Python u strojnom učenju - Pandas

Šanovsky, Nikolina

SVEUČILIŠTE U DUBROVNIKU

ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

NIKOLINA ŠANOVSKY

PYTHON U STROJNOM UČENJU - PANDAS

ZAVRŠNI RAD

Dubrovnik, rujan, 2019.

SVEUČILIŠTE U DUBROVNIKU

ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

PYTHON U STROJNOM UČENJU - PANDAS

ZAVRŠNI RAD

Studij: Primijenjeno/poslovno računarstvo

Kolegij: Strojno učenje

Mentor: izv.prof.dr.sc. Mario Miličević

Student: Nikolina Šanovsky

Dubrovnik, rujan, 2019.

SAŽETAK

Razvojem tehnologije ljudi se okreću prema novim, traženijim poljima znanosti, kao što je strojno učenje. To su prepoznali i IT stručnjaci u Pythonu te su razvili skup biblioteka koji omogućuju rad s podacima na razne načine. S tim poduhvatom Python je izgubio lošu reputaciju ‘toy language-a’, jezika bez opće namjene. Python je već neko vrijeme pri vrhu ljestvice najpopularnijih programskih jezika te je cilj ovoga rada opravdati tu visoku poziciju. S obzirom da se skoro svaki posao svodi na kalkulacije, predviđanja i slično, cilj je ljudima približiti kako s jednostavnim programskim jezikom analizirati podatke i koristiti ih za svoje potrebe.

SUMMARY

With the development of technology, people are more interested in new, more in-demand science fields, like machine learning. The same is true for IT experts in Python language, and for this reason, they developed a set of libraries that simplifies data handling in a variety of ways. Because of this very reason Python lost it's bad "toy language" reputation, a language without general-purpose, and even got to the top of the list of most popular programming languages. The goal of this paper is to justify this position. Since every work with data is, in the foundation, related to calculations, predictions, and the similar. The goal is to describe how to manipulate, analyze and use data through a simple programming language.

Ključne riječi: Python, strojno učenje, programski jezik

Key words: Python, machine learning, programming language

SADRŽAJ

1. UVOD **6**

1.1 Značaj strojnog učenja 6

1.2 Primjena programskog jezika “Python” 6

1.3 Cilj rada 6

2 STROJNO UČENJE **7**

2.2 Statistika 7

2.2.1 Tipovi podataka 7

2.2.2 Uzorak i populacija 8

2.2.3 Mjere centralnih vrijednosti 9

2.2.4 Mjere raspršenja 9

2.3 Bayesov teorem 10

2.3.1. Uvjetna vjerojatnost 10

2.3.2 Potpuna vjerojatnost 11

2.3.3 Bayesova formula 11

2.4.1 Nadzirano strojno učenje (engl. Supervised learning) 12

2.4.3 Polu-nadzirano strojno učenje (engl. Semi-supervised learning) 13

2.4.4 Podržano strojno učenje (engl. Reinforcement learning) 13

2.5 Algoritmi strojnog učenja 14

2.5.1 Regresijski algoritmi (engl. Regression algorithms) 14

2.5.2 Algoritmi stabla odlučivanja (engl. Decision tree algorithms) 15

2.5.3 Bayesovi algoritmi (engl. Bayesian algorithms) 16

2.5.4 Deep learning algoritmi 16

3 PYTHON U STROJNOM UČENJU **18**

3.1 Jupyter notebook 18

3.2 Osnovne Python komponente 19

3.2.1. Liste 19

Slika 13. Primjeri rada s listama 20

3.2.2 Stringovi 20
3.2.3 Tuples 21
3.2.4 Rječnici (engl. Dictionaries) 21
3.2.5 Iteracija i uvjetne konstrukcije 21
3.3 Python biblioteke 22
3.3.1 NumPy (Numerical Python) - Osnove 23
3.3.2 Matplotlib - Osnove 27
3.3.3 Scikit-Learn - Osnove 29
4 PANDAS **33**
4.1 Data frame 33
4.2 Čitanje iz SQL baze podataka 34
4.3 Najvažnije operacije s data frame-om 34
4.4 Indeksiranje i uvjetno odabiranje 36
4.5 Primjenjivanje funkcija na skupovima podataka 38
4.6 Grafovi 38
4.7 Zanimljivosti 39
5 ZAKLJUČAK **41**
LITERATURA 42
POPIS SLIKA 44
IZJAVA 46

1. UVOD

1.1 Značaj strojnog učenja

U današnjem svijetu, dnevno se, putem različitih tehnologija, prikupi nebrojen broj podataka. Ljudi su zahvaljujući sveprisutnosti podataka prepoznali raznolike mogućnosti, od kojih je strojno učenje (engl. *machine learning*) u velikom porastu. Strojno učenje omogućuje da prikupljanjem i analiziranjem podataka, računala provode statističke testove i prate određene uzorke, a potom sami sebe uče i predviđaju određene situacije. Strojno učenje je prisutno u svim poljima života, od prepoznavanja slika i govora, detektiranja neželjene pošte pa sve do samovozećih autobusa i medicinskih dijagnoza.

1.2 Primjena programskog jezika "Python"

Python je objektno-orijentiran programski jezik opće namjene kojeg je stvorio danski programer Guido van Rossum 1991. godine. Python je osmišljen kao programski jezik kojega mogu koristiti i sami početnici u programiranju zbog jednostavne i intuitivne sintakse. Premda se Python ponajviše koristi za pisanje skripti i automatizaciju poslova, sve se više upotrebljava i kod web aplikacija, igrica i učenja. Zahvaljujući raznim bibliotekama koje posjeduje, Python je dobio zamaha u područjima kao što su analiziranje podataka i strojno učenje.

1.3 Cilj rada

Cilj ovoga rada je pokazati kako jednostavan programski jezik kao što je Python može doprinijeti analiziranju podataka i primjeni istih za strojno učenje. Prvotno će se opširnije opisati pojam strojnog učenja. potom navest i ukratko opisati sve Python biblioteke koje to omogućuju, a posebno će se obraditi Pandas biblioteka namijenjena manipulaciji i analiziranju podataka.

2 STROJNO UČENJE

Strojno učenje je skup disciplina i grana znanosti koje su potrebne da bi algoritmi učenja bili funkcionalni i što ispravniji. Neke od njih su: umjetna inteligencija, statistika, Bayesove metode, računarska teorija kompleksnosti, filozofija, psihologija, neurobiologija i slično.

U strojnom učenju poznate su četiri metode učenja: nadzirano, nenadzirano, polu-nadzirano i podržano pri čemu se koristimo algoritmom iz određene skupine algoritama primjerenoj odabranoj metodi.

2.1 Umjetna inteligencija

Umjetna inteligencija (engl. *artificial intelligence* - AI) je grana računalske znanosti koja se bavi razvojem inteligentnih alata, strojeva ili aplikacija. Ti alati raspoznaju govor, sliku, reakcije i motoriku te sami sebe uče rješavanju nekog problema promatrajući okolinu.

Jedni od najpoznatijih AI aparata su osobni asistenti poput Siri (Apple), Cortane (Microsoft), Google Assistant, Alexa (Amazon), itd. Oni pomažu u svakodnevnim radnjama tako što spajanjem na druge uređaje prate naše kretnje, navike, temperaturu tijela i slično, te nas upozoravaju ili navode na nešto, zavisno o čemu je riječ. [1]

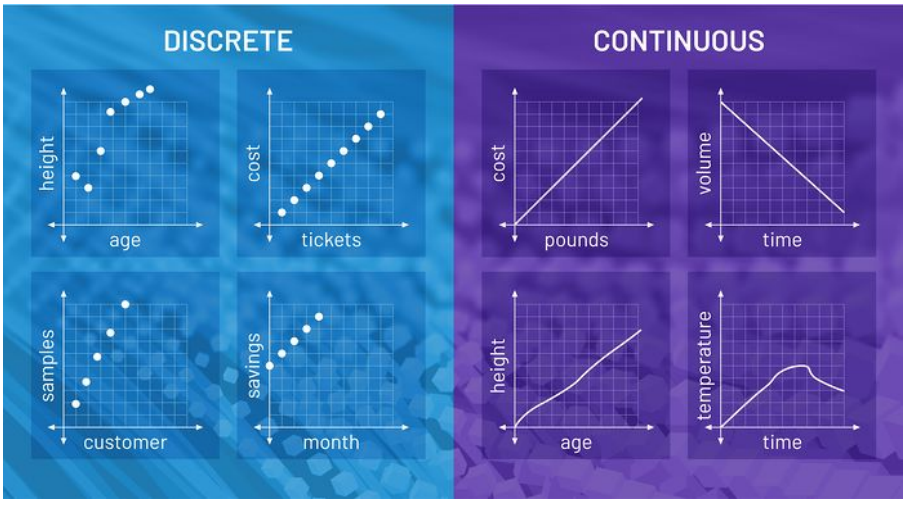
2.2 Statistika

Statistika je grana matematike koja proučavanjem uzoraka donosi zaključke o cijeloj populaciji. Ispravnost zaključka ovisi o kvaliteti izabranog uzorka, tj. o tome je li uzorak reprezentativan.

2.2.1 Tipovi podataka

Da bi odabrali ispravan algoritam za analiziranje podataka, potrebno je znati kojim tipovima podataka raspolažemo. Najučestaliji su numerički i kategorički podaci.

Numerički podaci predstavljaju stvari kojima možemo dodijeliti nekakav broj (mjerljivi su). Mogu biti diskretne ili kontinuirane varijable. Diskretne varijable mogu primiti samo konačan broj vrijednosti, npr. broj automobila koji su prošli ulicom u jednom satu, dok kontinuirane varijable mogu predstavljati cijeli interval ili pravac (u točno jednoj vrijednosti poprima vrijednost 0). Primjer kontinuirane varijable je količina padalina u jednoj godini. (slika 1.)



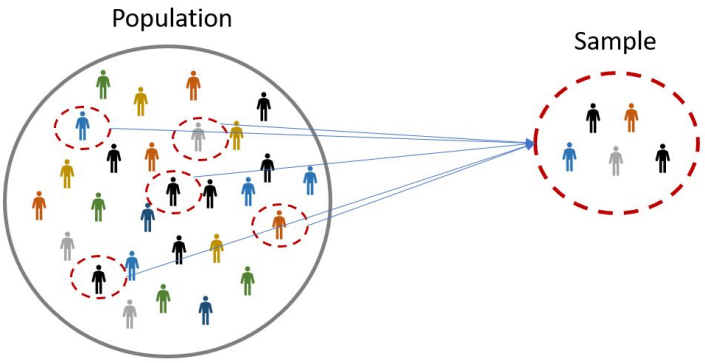
Slika 1. Razlika diskretnih i kontinuiranih varijabli[2]

Kategorički podaci nemaju nikakvo numeričko značenje, premda im se može dodijeliti brojčana vrijednost radi simplificiranja. Oni su najčešće neko obilježje, vrsta, spol, itd.

2.2.2 Uzorak i populacija

Populacija je osnovni skup jedinica s nekim zajedničkim obilježjima nad kojima se vrši neko ispitivanje.

Uzorak je dio populacije, na kojemu, ako je pomno izabran, možemo vršiti istraživanja i donositi zaključke o cijeloj populaciji. (slika 2.)



Slika 2. Uzimanje uzorka iz populacije[3]

2.2.3 Mjere centralnih vrijednosti

Skupovi podataka imaju tendenciju gomilanja oko neke centralne vrijednosti. Te vrijednosti opisuju taj skup podataka i zovemo ih srednje vrijednosti. [4].

Aritmetička sredina (prosječna vrijednost) je omjer zbroja numeričkih vrijednosti i ukupnog broja svih vrijednosti.

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Mod je vrijednost s najvećom frekvencijom u nizu. U slučaju kada su podaci podijeljeni u razrede, mod je aritmetička sredina razreda s najvećom frekvencijom.

Medijan je vrijednost središnjeg podatka u sortiranom nizu, što znači da 50% elemenata ima vrijednost manju ili jednaku njemu, dok ostalih 50% ima veću ili jednaku vrijednost. Ako je n neparan broj onda je medijan (M_e) vrijednost središnje vrijednosti. U slučaju da je n paran, medijan je aritmetička sredina dva središnja člana.

2.2.4 Mjere raspršenja

Raspon podataka (R) je razlika između najveće i najmanje vrijednosti.

$$R = x_{max} - x_{min}$$

Varijanca uzorka je prosječno kvadratno odstupanje od prosjeka gdje n predstavlja broj podataka u uzorku.

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}$$

Varijanca populacije je prosječno kvadratno odstupanje od prosjeka gdje N predstavlja broj podataka u populaciji.

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N}$$

Standardna devijacija uzorka je drugi korijen iz varijance uzorka.

$$s = \sqrt{s^2}$$

Standardna devijacija populacije je drugi korijen iz varijance populacije.

$$\sigma = \sqrt{\sigma^2}$$

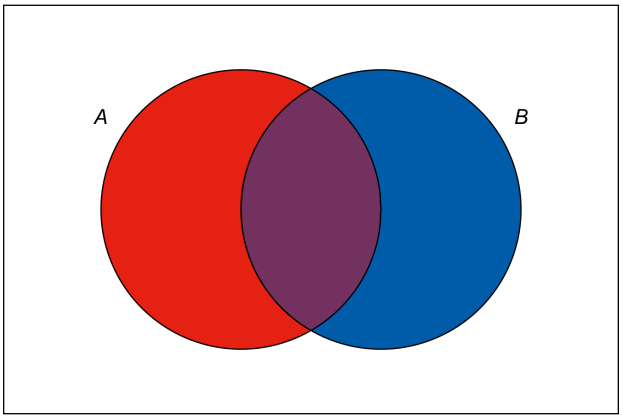
2.3 Bayesov teorem

Bayesov teorem omogućava nam da preispitamo postojeću predikciju (vjerojatnost) s novom informacijom (dokazom). Služeći se apriornom vjerojatnošću možemo izračunati posteriornu. Posteriorna vjerojatnost nam govori kolika je vjerojatnost nekog događaja uzimajući u obzir novu informaciju.[5]

2.3.1. Uvjetna vjerojatnost

Uvjetna vjerojatnost je definirana kao vjerojatnost da se dogodio događaj A ako znamo da se dogodio događaj B. (**p(A|B)**). (slika 3.)

$$p(A|B) = \frac{P(A \cap B)}{P(B)} \quad ili \quad p(A|B) = \frac{P(B|A)p(A)}{P(B)}$$



Slika 3. Uvjetna vjerojatnost[6]

2.3.2 Potpuna vjerojatnost

Kažemo da međusobno disjunktni događaji (hipoteze) $H_1, H_2, ..., H_n$ čine **potpuni sustav događaja** ako vrijedi $\bigcup_{i=1}^{n} H_i = \Omega \ \ i \ p(H_i) > 0$.[7]

Imamo proizvoljan događaj A, te iz prethodno navedenog svojstva potpunog događa vidimo da vrijedi:

$$p(A) = p(A \cap \Omega) = p(\bigcup_{i=1}^{n} = A \cap H_i) = \sum_{i=1}^{n} p(A \cap H_i) = \sum_{i=1}^{n} p(H_i)p(A|H_i)$$

2.3.3 Bayesova formula

Imamo potpun sustav događaja $H_1, H_2, ..., H_n$ te znamo da se nakon izvođenja pokusa pojavio događaj A kao njegov ishod. Postavljamo pitanje koliku vjerojatnost imaju hipoteze $H_1, H_2, ..., H_n$ znajući da se dogodio događaj A.[8]

$$p(H_i|A) = \frac{p(H_i)p(A|H_i)}{p(A)}$$

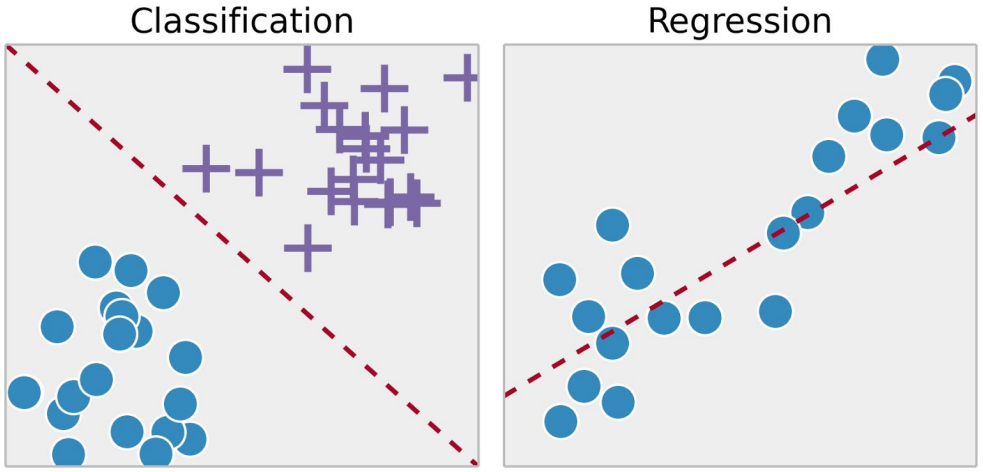
2.4 Metode strojnog učenja

Metode strojnog učenja razlikujemo po izlaznim podacima i kako se odnosimo prema njima. Razlikujemo ih po tome jesu li klasificirani, daju li rezultat i po strukturi modela.

2.4.1 Nadzirano strojno učenje (engl. *Supervised learning*)

Nadzirano strojno učenje koristi se kad su izlazni podaci klasificirani te kad algoritmi uče na temelju ulaznih podataka (engl. *training data*). Model je zasnovan na predikcijama i ispravljanju istih u slučaju pogreške, što opisuje i sami proces učenja. Proces se ponavlja sve dok postoji značajna vjerojatnost pogreške.[9]

Primjeri problema kod kojih se koristi nadzirano učenje su klasifikacija (npr. slika) i regresija. (slika 4.)

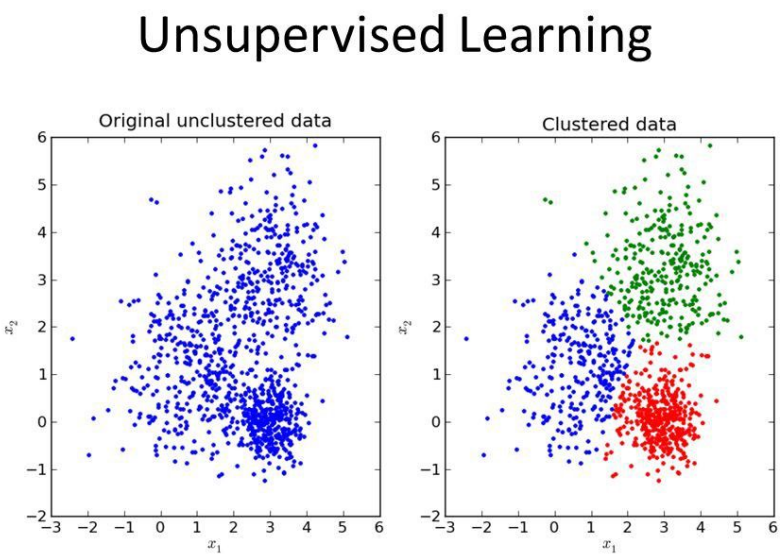


Slika 4. Klasifikacija i regresija[10]

2.4.2 Nenadzirano strojno učenje (engl. *Unsupervised learning*)

Nenadzirano učenje koristi se kada se iz neoznačenih podataka donose zaključci deduciranjem osnovne strukture te raspoznavanjem određenih uzoraka. Cilj ovog učenja nije donošenje ispravnih predikcija, već pokazivanje skrivenih uzoraka koji nisu bili uočljivi prije. [9]

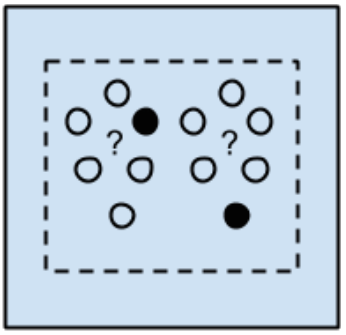
Primjeri problema kod kojih se koristi nenadzirano učenje su grupiranje (slika 5.), redukcija dimenzionalnosti, udruživanje i slično. [9]



Slika 5. Nenadzirano učenje[11]

2.4.3 Polu-nadzirano strojno učenje (engl. *Semi-supervised learning*)

U polu-nadziranom učenju ulazni podaci su mješavina označenih i neoznačenih podataka. Zasniva se na donošenju predikcija, ali tek nakon strukturiranja modela (problemi klasifikacije i regresije). [9](slika 6.)

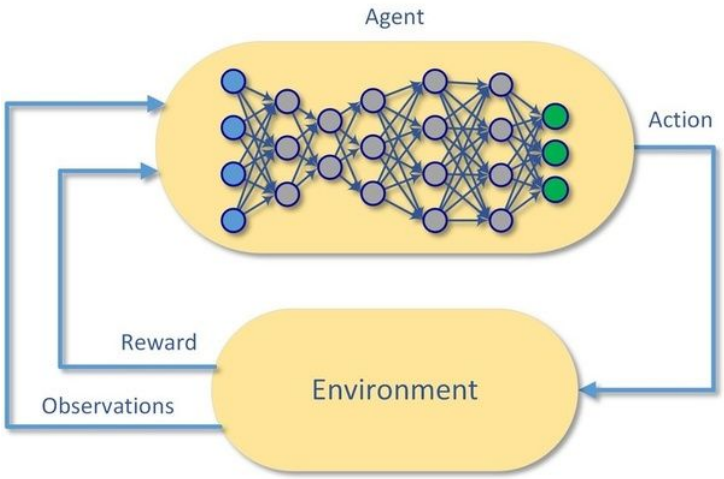


Slika 6. Polu-nadzirano strojno učenje[9]

2.4.4 Podržano strojno učenje (engl. *Reinforcement learning*)

Podržano strojno učenje je učenje bez označenih ulaznih podataka, gdje se učenje zasniva na temelju izlaznih podataka. Cilj tog učenja je pronaći optimalnu akciju, a produkt

toga je dobivanje nagrade. Najčešće se koristi za učenje *botova* u igricama, tako da bi dočarali igranje stvarne osobe. (slika 7.)



Slika 7. Podržano strojno učenje[12]

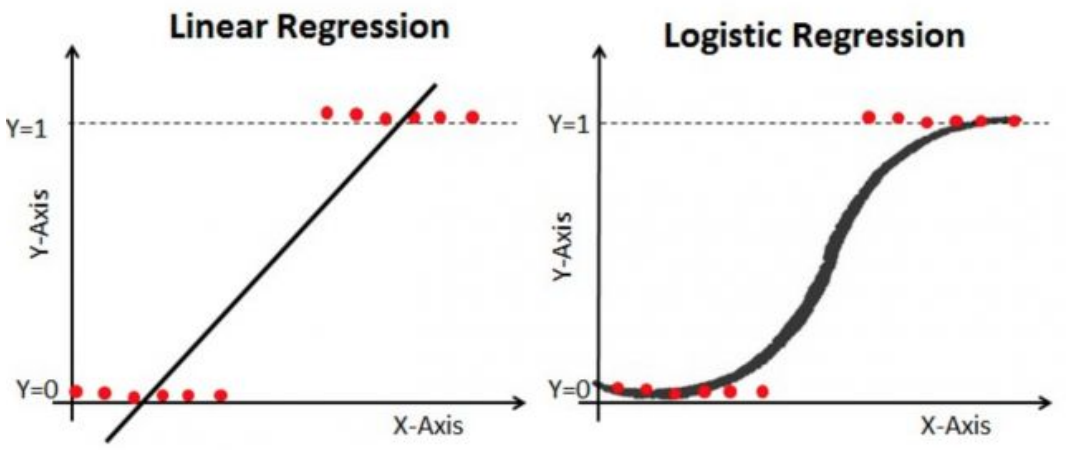
2.5 Algoritmi strojnog učenja

Algoritmi strojnog učenja najčešće su grupirani po svojim funkcionalnostima. Neki od njih su regresijski algoritmi, stablo odlučivanja, Bayesovi algoritmi, *deep learning* algoritmi, itd.

2.5.1 Regresijski algoritmi (engl. *Regression algorithms)*

Regresijski algoritmi služe za razumijevanje veza među varijablama, koje se postepeno 'poliraju' učeći na pogreškama kod predikcije, dok se u cijelom procesu najviše služe statističkim metodama. [9]

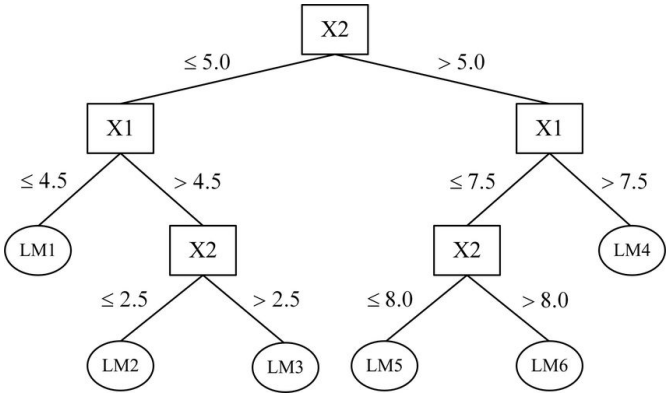
Najpoznatiji regresijski algoritmi su: *Ordinary Least Squares Regression (OLSR)*, linearna regresija (slika 8.) , logistička regresija (slika 8.), postepena regresija, *Multivariate Adaptive Regression Splines, itd. [9]*



Slika 8. Linearna i logistička regresija[13]

2.5.2 Algoritmi stabla odlučivanja (engl. *Decision tree algorithms)*

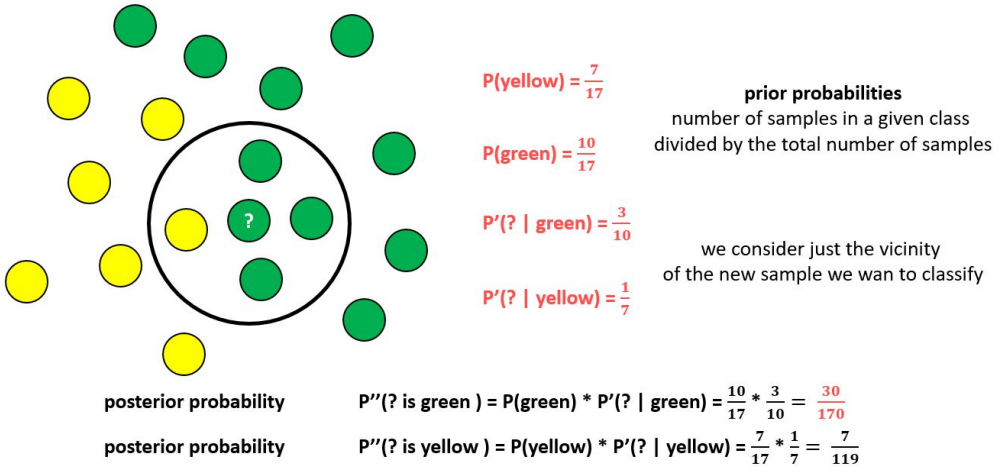
Na temelju vrijednosti podataka i donešenih odluka algoritmi stabla odlučivanja slažu model. Struktura se grana kao stablo sve dok se ne zaključi predikcija na temelju podataka. Najčešće se koriste za klasifikaciju ili regresijske modele, te su poznati zbog brzine izvođenja. [9] Najpoznatiji algoritmi ove skupine su: klasifikacijsko i regresijsko stablo, *M5* (slika 9.), *Conditional Decisions Tree, Iterative Dichotomiser 3* (ID3) i drugi.



Slika 9. MP5 stablo odlučivanja[14]

2.5.3 Bayesovi algoritmi (*engl. Bayesian algorithms*)

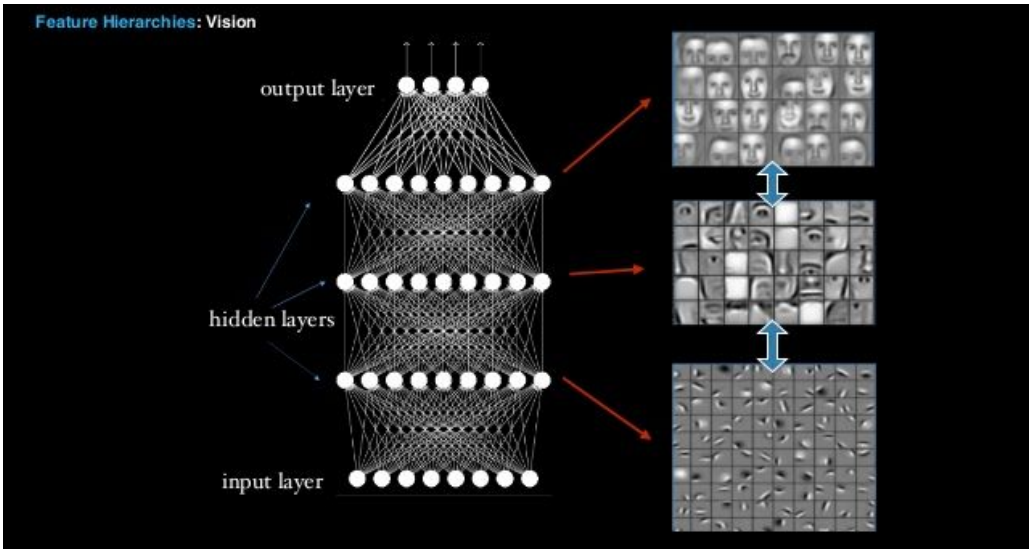
Bayesovi algoritmi su oni koji, kao što sama riječ govori, koriste Bayesov teorem za klasifikaciju ili regresijsku analizu. [9]. Najpoznatiji su: *Naive Bayes* (slika 9.), *Gaussian Naive Bayes, Multinomial Naive Bayes, Bayesian Network* (BN) i drugi.



Slika 9. *Naive Bayes* algoritam[15]

2.5.4 *Deep learning* algoritmi

Deep learning algoritmi se koriste kod velikih baza podataka te služe za pravljenje kompleksnih i velikih neuronskih mreža. Prolazeći kroz više slojeva, algoritam, iz sirovih materijala, dolazi do naprednijih svojstava. (slika 10.)



Slika 10. *Deep learning* algoritam - slojevi[16]

Najpoznatiji algoritmi su: *Convolutional Neural Network* (CNN)*, Recurrent Neural Networks* (RNNs)*, Long Short-Term Memory Networks* (LSTMs*), Stacked Auto-Encoders, Deep Boltzmann Machine* (DBM)*, Deep Belief Networks* (DBN). [9]

3 PYTHON U STROJNOM UČENJU

Python, programski jezik visoke razine, došao je na vrh ljestvice programskih jezika korištenih za *Data science*. Razlog zašto je s vrha ljestvice maknuo i popularni R programski jezik, je opsežan skup raznolikih biblioteka za *Data science*.

Python nam omogućuje vađenje i manipulaciju podataka, analizu podataka te modeliranje, evaluiranje, razvoj i ažuriranje različitih rješenja. [17]

Da bi iskoristili Pythonov potpuni potencijal kod strojnog učenja koristit ćemo se *open source* web aplikacijom **Jupyter Notebook**.

3.1 Jupyter notebook

Jupyter Notebook omogućava kombiniranje koda, vizualizacije, jednadžbi i slično, dok u isto vrijeme možemo kod djeliti s drugima, s obzirom na to da se radi o web aplikaciji. Samo ime Jupyter dolazi od programa koje podržava, a to su: Julia, Python i R. [18]

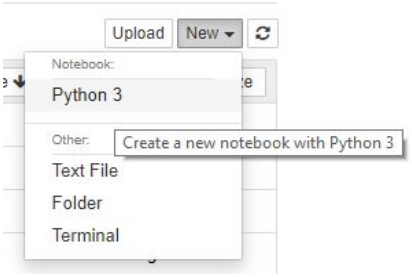
Za instalaciju Jupyter-a, koristimo se Python alatom zvanim *pip* u komandnoj liniji. Na primjer:

$ pip install jupyter

Pokrećemo ga naredbom:

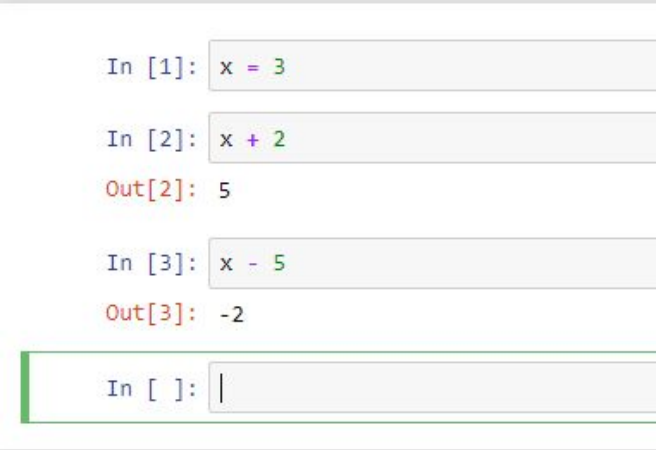
$ jupyter notebook

Nakon naredbe otvorit će nam se prozor u *defaultnom* pregledniku. Zatim stvaramo novi *notebook* klikom na *new* i odabirom na jednu od opcija. U ovom slučaju to će biti instalirana verzija Pythona. (slika 11.)



Slika 11. Stvaranje novog *notebooka*

Svaka ćelija ima mogućnost pisanja u više redaka, a pokreće se naredbom **SHIFT + ENTER**. Broj u uglatim zagradama predstavlja koja se ćelija kad izvršila. Ćelije nam omogućuju da razdvojimo dijelove koda radi preglednosti, a varijable se prenose. (slika 12.)



Slika 12. Ćelije u Jupyter-u

3.2 Osnovne Python komponente

Da bi u Pythonu započeli rad s podacima, potrebno je znati neke osnovne strukture podataka kao što su liste, stringovi, *tuples*, rječnici (*dictionaries*), iteracije i uvjetne konstrukcije. [19]

3.2.1. Liste

Liste mogu sadržavati različite tipove podataka, premda je praksa da su sastavljene od istoga tipa. Liste su promjenjivog tipa, a svaki element se može zasebno mijenjati. Definiramo ih u uglatim zagradama, odvajajući elemente zarezom. (slika 13.)[19]

Za pristup pojedinom elementu uz ime lista u uglatim zagradama dodamo poziciju tog elementa, dok u slučaju kad želimo pristupiti djelu liste napišemo početnu i završnu poziciju odvajajući ih znakom **:** . (slika 13.)

Dodavanje novog elementa na kraj liste postižemo funkcijom **append** stavljajući novu vrijednost u normalne zagrade (slika 13.)

```
In [2]: lista = [1,2,3,4,5]

In [3]: lista[0] = 2

In [4]: lista
Out[4]: [2, 2, 3, 4, 5]

In [6]: lista.append(6)

In [7]: lista
Out[7]: [2, 2, 3, 4, 5, 6]

In [8]: lista[2:4]
Out[8]: [3, 4]
```

Slika 13. Primjeri rada s listama

3.2.2 Stringovi

Stringovi se navode u jednostrukim, dvostrukim ili trostrukim navodnicima (trostruki ako želimo u više redaka) te su, za razliku od lista, nepromjenjivi. (slika 14.)

```
In [1]: string = 'Ime'

In [2]: string2 = "Prezime"

In [3]: string3 = '''
        Godina
        Rođenja'''

In [4]: print(string, string2, string3)
        Ime Prezime
        Godina
        Rođenja
```

Slika 14. Primjer stringova

3.2.3 *Tuples*

Tuple je nepromjenjiva, uređena kolekcija. Navodi se u običnim zagradama, a elementi se odvajaju zarezom. Pojedinom elementu pristupamo kao i u listama. (slika 15.)

```
In [1]: tuple = ('prvi', 'drugi', 'treći')

In [2]: tuple[2]
Out[2]: 'treći'
```

Slika 15. Primjer *tupl-a*

3.2.4 Rječnici (engl. *Dictionaries*)

Rječnici su nesortirani parovi **ključ - vrijednost**, uz uvjet da je svaki ključ jedinstven. Definiraju se u vitičastim zagradama ({}), ključ - vrijednost se odvaja znakom **:** , a parovi se odvajaju zarezom. (slika 16.)

```
In [1]: dictionary = { 'ključ1' : 1 , 'ključ2' : 2, 'ključ3' : 3 }

In [2]: dictionary
Out[2]: {'ključ1': 1, 'ključ2': 2, 'ključ3': 3}

In [3]: dictionary.keys()
Out[3]: dict_keys(['ključ1', 'ključ2', 'ključ3'])

In [4]: dictionary.values()
Out[4]: dict_values([1, 2, 3])
```

Slika 16. Rad s rječnicima

3.2.5 Iteracija i uvjetne konstrukcije

Najpoznatije iteracije je **FOR** petlja, dok je **IF-ELSE** najučestalija uvjetna konstrukcija. U **FOR** petlji iteracija može biti lista ili opseg (od-do). (slika 17.)

```
In [1]: lista = [1,2,3,4]

        for x in lista:
            print(x+1)

        2
        3
        4
        5

In [2]: for i in range (0,5):
            print(i)

        0
        1
        2
        3
        4
```

Slika 17. Primjer FOR petlje s listom i opsegom kao iteracijom

3.3 Python biblioteke

Biblioteke su skup funkcija i modula, napravljenih da bi olakšali pisanje koda tako da već postojeće funkcionalnosti iskoristimo te da ih sami ne moramo pisati.

Python biblioteke koje se najčešće koriste za rad s podacima (*data science/analysis*) su **NumPy**, **Pandas, SciPy**, **Scikit Learn, Matplotlib**, **Seaborn** i mnoge druge.

Biblioteku možemo uključiti tako da joj dodijelimo pseudonim ili uključujući sve ili pojedine funkcionalnosti iz biblioteke bez pseudonima, nakon što ih možemo pozvati samo nazivom funkcije. (slika 18.)

```
In [1]: import math as m
        m.sqrt(16)

Out[1]: 4.0

In [2]: from math import *
        sqrt(16)

Out[2]: 4.0
```

Slika 18. Uključivanje biblioteke u kod

3.3.1 NumPy (Numerical Python) - Osnove

Numpy je biblioteka sastavljena od višedimenzionalnih vektora i funkcionalnosti potrebnih za rad s njima.

Najčešće se koristi za matematičke i logičke operacije nad vektorima, Fourierove transformacije, manipulacije oblicima, operacije potrebne za linearnu algebru, te u sebi sadrži još i slučajni generator brojeva. [20]

Paket ćemo instalirati sljedećom naredbom:

$ pip install numpy

Najvažnija komponenta je **ndarray** (višedimenzionalni vektor), koja predstavlja 'kontejner' koji sadrži elemente iste veličine i tipa. Svaki element je podatkovnog tipa, **dtype objekt.** Pozivajući parametar **shape** provjeravamo dimenzije vektora, dok pozivajući parametar **dtype** otkrivamo kojeg su tipa podaci unutar vektora. (slika 19.)

```
In [1]: import numpy as np

In [3]: a = np.array([1,2,3,4])

In [4]: a
Out[4]: array([1, 2, 3, 4])

In [6]: a.shape
Out[6]: (4,)

In [9]: b = np.array([[1,2,3], [4,5,6]])

In [10]: b
Out[10]: array([[1, 2, 3],
                [4, 5, 6]])

In [14]: b.dtype
Out[14]: dtype('int32')

In [15]: b.shape
Out[15]: (2, 3)
```

Slika 19. ndarray parametri

NumPy sadrži puno veći broj numeričkih tipova podataka od samog Pythona (slika 20.)

bool_	Boolean (True or False) stored as a byte
int_	Default integer type
intc	Identical to C int e.g int32 in64
intp	Integer used for indexing
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
complex128	Complex number, represented by two 64-bit floats (real and imaginary components)

Slika 20. Numerički tipovi podataka u NumPy-u

***Data type* objekt** predstavlja fiksni blok memorije namijenjenog vektoru. Koliki je blok koji zauzima ovisi o tipu podatka, veličini podatka, redu bajtova i slično.

NumPy sadrži brojne matematičke operacije, kao što su trigonometrijske (sin, cos, arcsin, arccos, degrees,..) aritmetičke (add, subtract, multiply, divide, mod), operacije s kompleksnim brojevima (real, imag, conj, angle) i mnoge druge.

Sadrži i statističke operacije kao što su minimum, maksimum, percentili, aritmetička sredina, standardna devijacija i varijanca danih elemenata u vektoru.[21] (slika 21.)

Olakšava nam i rješavanje linearnih struktura, kao što su matrice. Sadrži operacije za množenje matrica, računanje determinante, traženje inverza i slično.

```
In [1]: import numpy as np
        a = np.array([1,2,3,4])

In [2]: a
Out[2]: array([1, 2, 3, 4])

In [3]: np.amin(a)
        #minimum
Out[3]: 1

In [4]: np.amax(a)
        #maksimum
Out[4]: 4

In [5]: np.ptp(a)
        #maximum-minimum
Out[5]: 3

In [6]: np.percentile(a,75,0)
        #percentile(array,percentage,axis)
Out[6]: 3.25

In [7]: np.median(a)
        #medijan
Out[7]: 2.5

In [8]: np.mean(a)
        #aritmetička sredina
Out[8]: 2.5

In [9]: np.average(a)
        #prosjek
Out[9]: 2.5

In [10]: np.std(a)
         #standardna devijacija
Out[10]: 1.118033988749895

In [11]: np.var(a)
         #standardna varijanca
Out[11]: 1.25
```

Slika 21. Primjeri statističkih operacija

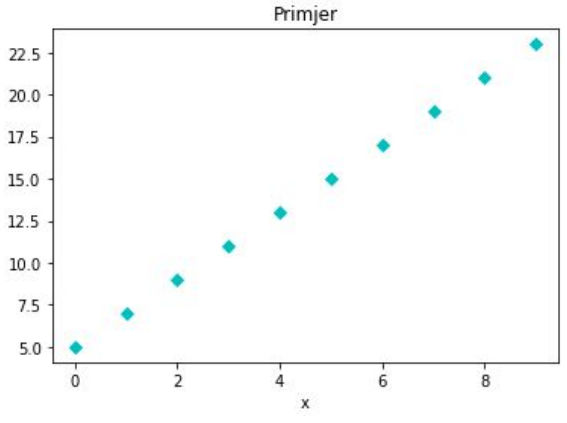
Kombinacija **NumPy-a** i **Matplotlib-a** daje nam mogućnost vizualiziranja funkcija i grafova. Uz pomoć numpy-a definiramo x i y os dok s **pyplot()** funkcijom, koju smo importali iz matplotlib-a, crtamo graf. Graf možemo dodatno 'ukrasiti' dodavanjem trećeg parametra u kojem je prvi dio izgled funkcije, a drugi boja funkcije, sve u navodnicima. Sve oznake pogledati ovdje https://www.tutorialspoint.com/numpy/numpy_matplotlib.htm. (slika 22.)

```
import numpy as np
from matplotlib import pyplot as plt

x = np.arange(10)
y= 2*x + 5

plt.title("Primjer")
plt.xlabel("x")
plt.ylabel("y")

plt.plot(x,y, "Dc")
plt.show()
```



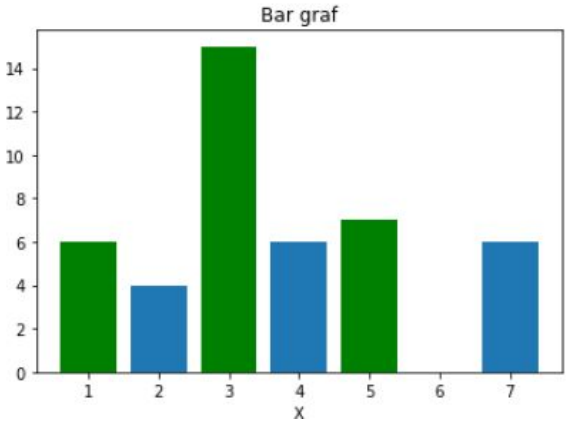
Slika 22. Crtanje grafa s NumPy-om i Matplotlib-om

Pyplot funkcija ima mogućnost crtanja stupčastih grafova (bar) , što možemo vidjeti na slici 23.

```
from matplotlib import pyplot as plt
x = [2,4,7]
y = [4,6,6]

x2 = [1,3,5]
y2 = [6,15,7]
plt.bar(x, y, align = 'center')
plt.bar(x2, y2, color = 'g', align = 'center')
plt.title('Bar graf')
plt.ylabel('Y')
plt.xlabel('X')

plt.show()
```



Slika 23. Crtanje stupčastih dijagrama

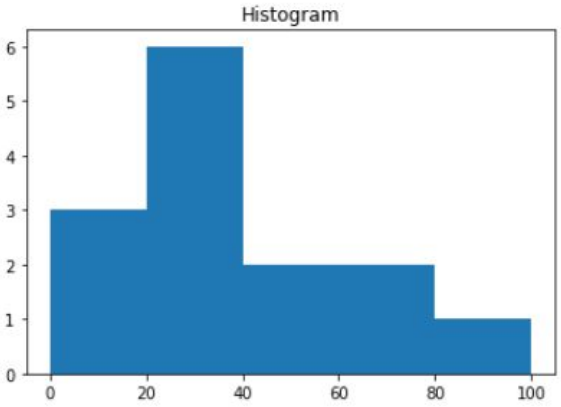
Najčešća vizualizacija frekvencije i distribucije podataka je histogram, koji također možemo prikazati pomoću numPy-a, definirajući vrijednosti i intervale. (slika 24.)

```
import numpy as np
from matplotlib import pyplot as plt

a = np.array([30,12,14,55,33,88,23,43,29,39,20,6,77,62])
hist,bins = np.histogram(a, bins = [0,20,40,60,80,100])
print (hist)
print (bins)
```

```
[3 6 2 2 1]
[  0  20  40  60  80 100]
```

```
plt.hist(a,bins)
plt.title("Histogram")
plt.show()
```



Slika 24. Histogram pomoću numPy-a i matplotlib-a

3.3.2 Matplotlib - Osnove

Matplotlib biblioteka služi za vizualizaciju podataka i crtanje 2D grafova. U potpoglavlju o NumPy-u vidjeli smo osnove crtanja grafa, a sad ćemo vidjeti i ostale mogućnosti.

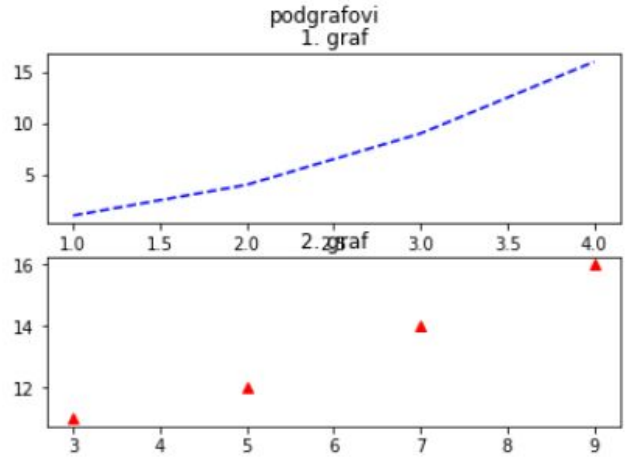
Prikazivanje više skupova podataka, dobit ćemo tako da u metodu **plot()** dodamo još jedan par parametara za drugi skup podataka, dok prikazivanje više pod-grafova na istoj 'figuri' (*figure*) postižemo metodom **subplot(broj redaka, broj stupaca, redni broj)**. (slika 25.)

```
from matplotlib import pyplot as plt

plt.subplot(2,1,1)
plt.plot([1,2,3,4],[1,4,9,16],'b--')
plt.title('1. graf')

plt.subplot(2,1,2)
plt.plot([3,5,7,9],[11,12,14,16],'r^')
plt.title('2. graf')

plt.suptitle('podgrafovi')
plt.show()
```



Slika 25. Crtanje pod-grafova

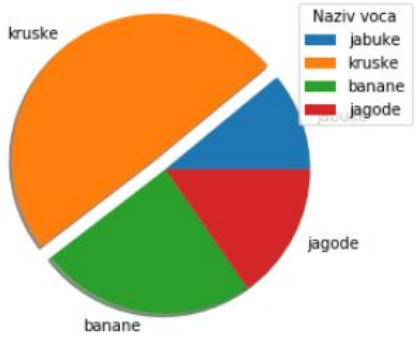
Za crtanje većeg broja grafova, ova metoda nije optimalna, te se stoga koristi metoda **subplots(*nrows, ncolumns, figsize*).** Pojedinom grafu pristupamo kao u matričnom sustavu (a[row:column)).

Stupčasti dijagram (spomenut u 3.3.1) sadrži još dvije važne metode: **legend()** - prikazuje što prikazuje koja boja i **xticks()** - namještanje labela stupaca na x-osi.

pie() metodu koristimo za pita dijagram (engl. *pie chart*). Dijagram možemo osjenčiti, a pojedine elemente naglasiti s parametrom explode. (slika 26.)

```
from matplotlib import pyplot as plt
import numpy as np

voce = ["jabuke","kruske","banane","jagode"]
frekvencija = [10,45,22,14]
Explode = [0,0.1,0,0]
plt.pie(frekvencija,explode=Explode,labels=voce,shadow=True)
plt.axis('equal')
plt.legend(title='Naziv voca')
plt.show()
```

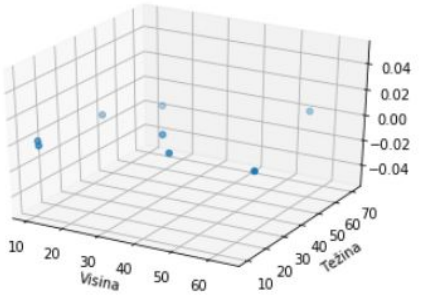


Slika 26. Pita dijagram (*Pie chart*)

Da bi vizualizirali raspršeni graf (problemi regresije) koristimo metode **scatter(*height, weight*), xlim(*int*) te ylim(*int*).** Raspršeni graf možemo vizualizirati i preko 3D modela uključujući modul **mplot3d** te dodjeljivanjem parametra **projection = '3d'** **axes** metodi. (slika 27.)[22]

```
from matplotlib import pyplot as plt
import numpy as np
from mpl_toolkits import mplot3d

ax = plt.axes(projection='3d')
ax.scatter3D([41,13,10,22,55,64,12,33],[22,40,13,55,71,21,10,34])
ax.set_xlabel("Visina")
ax.set_ylabel("Težina")
plt.show()
```



Slika 27. 3D Graf

3.3.3 Scikit-Learn - Osnove

Scikit-Learn,biblioteka najčešće korištena uz NumPy i Pandas, je namijenjena za modeliranje podataka. Koristi se za klasifikaciju, regresiju i grupiranje.

Scikit dolazi s nekoliko standardnih podatkovnih skupova (engl. *datasets*), poput *iris* i *digits.* (slika 28.)

```
from sklearn import datasets
digits = datasets.load_digits()
digits.data
```

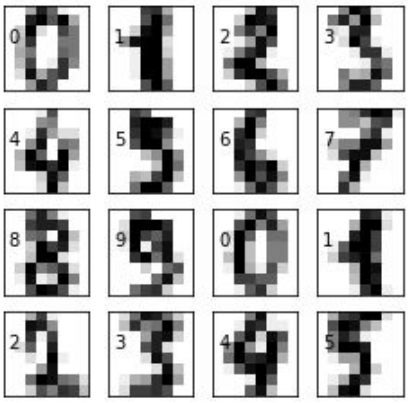
```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0.,  2., ..., 12.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

Slika 28. *Digits dataset*

Podaci su uvijek u 2D-matrici (n uzoraka, n značajki), bez obzira na originalan oblik. Podatke možemo i vizualizirati korištenjem subplot metode. U primjeri ćemo uzeti prvih 16 primjeraka, smjestiti ih u 4x4 matricu te izraditi njihove ‘slike’ (*images*). (slika 29.)

```
## Prvih 16 primjera, matrica 4x4
fig = plt.figure(figsize = (4,4))

for i in range(16):
    ax = fig.add_subplot(4, 4, i+1, xticks=[], yticks=[])
    ax.imshow(digits.images[i],cmap=plt.cm.binary,interpolation='nearest')
    ax.text(0, 3, str(digits.target[i]))
```



Slika 29. Primjer vizualizacije podataka kombinacijom matplotlib-a i sklearn-a

Linearna regresija predviđa vrijednost zavisne varijable **y** u odnosu na nezavisnu varijablu **x.** [23]. Podatke učitavamo **pandas** funkcijom read_csv(*path*), oblik skupa podataka provjeravamo atributom **shape,** koji nam vraća broj podataka i značajki, a stvarne podatke ispisujemo metodom **head()**. Statističke podatke o skupu možemo vidjeti pozivanjem metode **describe()**. (slika 30.)

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

dataset = pd.read_csv('C:\\Users\\Nikolina Šanovsky\\Desktop\\student_scores.csv')
print(dataset.head())
print('\n')

dataset.describe()
```

```
   Hours  Scores
0    2.5      21
1    5.1      47
2    3.2      27
3    8.5      75
4    3.5      30
```

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

Slika 30. Primjeri head() i describe() metoda na skupu podataka

Prvi korak prema linearnoj regresiji je odvajanje podataka u zavisne i nezavisne varijable, te odvajanje podataka za učenje i podataka za testiranje. U ovom primjeru odvojit ćemo ⅓ podatka za učenje, a ostatak za testiranje. Zatim stvaramo objekt linearne regresije, kojemu naknadno opisujemo model za učenje. Testiranje vršimo pozivajući metodu **predict().** (slika 31.) [24]

Preostaje nam samo skicirati dva modela linearne regresije, jedan s podacima za učenje. a drugi s testnim podacima. (slika 32.)

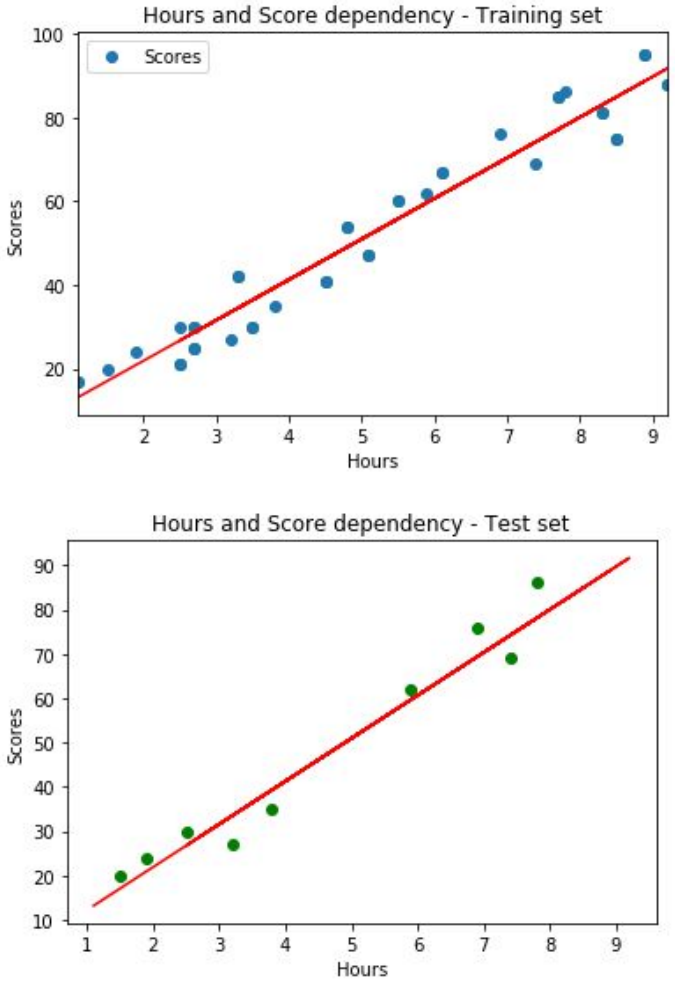
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plot
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

%matplotlib inline

dataset = pd.read_csv('C:\\Users\\Nikolina Šanovsky\\Desktop\\student_scores.csv')
dataset.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values

xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size = 1/3, random_state = 0)
linearRegressor = LinearRegression()
linearRegressor.fit(xTrain, yTrain)
yPrediction = linearRegressor.predict(xTest)
```

Slika 31. Priprema za model linearne regresije



Slika 32. Usporedba podataka za učenje i testiranje

4 PANDAS

Pandas je jedna od najkorištenijih biblioteka za analizu podataka, jer uzimajući podatke iz CSV, TSV datoteke ili SQL baze podataka, stvara objekt zvan ***data frame*** **(tablicu podataka)**. [25]

4.1 *Data frame*

Podaci se učitavaju kao python komponente, lokalne datoteke ili preko URL-a. a potom se pretvaraju u *data frame.*

pd.read_filetype()

pd.DataFrame()

Prilikom rada na podacima, *data frame* možemo i spremiti metodom **to_filetype(filename)**.

Možemo stvoriti i svoj *data frame* tako da u *dictionary* spremimo parove ključ-vrijednost, tj. u ovom slučaju naziv stupca-vrijednosti stupca, te onda *data frame-u* dodijeliti taj *dictionary*. (slika 33.)

```
import pandas as pd
data={
    'visina':[158,160,163,175,180,180],
    'tezina':[50,54,68,60,63,64]
}
podaci = pd.DataFrame(data, index=['Lea', 'Tea', 'Rea', 'Mara','Sara','Vanja'])
podaci
```

	visina	tezina
Lea	158	50
Tea	160	54
Rea	163	68
Mara	175	60
Sara	180	63
Vanja	180	64

Sika 33. Stvaranje *data frame-a*

Pojedinom podatku pristupamo s atributom **loc[‘ključ’].**

4.2 Čitanje iz SQL baze podataka

Da bismo čitali iz baze podataka, trebamo uspostaviti vezu s bazom, dakle, prvo trebamo skinuti python biblioteku za bazu. U ovom radu koristit ćemo se SQLite bazom podataka, stoga ćemo instalirati **pysqlite3** biblioteku.

```
import sqlite3

con = sqlite3.connect("database.db")

df = pd.read_sql_query("SELECT * FROM table_name", con)
```

4.3 Najvažnije operacije s *data frame-om*

Pri učitavanju skupa podataka, korisno je vidjeti prvih par redaka, da vidimo strukturu podataka. To možemo postići metodom **head(n)**, koja po *defaultu* izbacuje prvih par redaka, ali možemo joj dodijeliti i proizvoljan broj. Da bi vidjeli zadnjih par redaka, koristimo metodu **tail(n).**

Pozivajući metodu **info()**, dobijamo osnovne informacije o skupu podataka, kao što su broj varijabli i značajki, tip podataka i koliko prostora zauzima. (slika 34.)

```
podaci.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6 entries, Lea to Vanja
Data columns (total 2 columns):
visina    6 non-null int64
tezina    6 non-null int64
dtypes: int64(2)
memory usage: 120.0+ bytes
```

Slika 34. Informacije o *data frame-u*

Atribut **shape** daje nam broj redaka i stupaca, što je korisno kod sređivanja i filtriranja podataka (da saznamo koliko nam je podataka ostalo nakon).

Metoda **append()** vraća kopiju originalnog *data frame-a* s novim vrijednostima, a metoda **drop_duplicates()** vraća kopiju s uklonjenim duplikatima.

Ako ne želimo konstantno pridjeljivati novom *data frame-u* izmijenjeni skup podataka, onda koristimo argument **inplace = True** unutar metoda kojima radimo promjene.

```
temp_df.drop_duplicates(inplace=True)
```

Drop_duplicates() sadrži i argument **keep** s tri opcije:

1. **first -** obriši sve duplikate osim prvog
2. **last -** obriši sve duplikate osim zadnjeg
3. **false -** obriši sve duplikate

Ako želimo preimenovati stupac koristit ćemo metodu rename() putem *dictionary-a*. (slika 35.)[26]

```
podaci.rename(columns={
        'visina': 'visina_daka',
        'tezina': 'tezina_daka'
    }, inplace=True)


podaci.columns
```

```
Index(['visina_daka', 'tezina_daka'], dtype='object')
```

Slika 35. Preimenovanje stupaca

Kako bi provjerili koliko imamo *null* vrijednosti u skupu podataka koristimo agregaciju **isnull()** i **sum()** metode.

podaci.isnull().sum()

Da bi uklonili *null* vrijednosti koristimo **dropna()** metodu koja će obrisati svaki redak koji ima bar jednu *null* vrijednost i spremiti novi skup podataka u kopiju. Osim redaka, možemo brisati i stupce s *null* vrijednostima, dodavajući parametar **axis=1.**

Kako ne bi brisali stupce koji sadrže vrijedne podatke, samo zato što imaju jednu/dvije *null* vrijednosti, koristit ćemo se tehnikom zvanom *Imputation* (‘podmetanje’). Njome se često koriste tehnički inženjeri tako da svaku *null* vrijednost zamijene

aritmetičkom sredinom ili medijanom tog stupca. [26] Popunjavanje praznih vrijednosti postižemo metodom **fillna(nova vrijednost).**

U poglavlju Scikit-Learn - Osnove smo vidjeli da metodom **describe()** vidimo distribuciju kontinuiranih varijabla. Osim na cijelom skupu podataka, može se koristiti i na kategoričkoj značajki (stupcu), gdje potom dobijemo detaljan pregled jedinstvenih vrijednosti, kategorija, i slično.

Koeficijent korelacije pokazuje nam odnos između dvije varijable. Ako je pozitivan, to znači da ako jedna varijabla raste i druga raste (jedna pada i druga pada), a ako je negativan to znači da im je odnos takav da ako jedna pada, druga raste. Broj 1.0 označava najbolju korelaciju između varijabli. Koristimo metodu **corr()** da bi ispisali koeficijente korelacije. (slika 36.)

```
In [22]: podaci.corr()
```

Out[22]:

	visina_daka	tezina_daka
visina_daka	1.000000	0.567979
tezina_daka	0.567979	1.000000

Slika 36. Koeficijent korelacije - primjer

4.4 Indeksiranje i uvjetno odabiranje

Da bi pristupili retku, koristimo dva argumenta: **loc** i **iloc. loc** se koristi za traženje po imenu, a **iloc** za traženje po indeksu. Možemo odabrati više redaka tako da napišemo od-do, isto kao u Python listama. Jedina razlika je što se kod indeksiranja s imenima (loc) uključuje i krajnji navedeni rub, što Python liste i iloc to ne rade. (slika 37.)

```
podaci.loc['Rea':'Sara']
```

	visina_daka	tezina_daka
Rea	163	68
Mara	175	60
Sara	180	63

Slika 37. Indeksiranje podataka

Ako želimo retke s točno određenom vrijednošću, primjenjujemo uvjetno odabiranje (*Boolean condition*). Ono nam vraća TRUE ili FALSE, ovisno o tome zadovoljava li uvjet ili ne. Želimo li filtrirati *data frame* da nema FALSE vrijednosti, onda uvjet postavljamo samom *data frame-u*. (slika 38.) [26]

```
uvjet = (podaci['visina_daka'] < 165.0)
```

```
uvjet
```

```
Lea      True
Tea      True
Rea      True
Mara    False
Sara    False
Vanja   False
Name: visina_daka, dtype: bool
```

```
podaci[podaci['visina_daka'] < 165.0]
```

	visina_daka	tezina_daka
Lea	158	50
Tea	160	54
Rea	163	68

Slika 38. *Boolean condition*

Za kompliciranije uvjete koristimo operatore i (**&**) i ili(|) ili ih pojednostavljujemo metodom **isin().** (slika 39.)

```
podaci[podaci['visina_daka'].isin([175,180])]
```

	visina_daka	tezina_daka
Mara	175	60
Sara	180	63
Vanja	180	64

Slika 39. isin() metoda - primjer

4.5 Primjenjivanje funkcija na skupovima podataka

Kada želimo nešto izmijeniti ili dodati novi stupac, čije vrijednosti dobivamo pomoću originalnog skupa podataka, optimalno je napisati funkciju koju ćemo kasnije primjeniti na *data frame* metodom **apply()**. (slika 40.)

```
def height_function(x):
    if x > 168.0:
        return "visoka"
    else:
        return "niska"

podaci["visok/nizak"] = podaci["visina_daka"].apply(height_function)

podaci
```

	visina_daka	tezina_daka	visok/nizak
Lea	158	50	niska
Tea	160	54	niska
Rea	163	68	niska
Mara	175	60	visoka
Sara	180	63	visoka
Vanja	180	64	visoka

Slika 40. Primjenjivanje funkcije na skup podataka

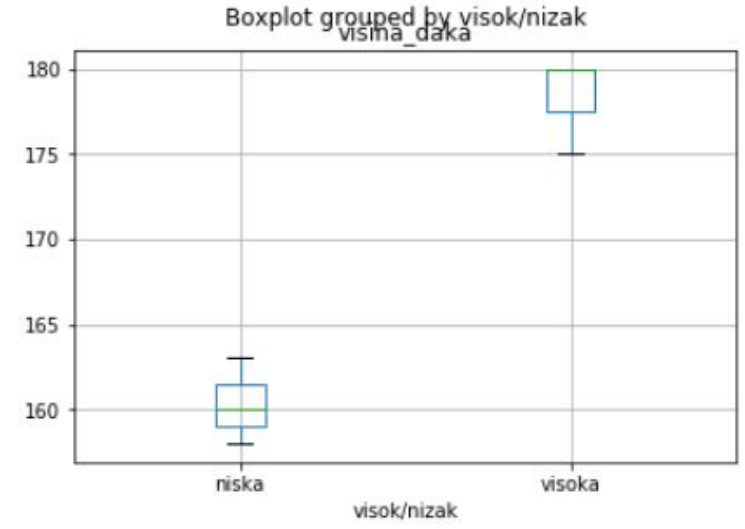
4.6 Grafovi

U kombinaciji s Matplotlib-om, Pandas može vizualizirati *data frame-ove* raznolikim grafovima. Raspršeni graf, histogram i stupčasti dijagram smo već prošli u poglavlju o Matplotlib-u.

Pravokutni dijagram (engl. *Boxplot*) zasniva se na skupu različitih brojeva koji opisuju skup podataka (minimum, prvi kvartil Q_1, medijan, treći kvartil Q_3 i maksimum). Govori nam koliko je skup podataka simetričan, kolika je gustoća distribucije i slično. [27]

Pravokutni dijagram možemo prikazati na dva načina. Prvi je da u metodi plot stavimo **kind="box"** ili da pozovemo metodu **boxplot(colum="*column_name*", by="*sort_by_column*").** (slika 41.)

```
podaci.boxplot(column="visina_daka", by="visok/nizak");
```



Slika 41. Pravokutni dijagram

4.7 Zanimljivosti

Kada nemamo već spremljenu datoteku s podacima i kada ne želimo gubiti vrijeme na spremanje datoteke, Pandas nudi mogućnost čitanje podataka iz *clipboarda*. Tako možemo kopirati tablicu iz Excel datoteke i pozvati metodu read_clipboard() i Pandas će je učitati kao i svaku drugu datoteku.

Nakon rada s podacima, datoteku možemo odmah spremiti u gzip, zip ili bz2 format dodavanjem parametra **compression = "gzip/zip/bz2"** u metodu **to_filetype(filename)**.

Za stiliziranje tablica možemo koristiti CSS svojstva. (slika 42.)

```
podaci.style.set_table_styles(
[{'selector': 'th:first-child',
  'props': [('background', 'blue')]},
 {'selector': 'tr:nth-of-type(odd)',
  'props': [('background', '#eee')]},
 {'selector': 'tr:nth-of-type(even)',
  'props': [('background', 'white')]},
 {'selector': 'th',
  'props': [('background', '#606060'),
            ('color', 'white'),
            ('font-family', 'verdana')]},
 {'selector': 'td',
  'props': [('font-family', 'verdana')]},
]
)
```

	visina_daka	tezina_daka	visok/nizak
Lea	158	50	niska
Tea	160	54	niska
Rea	163	68	niska
Mara	175	60	visoka
Sara	180	63	visoka
Vanja	180	64	visoka

Slika 42. Stiliziranje tablica u Pandas-u

5 ZAKLJUČAK

Kako bi stupili u svijet analiziranja podataka s Python bibliotekama, naročito s Pandas-om, prvo smo prošli kroz same osnove strojnog učenja i Python programskog jezika. Dotakli smo se disciplina bez kojih strojno učenje ne bi ni postojalo.

Ponovili smo osnovne statističke pojmove, kao što su mjere centralnih vrijednost, mjere raspršenja, uvjetna vjerojatnost te Bayesova formula. Proučili smo četiri tipa strojnog učenja te četiri vrste algoritama (regresijski algoritmi, stabla odlučivanja, Bayesovi, *deep learning*).

Nakon što smo naučili osnove korištenja Jupyter Notebook-a, prešli smo na osnove tri Python biblioteke koje će nam pomoći u razumijevanju samog Pandas-a, a to su NumPy, Matplotlib, Scikit-Learn. Proučili smo kako raditi s višedimenzionalnim vektorima, kako crtati različite vrste grafove te kako modelirati skupove podataka.

Nakon svih osnova, razradili smo Pandas biblioteku. Vidjeli smo da se jako rijetko koristi sama, tj. najčešće je u kombinaciji s jednom od tri, prethodno navedene, biblioteke. Pandas olakšava pripremu podataka s funkcionalnostima kao što su zamjena *null* vrijednosti s nekom od mjera centralnih vrijednosti. Na taj način ne ostajemo bez vrijednih podataka, koje bi inače izbrisali jer su nepotpuni.

Pokazali smo da Python, kao jednostavan, *open source* programski jezik, može pružiti raznolike mogućnosti za rad s podacima i da je s razlogom prestigao jezik R i slične njemu.

LITERATURA

[1] N. Blagojević, „Što je to AI (umjetna inteligencija) i trebamo li je se bojati? - Hrvatska - European Commission“, *Hrvatska - European Commission*, 17-sij-2019. [Online]. Dostupno na: https://ec.europa.eu/croatia/basic/what_is_artificial_intelligence_hr. [Pristupljeno: 05-ruj-2019]

[2] A. Zangre, „Discrete vs Continuous Data – What’s the Difference?“ [Online]. Dostupno na: https://learn.g2.com/discrete-vs-continuous-data. [Pristupljeno: 12-ruj-2019]

[3] Rapid Sigma Solutions LLP, „Online Sample Size Calculators - Users Beware“. [Online]. Dostupno na: https://www.sigmamagic.com/blogs/online-sample-size-calculators/. [Pristupljeno: 12-ruj-2019]

[4] „06_5h_prikaz_i_analiza_podataka_6_2.pdf“ [Online]. Dostupno na: https://www.e-sfera.hr/dodatni-digitalni-sadrzaji/7b47ae81-43da-4d96-a7ad-3cce66310eeb/assets/download/06_5h_prikaz_i_analiza_podataka_6_2.pdf

[5] A. Hayes, „Understanding Posterior Probability“, *Investopedia*, 24-tra-2019. [Online]. Dostupno na: https://www.investopedia.com/terms/p/posterior-probability.asp. [Pristupljeno: 07-ruj-2019]

[6] J. Weisberg, „6 Conditional Probability | Odds & Ends“. [Online]. Dostupno na: https://jonathanweisberg.org/vip/conditional-probability.html. [Pristupljeno: 12-ruj-2019]

[7] uniZg, „Osnove teorije vjerojatnosti“.

[8] Z. Rad, „Formula potpune vjerojatnosti i Bayesova formula“ [Online]. Dostupno na: https://repozitorij.mathos.hr/islandora/object/mathos:51/preview

[9] J. Brownlee, „A Tour of Machine Learning Algorithms“, *Machine Learning Mastery*, 11-kol-2019. [Online]. Dostupno na: https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/. [Pristupljeno: 07-ruj-2019]

[10] D. Korbut, „Machine Learning Algorithms: Which One to Choose for Your Problem“, *Medium*, 26-lis-2017. [Online]. Dostupno na: https://blog.statsbot.co/machine-learning-algorithms-183cc73197c. [Pristupljeno: 12-ruj-2019]

[11] „Unsupervised Learning“, *DeepAI*, 17-svi-2019. [Online]. Dostupno na: https://deepai.org/machine-learning-glossary-and-terms/unsupervised-learning. [Pristupljeno: 12-ruj-2019]

[12] R. Ring, „Deep Reinforcement Learning with TensorFlow 2.0 | Roman Ring“, *Roman Ring*, 20-sij-2019. [Online]. Dostupno na: http://inoryy.com/post/tensorflow2-deep-reinforcement-learning/. [Pristupljeno: 12-ruj-2019]

[13] V. Maheshwari, „LOGISTIC REGRESSION“, *Medium*, 21-pros-2018. [Online]. Dostupno na: https://medium.com/datadriveninvestor/logistic-regression-18afd48779ce. [Pristupljeno: 12-ruj-2019]

[14] P. Ditthakit i C. Chinnarasri, „Fig. 1: Example of M5 model tree algorithm with 6 linear regression models“, *ResearchGate*, 01-sij-2012. [Online]. Dostupno na: https://www.researchgate.net/figure/Example-of-M5-model-tree-algorithm-with-6-linear-regression-models_fig2_289078955. [Pristupljeno: 12-ruj-2019]

[15] Global Software Support, „Naive Bayes Classifier Explained Step by Step“, *Global Software Support*. [Online]. Dostupno na: https://www.globalsoftwaresupport.com/naive-bayes-classifier-explained-step-step/. [Pristupljeno: 12-ruj-2019]

[16] R. Pieters, „Python for Image Understanding: Deep Learning with Convolutional Neur…“,

21-lip-2015. [Online]. Dostupno na: https://www.slideshare.net/roelofp/python-for-image-understanding-deep-learning-with-convolutional-neural-nets. [Pristupljeno: 12-ruj-2019]
[17] „Machine Learning with Python - Ecosystem - Tutorialspoint“. [Online]. Dostupno na: https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_ecosystem.htm. [Pristupljeno: 07-ruj-2019]
[18] R. Python, „Jupyter Notebook: An Introduction – Real Python“, *Real Python*, 28-sij-2019. [Online]. Dostupno na: https://realpython.com/jupyter-notebook-introduction/. [Pristupljeno: 07-ruj-2019]
[19] K. Jain, „A Complete Tutorial to Learn Python for Data Science from Scratch“, *Analytics Vidhya*, 14-sij-2016. [Online]. Dostupno na: https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/. [Pristupljeno: 07-ruj-2019]
[20] „NumPy - Introduction - Tutorialspoint“. [Online]. Dostupno na: https://www.tutorialspoint.com/numpy/numpy_introduction.htm. [Pristupljeno: 08-ruj-2019]
[21] „NumPy - Statistical Functions - Tutorialspoint“. [Online]. Dostupno na: https://www.tutorialspoint.com/numpy/numpy_statistical_functions.htm. [Pristupljeno: 08-ruj-2019]
[22] K. Govani, „Matplotlib Tutorial: Learn basics of Python’s powerful Plotting library“, *Medium*, 03-velj-2019. [Online]. Dostupno na: https://towardsdatascience.com/matplotlib-tutorial-learn-basics-of-pythons-powerful-plotting-library-b5d1b8f67596. [Pristupljeno: 09-ruj-2019]
[23] N. S. Chauhan, „A beginner’s guide to Linear Regression in Python with Scikit-Learn“, *Medium*, 25-velj-2019. [Online]. Dostupno na: https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-with-scikit-learn-83a8f7ae2b4f. [Pristupljeno: 10-ruj-2019]
[24] S. Srinidhi, „Linear Regression in Python using SciKit Learn“, *Medium*, 30-srp-2018. [Online]. Dostupno na: https://medium.com/@contactsunny/linear-regression-in-python-using-scikit-learn-f0f7b125a204. [Pristupljeno: 10-ruj-2019]
[25] A. Bronshtein, „A Quick Introduction to the “Pandas” Python Library“, *Medium*, 18-tra-2017. [Online]. Dostupno na: https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673. [Pristupljeno: 10-ruj-2019]
[26] „Python Pandas Tutorial: A Complete Introduction for Beginners“. [Online]. Dostupno na: https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/. [Pristupljeno: 10-ruj-2019]
[27] M. Galarnyk, „Understanding Boxplots“, *Medium*, 12-ruj-2018. [Online]. Dostupno na: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51. [Pristupljeno: 10-ruj-2019]

POPIS SLIKA

Slika 1. Razlika diskretnih i kontinuiranih varijabli

Slika 2. Uzimanje uzorka iz populacije

Slika 3. Uvjetna vjerojatnost

Slika 4. Klasifikacija i regresija

Slika 5. Nenadzirano učenje

Slika 6. Polu-nadzirano strojno učenje

Slika 7. Podržano strojno učenje

Slika 8. Linearna i logistička regresija

Slika 9. MP5 stablo odlučivanja

Slika 10. *Deep learning* algoritam - slojevi

Slika 11. Stvaranje novog *notebooka*

Slika 12. Ćelije u Jupyter-u

Slika 13. Primjeri rada s listama

Slika 14. Primjer stringova

Slika 15. Primjer tupl-a

Slika 16. Rad s rječnicima

Slika 17. Primjer FOR petlje s listom i opsegom kao iteracijom

Slika 18. Uključivanje biblioteke u kod

Slika 19. ndarray parametri

Slika 20. Numerički tipovi podataka u NumPy-u

Slika 21. Primjeri statističkih operacija

Slika 22. Crtanje grafa s NumPy-om i Matplotlib-om

Slika 23. Crtanje stupčastih dijagrama

Slika 24. Histogram pomoću numPy-a i matplotlib-a

Slika 25. Crtanje pod-grafova

Slika 26. Pita dijagram (Pie chart)

Slika 27. 3D Graf

Slika 28. Digits dataset

Slika 29. Primjer vizualizacije podataka kombinacijom matplotlib-a i sklearn-a

Slika 30. Primjeri head() i describe() metoda na skupu podataka

Slika 31. Priprema za model linearne regresije

Slika 32. Usporedba podataka za učenje i testiranje

Slika 33. Stvaranje data frame-a

Slika 34. Informacije o data frame-u

Slika 35. Preimenovanje stupaca

Slika 36. Koeficijent korelacije - primjer

Slika 37. Indeksiranje podataka

Slika 38. Boolean condition

Slika 39. isin() metoda - primjer

Slika 40. Primjenjivanje funkcije na skup podataka

Slika 41. Pravokutni dijagram

Slika 42. Stiliziranje tablica u Pandas-u

IZJAVA

Izjavljujem pod punom moralnom odgovornošću da sam završni rad izradila samostalno, isključivo znanjem stečenim na studijima Sveučilišta u Dubrovniku, služeći se navedenim izvorima podataka i uz stručno vodstvo mentora doc.dr.sc Maria Miličevića, kome se još jednom srdačno zahvaljujem.

Nikolina Šanovsky