



Univerzitet u Beogradu – Elektrotehnički fakultet

Softversko inženjerstvo velikih baza podataka

Domaći zadatak

Student

Anica Čančar, 2020/3125

Beograd, *Januar* 2021. godine

SADRŽAJ

1 UVOD	3
2 OPIS PROBLEMA.....	3
3 OBRADA PODATAKA	5
4 LOGISTIČKA REGRESIJA	6
5 NAIVNI BAYESOV KLASIFIKATOR	10
6 K NAJBЛИŽIH SUSJEDA (KNN)	13
7 NEURALNE MREŽE	16
8 ZAKLJUČAK	23

1 UVOD

Ideja ovog projekta je isprobavanje više različitih algoritama mašinskog učenja na istom setu podataka. *Dataset* koji se koristi je preuzet sa *Kaggle* – a, a naziva se Mobile Price Classification.

U ovom projektu ćemo obraditi sledeće algoritame:

Logističku regresiju, Naivni Bayes – ov klasifikator, K najbližih susjeda i neuralnu mrežu.

Na početku će biti riječi o datasetu, a zatim ćemo redom proći kroz algoritme i vidjeti kakve oni daju rezultate za naš problem.

2 OPIS PROBLEMA

Dataset korišten u ovom projektu je Mobile Price Classification, odnosno predikcija cijene telefona. Dataset je preuzet sa sajta Kaggle:

<https://www.kaggle.com/iabhishekofficial/mobile-price-classification>

Kao što iz naziva zaključujemo, ovaj dataset predstavlja problem klasifikacije. Problem se zasniva na tome da na osnovu nekih karakteristika telefona, kao što su jačina baterije, količina memorije i drugih klasifikujemo telefon u jednu od 4 klase, koje označavaju opseg cijena.

U datasetu imamo sledeća obilježja:

1. battery - power: Ukupna energija koju baterija može da skladišti u mAh
2. blue: Da li telefon ima bluetooth ili ne
3. clock_speed: Brzina kojom mikroprocesor izvršava operacije
4. dual_sim: Da li ima mogućnost dvije kartice ili ne
5. fc: Prednja kamera u megapixelima
6. four_g: Da li ima 4G ili ne
7. int_memory: Interna memorija u gigabajtima
8. m_dep: Dubina mobilnog telefona
9. mobile_wt: Težina mobilnog telefona
10. n_cores: Broj jezgara procesora
11. pc: Glavna kamera u megapixelima

12. px_height: Rezolucija piksela visina
13. px_width: Rezolucija piksela širina
14. ram: RAM memorija u megabajtima
15. sc_h: Visina ekrana u cm
16. sc_w: Širina ekrana u cm
17. talk_time: Najduže vrijeme koje će trajati napunjena baterija
18. three_g: Da li ima 3G ili ne
19. touch_screen: Da li je na dodir ili ne
20. wifi: Da li ima wifi ili ne

Dakle, imamo 20 obilježja za svaki telefon.

U datasetu poslednja kolona predstavlja price_range, tj. opseg cijena telefona i može da ima 4 vrijednosti: {0, 1, 2, 3} u zavisnosti od cijene telefona. U našem problemu ovo će da bude izlaz koji pokušavamo da predvidimo.

Sam dataset se sastoji od dva fajla: *train.csv* i *test.csv*. Za obradu podataka koristimo dataset *train.csv* koji ima 2000 redova i 21 kolona. Problem se svodi na projektovanje algoritama koji će moći da izvrši klasifikaciju telefona na osnovu datih obilježja u jednu od 4 klase koji odgovaraju određenim opsezima cijena.

OCIJENIVANJE PERFORMANSI MODELA

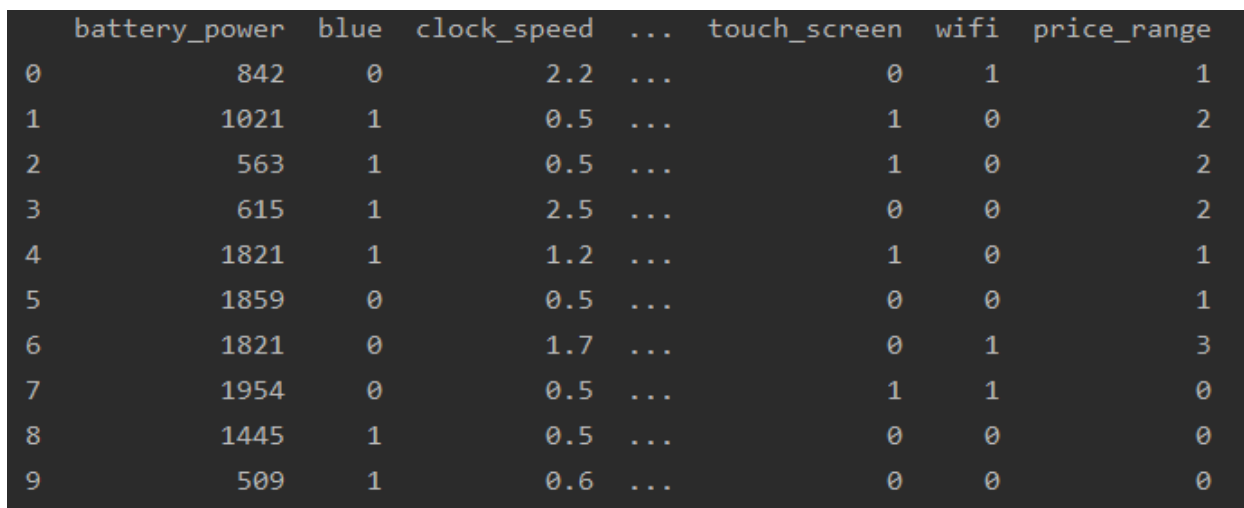
- **Tačnost (Accuracy)** – predstavlja količnik tačno klasifikovanih kroz ukupan broj posmatranih opservacija. Dobra je mjera performansi samo onda kada su podaci balansirani.
- **Preciznost (Precision)** – predstavlja količnik tačan klasifikovanih, kroz ukupan broj opservacija iz te klase.
- **Osjetljivost (Sensitivity)** – predstavlja količnik tačno klasifikovanih kroz ukupan broj opservacija koji su klasifikovani u tu klasu.
- **F1 score** – predstavlja prosjek preciznosti i osjetljivosti. Ovo je odlična mjera performansi ako su podaci nebalansirani. Računa se po sledećoj formuli:

$$F1\ Score = \frac{2(Sensitivity * Precision)}{Sensitivity + Precision} \quad (1)$$

3 OBRADA PODATAKA

Da bi mogli da uporedimo performanse naših algoritama, moramo da imamo isti set podataka za trenirajući i testirajući skup. Zbog toga prije bilo kog algoritma neophodno je da uradimo učitavanje podataka, njihovu obradu i podjelu na trenirajući i testirajući skup i nakon toga da ih sačuvamo u poseban fajl.

Prvi korak je da pomoću biblioteke *pandas* učitamo podatke sa metodom *read_csv*. Nakon toga u promjenjivu *X* smještamo obilježja, a u *y* odgovarajuće izlaze. S obzirom da su vrijednosti i skale naših podataka u *X* dosta različite (npr. Jačina baterije ima vrijednosti oko 1000, a bluetooth 0 ili 1) kao što možemo vidjeti na slici ispod:



	battery_power	blue	clock_speed	...	touch_screen	wifi	price_range
0	842	0	2.2	...	0	1	1
1	1021	1	0.5	...	1	0	2
2	563	1	0.5	...	1	0	2
3	615	1	2.5	...	0	0	2
4	1821	1	1.2	...	1	0	1
5	1859	0	0.5	...	0	0	1
6	1821	0	1.7	...	0	1	3
7	1954	0	0.5	...	1	1	0
8	1445	1	0.5	...	0	0	0
9	509	1	0.6	...	0	0	0

Slika 1. Prikaz dataseta

Dolazimo do toga da neka obilježja su jako dominantna u odnosu na ostala, pa je neophodno da ih nekako skaliramo. U tu svrhu koristimo standardizaciju, koja se radi po formuli:

$$X' = \frac{x - \mu}{\sigma} \quad (2)$$

Na ovaj način obezbjeđujemo da sva mjerenja imaju srednju vrijednost nula i jediničnu varijansu. Ovo je jako bitan korak kako neka obilježja ne bi bila dominantna u odnosu na ostala. U programu ovo radimo pomoću *StandardScaler()*.

Nakon toga, pomoću metode *train_test_split()* podatke dijelimo na trenirajući i testirajući set i to u opsegu 90% trenirajućih i 10% testirajućih podataka. Čuvamo ih pomoću alata *hickle* u novi fajl *data.hkl.2*

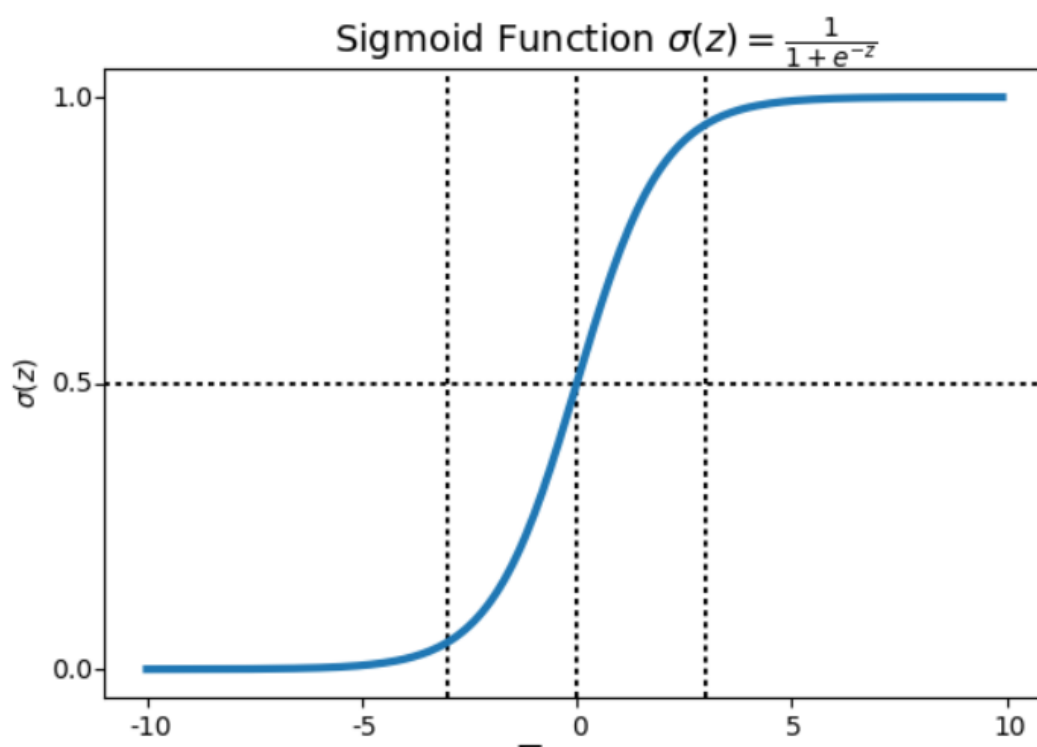
4 LOGISTIČKA REGRESIJA

Prvi algoritam koji ćemo koristiti se naziva logistička regresija. To je algoritam klasifikacije koji se koristi da bi observacije smjestio u diskretan skup klasa. Neki primjeri ovakvih problema su da li je neka e – mail poruka spam ili ne, da li je neki tumor maligni ili benigni itd. Logistička regresija transformiše svoj izlaz koristeći logističku sigmoidnu funkciju, kako bi vratila vjerovatnoću.

Postoje dva tipa logističke regresije:

1. Binarna (npr. Da li je tumor benigni ili maligni)
2. Multiklasne (npr. Da li je životinja mačka, pas, ovca itd.)

Možemo da kažemo da je Logistička regresija, model linearne regresija, ali kao funkciju troška koristi Sigmoidnu funkciju. Ta funkcija mapira predviđene vrijednosti u vjerovatnoće.



Slika 2. Sigmoid funkcija

Logistička regresija za binarnu klasifikaciju radi tako što definišemo granicu vjerovatnoće i u zavisnosti da li dobijemo vrijednost koja je veća ili manja od te granice mi vršimo klasifikaciju u određenu klasu.

Multiklasna klasifikacija, koja se drugačije naziva „Jedan protiv ostalih“ (*One-vs-All*) se koristi u ovom zadatku. Ona cijeli problem multiklasne klasifikacije dijeli na više binarnih problema. Postupak se sastoji da uzmemo jednu klasu, i nju smatramo npr. 1, dok pripadnike svih ostalih klasa smatramo 0. Na ovaj način pravimo „lažni“ obučavajući skup i projektujemo klasifikator:

$$h_{\theta}^{(1)}(x) \quad (3)$$

Nastavljamo za sve ostale klase na sličan način i dobijamo set klasifikatora:

$$h_{\theta}^{(i)}(x), i = 1, \dots, N \quad (4)$$

Gdje je N, broj klasa

Sada, kada želimo da neki podatak x klasifikujemo, gledamo koji od projektovanih klasifikatora će imati najveću vrijednost i u tu klasu ga smiještamo

POSTUPAK PROJEKTOVANJA

Prvo pomoću *hkl.load* učitamo obrađene podatke i iz njega izvlačimo podatke za treniranje i testiranje.

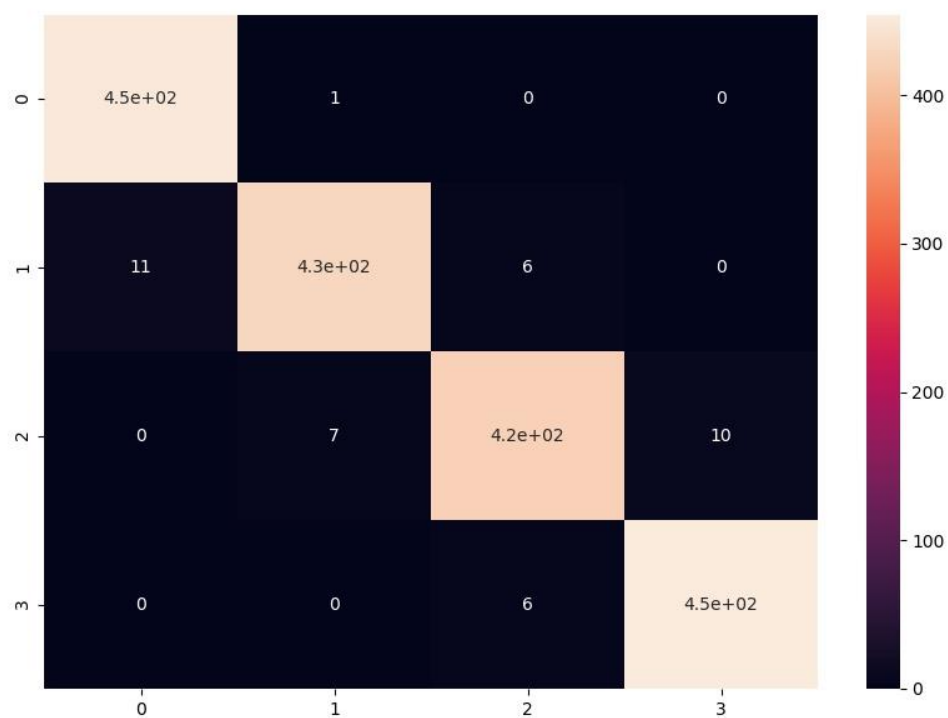
Pravimo logistički klasifikator pomoću `LogisticRegression()` i vršimo treniranje, kako bismo dobili rezultate.

REZULTATI

Trenirajući skup:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	453
1	0.98	0.96	0.97	447
2	0.97	0.96	0.97	440
3	0.98	0.99	0.98	460
accuracy				0.98 1800
macro avg				0.98 0.98 0.98 1800
weighted avg				0.98 0.98 0.98 1800

Konfuzionna matrica za trenirajući skup:



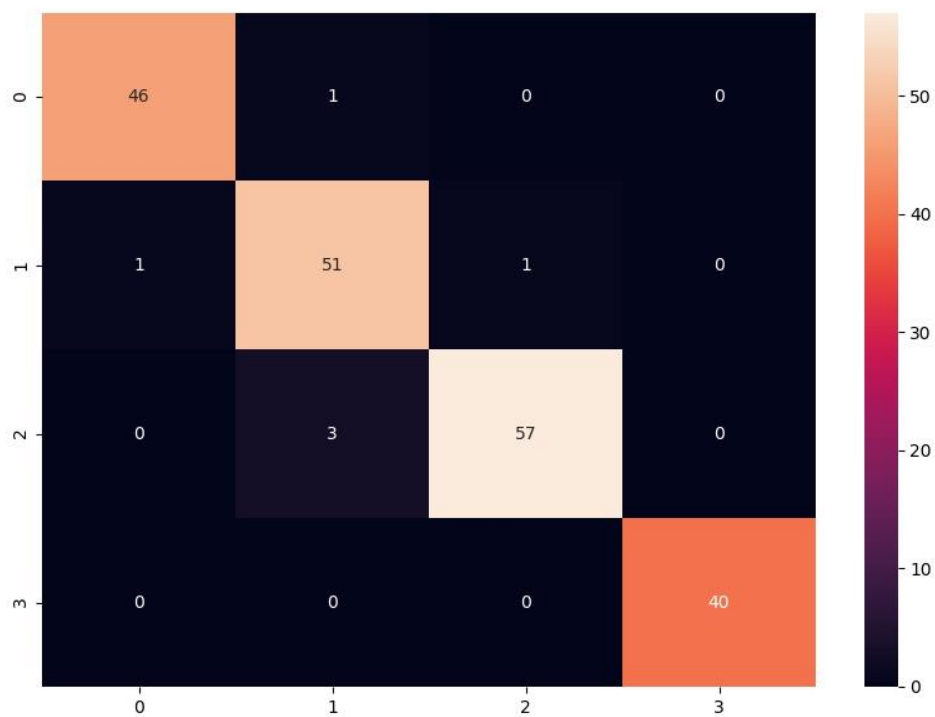
Slika 3. Konfuzionna matrica za trenirajući skup – logistička regresija

Testirajući skup:

0	0.98	0.98	0.98	47
1	0.93	0.96	0.94	53
2	0.98	0.95	0.97	60
3	1.00	1.00	1.00	40

accuracy		0.97		200
macro avg	0.97	0.97	0.97	200
weighted avg	0.97	0.97	0.97	200

Konfuzionna matrica za testirajući skup:



Slika 4. Konfuzionna matrica za testirajući skup – Logistička regresija

Dakle, uz obradu podataka za logističku regresiju dobijamo tačnost 97%.

5 NAIVNI BAYESOV KLASIFIKATOR

To je klasifikaciona tehnika zasnovana na Bayesovoj teoremi sa pretpostavkom nezavisnosti među podacima. Na najjednostavniji način to znači da NBK pretpostavlja da prisustvo određenog obilježja nije povezano sa prisustvom bilo kog drugog obilježja i zbog toga se naziva naivni.

Bayesova teorema je način računanja posteriornih vjerovatnoća $P(c|x)$ na osnovu $P(c), P(x) P(x|c)$

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \quad (6)$$

Za probleme klasifikacije možemo ovu teoremu napisati u sledećoj formi:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \quad (7)$$

Promjenjiva y je klasna promjenjiva, a X su obilježja:

$$X = (x_1, x_2, x_3, \dots, x_n) \quad (8)$$

Zbog pretpostavke o nezavisnosti ovo se može pisati:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)} \quad (9)$$

Sada konačnu klasifikaciju vršimo prema formuli

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y) \quad (10)$$

Odnosno, biramo onu klasu za koju je izraz maksimalan.

POSTUPAK

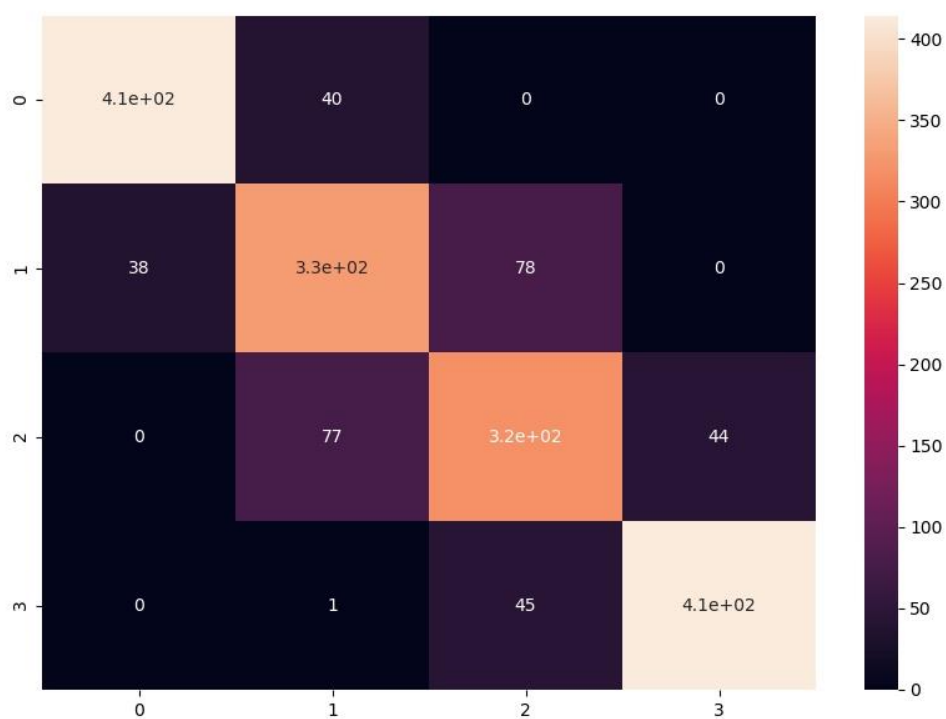
Kod ovog algoritma preprocesiranje i podijelu podataka radimo na isti način kao i kod prethodnog algoritma.

Koristimo *NBGaussian()* klasu kako bismo napravili naivni Bayesov klasifikator. Odlučili smo se za ovaj tip klasifikatora jer su naši podaci nakon obrade kontinualni.

REZULTATI

Trenirajući skup:

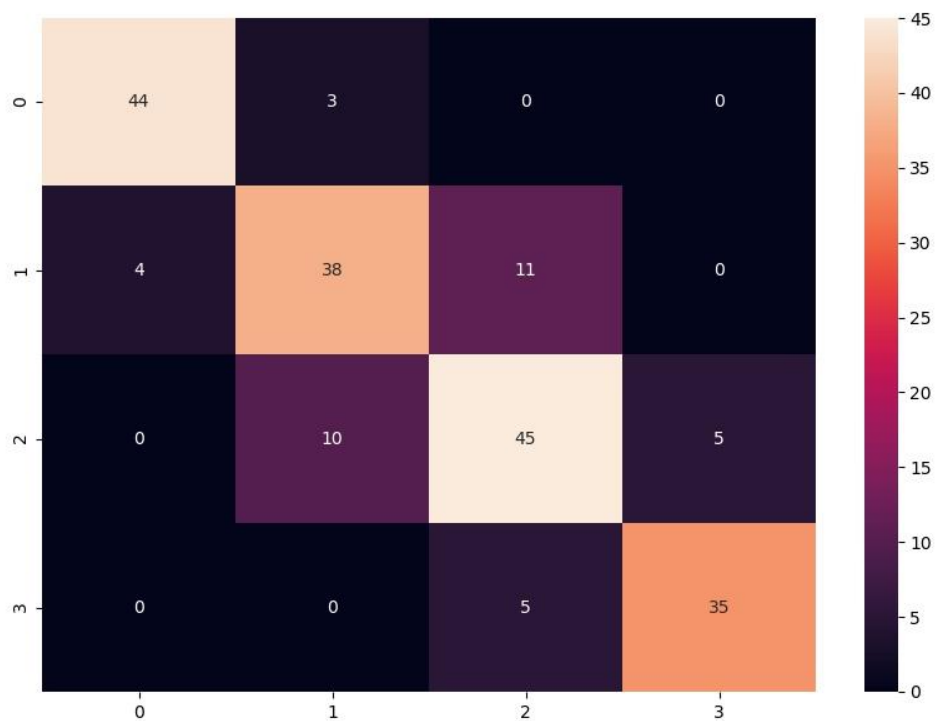
	precision	recall	f1-score	support
0	0.92	0.91	0.91	453
1	0.74	0.74	0.74	447
2	0.72	0.72	0.72	440
3	0.90	0.90	0.90	460
accuracy		0.82		1800
macro avg	0.82	0.82	0.82	1800
weighted avg	0.82	0.82	0.82	1800



Slika 5. Konfuziona matrica za testirajući skup – Naivni Bayesov klasifikator

Testirajući skup:

	precision	recall	f1-score	support
0	0.92	0.94	0.93	47
1	0.75	0.72	0.73	53
2	0.74	0.75	0.74	60
3	0.88	0.88	0.88	40
accuracy		0.81		200
macro avg	0.82	0.82	0.82	200
weighted avg	0.81	0.81	0.81	200



Slika 6. Konfuziona matrica za testirajući skup – Naivni Bayesov klasifikator

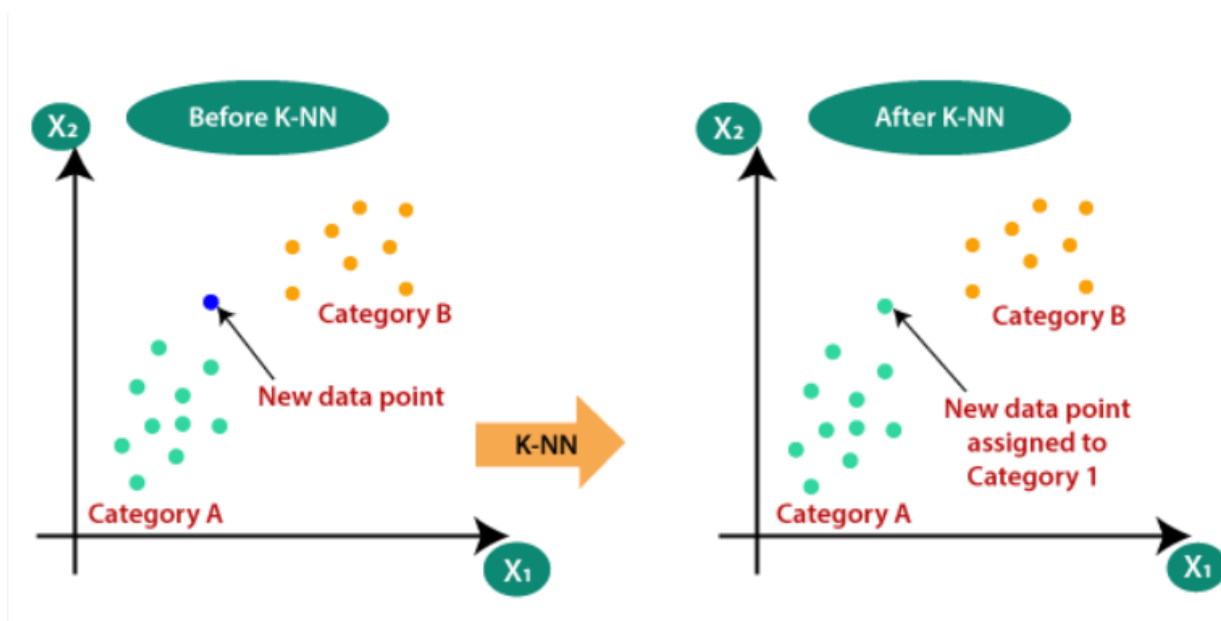
Dakle, Naivni Bayesov klasifikator ima tačnost 81%.

6 K NAJBЛИŽIH SUSJEDA (KNN)

KNN je jedan od najjednostavnijih algoritama mašinskog učenja. On proračunava sličnost između novog podatka i datih klasa i stavlja podatak u onu klasu kojoj je najbližnja. Koristi se i za regresiju i za klasifikaciju.

Koraci KNN algoritma:

1. Izaberemo broj susjeda(k)
2. Izračunamo Euklidsko rastojanje za k susjeda
3. Za ovih k najbližih susjeda izračunamo broj susjeda iz svake kategorije
4. Smjestimo naš podatak u onu klasu iz koje ima najveći broj susjeda.
5. Naš model je spreman.



Slika 7. KNN algoritam

POSTUPAK

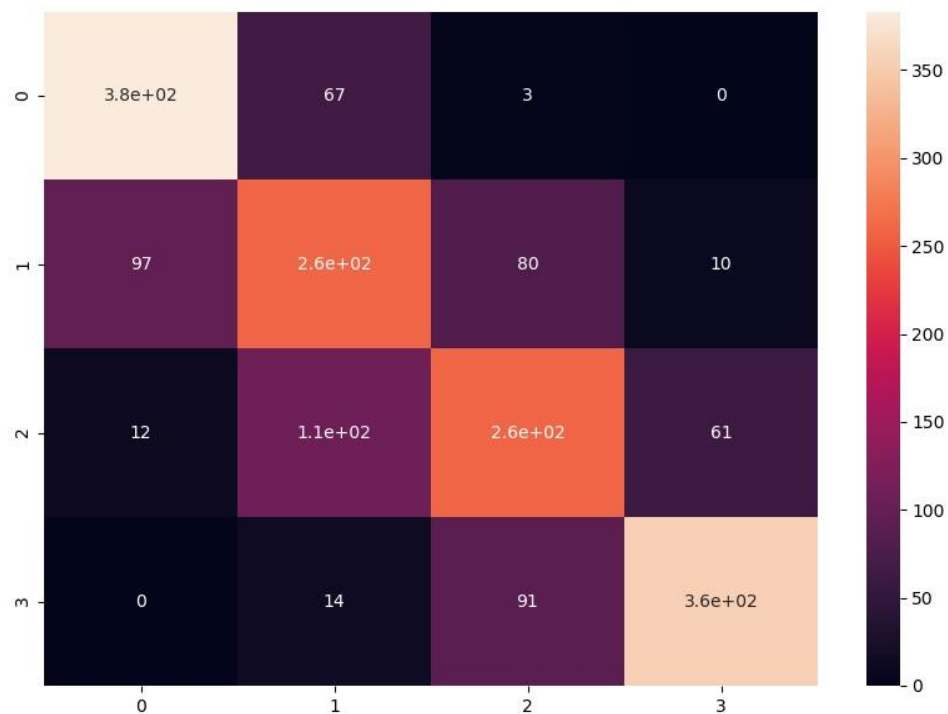
Da bi odredili optimalan broj susjeda k , provjeravamo sve brojeve od 1 do 30, i gledamo tačnost modela, te za optimalan broj k uzimamo onaj za koji je tačnost najveća.

REZULTATI

Najbolji model je za $k = 27$, sa tačnoscu 0.62

Trenirajući skup:

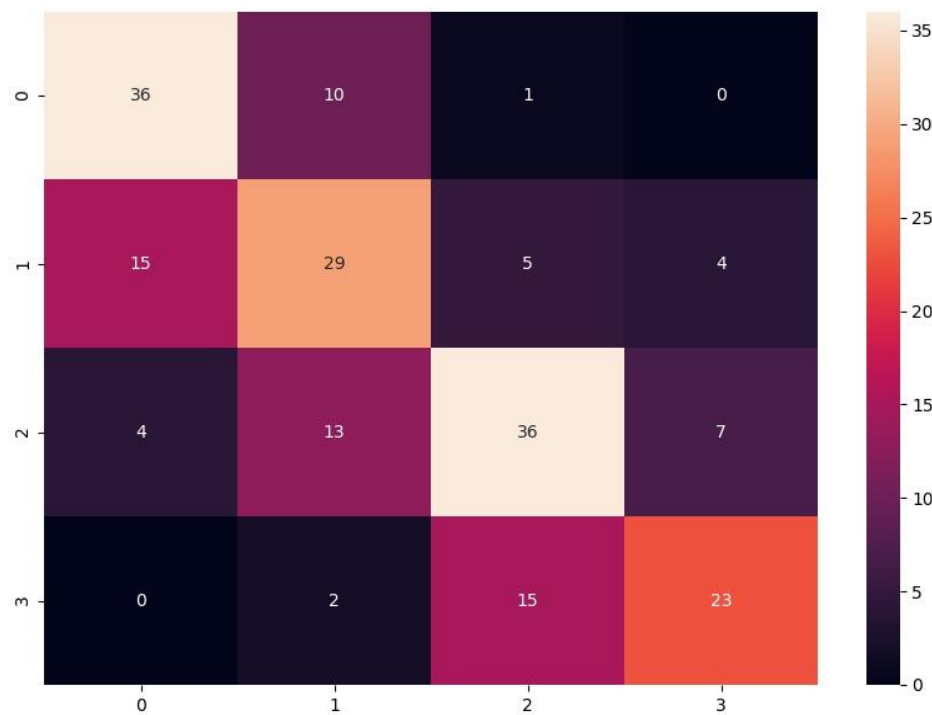
	precision	recall	f1-score	support
0	0.78	0.85	0.81	453
1	0.58	0.58	0.58	447
2	0.60	0.59	0.59	440
3	0.83	0.77	0.80	460
accuracy		0.70		1800
macro avg	0.70	0.70	0.70	1800
weighted avg	0.70	0.70	0.70	1800



Slika 8. Konfuziona matrica za trenirajući skup – KNN algoritam

Testirajući skup:

	precision	recall	f1-score	support
0	0.65	0.77	0.71	47
1	0.54	0.55	0.54	53
2	0.63	0.60	0.62	60
3	0.68	0.57	0.62	40
accuracy				0.62 200
macro avg				0.62 0.62 0.62 200
weighted avg				0.62 0.62 0.62 200



Slika 9. Konfuzionna matrica za testirajući skup – KNN algoritam

Kao što vidimo iz priloženih rezultata, tačnost koju dobijamo ovim algoritmom je 62%, za $k = 27$. Treba napomenuti da bi ovaj algoritam davao još bolje rezultate kad bi imali veći trenirajući skup.

7 NEURALNE MREŽE

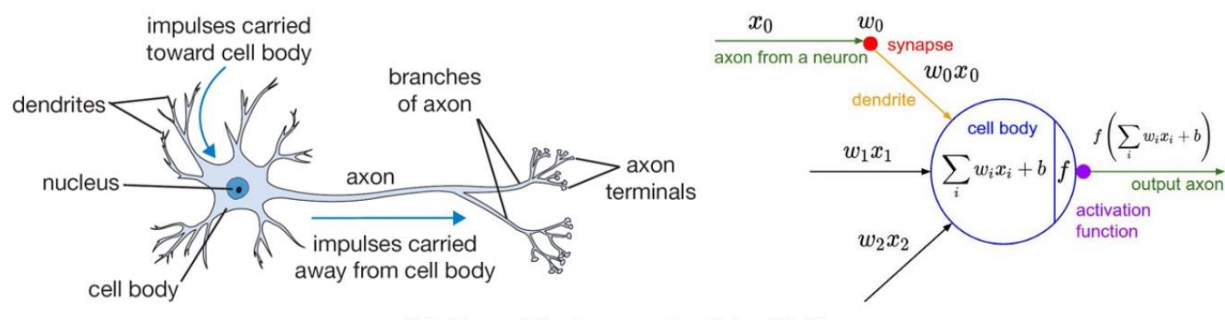
Neuralne mreže su jedna od najpopularnijih tema u datascience - u danas. One trenutno predstavljaju jedno od najboljih rješenja za prepoznavanje slika, govora i jezika.

Neuralna mreža je računarski sistem koji se sastoji od jednostavnih, međusobno povezanih procesnih elemenata, koji obrađuju informacije svojim odzivom stanja na spoljne ulaze.

Neuralna mreža je inspirisana biološkom neuralnom mrežom u ljudskom mozgu.

7.1 BIOLOŠKA MOTIVACIJA I KONEKCIJE

Osnovna računaska jedinica mozga je neuron. Oko 86 biliona neurona se može naći u ljudskom nervnom sistemu i povezani su sa otprilike $10^{14} - 10^{15}$ sinapsi. Dijagram ispod prikazuje crtež biološkog neurona(lijevo) i uobičajni matematički model(desno).



Slika 10. Biološki neuron

Osnovna računaska jedinica u neuralnoj mreži je neuron, koji se obično zove čvor ili jedinica. On prima ulaze nekih drugih čvorova ili nekog drugog spoljnog izvora i računa izlaz. Svaki ulaz ima određeno pojačanje (w), koje se dodjeljuje u odnosu na njegovu važnost u odnosu na ostale ulaze. Čvor primjenjuje funkciju sume svojih ulaza.

U osnovnom modelu, dendriti nose signal do tijela ćelije, gdje se oni sabiraju. Ako je konačna suma veća od određenog praga, neuron šalje impuls kroz akson. U računskom modelu mi smatramo da precizno određivanje vremena impusla nije važno, već samo frekvencija slanja impulsa. Modelirao brzinu slanja impulsa neurona sa određenom aktivacionom funkcijom koja predstavlja učestalost impulsa duž aksona.

7.2 ARHIKTETURA NEURALNE MREŽE

Iz prethodne priče mogli smo zaključiti da je neuralna mreža napravljena od neurona, biološki su neuroni povezani kroz sinapse kroz koje informacija prolazi. Kada treniramo neuralnu mrežu želimo da daju impuls svaki put kada nauče nešto iz podataka i modeliramo to kroz aktivacionu funkciju.

Komponente neuralne mreže:

- **Ulazni čvorovi (ulazni sloj):** Nemamo nikakvo računanje ovdje, oni samo prosljeđuju informaciju sledećem sloju.
- **Skriveni čvorovi (skriveni sloj):** Oni rade srednju obradu i proračune, a zatim težine prenose sa ulaznog sloja na sledeći sloj (skriveni ili izlazni).
- **Izlazni čvorovi (izlazni sloj):** Ovdje konačno koristimo funkciju aktivacije koja se preslikava u željeni izlazni format (npr. Softmax za klasifikaciju)
- **Konekcije i pojačanja:** Mreža se sastoji od veza, a svaka veza prenosi izlaz neurona i ka ulazu neurona j . U tom smislu i je prethodnik od j , a j je naslijednik i . Svaka veza ima težinu w_{ij} .
- **Funkcija aktivacije:** Ona definiše izlaz tog čvora kome su dati ulazi ili setovi ulaza. Standardno kolo računarskog čipa može se posmatrati kao digitalna mreža funkcije aktiviranja koja može biti „ON“ (1) ili „OFF“ (0), u zavisnosti od ulaza. Ovo ponašanje je slično ponašanju linearnog perceptrona u neuralnim mrežama. Međutim, nelinearne funkcije aktivacije omogućavaju takvim mrežama da izračunaju netrivialne probleme koristeći mali broj čvorova.
- **Pravilo učenja:** To je algoritam koji modifikuje parametre neuralne mreže kako bi dati ulazi imali favorizovani izlaz. Ovaj proces učenja obično predstavlja modifikovanje težine i pragova.

Tipovi neuralnih mreža:

- **Feedforward neuralna mreža**
 - Jednoslojni perceptron
 - Višeslojni perceptron
 - Konvoluciona neuralna mreža
- **Rekurentna neuralna mreža**

Tipovi aktivacionih funkcija:

- Sigmoid
- Tanh
- ReLU

7.3 NEURALNA MREŽA – PRIMJENA NA DATASET – U

Na početku analize problema klasifikacije za naš dataset, zaključujemo da u svakom slučaju naša mreža mora imati 20 ulaza i 4 izlaza. Koristićemo *Keras* framework za izradu neuralne mreže koji se zasniva na *tensorflow* – u. Takođe, kako bismo pronašli najbolje hiperparametre koje naša neuralna mreža treba da ima koristimo *Kerastuner*.

Postupak rada:

- Pretprocesiranje i učitavanje podataka
- Definisanje modela
- Definisanje funkcije i optimizatora
- Treniranje modela
- Testiranje modela

7.3.1 PRETPROCESIRANJE PODATAKA

Nakon učitavanja podataka jako bitno da uradimo kodiranje da bismo naše klase pretvorili u binarne vrijednosti. Ovaj korak je bitan da bi naša mreža mogla da radi sa kategoričkim podacima. To radimo sa *OneHotEncoder()*. Nakon ovog koraka dobijamo niz dimenzija (2000, 4) i izgleda kao na slici:

```
[[0.  1.  0.  0.]  
 [0.  0.  1.  0.]  
 [0.  0.  1.  0.]  
 ...  
 [0.  0.  0.  1.]  
 [1.  0.  0.  0.]  
 [0.  0.  0.  1.]
```

Slika 11. Enkodirane vrijednosti y

Sa ovim smo završili pretprocesiranje podataka.

7.3.2 PROJEKTOVANJE NEURALNE MREŽE

Da bismo pronašli optimalne parametre za našu neuralnu mrežu koristimo metodu pronlaska optimlnih hiperparametara. To radimo pomoću alata *Kerastuner*. Ova metoda se sastoji na isprobavanju različitih struktura, aktivacionih funkcija, koeficijenata regularizacije kao i same strukture mreže u cilju dobijanja najboljih rezultata. S obzirom da su podaci balansirani, ovu metodu radimo prema tačnosti (*accuracy*). U slučaju nebalansiranih podataka kao mjeru performansi gledamo *f1 score*.

Dio koda kojim se vrši podešavanje parametara je prikazan u nastavku:

```
def build_model(hp):
    model = keras.models.Sequential()

    model.add(Dense(hp.Int("input units", min_value=8, max_value=32, step=2), input_dim=20,
                        activation=hp.Choice('dense_activation', values=['relu', 'tanh', 'sigmoid'], default='relu'))
    for i in range(hp.Int("n_layers", 1, 4)):
        model.add(Dense(hp.Int("input units1", min_value=8, max_value=32, step=2),
                        activation=hp.Choice('dense_activation1', values=['relu', 'tanh', 'sigmoid'], default='relu'))
    model.add(
        Dropout(
            rate=hp.Float(
                'dropout_3',
                min_value=0.0,
                max_value=0.5,
                default=0.25,
                step=0.05
            )
        )
    )
    model.add(Dense(4, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

Prvo definišemo funkciju *build_model(hp)* u kojoj definišemo koje parametre želimo da ispitamo za našu mrežu. Kao model neuralne mreže koristimo *Sequential()*. To označava da je naš model sekvencijalan i da svaki izlaz sloja koji definišemo, ulaz u sledeći sloj.

Nakon toga dodajemo sloj sa metodom *model.add()*, koji kao parametar uzima tip sloja koji želimo. Mi koristimo *Dense()* koji označava da je u pitanju potpuno konektovan sloj. U prvom sloju imamo 20 ulaza koja su naša obilježja, a hoćemo da ispitamo koji sloj je najbolji kao izlazni

koji kreće od 8 do 32, sa korakom 2. Takođe u tom sloju ispitujemo najbolju aktivacionu funkciju između *{relu, tanh, sigmoid}*.

Nakon toga hoćemo da ispitamo koliko je najbolje slojeva da imamo u opsegu od 1 – 4. Ovaj broj sam izabrala, jer imamo poprilično mali skup podataka, pa nema potrebe za ogromnom arhitekturom mreže. Slično kao u prethodnom koraku provjeravamo broj izlaznih čvorova i aktivacione funkcije. Kao dodatak je da testiramo i koeficijent regularizacije (*Dropout rate*). To je način kojim se sprječava preobučavanje mreže. Ovo radi na principu što se eliminiše određeni čvor u svakoj epohi, a to se radi sa vjerovatnoćom koja odgovara ovom koeficijentu.

Izlazni sloj ima 4 izlaza i kao funkciju aktivacije koristi *softmax* koja je dobar izbor iz razloga što se radi o multiklasnom problemu. Ova funkcija vraća vjerovatnoće pripadnosti za svaku od klasa što pokazuje zbog čega je najčeće korištena kod višeklasnih problema.

Na kraju definišemo da kao funkciju troška koristimo *categorical_crossentropy*. Ovu funkciju koristimo jer je jako pogodna za klasifikaciju, a pošto imamo više klasa onda koristimo kategoričku. Kao optimizatora koristimo *adam*, a s obzirom da imamo balansirane podatke koristimo tačnost kao mjeru performansi.

Nakon toga pokrećemo pretragu i tražimo najbolje hiperparametre. Koristimo tip *RandomSearch*. Za validaciju podataka koristimo testirajuće podatke. Validacija nam pomaže da vidimo da li je došlo do preobučavanja, odnosno da li za trenirajući skup dobijamo dobre rezultate, a za validacione loše. To bi ukazalo da je došlo do preobučavanja naše mreže. Kao *batch_size* koristimo 64, a broj epoha je 150.

7.3.3 REZULTATI

Kao najbolji rezultat dobili smo tačnost od 98% na testirajućem skupu. Hiperparametri mreže koja je dala ove rezultate su sledeći:

Hyperparameters:

input units: 18

dense_activation: sigmoid

n_layers: 2

input units1: 18

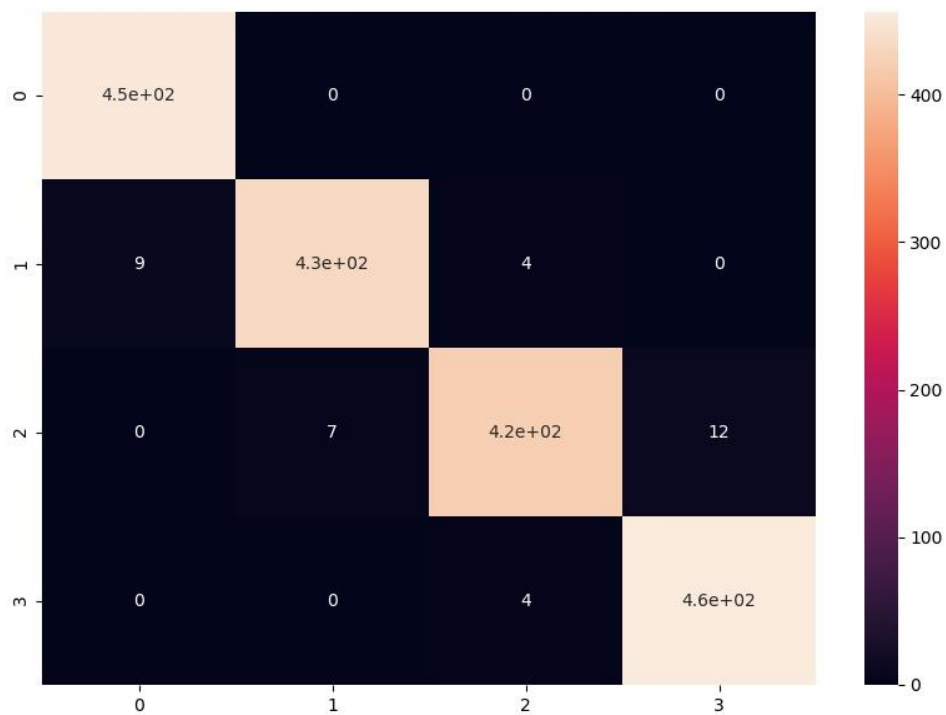
dense_activation1: relu

dropout_3: 0.45

Score: 0.9888888597488403

Trenirajući skup:

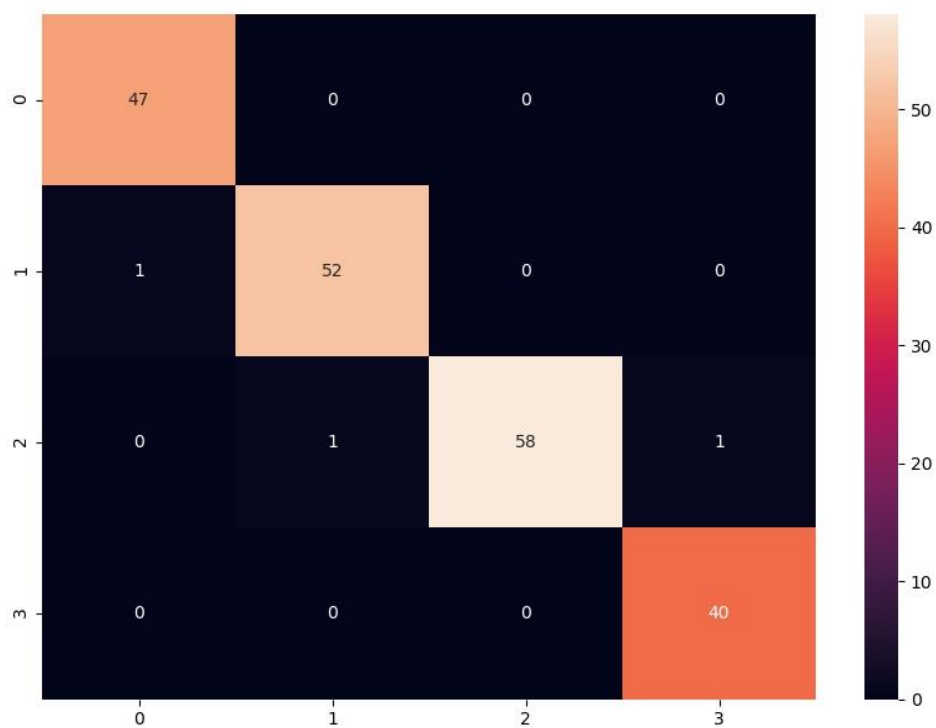
	precision	recall	f1-score	support
0	0.98	1.00	0.99	453
1	0.98	0.97	0.98	447
2	0.98	0.96	0.97	440
3	0.97	0.99	0.98	460
accuracy				0.98 1800
macro avg				0.98 0.98 0.98 1800
weighted avg				0.98 0.98 0.98 1800



Slika 12. Konfuziona matrica za trenirajući skup – neuralna mreža

Testirajući skup:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	47
1	0.98	0.98	0.98	53
2	1.00	0.97	0.98	60
3	0.98	1.00	0.99	40
accuracy			0.98	200
macro avg	0.98	0.99	0.99	200
weighted avg	0.99	0.98	0.98	200



Slika 13 . Konfuziona matrica za testirajući skup – Neuralna mreža

Kao što vidimo za neuralnu mrežu dobijamo tačnost 98%.

8 ZAKLJUČAK

TABELARAN PRIKAZ REZULTATA

ALGORITAM	Logistička regresija	Naivni Bayesov	KNN	Neuralna mreža
TAČNOST	0.97	0.81	0.62	0.98

Za naš dataset najbolje rezultate daje neuralna mreža, a najlošije KNN.