**Student:** Anja Vujačić

**Index:** RA 209/2021

# Regresija za određivanje cijene medicinskih troškova pacijenata

# Sadržaj

# UVOD

Razumijevanje faktora koji utiču na medicinske troškove pacijenata je od suštinskog značaja za planiranje i upravljanje zdravstvenim resursima. Uz sve veće troškove zdravstvene zaštite, tačna procjena medicinskih troškova postaje ključna za osiguravajuće kompanije, zdravstvene ustanove i same pacijente. Ovaj projekat ima za cilj razvoj preciznog regresionog modela koji može predvidjeti troškove liječenja na osnovu različitih karakteristika pacijenata.

Projekat se fokusira na analizu i modeliranje medicinskih troškova koristeći skup podataka koji uključuje informacije o starosti, polu, indeksu tjelesne mase (BMI), broju djece, statusu pušenja i regionu prebivališta pacijenata. Korišćenjem tehnika mašinskog učenja, kao što su Linear Regression, Lasso i Ridge Regression, SVM Regression, Decision Tree Regression i neuronske mreže, projekat ima za cilj da identifikuje faktore koji doprinose varijabilnosti troškova i da razvije model koji može pružiti tačne predikcije.

U ovom radu, posebna pažnja posvećena je procesu pretprocesiranja podataka, uključujući rješavanje problema sa nedostajućim vrijednostima, transformaciju kategoričkih promjenljivih i skaliranje podataka. Korišćenjem različitih metričkih vrijednosti kao što su Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE) i Coefficient of Variation (CV), evaluiraćemo performanse modela i iterativno ih poboljšavati.

Cilj ovog projekta je da se razvije model koji može podržati donošenje odluka u zdravstvenoj industriji, smanjujući neizvjesnost i poboljšavajući alokaciju resursa.

## Način pretprocesiranja podataka

U procesu pretprocesiranja podataka, prvo smo se suočili s problemom nedostajućih vrijednosti.

Prije nego što započnemo s pretprocesiranjem, podaci moraju biti učitani u DataFrame (Slika 1.). Za ovaj korak koristimo biblioteku pandas . Pandas omogućava jednostavno i efikasno upravljanje, analiziranje i manipulaciju podataka. Učitavanje podataka iz CSV datoteke vrši se sljedećim kodom:

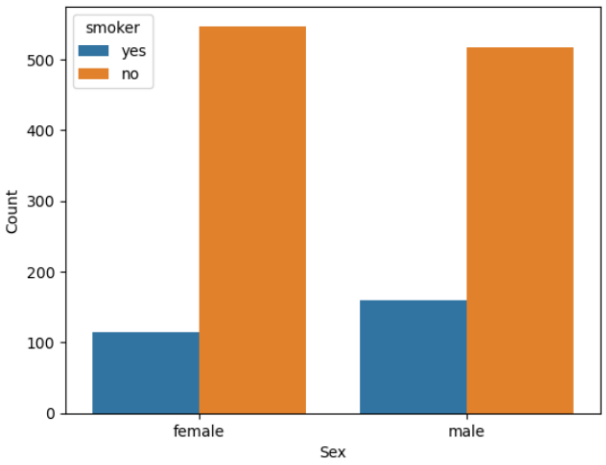
```
df=pd.read_excel("insurance.csv")
```

| | age | sex | bmi | children | smoker | region | charges | Medical id |
|---|---|---|---|---|---|---|---|---|
| **0** | 19 | female | 27.900 | 0 | yes | southwest | 16884.92400 | 1.0 |
| **1** | 18 | male | 33.770 | 1 | no | southeast | 1725.55230 | NaN |
| **2** | 28 | male | 33.000 | 3 | no | southeast | 4449.46200 | NaN |
| **3** | 33 | male | 22.705 | 0 | no | northwest | 21984.47061 | NaN |
| **4** | 32 | male | 28.880 | 0 | no | northwest | 3866.85520 | NaN |

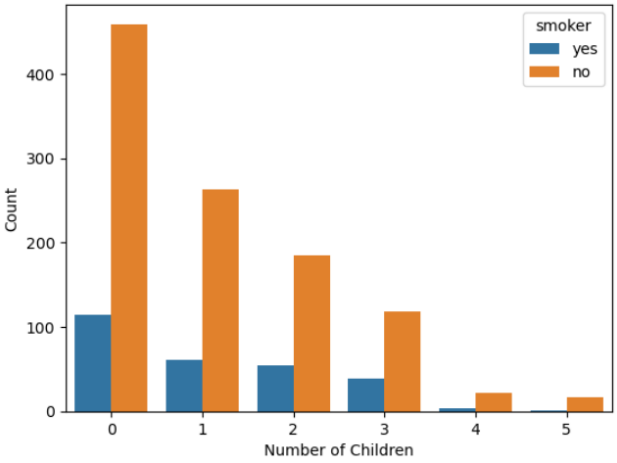
Slika 1.

## Vizualizacija podataka

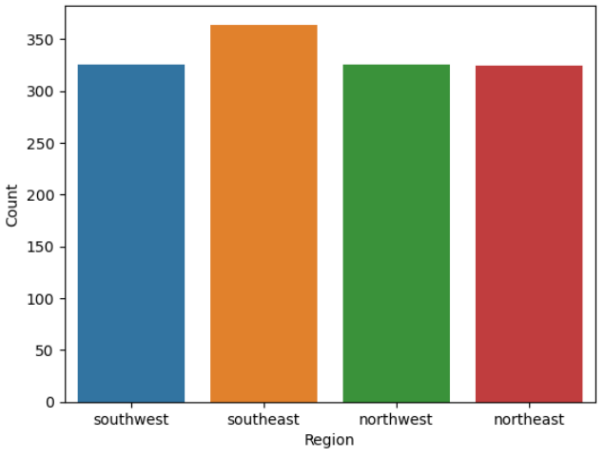
Nakon pretprocesiranja podataka, generisane su različite vizualizacije kako bismo bolje razumjeli distribuciju i odnose između različitih varijabli. Ovi grafici pružaju uvide u podatke i pomažu u daljoj analizi.



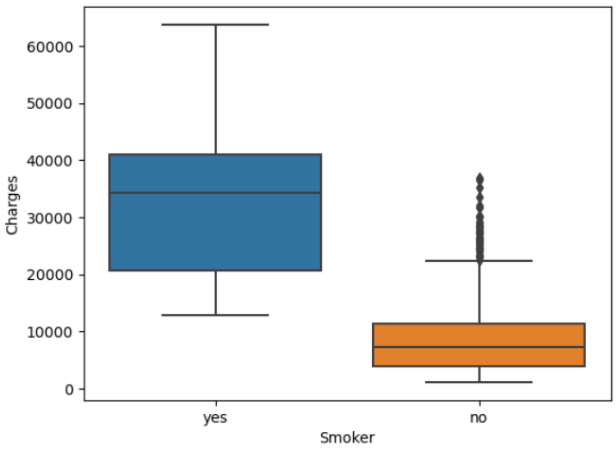

Broj pušača i nepušača za svaki pol



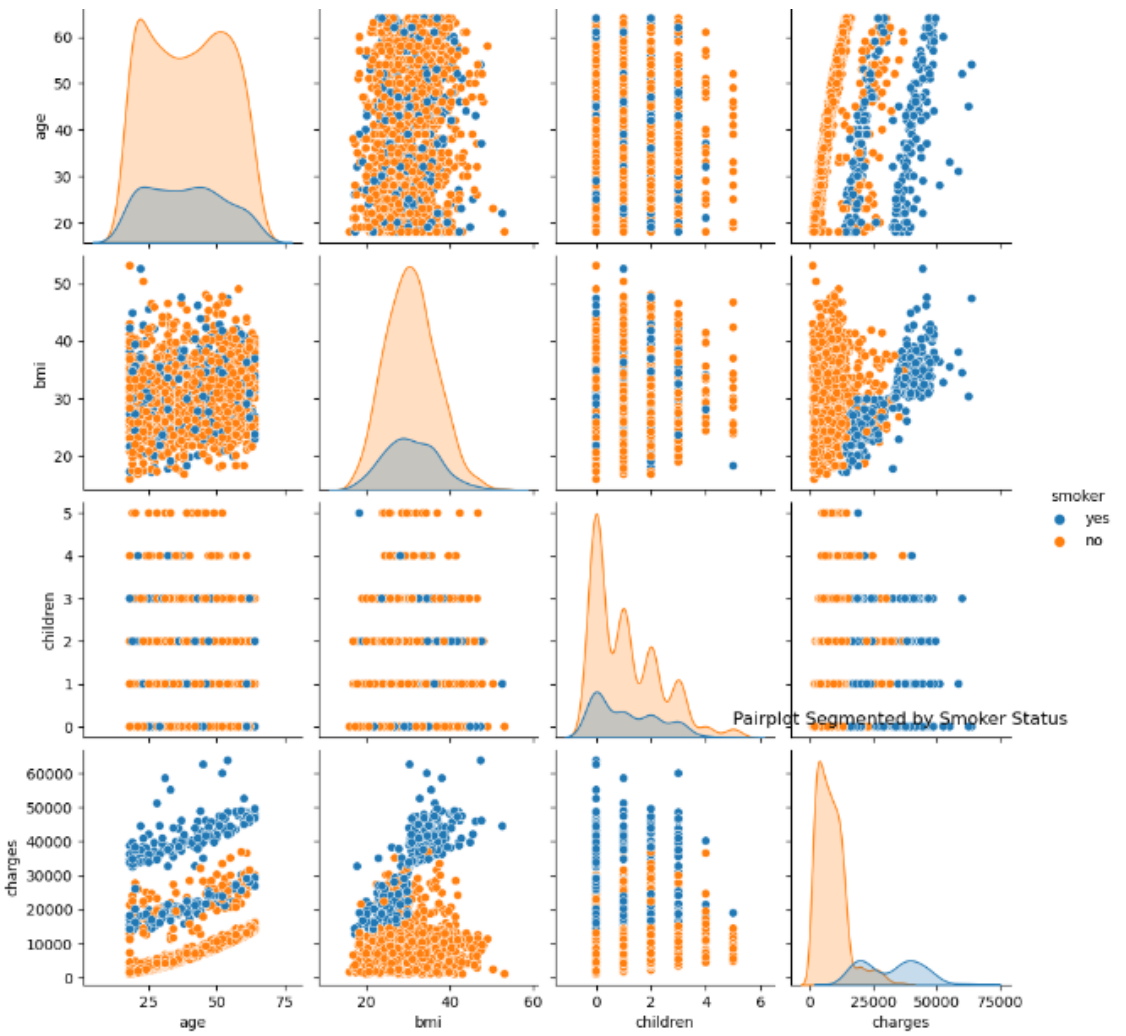

Broj djece za različite statuse pušenja




Broj pojedinaca u svakom regionu

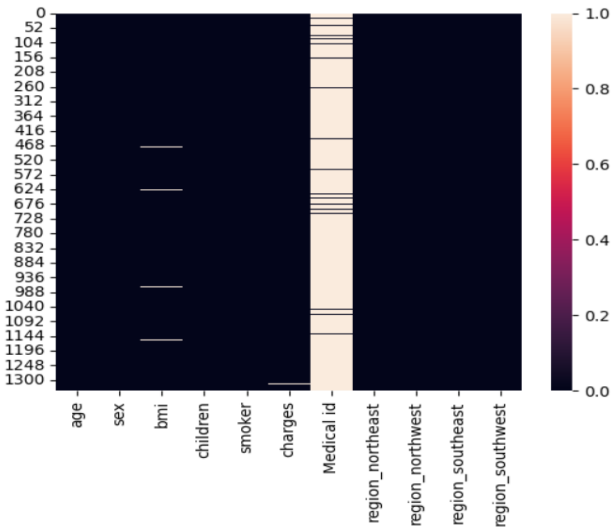



Raspodjela troškova po statusu pušenja




Odnosi između varijabli age, bmi, children i charges, segmentirani po statusu pušenja

Na slici (Slika 2.) može se vidjeti kako izgleda heatmap prije uklanjanja i popunjavanja nedostajućih vrijednosti:




Slika 2.

Analizom skupa podataka uočeno je nekoliko problema i poduzete su sljedeće akcije:

## Eliminacija kolone "Medical ID"

Većina informacija u koloni "Medical ID" je nedostajuća, što se vidi na slici 3., te je ova kolona eliminisana iz daljnjeg istraživanja. Ovaj korak je poduzet kako bi se smanjila složenost modela i uklonili irelevantni podaci.

| | age | sex | bmi | children | smoker | region | charges | Medical id |
|---|---|---|---|---|---|---|---|---|
| **0** | 19 | female | 27.900 | 0 | yes | southwest | 16884.92400 | 1.0 |
| **1** | 18 | male | 33.770 | 1 | no | southeast | 1725.55230 | NaN |
| **2** | 28 | male | 33.000 | 3 | no | southeast | 4449.46200 | NaN |
| **3** | 33 | male | 22.705 | 0 | no | northwest | 21984.47061 | NaN |
| **4** | 32 | male | 28.880 | 0 | no | northwest | 3866.85520 | NaN |

\

Slika 3.

Za uklanjanje ove kolone korišćen je sljedeći kod:

```
df=df.drop('Medical id',axis=1)
```

## Label Encoding

Kolone "sex" i "smoker" su kategoričke i stoga su transformisane u numeričke vrijednosti pomoću label encodinga. Ova transformacija omogućava da modeli mašinskog učenja bolje razumiju i obrađuju ove podatke.

| | age | sex | bmi | children | smoker | region | charges |
|---|---|---|---|---|---|---|---|
| **0** | 19 | female | 27.900 | 0 | yes | southwest | 16884.92400 |
| **1** | 18 | male | 33.770 | 1 | no | southeast | 1725.55230 |
| **2** | 28 | male | 33.000 | 3 | no | southeast | 4449.46200 |
| **3** | 33 | male | 22.705 | 0 | no | northwest | 21984.47061 |
| **4** | 32 | male | 28.880 | 0 | no | northwest | 3866.85520 |

**Primjer:**

- "sex" je transformisan u 0 za muškarce i 1 za žene.
- "smoker" je transformisan u 0 za nepušače i 1 za pušače.

Za ovu transformaciju koristili smo rječnik koji mapira ove vrijednosti na brojeve (Slika 4.), koristeći sledeći kod:

```
rjecnik = {"male": 0, "female": 1}

df['sex'] = df['sex'].map(rjecnik)
```

```
rjecnikPusaci={"yes":1,"no":0}

df['smoker']=df['smoker'].map(rjecnikPusaci)
```

| | age | sex | bmi | children | smoker | region | charges |
|---|---|---|---|---|---|---|---|
| **0** | 19 | 1 | 27.900 | 0 | 1 | southwest | 16884.92400 |
| **1** | 18 | 0 | 33.770 | 1 | 0 | southeast | 1725.55230 |
| **2** | 28 | 0 | 33.000 | 3 | 0 | southeast | 4449.46200 |
| **3** | 33 | 0 | 22.705 | 0 | 0 | northwest | 21984.47061 |
| **4** | 32 | 0 | 28.880 | 0 | 0 | northwest | 3866.85520 |

Slika 4.

## One Hot Encoding

Kolona "region" sadrži više kategoričkih vrijednosti (“southwest”, “southeast”, " northwest" i " northeast") (Slika 4.). Da bi se efikasno reprezentovali različiti regioni, bez uvođenja redoslijeda ili hijerarhije, koristi se one hot encoding. One hot encoding transformiše svaku kategoričku vrijednost u novu binarnu kolonu, gdje svaka kolona predstavlja jednu moguću vrijednost kategorije.

**Primjer:**

- "region" je transformisan u više binarnih kolona, npr. "region_southwest ", "region_southeast", itd. (Slika 5.)

Za ovu transformaciju koristi se sljedeći kod:

```
region_dummies = pd.get_dummies(df['region'], prefix='region')
```

*get_dummies* je funkcija iz biblioteke pandas koja se koristi za konvertovanje

| | region_northeast | region_northwest | region_southeast | region_southwest |
|---|---|---|---|---|
| **0** | False | False | False | True |
| **1** | False | False | True | False |
| **2** | False | False | True | False |
| **3** | False | True | False | False |
| **4** | False | True | False | False |
| **...** | ... | ... | ... | ... |

Slika 5.

Sledeći korak je spajanje originalnog DataFrame-a sa novim binarnim kolonama i uklanjanje originalne kolone "region":

```
df = pd.concat([df, region_dummies], axis=1)
df = df.drop('region', axis=1)
```

**axis=0**: Predstavlja redove.

**axis=1**: Predstavlja kolone.

Nakon ovih koraka, tabela će izgledati kao na slici 6.

| | age | sex | bmi | children | smoker | charges | region_northeast | region_northwest | region_southeast | region_southwest |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 19 | 1 | 27.900 | 0 | 1 | 16884.92400 | False | False | False | True |
| **1** | 18 | 0 | 33.770 | 1 | 0 | 1725.55230 | False | False | True | False |
| **2** | 28 | 0 | 33.000 | 3 | 0 | 4449.46200 | False | False | True | False |
| **3** | 33 | 0 | 22.705 | 0 | 0 | 21984.47061 | False | True | False | False |
| **4** | 32 | 0 | 28.880 | 0 | 0 | 3866.85520 | False | True | False | False |

Slika 6.

## Popunjavanje nedostajućih vrijednosti u koloni "BMI"

Kolona "BMI" ima određeni broj nedostajućih vrijednosti. Da bi se osigurao kontinuitet analize i izbjegao gubitak podataka, nedostajuće vrijednosti su popunjene prosječnom vrijednošću za odgovarajuće godine i pol. Ova metoda osigurava da popunjene vrijednosti budu reprezentativne za odgovarajuću podgrupu podataka.

Za ovu metodu koristi se sljedeći kod:

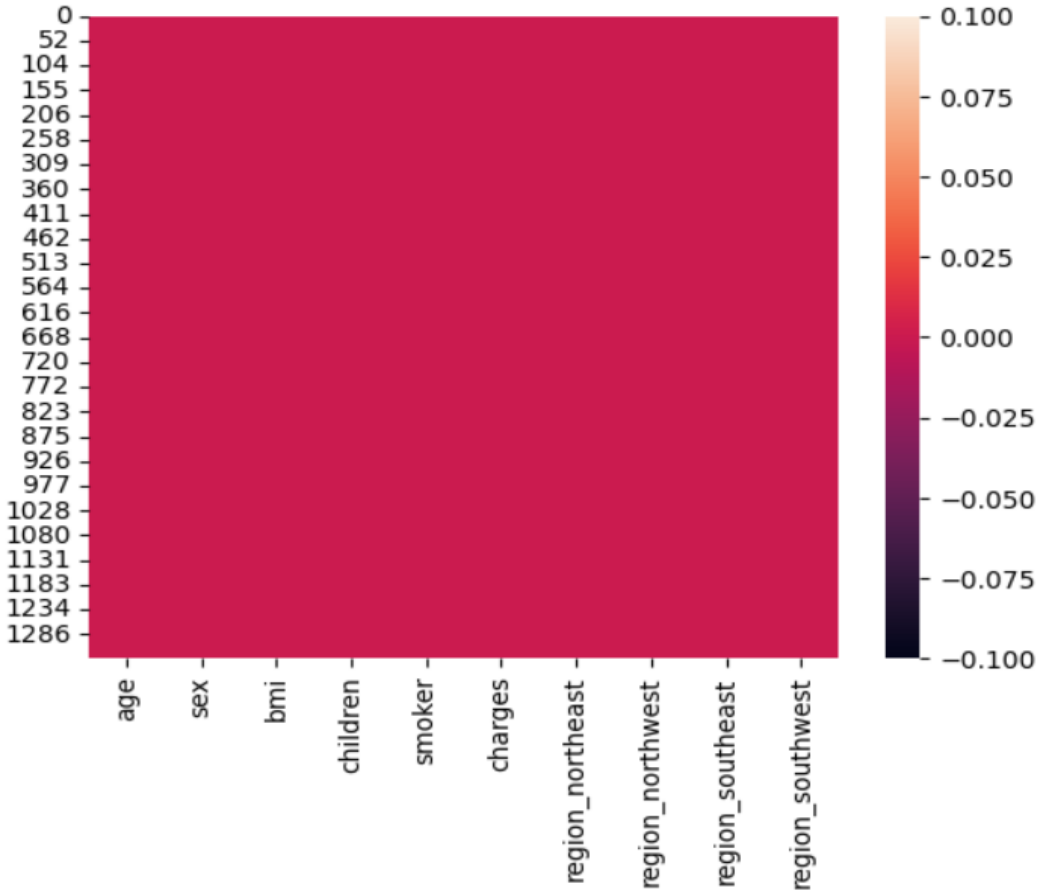
```
mean_bmi = df['bmi'].mean()
df['bmi'] = df.groupby(['age', 'sex'])['bmi'].transform(lambda x: x.fillna(x.mean()))
```

## Eliminacija redova s nedostajućim vrijednostima u koloni "charges"

Kolona "charges" je ključna za našu analizu jer predstavlja ciljnu promjenljivu čiju vrijednost treba predvidjeti. Redovi koji sadrže nedostajuće vrijednosti u ovoj koloni su eliminisani kako bi se osiguralo da model ima sve potrebne informacije za treniranje i evaluaciju.
Implementacija ovog koda je jednostavna, jer smo već riješili prethodne probleme s podacima. S obzirom na to da je broj redova sa nedostajućim vrijednostima u koloni "charges" bio relativno mali, uklonićemo ih:

```
df=df.dropna()
```

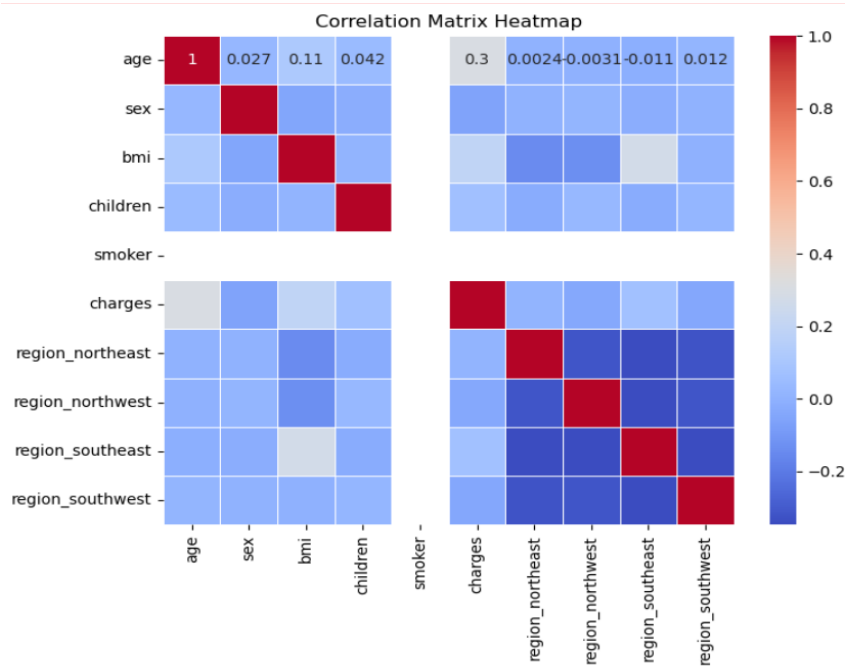
Nakon što smo završili sa svim koracima pretprocesiranja podataka, heatmap izgleda ovako:



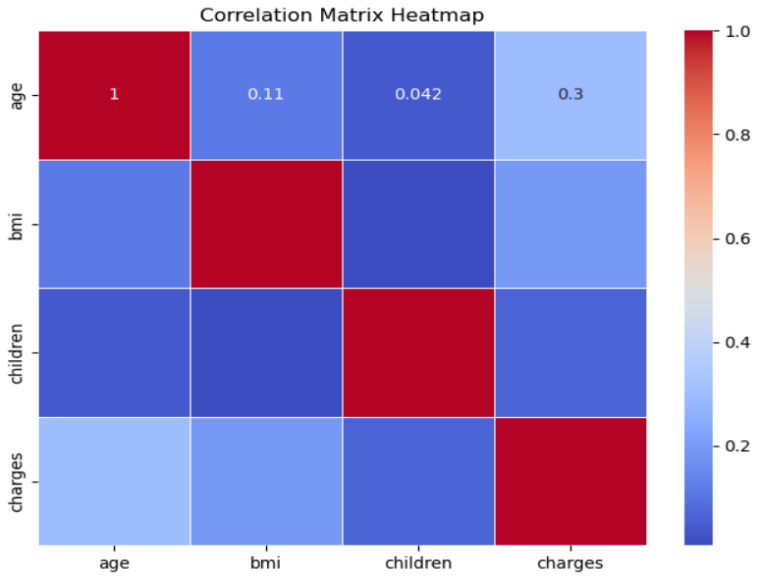

Slika 7.

## Korelacija između kolona

Nakon pretprocesiranja podataka, dodatno smo analizirali korelacije između različitih kolona u DataFrame-u koristeći heatmap. Ove vizualizacije nam pomažu da bolje razumijemo odnose između varijabli.




Korelacija između svih kolona




Korelacija između specifičnih kolona

## Podjela podataka na skupove

Nakon pretprocesiranja podataka i vizualizacije različitih varijabli, sledeći korak je podjela podataka na skupove za treniranje i testiranje. Ova podjela omogućava da se model trenira na jednom dijelu podataka, dok se drugi dio koristi za evaluaciju performansi modela.

Za podjelu podataka koristimo funkciju *train_test_split* iz biblioteke scikit-learn koju je potrebno importovati.

Prvo, definišemo ulazne varijable (X) i ciljnu varijablu (y):

```
X = df.drop('charges', axis=1)
y = df['charges']
```

Ovim kodom se uklanja kolona 'charges' iz DataFrame-a i sve preostale kolone se koriste kao ulazne varijable (X), dok kolona 'charges' postaje ciljana varijabla (y).
Zatim dijelimo podatke na trenirajući i testirajući skup:

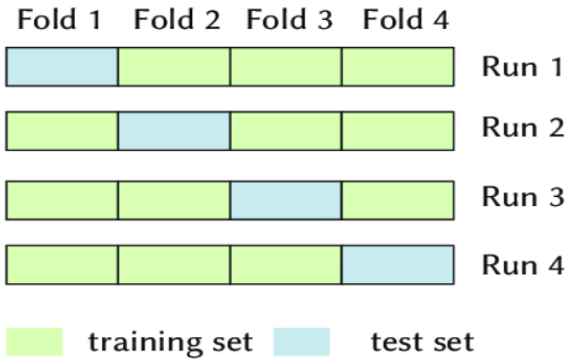
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

test_size=0.2: Određuje da 20% podataka bude izdvojeno za testiranje, dok će 80% podataka biti korišćeno za treniranje modela.

### DODATNO

### K-Fold unakrsna validacija/ Cross-Validation

K-fold unakrsna validacija je tehnika evaluacije modela koja pomaže u procjeni kako će se model performisati na neviđenim podacima. Podaci se dijele na 'k' jednakih djelova, gde se model trenira na 'k-1' djelova dok se preostali dio koristi za testiranje (Slika 10.). Ovaj proces se ponavlja 'k' puta, sa svakim dijelom podataka koji se jednom koristi kao test set. U praksi se često koriste ugrađene funkcije poput cross_val_score iz scikit-learn biblioteke koje automatizuju ovaj proces.




Slika 8.

## Skaliranje podataka

Nakon pretprocesiranja, primenjuje se skaliranje podataka kako bi se osiguralo da su sve karakteristike podataka u odgovarajućem opsegu. Skaliranje podataka je važan korak u pretprocesiranju, jer mnogi algoritmi mašinskog učenja bolje funkcionišu kada su ulazni podaci na sličnom opsegu.
Koriste se dvije različite metode skaliranja podataka: MinMaxScaler i StandardScaler. Nakon primjene obje metode, bira se optimalni pristup koji daje najbolje rezultate na trenirajućim i testirajućim podacima.

### StandardScaler

StandardScaler transformiše podatke tako da imaju srednju vrijednost 0 i standardnu devijaciju 1

```
standardScaler=StandardScaler()
X_train_scaled=standardScaler.fit_transform(X_train)
X_test_scaled=standardScaler.transform(X_test)
```

### MinMaxScaler

MinMaxScaler transformiše podatke tako da se svi podaci nalaze u opsegu između 0 i 1.

```
minMaxScaler=MinMaxScaler()
X_train_minMaxScaled=minMaxScaler.fit_transform(X_train)
X_test_minMaxScaled=minMaxScaler.transform(X_test)
```

## Izbor modela/Evaluacija greške

Za rješavanje problema procjene medicinskih troškova koriste se različiti regresioni modeli. Svaki od ovih modela može imati različite performanse, pa je važno analizirati i uporediti njihove rezultate. Sledeći modeli su korišćeni za analizu: Linear Regression, Lasso Regression, Ridge Regression, SVM Regression, Decision Tree Regression i neuronska mreža za regresione probleme. Evaluacija performansi svakog modela vrši se korišćenjem metrika kao što su Mean Absolute Error (MAE), Mean Squared Error (MSE) i Root Mean Squared Error (RMSE).

Kako bismo olakšali evaluaciju performansi različitih modela, definisali smo funkciju *print_metrics* koja izračunava, ispisuje i vizualizuje osnovne metričke vrijednosti. Funkcija print_metrics je koristi za analizu performansi predikcijskih modela kroz izračunavanje, ispisivanje, čuvanje i vizualizaciju različitih metričkih vrijednosti. Omogućava nam da efikasno upoređujemo rezultate različitih modela i identifikujemo najbolje performanse za zadate podatke.

### Linearna regresija

Linearna regresija je osnovni model za regresione probleme koji pretpostavlja linearnu vezu između ulaznih varijabli i ciljne varijable.

Potrebno je importovati :

```
from sklearn.linear_model import LinearRegression
```

STANDARD SCALER (Slika 9.)

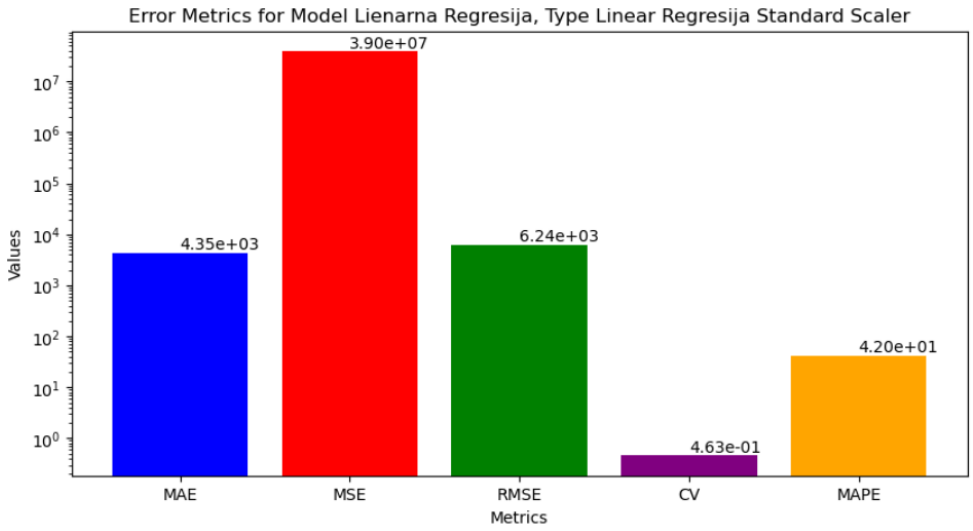
```
lr=LinearRegression()
lr.fit(X_train_scaled,y_train)
predictions=lr.predict(X_test_scaled)
print_metrics(y_test,predictions,"Lienarna Regresija","Linear Regresija Standard Scaler")
```

MIN MAX SCALER (Slika 10.)

```
lrMinMax=LinearRegression()
lrMinMax.fit(X_train_minMaxScaled,y_train)
predictionsMinMax=lrMinMax.predict(X_test_minMaxScaled)
print_metrics(y_test,predictionsMinMax,"Lienarna Regresija","Linear Regresija MinMax Scaler")
```

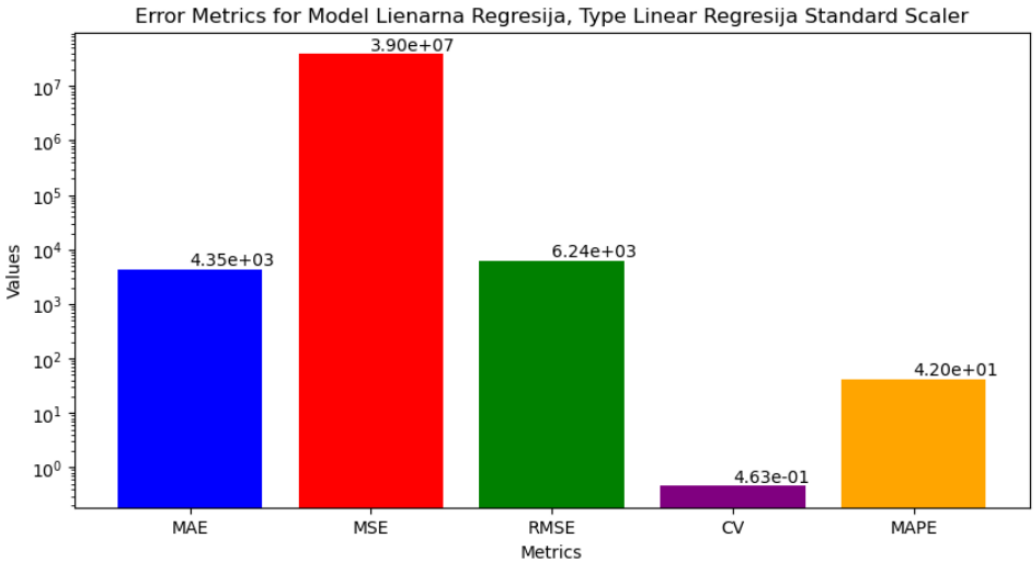
Grafikon koji funkcija print_metrics generiše je grafikon koji vizualizuje osnovne metričke vrijednosti: MAE, MSE, RMSE i CV.

## STANDARD SCALER




Slika 9.

## MIN MAX SCALER




Slika 10.

## Lasso regresija

Lasso (Least Absolute Shrinkage and Selection Operator) regresija je metoda linearne regresije koja koristi L1 regularizaciju kako bi poboljšala model. L1 regularizacija dodaje penalizaciju apsolutnoj vrijednosti koeficijenata modela, što može rezultovati postavljanjem nekih koeficijenata na nulu.

Lasso regresija se koristi za procjenu medicinskih troškova pacijenata na osnovu različitih ulaznih podataka (kao što su starost, pol, BMI, broj djece, status pušenja, region itd.). Primjena Lasso regresije može pomoći u identifikaciji koje varijable imaju najveći uticaj na medicinske troškove, istovremeno smanjujući uticaj nebitnih varijabli.

Prvo je potrebno importovati Lasso klasu iz scikit-learn biblioteke:

```
from sklearn.linear_model import Lasso
```

STANDARD SCALER (Slika 11.)

```
lasso=Lasso(alpha=0.1)
 lasso.fit(X_train_scaled, y_train)
predictions=lasso.predict(X_test_scaled)
print_metrics(y_test,predictions,"Lienarna Regresija","Lasso Standard Scaler")
```

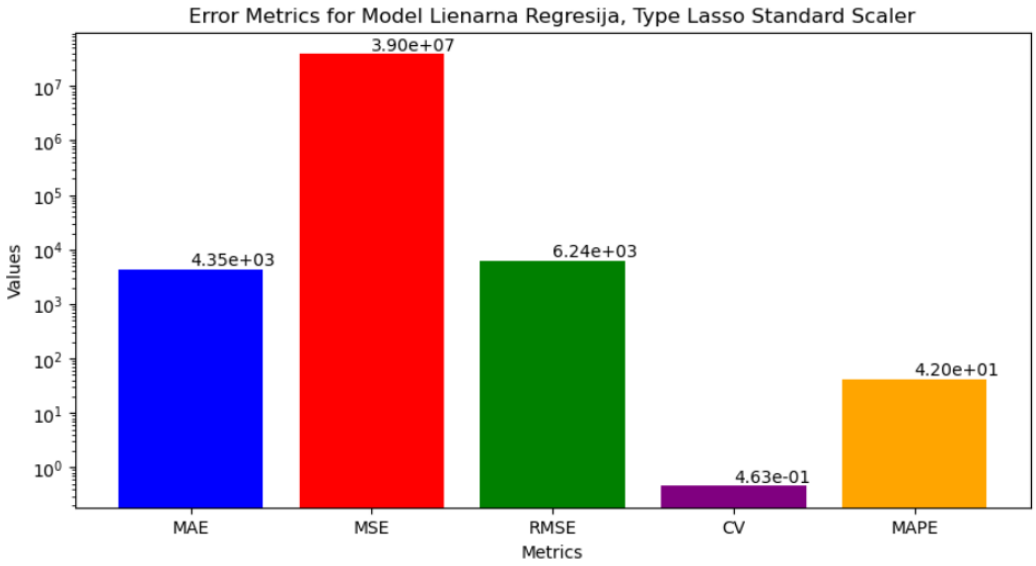
Parametar alpha predstavlja stepen regularizacije koja se primenjuje na model.
**Veća vrijednost za alpha** znači veći stepen regularizacije.
**Manja vrijednost za alpha** znači manji stepen regularizacije.

MIN MAX SCALER (Slika 12.)

```
lassoMinMax.fit(X_train_minMaxScaled, y_train)
lassoMinMax.fit(X_train_minMaxScaled, y_train)
predictionsMinMaxLasso = lassoMinMax.predict(X_test_minMaxScaled)
print_metrics(y_test, predictionsMinMaxLasso, "Linearna Regresija", "Lasso MinMax Scaler")
```

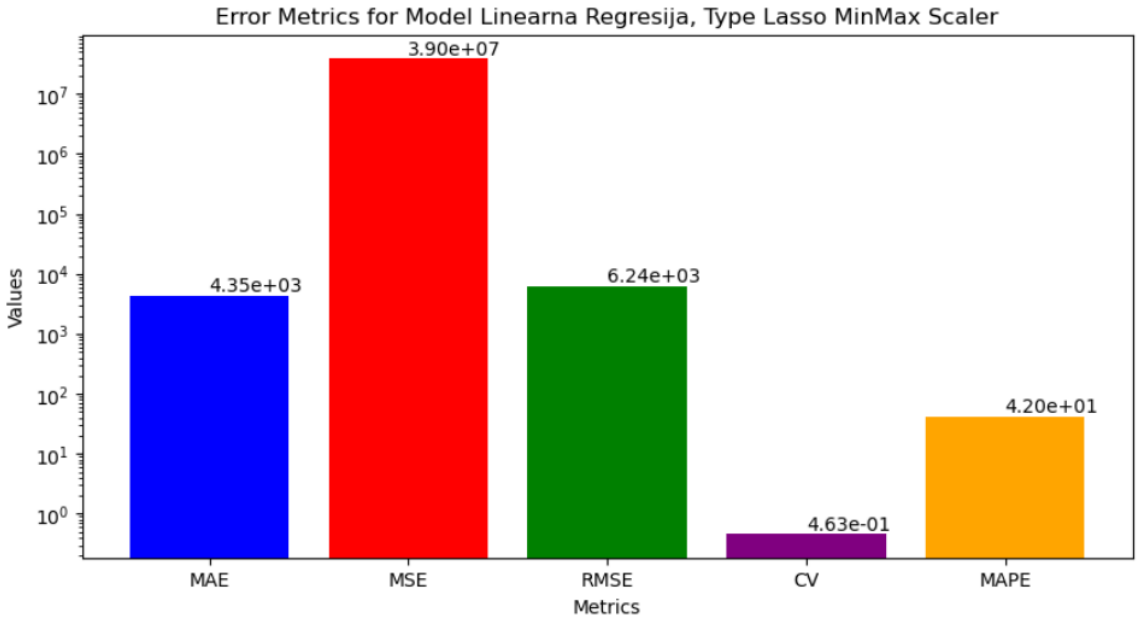
Grafikon koji funkcija print_metrics generiše je grafikon koji vizualizuje osnovne metričke vrijednosti: MAE, MSE, RMSE i CV.

## STANDARD SCALER




Slika 11.

## MIN MAX SCALER




Slika 12.

## Ridge regresija

Ridge regresija je oblik linearne regresije koji koristi L2 regularizaciju za poboljšanje predikcija modela, naročito kada postoji multikolinearnost između varijabli ili kad su podaci ograničeni. Za razliku od Lasso regresije, koja može svesti neke koeficijente na nulu, Ridge regresija proporcionalno smanjuje sve koeficijente, ali ih ne eliminiše. Ovo je korisno u situacijama gdje su sve karakteristike važne za predikciju

.

Prvo je potrebno importovati Ridge klasu iz scikit-learn biblioteke:

```
from sklearn.linear_model import Ridge
```

STANDARD SCALER (Slika 13.)

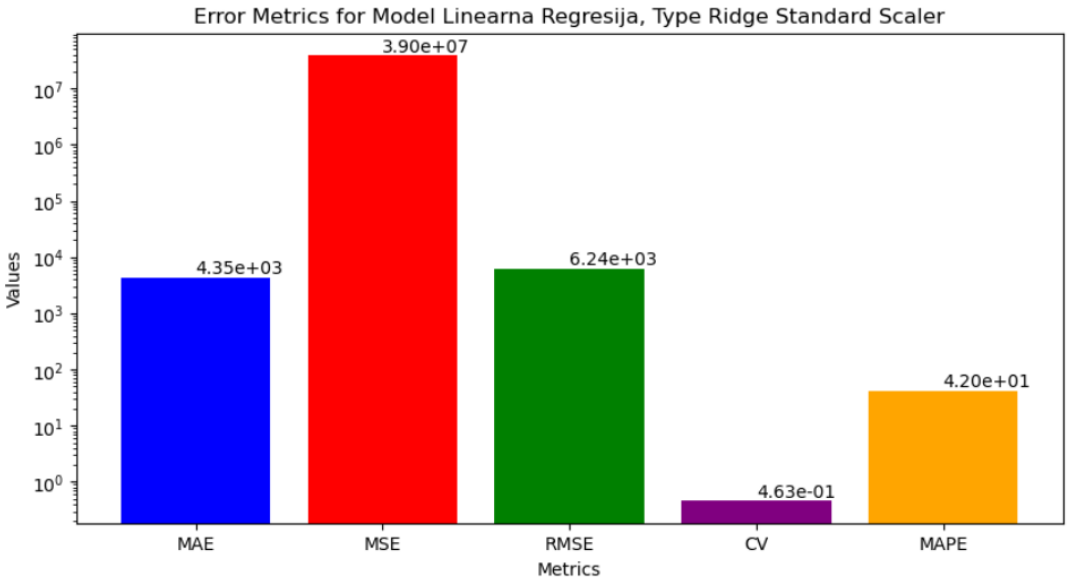
```
ridgeModel = Ridge(alpha=1.0)
ridgeModel.fit(X_train_scaled, y_train)
predictions = ridgeModel.predict(X_test_scaled)
print_metrics(y_test, predictions, "Linearna Regresija", "Ridge Standard Scaler")
```

MIN MAX SCALER (Slika 14.)

```
ridgeModel.fit(X_train_minMaxScaled, y_train)
predictions = ridgeModel.predict(X_test_minMaxScaled)
print_metrics(y_test, predictions, "Linearna Regresija", "Ridge MinMax Scaler")
```

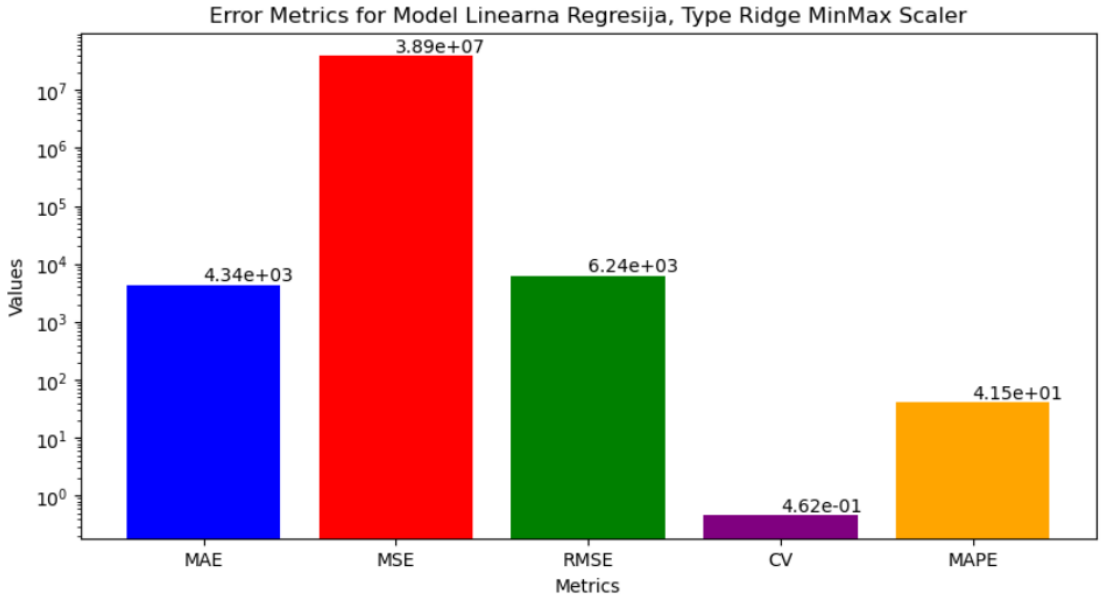
Grafikon koji funkcija print_metrics generiše je grafikon koji vizualizuje osnovne metričke vrijednosti: MAE, MSE, RMSE i CV.

## STANDARD SCALER




Slika 13.

## MIN MAX SCALER




Slika 14.

## Support Vector Regression (SVR)

Support Vector Regression (SVR) je algoritam mašinskog učenja koji koristi koncepte iz Support Vector Machines (SVM) za rješavanje regresionih problema. SVR pruža fleksibilan pristup u predikciji kontinuiranih varijabli, razvijen na osnovu principa maksimizacije margine i minimizacije greške.

Prvo je potrebno importovati:

```
from sklearn.svm import SVR
```

STANDARD SCALER (Slika 15.)

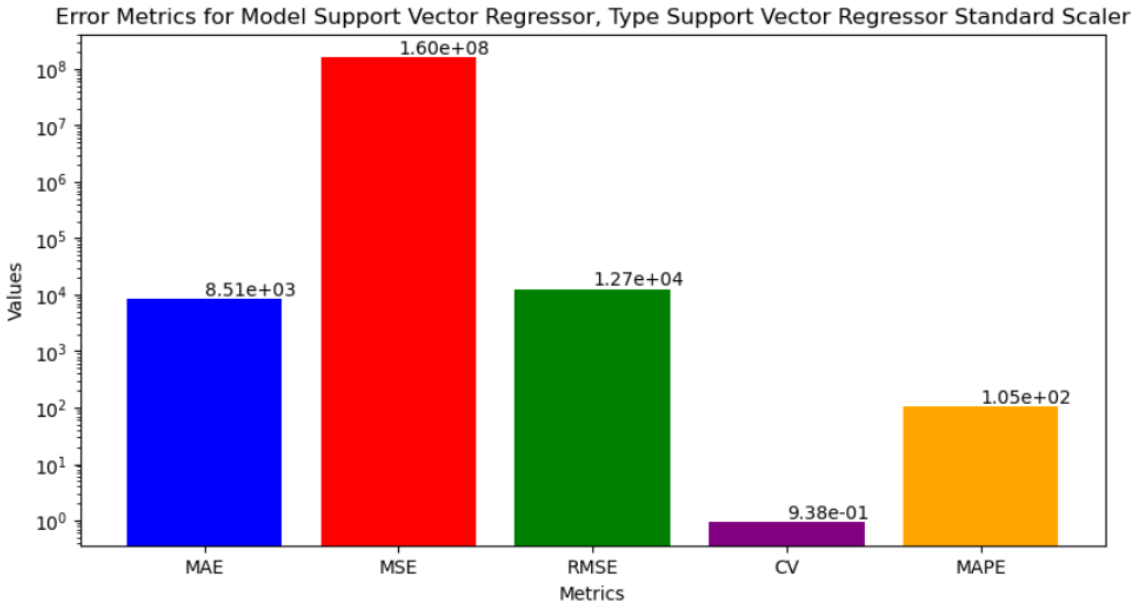
```
svr = SVR()
svr.fit(X_train_scaled,y_train)
predictionsSVR=svr.predict(X_test_scaled)
print_metrics(y_test,predictionsSVR,"Support Vector Regressor","Support Vector Regressor Standard Scaler")
```

MIN MAX SCALER (Slika 16.)

```
svr.fit(X_train_minMaxScaled,y_train)
predictionsSVR=svr.predict(X_test_minMaxScaled)
print_metrics(y_test,predictionsSVR,"Support Vector Regressor","Support Vector Regressor MinMax Scaler")
```

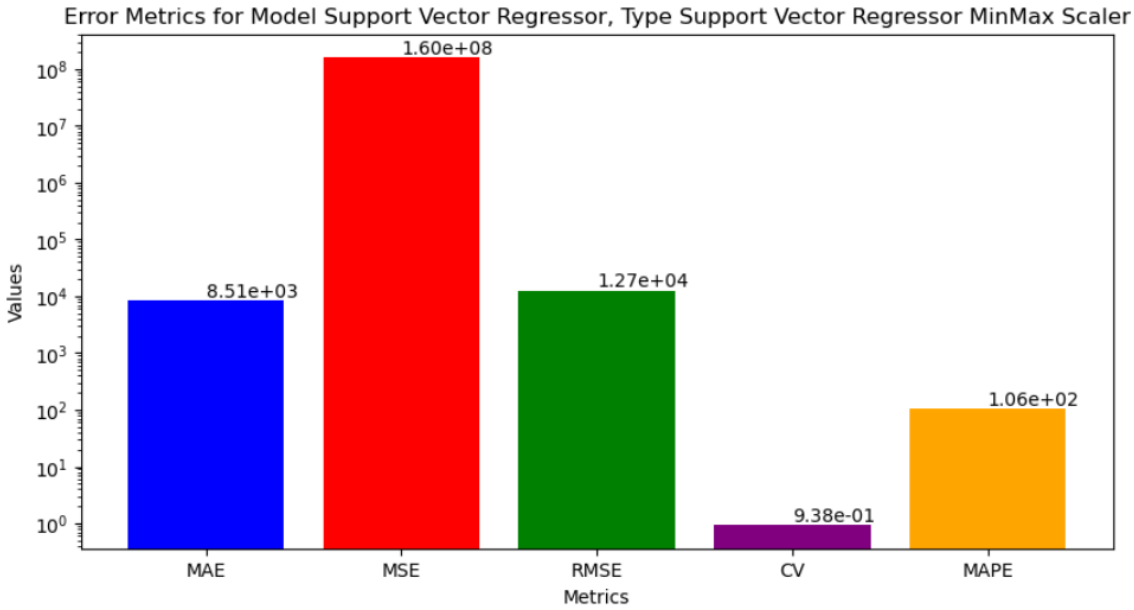
Grafikon koji funkcija print_metrics generiše je grafikon koji vizualizuje osnovne metričke vrijednosti: MAE, MSE, RMSE i CV.

## STANDARD SCALER




Slika 15.

## MIN MAX SCALER




Slika 16.

## Decision Tree Regression

Decision Tree Regression koristi model stabla odlučivanja za predviđanje kontinuiranih vrijednosti. Ova metoda se oslanja na konstrukciju stabla odlučivanja gdje čvorovi predstavljaju pitanja na osnovu karakteristika, a listovi predstavljaju prognozirane vrijednosti.

Prvo je potrebno importovati:

```
from sklearn.tree import DecisionTreeRegressor
```

STANDARD SCALER (Slika 17.)

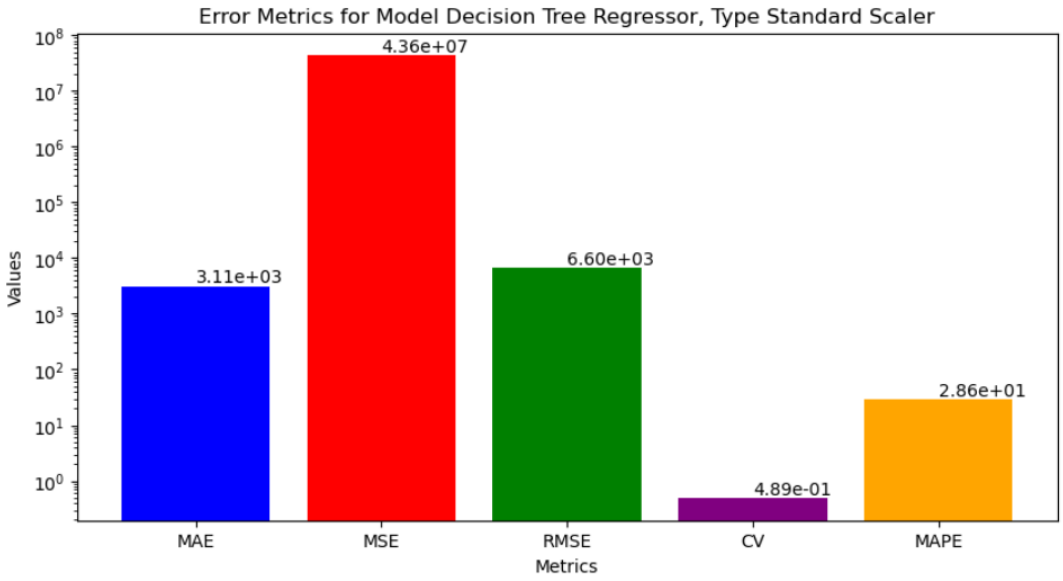
```
DTR=DecisionTreeRegressor()
DTR.fit(X_train_scaled,y_train)
predictions=DTR.predict(X_test_scaled)
print_metrics(y_test,predictions,"Decision Tree Regressor", "Standard Scaler")
```

MIN MAX SCALER (Slika 18.)

```
DTR.fit(X_train_minMaxScaled,y_train)
predictions=DTR.predict(X_test_minMaxScaled)
print_metrics(y_test,predictions,"Decision Tree Regressor", "MinMax Scaler")
```

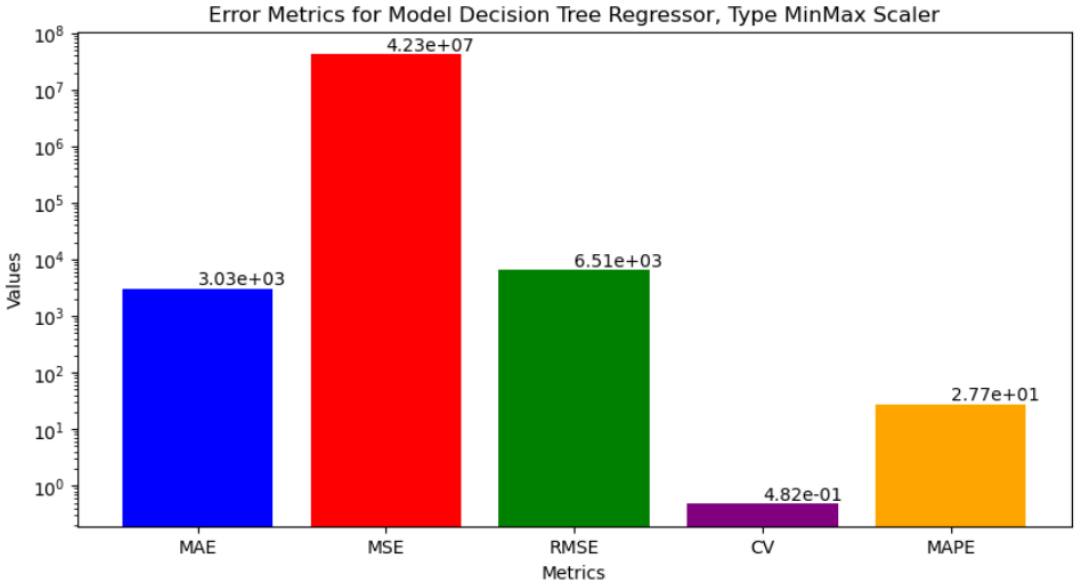
Grafikon koji funkcija print_metrics generiše je bar grafikon koji vizualizuje osnovne metričke vrijednosti: MAE, MSE, RMSE i CV.

## STANDARD SCALER




Slika 17.

## MIN MAX SCALER




Slika 18.

## Neuronske mreže za regresiju

U ovom dijelu izveštaja razmatraćemo primjenu neuronskih mreža za rješavanje regresionih problema, koristeći TensorFlow i Keras biblioteke za izgradnju i obuku modela. Neuronske mreže pružaju fleksibilnost i visoku efikasnost u modelovanju kompleksnih ne-linearitеta među podacima, što ih čini idealnim za složene zadatke poput predviđanja medicinskih troškova.

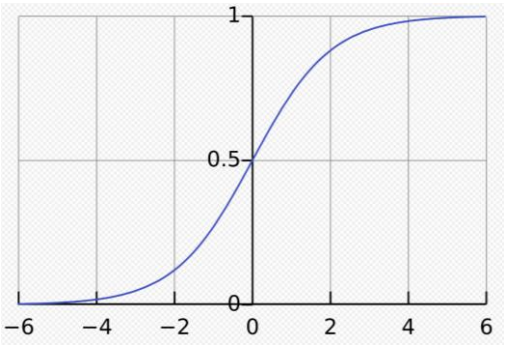
Zahvaljujući svojoj sposobnosti da se nose sa velikim i kompleksnim skupovima podataka, neuronske mreže obećavaju značajna unapređenja u preciznosti predviđanja i efikasnosti zdravstvenog sistema.

Importovanje TensorFlow-a:

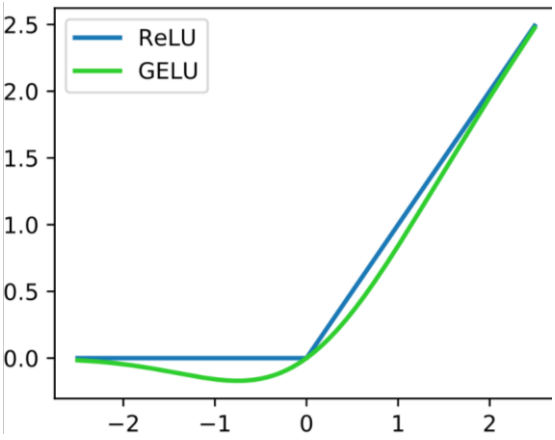
```
import tensorflow as tf
```

**Dense** sloj je potpuno povezani sloj (fully connected layer).Predstavlja osnovni blok u većini neuronskih mreža. U Dense sloju, svaki neuron je povezan sa svim neuronima u prethodnom sloju, što znači da su informacije sa svih neurona u prethodnom sloju dostupne svakom neuronu u Dense sloju.

**Sigmoid**
Sigmoidna funkcija transformiše ulazne vrednosti u opseg između 0 i 1.




**Relu**
Relu funkcija aktivacije omogućava prolaz pozitivnih vrednosti bez promene, dok su sve negativne vrednosti postavljene na nulu.



Metoda compile() konfiguriše model specificiranjem optimizatora, funkcije gubitka i metrika za evaluaciju, pripremajući model za obuku.

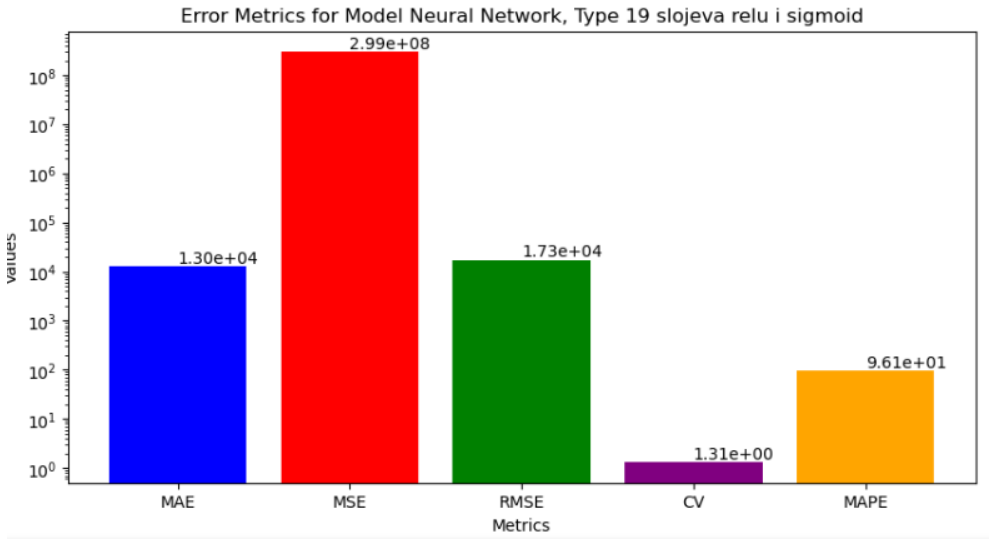
```
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])
```

- Optimizer specificira algoritam koji će se koristiti za ažuriranje težina mreže tokom obuke.
- Loss funkcija (funkcija gubitka) koristi se za izračunavanje količine greške između predikcija modela i stvarnih vrijednosti.
- Metrics definisu koje mjerenje performansi će se koristiti za evaluaciju modela tokom treninga i testiranja.

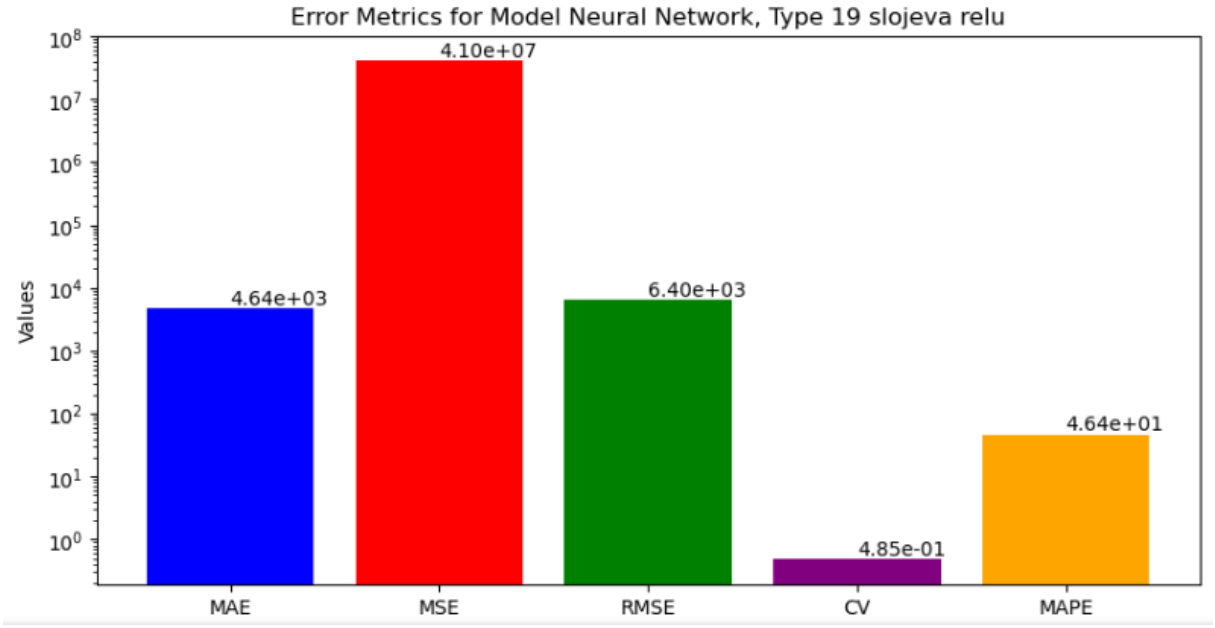
Metoda fit() zatim trenira model na određenom skupu podataka kroz definisani broj epoha, koristeći podatke za obuku i opcionalno, podatke za validaciju.

```
model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))
```

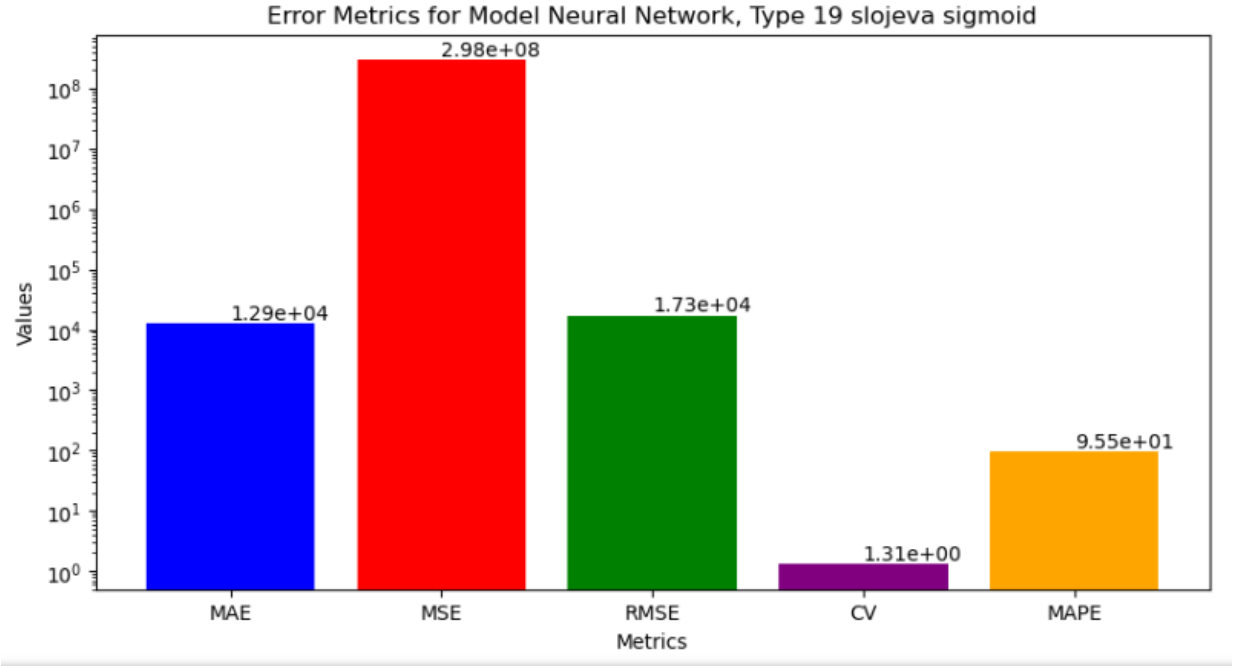
- X_train predstavlja skup podataka koji se koristi za obuku modela.
- y_train je skup ciljnih vrijednosti
- Epochs određuje broj puta da se kompletni skup podataka X_train koristi za obuku modela.
- Validation_data se koristi za evaluaciju modela na neviđenim podacima nakon svake epohe.




19 slojeva (relu i sigmoid)




19 slojeva (relu)




19 slojeva(sigmoid)

## Podešavanje hiperparametara izabranog modela

Izbor hiperparametara pomoću GridSearchCV predstavlja sistematičan način testiranja kombinacija različitih vrijednosti hiperparametara kako bi se našla ona konfiguracija koja omogućava modelu najbolju performansu.

### Linearna regresija

Za model linearne regresije, korišćeni hiperparametri uključuju positive i fit_intercept. Hiperparametar positive osigurava da svi koeficijenti u modelu linearne regresije budu pozitivni, dok fit_intercept kontroliše da li će model uključivati presretač (intercept) ili ne.

### Lasso regresija

Za Lasso regresiju, korišćeni hiperparametri uključuju alpha, positive i fit_intercept. Hiperparametar alpha kontroliše snagu regularizacije. Veće vrijednosti alpha dovode do veće penalizacije za velike koeficijente, što može pomoći u sprečavanju overfittinga modela.

Hiperparametar positive osigurava da svi koeficijenti u modelu budu pozitivni, što može biti korisno u određenim situacijama kada negativni koeficijenti nemaju smisla za određeni problem. Hiperparametar fit_intercept kontroliše da li će model uključivati presretač (intercept) ili ne.

### Ridge regresija

Za Ridge regresiju, korišćeni hiperparametri uključuju alpha, positive i fit_intercept, isti kao i kod Lasso regresije.

### Support Vector Regression

Za SVR (Support Vector Regression), korišćeni hiperparametri uključuju C, epsilon i kernel. Hiperparametar C kontroliše regularizaciju modela. Veće vrednosti C omogućavaju modelu da se bolje prilagodi trenirajućim podacima. Niže vrednosti C čine model otpornijim na overfitting, ali mogu smanjiti njegovu tačnost na trenirajućim podacima.

Hiperparametar epsilon definiše marginu unutar koje se greške predikcije ne penalizuju. Hiperparametar kernel određuje tip funkcije jezgra koja se koristi za mapiranje podataka u viši dimenzionalni prostor.

### Decision Tree Regression

Za DTR (Decision Tree Regression), korišćeni hiperparametri uključuju criterion, splitter, max_depth, min_samples_split i min_samples_leaf.

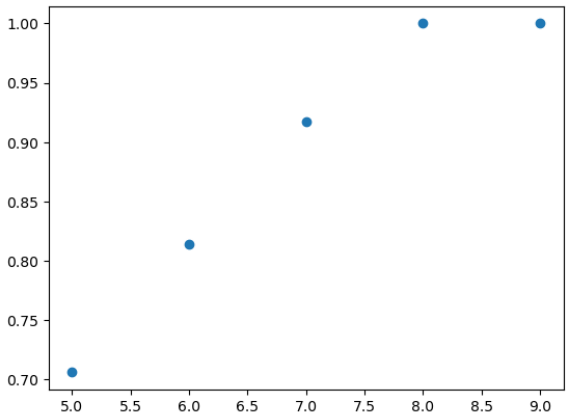
Hiperparametar criterion određuje funkciju koja se koristi za mjerenje kvaliteta podjele. Hiperparametar splitter određuje strategiju koja se koristi za dijeljenje čvorova. Hiperparametar max_depth određuje maksimalnu dubinu stabla.

Hiperparametar min_samples_split određuje minimalan broj uzoraka potrebnih za dijeljenje internog čvora.
Hiperparametar min_samples_leaf određuje minimalan broj uzoraka koji svaki list mora da sadrži.

Koriscenjem hiperparametara dobili smo bolje rezultate samo za SVR model.

## Principalna komponentna analiza (PCA)

Principalna komponentna analiza (PCA) je statistička tehnika koja se koristi za smanjenje dimenzionalnosti podataka, istovremeno zadržavajući što je moguće više informacija. PCA to postiže identifikovanjem pravaca (komponenti) duž kojih se podaci najviše razlikuju, što pomaže u otkrivanju glavnih struktura unutar podataka.




Slika 19.

Na osnovu analize komponenti, zaključili smo da izaberemo 7 komponenti, jer smanjujemo dimenzionalnost za 2 , a zadržavamo varijansu od 0.9168505961403511.

# ZAKLJUČAK

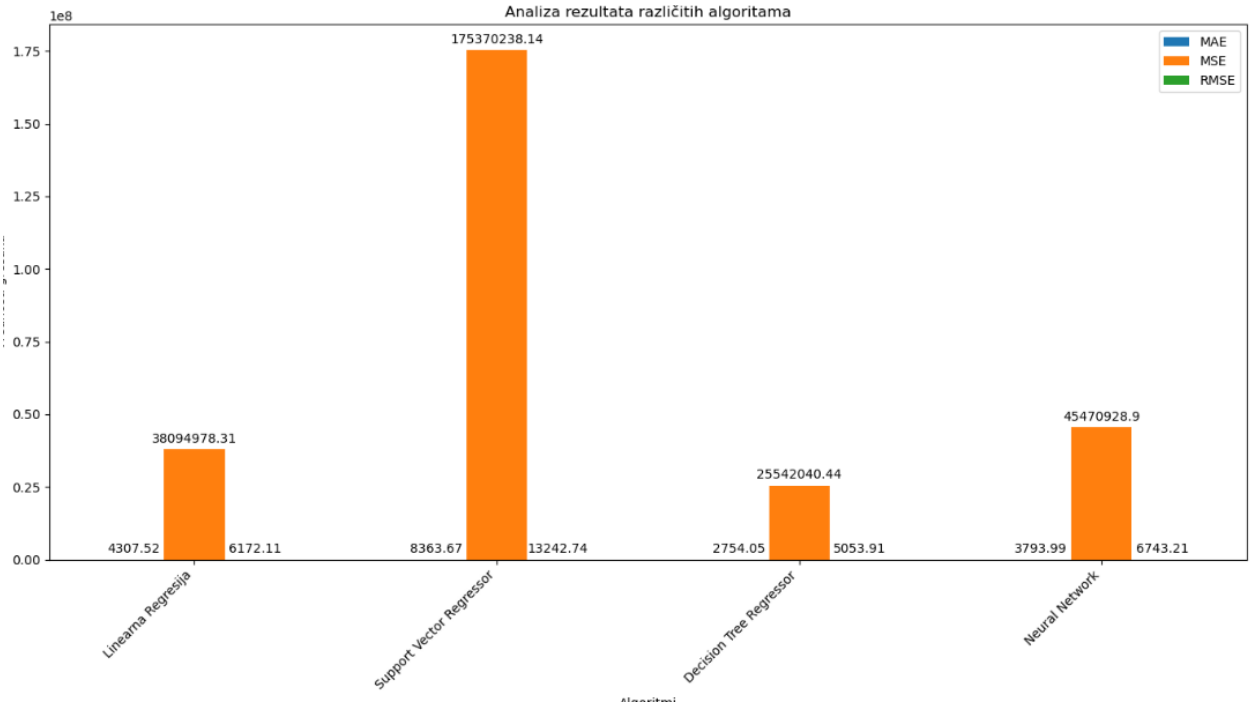
Na osnovu prikazane slike(Slika 20.), koja predstavlja analizu rezultata različitih algoritama za procjenu vrijednosti grešaka (MAE, MSE, RMSE), možemo doneti sledeće zaključke:

Linearna regresija je pokazala umjeren nivo grešaka u sve tri metrike (MAE, MSE, RMSE). Najniži MAE (4307.52) i RMSE (6172.11) sugerišu da je ovaj model relativno precizan u predikcijama u poređenju sa ostalim modelima.

Support Vector Regression (SVR) model je pokazao visoke vrijednosti za sve metrike grešaka, naročito MSE (17570302.14) i RMSE (13242.74). Ovo ukazuje na to da SVR nije adekvatan za ovakav tip podataka.

Decision Tree Regression (DTR) model pokazuje najbolje rezultate među svim analiziranim modelima sa najnižim vrijednostima grešaka za sve tri metrike (MAE: 2754.05, MSE: 25544002.44, RMSE: 5053.91). Ovo sugeriše da je DTR model najprecizniji u predikcijama za dati skup podataka.

Neuronska mreža pokazuje umjerene vrijednosti grešaka sa MAE od 3739.99, MSE od 43457029.9, i RMSE od 6743.21. Iako nije bolji od DTR modela, pokazuje bolje rezultate od Linearne regresije i SVR.




Slika 20.