviti te direktno pozivati stupce. Konačno funkcijom „head“ ispisujemo prvih par redaka iz učitanog podatkovnog okvira kako bismo potvrdili ispravnost prethodnih postupaka.

```
udruge.groupby(['davatelj'])[['iznos']].sum()
```

| davatelj | iznos |
|---|---|
| Bjelovarsko-bilogorska županija | 3.867573e+06 |
| Brodsko-posavska županija | 4.236285e+06 |
| Dubrovačko-neretvanska županija | 4.871947e+06 |
| Grad Zagreb | 2.620671e+08 |

**Slika 11: Deskriptivna statistika u Python programskom jeziku 2. dio**

Izvor: autor

Na slici br.11 sumu iznosa koju je dala svaka županija računamo s funkcijom „sum()“, no moramo voditi računa o implementaciji, jer sama funkcija bez specifikacije zbraja sve brojčane iznose u stupcima, što bi u konkretnom slučaju značilo sumu godina i OIB-a, što svakako želimo izbjeći. Specifikaciju vršimo funkcijom „groupby()“ gdje u prvim uglatim zagradama specificiramo indeks grupiranja, a u drugim uglatim zagradama stupce čiju sumu želimo, obratite pozornost na to da je funkcija isprogramirana tako da sama prepoznaje jednake tekstualne vrijednosti u zadanom indeksu, te u konačnici prikazuje konačnu sumu po županiji.

```
udruge.groupby(['godina'])[['iznos']].median()
```

| godina | iznos |
|---|---|
| 2016 | 8039.85 |

**Slika 12: Deskriptivna statistika u Python programskom jeziku 3. dio**

Izvor: autor

Na slici br.12 računamo medijan na sličan način kao i prethodno, samo što biramo godinu kao glavni indeks, a kao naredbu koristimo median().

```
udruge.groupby(['godina'])[['iznos']].mean()
```

| godina | iznos |
|---|---|
| 2016 | 52101.839375 |

**Slika 13: Deskriptivna statistika u Python programskom jeziku 4. dio**

Izvor: autor

Na slici br.13 aritmetičku sredinu računamo s funkcijom mean().

```
udruge.groupby(['godina'])[['iznos']].max()
```

| godina | iznos |
|---|---|
| 2016 | 51185105.39 |

**Slika 14: Deskriptivna statistika u Python programskom jeziku 5. dio**

Izvor: autor

Na slici br.14 računamo najmanji i najveći broj u podatkovnom skupu s funkcijama min() i max() .

```
udruge.groupby(['godina'])[['iznos']].std()
```

| godina | iznos |
|---|---|
| 2016 | 801968.024923 |

**Slika 15: Deskriptivna statistika u Python programskom jeziku 6. dio**

Izvor: autor

Na slici br.15 standardnu devijaciju računamo s funkcijom std().

```
udruge.groupby(['godina'])[['iznos']].var()
```

| godina | iznos |
|---|---|
| 2016 | 6.431527e+11 |

**Slika 16: Deskriptivna statistika u Python programskom jeziku 7. dio**

Izvor: autor

Na slici br.16 varijancu računamo s funkcijom var().

```
udruge.groupby(['godina'])[['iznos']].quantile(0.25)
```

| | 0.25 |
|---|---|
| godina | iznos |
| 2016 | 4000.0 |

**Slika 17: Deskriptivna statistika u Python programskom jeziku 8. dio**

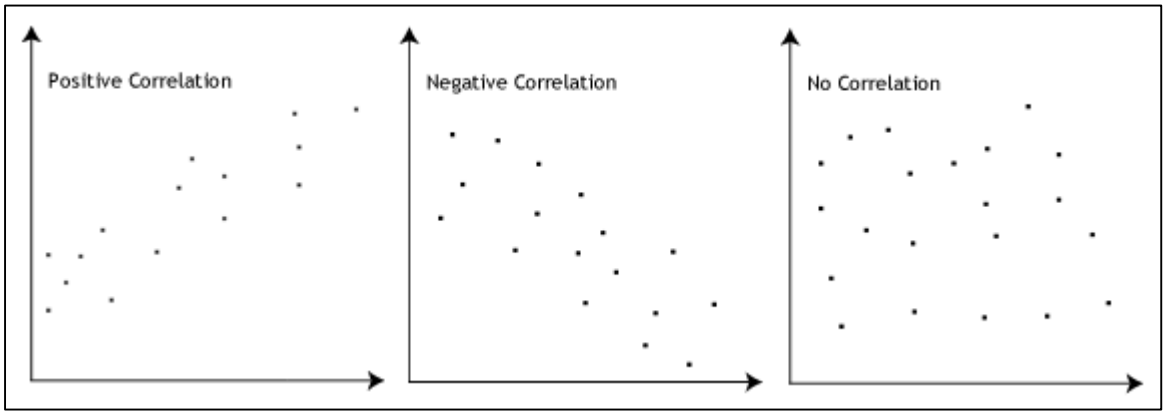
Izvor: autor

Na slici br.17 interkvartil računamo tako da funkcijom quantile() izračunamo gornji (Q3) i donji (Q1) kvartil , te potom oduzmemo jedan od drugoga za dobivanje interkvartila.

## 4.2 Linearna korelacija i regresija u Python programskom jeziku

### 4.2.1 Koeficijent linearne korelacije i postupak njegovog izračuna

Pod pojmom korelacije podrazumijevamo međuzavisnost, odnosno povezanost slučajnih varijabli. Mjera stupnja podudarnosti slučajnih varijabli predstavlja mjeru korelacije. Kao što prikazano na grafikonu br. 4 korelacija može biti pozitivna i negativna po smjeru. Pozitivna korelacija je prisutna onda kada rast jedne varijable prati rast druge varijable, odnosno pad jedne varijable prati pad druge. Negativna korelacija znači da rast jedne varijable prati pad druge varijable. Ako je korelacija potpuna, govorimo o funkcionalnoj povezanosti varijabli. U tome slučaju vrijednost jedne slučajne varijable može se s potpunom sigurnošću odrediti pomoću druge vrijednosti druge varijable. Ako je korelacija djelomična, govorimo o stohastičkoj ili statističkoj vezi.[23]




**Grafikon 4: Vrste korelacija**

Izvor: Statistics.laerd.com. (2019). Pearson Product-Moment Correlation - When you should run this test, the range of values the coefficient can take and how to measure strength of association.. [online] Dostupno na: https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php [Pristupljeno 10 Svibanj. 2019].

Koeficijent linearne korelacije najvažnija je mjera linearne korelacije među slučajnim varijablama, a naziva se Pearsonov koeficijent linearne korelacije. Primjenjuje se na omjerne i intervalne skale mjerenja(numeričke varijable). Pretpostavka za njegovu primjenu je da su

[23] Ante Rozga (2009). Statistika za Ekonomiste, 5 izdanje, str. 179-180

obje slučajne varijable X i Y normalno distribuirane. Međutim, može se primijeniti ako i nisu normalno distribuirane. Računa se sljedećom formulom:

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}}$$

Za računanje linearnog koeficijenta korelacije u Pythonu koristit ćemo biblioteke Pandas i SciPy. Podatkovne skupove pod naslovom „SP 500“ i „MSFT“ skinut ćemo sa stranice Kaggle, u tabličnoj datoteci SP 500[24] su sadržani podatci o vrijednostima SP 500 indeksa kroz povijest, dok su u datoteci MSFT[25] sadržani datumi i cijene dionica Microsofta od početka kotiranja na burzi 1998.

```
import scipy
from scipy.stats.stats import  pearsonr


indexsp = pd.read_csv("spx.csv")
msft = pd.read_csv("msft.csv")

indexsp.columns = ['date', 'value']
msft.columns = ['date','open','high','low','close','volume']

indexsp.head()
```

| | date | value |
|---|---|---|
| 0 | 02-Jan-98 | 975.04 |
| 1 | 05-Jan-98 | 977.07 |

**Slika 18: Koeficijent linearne korelacije i postupak njegovog izračuna 1. dio**

Izvor: autor

Kao što je prikazano na slici br.18 prvo uvozimo potrebne biblioteku SciPy. Zatim u podatkovne okvire „indexsp“ i „msft“ uvozimo potrebne tablične datoteke, te definiramo stupce u podatkovnim okvirima. Jednostavnom funkcijom „head()“ provjeravamo ispravnost prethodnih postupaka..

[24] Kaggle.com. (2019). S&P 500 stock data. [online] Dostupno na: https://www.kaggle.com/camnugent/sandp500 [Pristupljeno 24 Svibanj. 2019].
[25] Kaggle.com. (2019). MSFT-stock-data. [online] Dostupno na: https://www.kaggle.com/ryanforbes/msftstockdata [Pristupljeno 24 Svibanj. 2019].

```
index_value = indexsp['value'].astype(np.float)
msft_price = msft['close'].astype(np.float)

linearni_koeficijent, p_value = pearsonr(msft_price, index_value)
print ("Koeficijent linearne korelacije iznosi %0.5f" % linearni_koeficijent)
```

```
Koeficijent linearne korelacije iznosi 0.36231
```

**Slika 19: Koeficijent linearne korelacije i postupak njegovog izračuna 2. dio**

Izvor: autor

Na slici br.19 računamo linearni koeficijent tako da definiramo varijable „index_value“ i „msft_price“, kojima dodjeljujemo stupce iz podatkovnih okvira, te ih definiramo kao podatkovni tip float, koji u memoriji zauzima više bajtova od tvornički zadanog tipa Int, što praktično znači mogućnost znatno preciznijih rezultate.

Samo izračunavanje obavljamo tako da sadržaj varijabli „linearni_koeficijent“ i „p_value“ definiramo funkcijom „pearsonr“. Prva varijabla sadrži vrijednost koeficijenta, dok druga varijabla „p_value“ označava p-vrijednost, odnosno vjerojatnost da je nulta hipoteza istina.

Konačno funkcijom „print“ ispisujemo rezultate, valja napomenuti da oznaka „%0.5f“ označava pet decimalnih mjesta varijable tipa float, te njenu vrijednost možemo mijenjati proizvoljno, ovisno s koliko decimalnih mjesta rezultat želimo dobiti.

Kao rezultati dobivamo 0.36231, što je umjerena pozitivna korelacija, odnosno možemo zaključiti da dionice Microsofta blago koreliraju s burzovnim indeksom SP 500.

### 4.2.2 Jednostruka linearna regresija i postupak njezinog izračuna

Za razliku od korelacijske analize, kod regresijske analize treba odrediti koja varijabla će biti regresand (zavisna) varijabla. Nezavisne varijable u modelu obično se nazivaju regresorskim varijablama. U velikom broju primjena veza između varijabli je linearna, ali postoje i slučajevi nelinearne regresije.[26]

[26] Ante Rozga (2009). Statistika za Ekonomiste, 5 izdanje, str.186

Regresijski model izgleda ovako:

$$Y = \beta_0 + \beta_1 * X + e$$

Pri tomu slučajna greška „e“ mora udovoljavati sljedećim uvjetima (tzv. Gauss Markovljevi uvjeti)[27]:

1. E(e) = 0 $\forall i$

   To znači da je očekivana vrijednost slučajne varijable jednaka nuli. Slučajna greška je čas pozitivna, čas negativna, ali ne smije imati nikakvo sistematsko kretanje u bilo kojemu smjeru. Ako se radi s konstantnim članom, onda je gornji uvjet ispunjen automatski jer imamo:

   $$\sum_{i=1}^{N}(Y_i - \hat{Y}_i) = 0$$

2. $E(e_i, e_j) = \sigma_e^2 < +\infty$ za i =j

   $$\sigma_e^2 = const$$

   Radi se o uvjeti da varijanca reziduala bude konačna i čvrsta, tj. da se ne mijenja od opservacije do opservacije. Taj uvjet nazive se često uvjetom homoskedastičnosti varijance reziduala. Ako taj uvjet nije ispunjen radi se o tzv. heteroskedastičnosti reziduala. U tom slučaju varijanca može sistematski kovarirati s regresorskom varijablom. Ako ovaj uvjet nije ispunjen ocjena parametara standardnom metodom najmanjih kvadrata bit će neefikasna.

3. $E(e_i, e_j) = 0 \; \forall \; (i \neq j)$

   odnosno $\text{Cov}(e_i, e_j) = 0$ $(i \neq j)$

   Ovaj uvjet se odnosi na nepostojanje sustavnosti u slučajnim greškama. Naime, ako je greška doista slučajna ne smije postojati nikakva korelacija između vrijednosti varijable „e“ s pomakom. Ako ovaj uvjet nije ispunjen, standardna metoda najmanjih kvadrata dat će neefikasne ocjene.

[27] Ante Rozga (2009). Statistika za Ekonomiste, 5 izdanje, str.187

4. Slučajna varijabla mora biti distribuirana nezavisno od regresorske (eksplanatorne) varijable:

$$\mathrm{E}(e_i, X_i) = 0$$

Kao dodatak gornjim uvjetima još je važno napomenuti da slučajna varijabla mora biti distribuirana po normalnoj distribuciji. Ocjene parametara dobiju se metodom najmanjih kvadrata.

Ako se za regresand varijablu (zavisnu varijablu) postavi varijabla X umjesto varijable Y, dobiva se drugi regresijski pravac koji će se s pravcem Y poklapati samo u slučaju perfektne zavisnosti između varijabli X i Y. Ako se regresijski pravci Y i X bliži jedan drugome, veza između varijabli je jača, a ako tvore pravi kut, koeficijent korelacije jednak je nuli. Ako se pravci potpuno poklapaju veza je funkcionalna. Između tih ekstrema nalazi se stohastička veza koja postoji u praksi.

Za prikaz izračunavanja jednostruke linearne regresije u Pythonu koristit ćemo podatke o srčanim bolesnicima skinute sa stranice Kaggle[28]. Želimo uočiti poveznicu između godina starosti i razine kolesterola u krvi, odnosno napraviti prediktivni linearni model koji će nam pomoći da procijenimo razinu kolesterola u određenoj godini. U ovom slučaju godine su nezavisna varijabla, dok je razina kolesterola zavisna varijabla.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model

df = pd.read_csv("heart.csv")
reg = linear_model.LinearRegression()
reg.fit(df[['age']], df.trestbps)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

**Slika 20: Jednostruka linearna regresija i postupak njezinog izračuna 1. dio**

Izvor: autor

[28] Kaggle.com. (2019). Heart Disease UCI. [online] Dostupno na: https://www.kaggle.com/ronitf/heart-disease-uci [Pristupljeno 12 Lipanj. 2019].

Na slici br.20 prvo uvozimo potrebne biblioteke od kojih je najvažnija „sklearn (scikit-learn)“ iz koje uvozimo „linear_model“ klasu. Zatim, u podatkovni okvir „df“ učitavamo podatke iz tablične datoteke, potom kreiramo objekt „reg“ kojeg popunjavamo funkcijom „LinearRegression()“ koja je sastavni dio „linear_model“ klase. Funkcijom „fit“ objektu nalažemo da kao nezavisnu varijablu stavi godine, dok kao zavisnu stavljamo razinu kolesterola u krvi.

```
reg.predict([[45]])

array([126.57113672])
```

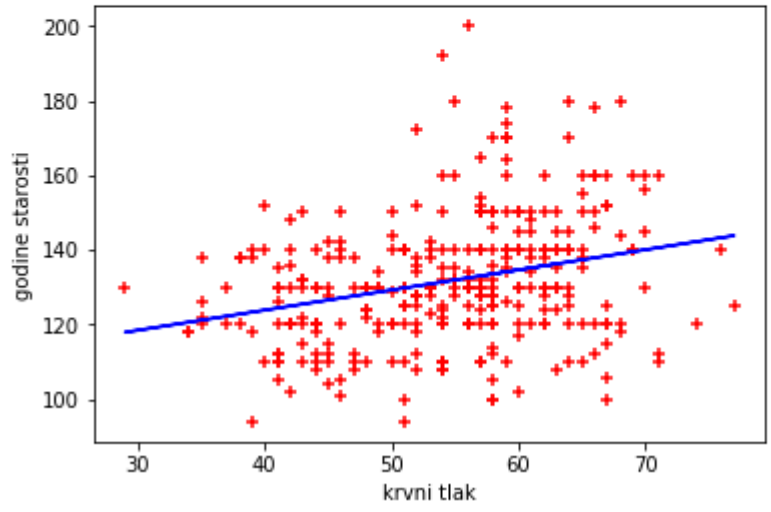
**Slika 21: Jednostruka linearna regresija i postupak njezinog izračuna 2. dio**

Izvor: autor

Nakon izračuna jednostruke linearne regresije, na slici br.21 funkcijom predict() i unosom proizvoljnog parametra, u ovom slučaju unosimo željene godine starosti, te kao odgovor dobivamo predviđenu razinu kolesterola.

```
%matplotlib inline
plt.xlabel('krvni tlak')
plt.ylabel('godine starosti')
plt.scatter(df.age,df.trestbps, color='red',marker='+')
plt.plot(df.age, reg.predict(df[['age']]), color= 'blue')
```

```
[<matplotlib.lines.Line2D at 0x23443960128>]
```




**Slika 22: Jednostruka linearna regresija i postupak njezinog izračuna 3. dio**

Izvor: autor

Kao što je prikazano na slici br.22 izračunatu jednostruku linearnu regresiju možemo i grafički prikazati putem matplotlib biblioteke, X i Y os označavamo zadanim varijablama, kreiramo dijagram rasipanja za naše dvije varijable funkcijom scatter(), te na kraju funkcijom plot() crtamo našu liniju linearne predikcije.

### 4.4.3 Višestruka linearna regresija i postupak njezinog izračuna

Model višestruke linearne regresije sastoji se od jedne zavisne (regresand) varijable i k „nezavisnih“ (regresorskih varijabli).

Model višestruke linearne regresije glasi:

$$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \ldots + \beta_k * X_k$$

Ocjene parametara dobivaju se na sličan način kao kod regresijskoga polinoma k-tog stupnja. Razlika je u tome što su na desnoj strani jednadžbe nalazi više regresorskih varijabli, a ne polinomi jedne regresorske varijable.[29]

$$\beta = (X^T X)^{-1} * (X^T Y)$$

Za izračun višestruke linearne regresije u Pythonu koristit ćemo podatke o zdravstvenom osiguranju i povezanim troškovima skinutim sa stranice Kaggle[30].

```
import pandas as pd
import numpy as np
from sklearn import linear_model

df = pd.read_csv("insurance.csv")
reg = linear_model.LinearRegression()
reg.fit(df[['age','bmi']],df.charges)
```

**Slika 23: Višestruka linearna regresija i postupak njezinog izračuna 1. dio**

Izvor: autor

[29] Ante Rozga (2009). Statistika za Ekonomiste, 5 izdanje, str.193-194
[30] Kaggle.com. (2019). Medical Cost Personal Datasets. [online] Dostupno na: https://www.kaggle.com/mirichoi0218/insurance [Pristupljeno 11 Lipanj. 2019].

Kao što je prikazano na slici br.23 prvo uvozimo potrebne biblioteke, te učitavamo dotičnu tabličnu datoteku. Kreiramo „reg“ koji predstavlja objekt klase „linear_model“. Zatim sami model višestruke regresije računamo funkcijom fit(), gdje kao nezavisne varijable u podatkovni okvir stavljamo godine starosti i indeks tjelesne mase, a kao zavisnu varijablu stavljamo troškove osiguranja.

```
reg.predict([[40,35]])

array([14906.20468231])
```

**Slika 24: Višestruka linearna regresija i postupak njezinog izračuna 2. dio**

Izvor: autor

Nakon izračuna višestruke linearne regresije, na slici br.24 funkcijom „predict()“ računamo predikciju za proizvoljno unesene parametre, u ovom slučaju unosimo željene godine starosti i indeks tjelesne mase, te kao odgovor dobivamo predviđenu cijenu osiguranja.

Grafičko prikazivanje s više od dvije regresorske varijable nije jednostavno jer se radi o tzv. hiperravninama. Trenutne Python biblioteke nisu optimalne za ovu zadaću, a i u praksi se višestruka regresija rijetko vizualizira.

## 4.3 Vizualizacija podataka u Python programskom jeziku

Za vizualizaciju podatka u Python programskom jeziku koristimo matplotlib biblioteku koja je zbog svoje fleksibilnosti i mogućnosti podešavanja velikog broja parametara postala najpopularnija programska biblioteka za vizualizaciju podataka.

Za prikaz vizualizacije podataka koristit ćemo već spomenute tablične podatke o kretnju burzovnog indeksa SP 500 i tablične podatke o srčanim bolesnicima skinute sa stranice kaggle.

```
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline

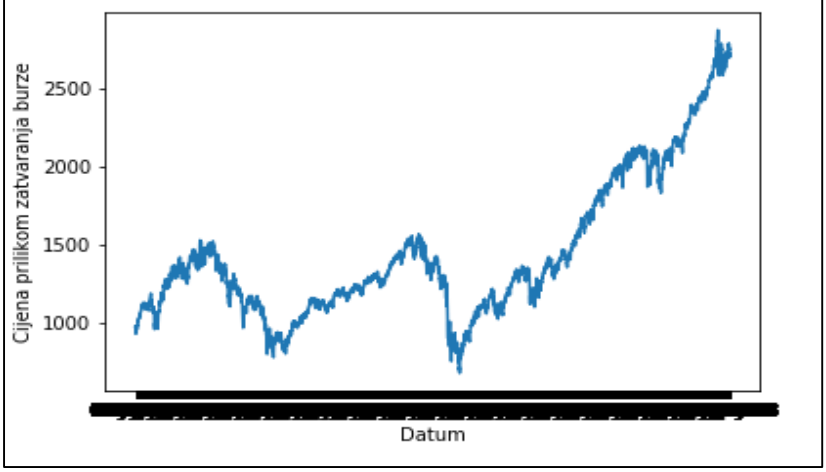
df1 = pd.read_csv("spx.csv")
df2 = pd.read_csv("heart.csv")
```

**Slika 25: Vizualizacija podataka u Python programskom jeziku 1. dio**

Izvor: autor

Kao što je prikazano na slici br.25 prvo uvozimo potrebne biblioteke, zatim popunjavamo podatkovne okvire tabličnim datotekama. Oznakom „%matplotlib inline“ dajemo do znanja prevoditelju da nam sve grafikone prikazuje u istome prozoru, odnosno da ne otvara dodatne prozore za prikaz grafikona.

```
plt.plot(df1.date, df1.close)
plt.xlabel('Datum')
plt.ylabel('Cijena prilikom zatvaranja burze')
plt.show()
```




**Slika 26: Vizualizacija podataka u Python programskom jeziku 2. dio**

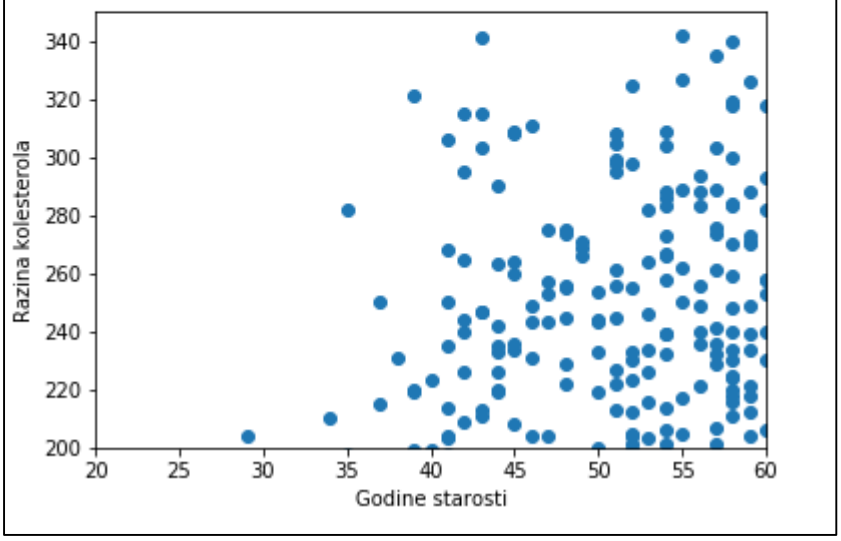
Izvor: autor

Na slici br.26 linijski graf kreiramo funkcijom „plot()“ gdje kao X os stavljamo podatkovni okvir koji sadrži stupac s datumima, a kao Y os stavljamo podatkovni okvir koji sadrži podatke o cijeni. Funkcijom „x/ylabel()“ označavamo X i Y os odgovarajućim imenom.

```
plt.scatter(df2.age, df2.chol)
plt.xlabel('Godine starosti')
plt.ylabel('Razina kolesterola')

plt.axis([20, 60, 200, 350 ])

plt.show()
```




**Slika 27: Vizualizacija podataka u Python programskom jeziku 3. dio**

Izvor: autor

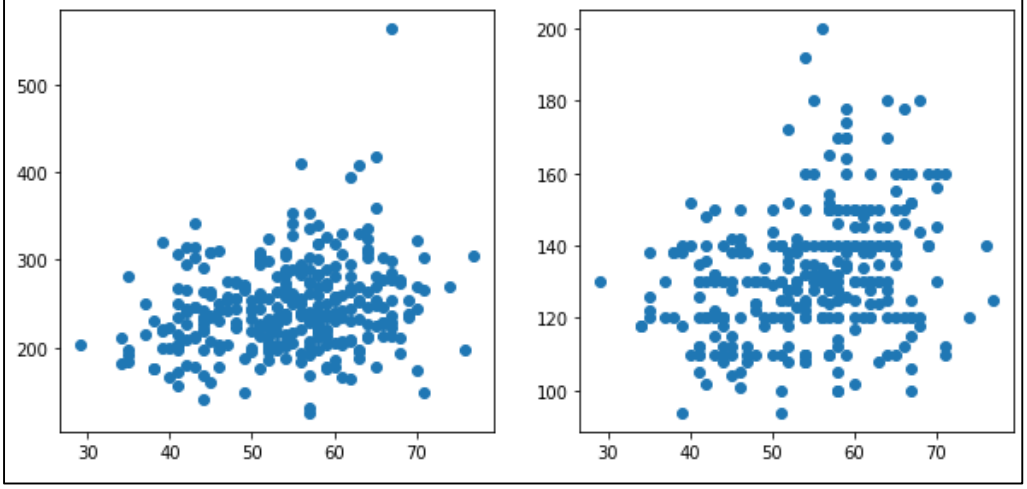
Na slici br.27 dijagram rasipanja označavamo funkcijom „scatter()“ koju popunjavamo odgovarajućim popratnim podatkovnim okvirima. Funkcijom „axis“ određujemo u kojim intervalima želimo da nam se podatci prikazuju na X i Y osi, odnosno u ovom konkretnom primjeru želimo prikazati osobe između 20 i 60 godina starosti i s razinom kolesterola između 200 i 350.

```
figure = plt.figure(1, figsize=(10, 10))

grafikon_1 = figure.add_subplot(221)
grafikon_2 = figure.add_subplot(222)

grafikon_1.scatter(df2.age, df2.chol)
grafikon_2.scatter(df2.age, df2.trestbps)
```

```
<matplotlib.collections.PathCollection at 0x1697b4559b0>
```




**Slika 28: Vizualizacija podataka u Python programskom jeziku 4. dio**

Izvor: autor

U matplolib-u je također moguće prikazivati više grafikona odjedanput, to se radi kreiranjem objekta „figure“ koji se predstavlja objekt klase „pyplot“, funkcijom figsize() unutar njega određujemo veličinu cjelokupnog grafičkog okvira. Nakon kreiranja glavnog objekta, kreiramo i dva podobjekta „grafikon_1“ i „grafikon_2“ koja funkcijom „add_subplot“ stavljamo pod nadležnost prethodno kreiranog „figure“ objekta, te određujemo veličinu i položaj njih samih unutar grafičkog okvira. Na kraju funkcijom „scatter()“ kreiramo dva grafikona rasipanja koja se pune prethodno definiranim podatcima. Rezultat i postupak grafičkog prikazivanja prikazan je na slici br.28.

## 4.4 Manipulacija podataka u Python programskom jeziku

Za prikaz mogućnosti manipulacije podataka koristit ćemo biblioteku Pandas koja se najčešće koristi za tu svrhu, također koristit ćemo i biblioteku matplotlib za vizualizaciju podataka.

### 4.4.1 Jednostavni postupci manipulacije podataka

Jednostavni postupci manipulacije podataka obično uključuju sortiranje, filtriranje i kreiranje novih stupaca kombinacijom postojećih podataka po nekim kriterijima. Za prikaz jednostavne manipulacije podatcima koristit ćemo već spomenute podatke o srčanim bolestima skinute sa stranice Kaggle.

```
import pandas as pd
df = pd.read_csv('heart.csv')
df.head()
```

| | age | sex | cp | trestbps | chol | fbs | restecg |
|---|---|---|---|---|---|---|---|
| **0** | 63 | 1 | 3 | 145 | 233 | 1 | 0 |
| **1** | 37 | 1 | 2 | 130 | 250 | 0 | 1 |

**Slika 29: Jednostavni postupci manipulacije podataka 1. dio**

Izvor: autor

Kao što je i prikazano na slici br.29 pravo uvozimo potrebnu Pandas biblioteku, zatim u podatkovni okvir učitavamo tabličnu datoteku. Podatkovni okvir osnovna je podatkovna struktura Pandas biblioteke s kojom vršimo obradu podataka. U računalnom smislu radi se o tzv. „container of arrays“ odnosno skladištu nizova.

```
print(df.sort_values('age', ascending=True))
```

| | age | sex | cp | trestbps | chol | fbs | restecg |
|---|---|---|---|---|---|---|---|
| 72 | 29 | 1 | 1 | 130 | 204 | 0 | 0 |
| 58 | 34 | 1 | 3 | 118 | 182 | 0 | 0 |
| 125 | 34 | 0 | 1 | 118 | 210 | 0 | 1 |
| 239 | 35 | 1 | 0 | 126 | 282 | 0 | 0 |
| 65 | 35 | 0 | 0 | 138 | 183 | 0 | 1 |

**Slika 30: Jednostavni postupci manipulacije podataka 2. dio**

Izvor: autor

Na slici br.30 podatke sortiramo funkcijom „sort_values()“ gdje kao parametar za sortiranje stavljamo stupac godine, te ga sortiramo uzlazno.

```
print(df.loc[3:8,['age', 'chol']])
```

```
   age  chol
3   56   236
4   57   354
5   57   192
6   56   294
7   44   263
8   52   199
```

**Slika 31: Jednostavni postupci manipulacije podataka 3. dio**

Izvor: autor

Na slici br.31 prikazan je postupak koji je poznat pod nazivom „slicing“ odnosno rezanje podataka kojeg provodimo s funkcijom „loc()“; funkciji prvo prosljeđujemo raspon u kojem želimo prikazati podatke, te u sljedećim uglatim zagradama prosljeđujemo parametre stupaca na koje primjenjujemo slicing postupak.

```
print(df[df.chol > 300])
```

```
     age  sex  cp  trestbps  chol
4     57    0   0       120   354
16    58    0   2       120   340
25    71    0   1       160   302
28    65    0   2       140   417
36    54    0   2       135   304
```

**Slika 32: Jednostavni postupci manipulacije podataka 4. dio**

Izvor: autor

Na slici br.32 filtriranje podataka možemo provoditi tako da na odabranom stupcu u podatkovnom okviru filtriranje vršimo po matematičkim ograničenjima, konkretno u ovom primjeru po ljudima čija razina kolesterola prelazi 300.

```
print(df[df['age'].isin(['58','59','60'])])
```

| | age | sex | cp | trestbps | chol | fbs | restecg |
|---|---|---|---|---|---|---|---|
| 14 | 58 | 0 | 3 | 150 | 283 | 1 | 0 |
| 16 | 58 | 0 | 2 | 120 | 340 | 0 | 1 |
| 20 | 59 | 1 | 0 | 135 | 234 | 0 | 1 |
| 26 | 59 | 1 | 2 | 150 | 212 | 1 | 1 |
| 64 | 58 | 1 | 2 | 140 | 211 | 1 | 0 |

**Slika 33: Jednostavni postupci manipulacije podataka 5. dio**

Izvor: autor

Na slici br.33 filtriranje također možemo vršiti koristeći funkciju „isin()“ u kojoj točno određujemo koje parametre od određenog stupca želimo, u ovom konkretnom primjeru filtriramo rezultate po točno određenim godinama.

```
df['prosjek_godina'] = (df.age.sum() / len(df.index))
print(df.prosjek_godina[:1])
```

```
0    54.366337
Name: prosjek_godina, dtype: float64
```

**Slika 34: Jednostavni postupci manipulacije podataka 6. dio**

Izvor: autor

Na slici br.34 prikazan je postupak izrađivanja novog stupca koristeći postojeće podatke tako da definiramo novi stupac u podatkovnom okviru nazvan prosjek godina kojeg popunjavamo zbrajajući godine funkcijom „sum()“ i podijelimo s brojem članova podatkovnog skupa. Cilj ovoga primjera je demonstrirati kreiranje novog stupca od postojećih podataka, a ne efikasno izračunavanje prosjeka, za to postoje specijalizirane funkcije koje su prethodno obrađene u ovom radu.

### 4.4.2 Napredni postupci manipulacije podataka

Napredni postupci manipulacije podataka podrazumijevaju kreiranje zasebnog podatkovnog okvira po zadanim kriterijima, što uključuje i korištenje računalnih petlji. Za prikaz napredne manipulacije koristit ćemo podatke o cijenama avokada u SAD-u skinute sa stranice kaggle.[31]

[31] Kaggle.com. (2019). Avocado Prices. [online] Dostupno na: https://www.kaggle.com/neuromusic/avocado-

```
import pandas as pd
import matplotlib as plt
df = pd.read_csv("datasets/avocado.csv")

df.head()
```

| | Unnamed: 0 | Date | AveragePrice | Total Volume | 4046 | type | year | region |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2015-12-27 | 1.33 | 64236.62 | 1036.74 | conventional | 2015 | Albany |
| 1 | 1 | 2015-12-20 | 1.35 | 54876.98 | 674.28 | conventional | 2015 | Albany |

**Slika 35: Napredni postupci manipulacije podataka 1. dio**

Izvor: autor

Kao što je i prikazano na slici br.35 prvo uvozimo potrebne biblioteke, te u podatkovni okvir učitavamo tabličnu datoteku, učitani podatci prikazuju cijene avokada po Američkim saveznim državama. Da bih smo prikazali svu moć ove biblioteke izračunati ćemo pomične prosjeke organskog avokada za razdoblje od 25 dana za svaku saveznu državu zasebno.

Pomični prosjek računamo tako da zbrojimo cijenu nekog dobra, u ovom slučaju avokada te podijelimo s brojem razdoblja koje želimo.

```
import pandas as pd

df = pd.read_csv("avocado.csv")
df = df.copy()[df['type'] == "organic"]
df['Date'] = pd.to_datetime(df["Date"])

df.sort_values(by="Date", ascending = True, inplace = True)

novi_okvir = pd.DataFrame()

for regija in df['region'].unique():
    print(regija)
    privremeni_okvir = df.copy()[df['region'] == regija]
    privremeni_okvir.set_index("Date", inplace = True)
    privremeni_okvir.sort_index(inplace= True)
    privremeni_okvir[f'{regija}_price25ma'] = privremeni_okvir['AveragePrice'].rolling(25).mean()


    if novi_okvir.empty:
        novi_okvir = privremeni_okvir[[f'{regija}_price25ma']]

    else:
        novi_okvir = novi_okvir.join(privremeni_okvir[f'{regija}_price25ma'])

novi_okvir.tail()
```

**Slika 36: Napredni postupci manipulacije podataka 2. dio**

Izvor: autor

prices [Pristupljeno 12 Lipanj. 2019].

Na slici br.36 prikazana je tehnička realizacija naprednog kreiranja novog podatkovnog okvira od postojećih podataka. Prvo učitavamo tabličnu datoteku u podatkovni okvir df, u stupcu „type“ filtriramo samo one vrste avokada koje su organske. U stupcu „Date“ funkcijom „to_datetime()“ dajemo do znanja prevoditelju da podatke u stupcu konvertira u tip podatka „Date“ odnosno kao datum, za kasniju lakšu obradu. Podatkovni okvir sortiramo po datumu uzlazno funkcijom „sort_values()“ .

Potom kreiramo novi podatkovni okvir koristeći funkciju „DataFrame()“, ova funkcija kreira podatkovnu strukturu koja se sastoji od dvodimenzionalnog niza kojeg ćemo kasnije popunjavati podatcima.

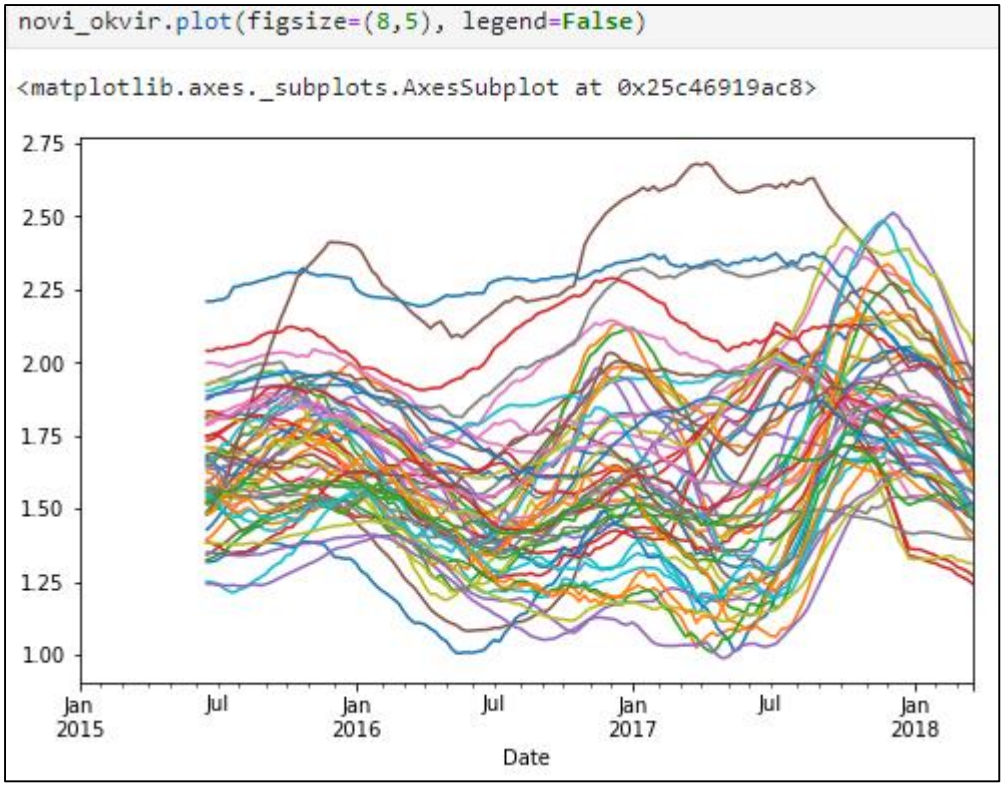
For petljom iteriramo kroz svaku saveznu državu jednom, za to koristimo funkciju „unique()“ koja  vraća samo unikatne vrijednosti u određenom nizu; zatim kreiramo jedan privremeni podatkovni okvir u koji kopiramo postojeći podatkovni okvir s novo formiranim stupcem regija. U privremeni okvir stavljamo datume kao glavni indeks funkcijom „set_index()“, te ih sortiramo funkcijom „sort_index()“. Na kraju u privremenom okviru kreiramo novi stupac „_price25ma“ koji se definira kao pomični prosjek posljednjih 25 razdoblja koristeći funkcije „rolling()“ i „mean()“ .

Novi podatkovni okvir popunjavamo tako da podatke iz prethodno kreiranog privremenog okvira smjestimo u novi podatkovni okvir koji je prethodno kreiran za tu namjenu. Ako je podatkovni okvir prazan, što provjeravamo petljom „if“ i funkcijom „empty()“, definiramo novi stupac u podatkovnom okviru, te ga popunjavamo podatcima. Ako za neku regiju podatci već postoje, funkcijom „join()“ pridodajemo podatke za tu istu regiju u podatkovni okvir.

| Date | California_price25ma | LasVegas_price25ma | PhoenixTucson_price25ma | BuffaloRochester_price25ma |
|---|---|---|---|---|
| 2018-02-25 | 1.9120 | 1.9120 | 1.9120 | 1.9120 |
| 2018-03-04 | 1.8748 | 1.8748 | 1.8748 | 1.8748 |
| 2018-03-11 | 1.8440 | 1.8440 | 1.8440 | 1.8440 |
| 2018-03-18 | 1.8204 | 1.8204 | 1.8204 | 1.8204 |

**Slika 37: Napredni postupci manipulacije podataka 3. dio**
Izvor: autor

Na slici br.37 prikazani su konačni rezultati novog okvira kreiranog po zadanim kriterijima, vidimo da smo dobili pomične prosjeke poredane po datumu u intervalima od dvadeset pet razdoblja.




**Slika 38: Napredni postupci manipulacije podataka 4. dio**
Izvor: autor

Grafički prikaz novog podatkovnog okvira vršimo standardno funkcijom „plot“, a konačni rezultati možemo vidjeti na slici br.38. Više o naprednim vrstama manipulacije podatcima moguće je pronaći na kanalu korisnika sentdex[32] i službenoj dokumentaciji Pandas[33] biblioteke.

[32] YouTube. (2019). Graphing/visualization - Data Analysis with Python and Pandas p.2. [online] Dostupno na: https://www.youtube.com/watch?v=DamIIzp41Jg&list=PLQVvvaa0QuDfSfqQuee6K8opKtZsh7sA9&index=2 [Pristupljeno 5 Lipanj. 2019].
[33] Pandas.pydata.org. (2019). User Guide — pandas 0.24.2 documentation. [online] Dostupno na: https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html [Pristupljeno 15 Svibanj. 2019].

# 5. ZAKLJUČAK

Zbog ogromnog broja uređaja i korisnika povezanih na Internet, količina podataka povećava se eksponencijalnom brzinom. Ona poduzeća koje budu implementirale sustave Velikih Podataka imat će značajnu konkurentsku prednost na tržištu; ta prednost prvenstveno proizlazi iz sposobnosti da subjekti uspješno donosi kratkoročne i dugoročne odluke na osnovu podataka koje sustav uspješno analizira i obrađuje, te u konačnici kvalitetno prezentira u jednu smislenu cjelinu.

Implementacija sustava velikih podataka nije jednostavna, zahtijeva značajna dugoročna financijska ulaganja. Prepreke s kojima se poduzeće susreće prilikom implementacije sustava velikih podataka nisu samo financijske prirode, već ljudske i tehničke. Naime, na tržištu nedostaje veliki broj stručnjaka, te zbog toga većina poduzeća kasni ili odgađa implementaciju. S tehničke strane veliku prijetnju predstavljaju kibernetički napadi čija uspješnost praktički može uništiti reputaciju nekog poduzeća i dovesti ga do ruba propasti. Zbog svega navedenog poduzeća koja se odluči na implementaciju sustava velikih podataka treba iznimno kvalitetno planirati i implementirati adekvatne sustave kontrole kvalitete i provedbe.

Jedan od najpopularnijih programskih jezika u različitim komponentama sustava velikih podataka je Python. Svoju popularnost je prvenstveno stekao svojom jednostavnošću i fleksibilnošću. Python-ov ekosustav čine desetina tisuća stručnih entuzijasta koji održavaju na tisuće različitih programskih biblioteka za različite svrhe, od analize podataka do umjetne inteligencije. Ono što je najvažnije za pravne i fizičke osobe jest to, što skoro sve programske biblioteke i funkcije koje ovaj programski jezik pruža dolaze s besplatnim licencama i otvorenog su koda; te posjeduju kvalitetnu i održavanu tehničku dokumentaciju što svakom poduzeću pruža značajna uštedu novca i vremena.

Fleksibilnost i moć programskog jezika Python možda je najvidljivija u podatkovnoj analizi i obradi. Kao što je i prikazano u praktičnom dijelu ovoga rada, naizgled kompliciranе postupke nad raznovrsnim skupovima podataka svodi na programske kodove od par linija, a potrebno vrijeme za korisnika da savlada većinu programskih biblioteka svodi na minimum, dapače možemo reći da je većina Python-ove sintakse iznimno intuitivna i lako shvatljiva, čak i onim ljudima koji nikada prije nisu programirali. Sve ove prethodno navedene činjenice čini Python iznimno popularnim programskim jezikom i za korisnike koji ne dolaze nužno iz I.T. sektora.

## LITERATURA:


1. Ante Rozga (2009). Statistika za Ekonomiste, 5 izdanje, str. 39-60
2. Ante Rozga (2009). Statistika za Ekonomiste, 5 izdanje, str. 179-187
3. Ante Rozga (2009). Statistika za Ekonomiste, 5 izdanje, str. 186
4. Ante Rozga (2009). Statistika za Ekonomiste, 5 izdanje, str. 187
5. Ante Rozga (2009). Statistika za Ekonomiste 5, izdanje, str. 193-194
6. AI Impacts. (2015). Trends in the cost of computing. [online] Dostupno na: https://aiimpacts.org/trends-in-the-cost-of-computing/ [Pristupljeno 19 Lipanj. 2019].
7. Anaconda. (2019). Anaconda Python/R Distribution - Anaconda. [online] Dostupno na: https://www.anaconda.com/distribution/ [Pristupljeno 20 Lipanj. 2019].
8. Data.gov.hr. (2019). Podaci o financiranju udruga - Skupovi podataka | data.gov.hr. [online] Dostupno na: https://data.gov.hr/dataset/podaci-o-financiranju-udruga/resource/57368e79-fa17-4fb8-99e3-eee47fe28768 [Pristupljeno 24 Svibanj. 2019].
9. Dzs.hr. (2019). DRŽAVNI ZAVOD ZA STATISTIKU - REPUBLIKA HRVATSKA. [online] Dostupno na: https://www.dzs.hr/ [Pristupljeno 18 May 2019].
10. Docs.scipy.org. (2019). SciPy — SciPy v1.3.0 Reference Guide. [online] Dostupno na: https://docs.scipy.org/doc/scipy/reference/ [Pristupljeno 21 Lipanj. 2019].
11. Docs.python.org. (2019). Using the Python Interpreter - Python 3.7.4 documentation. [online] Dostupno na: http://tiny.cc/t12pbz [Pristupljeno 15 Lipanj. 2019].
12. Jules J. Berman (2018). Principles and Practice of Big Data: Preparing, Sharing, and Analyzing Complex Information 2nd Edition, str. 16-21
13. Jules J. Berman (2018). Principles and Practice of Big Data: Preparing, Sharing, and Analyzing Complex Information 2nd Edition, str. 326-329
14. Jupyter.org. (2019). Project Jupyter. [online] Dostupno na: https://jupyter.org/about [Pristupljeno 15 Lipanj. 2019].
15. Kaggle.com. (2019). Kaggle: Your Home for Data Science. [online] Dostupno na: https://www.kaggle.com/ [Pristupljeno 18 May 2019].
16. Kaggle.com. (2019). S&P 500 stock data. [online] Dostupno na: https://www.kaggle.com/camnugent/sandp500 [Pristupljeno 24 Svibanj. 2019].
17. Kaggle.com. (2019). MSFT-stock-data. [online] Dostupno na: https://www.kaggle.com/ryanforbes/msftstockdata [Pristupljeno 24 Svibanj. 2019].
18. Kaggle.com. (2019). Heart Disease UCI. [online] Dostupno na:

https://www.kaggle.com/ronitf/heart-disease-uci [Pristupljeno 12 Lipanj. 2019].
19. Kaggle.com. (2019). Medical Cost Personal Datasets. [online] Dostupno na: https://www.kaggle.com/mirichoi0218/insurance [Pristupljeno 11 Lipanj. 2019].
20. Kaggle.com. (2019). Avocado Prices. [online] Dostupno na: https://www.kaggle.com/neuromusic/avocado-prices [Pristupljeno 12 Lipanj. 2019].
21. Laura Donnelly (2018). More than 900 NHS deaths yearly may be caused by IT failings . [online] The Telegraph. Dostupno na: https://bit.ly/2ZfNVyn [Pristupljeno 19 Lipanj. 2019].
22. Matematika.fkit.hr. (2019). [online] Dostupno na: http://matematika.fkit.hr/novo/statistika_i_vjerojatnost/vjezbe/cjeline/1Deskriptivnastatistika.pdf [Pristupljeno 18 Svibnja 2019].
23. Martin Hilbert M, Lopez P. (2011). The world's technological capacity to store, communicate, and compute information. Science, str. 332:60–5.
24. Pandas.pydata.org. (2019). User Guide — pandas 0.24.2 documentation. [online] Dostupno na: http://tiny.cc/rz2pbz [Pristupljeno 15 Svibanj. 2019].
25. PyPI. (2019). spyder. [online] Dostupno na: https://pypi.org/project/spyder/ [Pristupljeno 20 Lipanj. 2019].
26. PyPI. (2019). pandas. [online] Dostupno na: https://pypi.org/project/pandas/ [Pristupljeno 20 Lipanj. 2019].
27. Robert Johansson (2018). Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib 2nd ed. Edition, str. 1-2
28. Robert Johansson (2018). Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib 2nd ed. Edition, str. 2-4
29. Schmidt S. (2012). Data is exploding: the 3V's of Big Data. Business Computing World.
30. Scikit-learn.org. (2019). scikit-learn: machine learning in Python — scikit-learn 0.21.2 documentation. [online] Dostupno na: https://scikit-learn.org/stable/ [Pristupljeno 21 Lipanj. 2019].
31. Team, E. (2018). Infographic: The Data Scientist Shortage - insideBIGDATA. [online] insideBIGDATA. Dostupno na: http://tiny.cc/cx2pbz [Pristupljeno 18 Lipanj. 2019].
32. Viktor Mayer-Schönberger , Kenneth Cukier (2014). Big Data: A Revolution That Will Transform How We Live, Work, and Think, str. 6-14
33. YouTube. (2019). Graphing/visualization - Data Analysis with Python and Pandas p.2. [online] Dostupno na: http://y2u.be/DamIIzp41Jg [Pristupljeno 5 Lipanj. 2019].

# PRILOZI

## Popis Slika:

Slika 1: Količine generiranih podataka i kapacitet uređaja kroz godine ..... 6

Slika 2: Karakteristike velikih podataka ..... 7

Slika 3: Razlike sustava velikih podataka i podatkovno manje intezivnih sustava..... 9

Slika 4: Pythonov ekosustav ..... 16

Slika 5: Python interpreter..... 17

Slika 6: JupyterLab ..... 18

Slika 7: Spyder IDE..... 19

Slika 8: Primjer korištenja scikit-learn biblioteke..... 21

Slika 9: Primjer korištenja SciPy biblioteke…….....……22

Slika 10: Deskriptivna statistika u Python programskom jeziku 1. dio ..... 24

Slika 11: Deskriptivna statistika u Python programskom jeziku 2. dio ..... 25

Slika 12: Deskriptivna statistika u Python programskom jeziku 3. dio ..... 26

Slika 13: Deskriptivna statistika u Python programskom jeziku 4. dio ..... 26

Slika 14: Deskriptivna statistika u Python programskom jeziku 5. dio ..... 26

Slika 15: Deskriptivna statistika u Python programskom jeziku 6. dio ..... 27

Slika 16: Deskriptivna statistika u Python programskom jeziku 7. dio ..... 27

Slika 17: Deskriptivna statistika u Python programskom jeziku 8. dio ..... 27

Slika 18: Koeficijent linearne korelacije i postupak njegovog izračuna 1. dio..... 29

Slika 19: Koeficijent linearne korelacije i postupak njegovog izračuna 2. dio..... 30

Slika 20: Jednostruka linearna regresija i postupak njezinog izračuna 1. dio..... 32

Slika 21: Jednostruka linearna regresija i postupak njezinog izračuna 2. dio..... 33

Slika 22: Jednostruka linearna regresija i postupak njezinog izračuna 3. dio..... 33

Slika 23: Višestruka linearna regresija i postupak njezinog izračuna 1. dio..... 34

Slika 24: Višestruka linearna regresija i postupak njezinog izračuna 2. dio..... 35

Slika 25: Vizualizacija podataka u Python programskom jeziku 1. dio..... 36

Slika 26: Vizualizacija podataka u Python programskom jeziku 2. dio..... 36

Slika 27: Vizualizacija podataka u Python programskom jeziku 3. dio..... 37

Slika 28: Vizualizacija podataka u Python programskom jeziku 4. dio..... 38

Slika 29: Jednostavni postupci manipulacije podataka 1. dio..... 39

Slika 30: Jednostavni postupci manipulacije podataka 2. dio..... 39

Slika 31: Jednostavni postupci manipulacije podataka 3. dio..... 40

Slika 32: Jednostavni postupci manipulacije podataka 4. dio..... 40

Slika 33: Jednostavni postupci manipulacije podataka 5. dio..... 41

Slika 34: Jednostavni postupci manipulacije podataka 6. dio..... 41

Slika 35: Napredni postupci manipulacije podataka 1. dio..... 42

Slika 36: Napredni postupci manipulacije podataka 2. dio..... 42

Slika 37: Napredni postupci manipulacije podataka 3. dio..... 43

Slika 38: Napredni postupci manipulacije podataka 4. dio..... 44

## Popis grafikona:

Grafikon 1: Spremnost na implementaciju sustava velikih podataka i njihova integriranost po industrijskim granama……..12

Grafikon 2: Broj prijava i vrsta IT nezgoda od strane Američkih federalnih agencija..... 13

Grafikon 3: „Trade-off“ u računalnim resursima i vremenu razvoja između programskih jezika više i niže razine..... 15

Grafikon 4: Vrste korelacija..... 28

# SAŽETAK

Fenomen Velikih Podataka predstavlja jednu od najznačajnijih i najmanje vidljivih posljedica razvoja tehnologije i interneta. Naime, podatci koje generira današnji globalno povezani svijet rastu eksponencijalnom brzinom, te predstavljaju pravi rudnik zlata za one korisnike koje takve podatke znaju pravilno interpretirati i na osnovu njih donosi uspješne odluke. Ona poduzeća koje najefikasnije i najefektivnije budu implementirale i koristile sustave velikih podataka imat će značajnu konkurentsku prednost na tržištu. Podatkovna analiza i obrada čine jednu od najvažnijih komponenti sustava velikih podataka, a u toj branši podatkovne znanosti najpopularniji je programski jezik Python koji svojim korisnicima pruža velik broj konstantno održavanih programskih biblioteka i razvojnih okruženja otvorenog koda koji su potpuno besplatni. Konkretno, u praktičnom dijelu smo koristeći programske biblioteku Pandas prvenstveno manipulirali podatke po zadanim kriterijima; programsku biblioteku Scikit-learn koristili smo uglavnom za prikaz statističke analize podataka, dok je programska biblioteka Matplotlib korištena da bi se svi ti podatci u konačnici vizualizirali. Zbog svega prikazanoga u praktičnom dijelu gdje smo demonstrirali prednosti programskog jezika Python koje se najviše očituju u njegovoj fleksibilnosti i jednostavnoj programskoj sintaksi koja svakom korisniku pruža iznimno pristupačan način za analizu i obradu podataka uz relativno malu krivulju učenja. Zbog svih navedenih prednosti nije nimalo čudno što je Python programski jezik i njegov popratni ekosustav postao jedan od najpopularnijih programskih alata za korisnike koji nužno ne dolaze iz I.T. sektora, a žele olakšati i unaprijediti poslovanje u svojoj profesiji.



**KLJUČNE RIJEČI:** Veliki Podatci, Python programski jezik, podatkovna analiza i obrada

## SUMMARY

Big Data phenomenon represents one of the most significant and least visible consequences that rapid expansion of the Internet and technology has brought to us. Data that is generated by the globally connected world is growing at an exponential rate, and embodies a real gold mine for those companies that are capable of successfully implementing big data systems. Companies that most efficiently and effectively employ big data systems will be expected to gain substantial competitive advantage on the market. Data analysis and processing are one of the major components that make big data ecosystem; and in those branches of the data science Python programming language is the most prevalent one because it gives its user-free access to huge amount of constantly updated open source programming libraries and development environments. Namely, in the hands-on section of this thesis we have shown some brief capabilities of the most popular Python programming libraries like Pandas, Scikit-learn, SciPy and Matplotlib which are primarily used for data manipulation, statistics and visualization. Because of all demonstrated in the hands-on section, we hope that we have successfully shown advantages of Python programming language that are primarily reflected in its flexibility and versatility. With few lines of code, the user can transform complex heterogenous and unstructured input information in the logically connected and processable data. Python programming language with its gradual learning curve allows event the most unexperienced users to master its syntax in relatively short period of time; this feature makes Python the most popular programming language for users that are not coming from the IT sector, but want to advance and modernize procedures and tools used in their area of expertise.

**KEYWORDS:** Big Data, Python programming language, data analysis and manipulation