Fakultet inženjerskih nauka
Univerziteta u Kragujevcu

Tema:

Regresioni algoritmi sa nadgledanim učenjem na dataset-u California Housing

student:
Nikola Mitrevski
400/2021

predmetni profesor:
dr Nenad Filipović
predmetni asistent:
Tijana Šušteršič

Kragujevac 2021.

Sadržaj:

1 Uvod

U ovom radu biće reči o nekoliko regresionih algoritama(Simple Linear Regression, Multiple Linear Regression, Polynomial Regression, Decision Tree Regression, Random Forest Regression, Support Vector Regression) koji su primenjeni na California Housing skupu podataka. Cilj primene regresionih algoritama nad ovim skupom podataka je predviđanje vrednosti cene kuće.

Housing skup podataka sadrži 20640 instanci i 9 karakteristika (longitude, latitude, housing_median_age, total_rooms, total_bedrooms, population, households, median_income, ocean_proximity) + 1 ciljna promenljiva (target) -srednja vrednost_kuće (median_house_value).

2 Simple And Multiple Linear Regression

Razlika između Simple i Multiple Linear Regression je u broju nezavisnih promenljivih. Na primer ako koristimo Simple Linear Regression broj nezavisnih promenljivih je jedan, dok za Multiple Linear Regression, koristimo dve ili više nezavisnih promenljivih.

Model koji se koristi iz sklearn biblioteke za Simple i Multiple Linear Regression je isti (LinearRegression).

U kodu ispod je vršena Multiple Linear Regression.

2.1 Uvoz potrebih biblioteka

```
#Import Necessary Libraries:
import pandas as pd
import numpy as np

from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR

from sklearn.preprocessing import PolynomialFeatures

import statsmodels.formula.api as smf

from sklearn.metrics import mean_squared_error,r2_score
from math import sqrt

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')

from sklearn.model_selection import train_test_split
```

2.2 Uvoz California Housing skupa podataka

```
df_house = pd.read_csv('housing.csv')
df_house.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

Slika 1 Tabelarni prikaz California Housing dataset-a

2.3 Odbacivanje kolone “ocean_proximity”

```
data_refine = df_house.drop('ocean_proximity', axis = 1)
data_refine.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0

Slika 2 Tabelarni prikaz California Housing dataset-a bez kolone 'ocean_proximity'

2.4 Prebrojavanje null vrednosti za svaku kolonu

Potrebno je izvršiti proveru da li kolone sadrže null vrednosti.

```
data_refine.isnull().sum()
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms      207
population            0
households            0
median_income         0
median_house_value    0
dtype: int64
```

Slika 3 Ukupan broj null vrednosti za svaku kolonu

2.5 Uklanjanje vrsta sa null vrednostima

Ukoliko neka od kolona sadrži null vrednosti, potrebno je odbaciti sve vrste sa null vrednostima, jer ne doprinose tačnosti obučavanja modela.

```
#removing NA/NaN values
data_refine = data_refine.dropna(axis = 0)
data_refine.isnull().sum()
```

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
dtype: int64
```

Slika 4 Ukupan broj null vrednosti za svaku kolonu nakon filtriranja

2.6 Podela podataka na nezavisne i zavisnu promenljivu

Ovaj skup podataka je potrebno podeliti na nezavisne promenljive(X) i zavisnu promenljivu(Y), kako bi se mogao odrediti efekat nezavisnih promenljivih na zavisnu promenljivu.

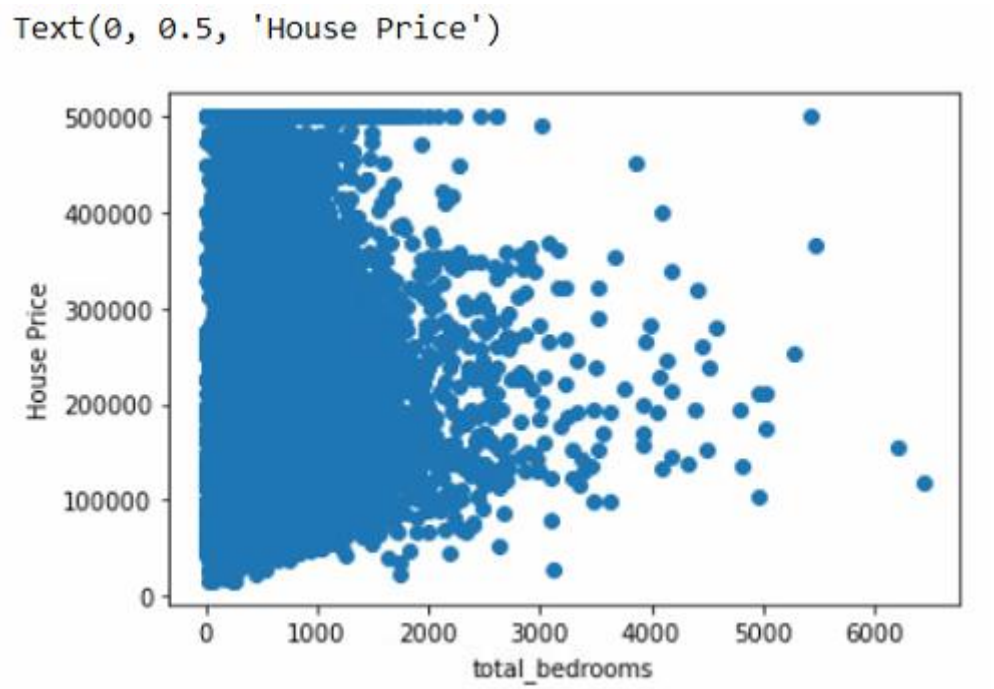
```
X = data_refine.drop('median_house_value', axis = 1)
Y = data_refine['median_house_value']

print(data_refine.shape)
print(X.shape)
print(Y.shape)
```

```
(20433, 9)
(20433, 8)
(20433,)
```

Slika 5 Podela podataka na zavisnu i nezavisne promenljive

2.7 Grafički prikaz zavisnosti zavisne promenljive(median_house_value) od nezavisne promenljive(total_bedrooms)

```
plt.scatter(X['total_bedrooms'], Y)
plt.xlabel('total_bedrooms')
plt.ylabel('House Price')
```



Slika 6 Grafički prikaz zavisnosti zavisne promenljive(median_house_value) od nezavisne promenljive(total_bedrooms)

2.8 Podela podataka na skup za treniranje i na skup za testiranje modela

Da bi se model mogao naučiti, potrebno je podeliti podatke na skup podataka za treniranje i na skup podataka za testiranje modela.

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25)
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(15324, 8)
(15324,)
(5109, 8)
(5109,)
```

Slika 7 Podela podataka na skup za treniranje i na skup za testiranje modela

2.9 Kreiranje i treniranje modela

```
LR = LinearRegression()
LR.fit(X_train, Y_train)
```

2.10 Testiranje modela

```
predict = LR.predict(X_test)
print('Predicted Value :',predict[3])
print('Actual Value :',Y_test.values[3])
```

```
Predicted Value : 170108.03250666987
Actual Value : 112500.0
```

Slika 8 Predviđena i stvarna cena

2.11 Srednja kvadratna greška

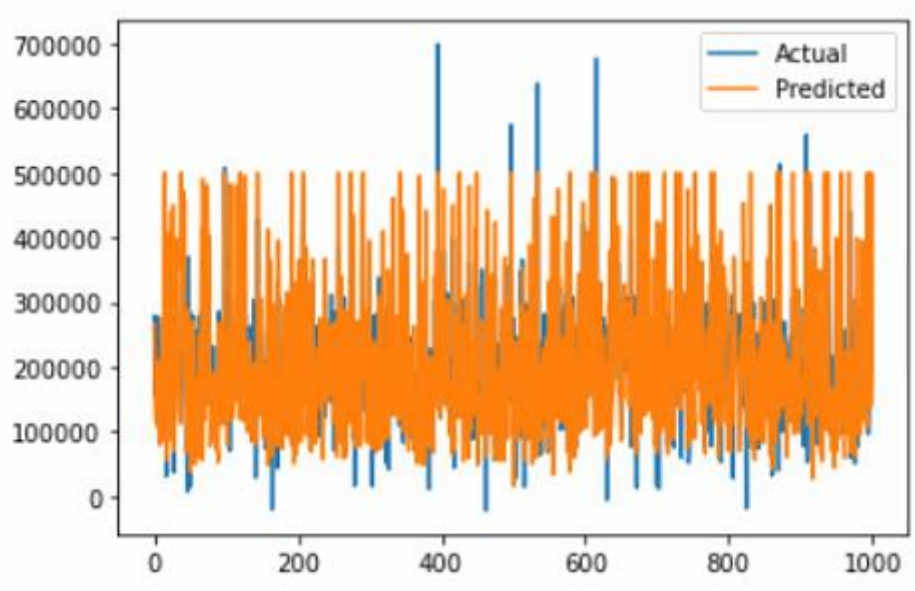
```
print(sqrt(mean_squared_error(Y_test, predict)))
print((r2_score(Y_test, predict)))
```

```
70567.51989526374
0.6180819574096446
```

Slika 9 Srednja kvadratna greška

2.12 Grafički prikaz stvarnih i predviđenih izlaza

```
gr = pd.DataFrame({'Predicted':predict,'Actual':Y_test})
gr = gr.reset_index()
gr = gr.drop(['index'],axis=1)
plt.plot(gr[:1000])
plt.legend(['Actual','Predicted'])
#gr.plot.bar();
```



Slika 10 Grafički prikaz stvarnih i predviđenih izlaza

3 Polynomial Regression

Razlika između Simple i Polynomial Regression je u skupu podataka za obučavanje i testiranje modela. Kod Polynomial Regression potrebno je da podaci budu u polinomskom obliku.

Model koji se koristi iz sklearn biblioteke za Simple i Polynomial Regression je isti (LinearRegression).

3.1 Generisanje polinomskih karakteristika

```
pft = PolynomialFeatures(degree = 2)
X_poly = pft.fit_transform(X)
```

3.2 Podela podataka na skup za treniranje i na skup za testiranje modela

```
X_train,X_test,Y_train,Y_test = train_test_split(X_poly,Y,test_size=0.25)
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(15324, 45)
(15324,)
(5109, 45)
(5109,)
```

Slika 11 Podela podataka na skup za treniranje i na skup za testiranje modela

3.3 Kreiranje i treniranje modela

```
LR = LinearRegression()
LR.fit(X_train, Y_train)
```

3.4 Testiranje modela

```
predict = LR.predict(X_test)
print('Predicted Value :',predict[3])
print('Actual Value :',Y_test.values[3])
```

```
Predicted Value : 64117.25447650626
Actual Value : 81800.0
```

Slika 12 Predviđena i stvarna vrednost

3.5 Srednja kvadratna greška

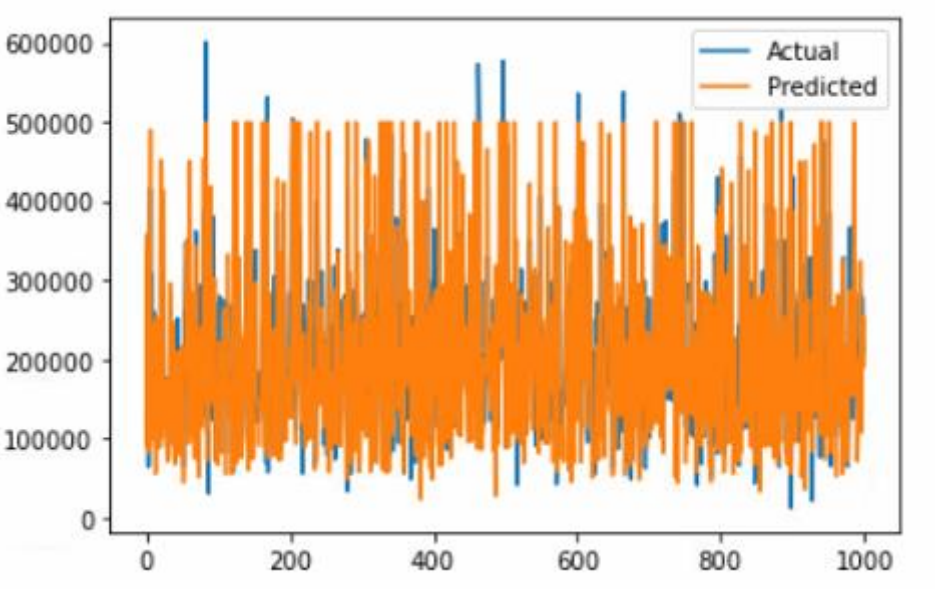
```
print(sqrt(mean_squared_error(Y_test, predict)))
print((r2_score(Y_test, predict)))
```

```
62322.66426980197
0.7085771025180712
```

Slika 13 Srednja kvadratna greška

3.6 Grafički prikaz stvarnih i predviđenih izlaza

```
gr = pd.DataFrame({'Predicted':predict,'Actual':Y_test})
gr = gr.reset_index()
gr = gr.drop(['index'],axis=1)
plt.plot(gr[:1000])
plt.legend(['Actual','Predicted'])
#gr.plot.bar();
```



Slika 14 Grafički prikaz stvarnih i predviđenih izlaza

4 Decision Tree Regression

4.1 Kreiranje i treniranje modela

```
dtreg=DecisionTreeRegressor()
dtreg.fit(X_train, Y_train)
```

4.2 Testiranje modela

```
predict = dtreg.predict(X_test)
print('Predicted Value :',predict[3])
print('Actual Value :',Y_test.values[3])
```

```
Predicted Value : 101200.0
Actual Value : 155900.0
```

Slika 15 Predviđena i stvarna cena

4.3 Srednja kvadratna greška

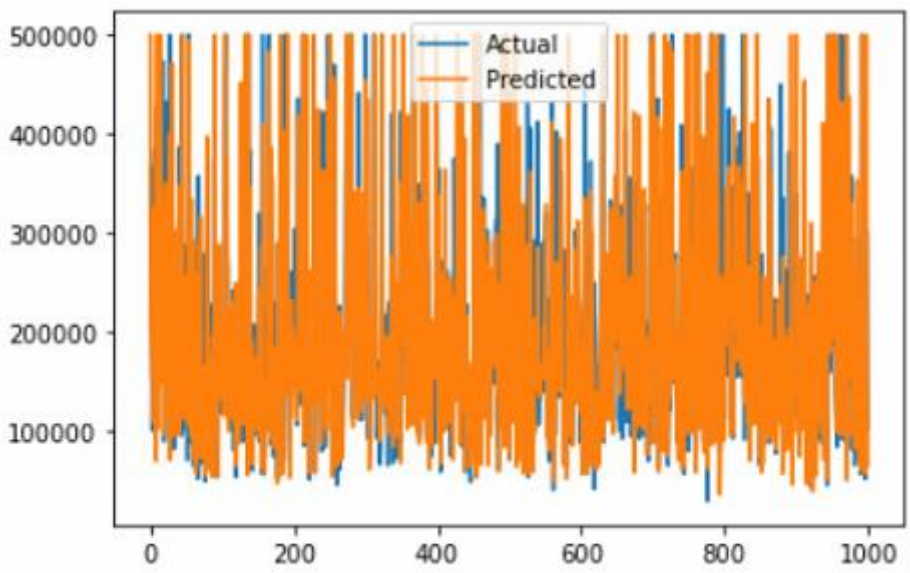
```
print(sqrt(mean_squared_error(Y_test, predict)))
print((r2_score(Y_test, predict)))
```

```
67343.26366993465
0.6523451445433155
```

Slika 16 Srednja kvadratna greška

4.4 Grafički prikaz stvarnih i predviđenih izlaza

```
gr = pd.DataFrame({'Predicted':predict,'Actual':Y_test})
gr = gr.reset_index()
gr = gr.drop(['index'],axis=1)
plt.plot(gr[:1000])
plt.legend(['Actual','Predicted'])
#gr.plot.bar();
```



Slika 17 Grafički prikaz stvarnih i predviđenih izlaza

5 Random Forest Regression

5.1 Kreiranje i treniranje modela

```
rfreg=RandomForestRegressor()
rfreg.fit(X_train,Y_train)
```

5.2 Testiranje modela

```
predict = rfreg.predict(X_test)
print('Predicted Value :',predict[3])
print('Actual Value :',Y_test.values[3])
```

```
Predicted Value : 273721.04
Actual Value : 233300.0
```

Slika 18 Predviđena i stvarna cena

5.3 Srednja kvadratna greška

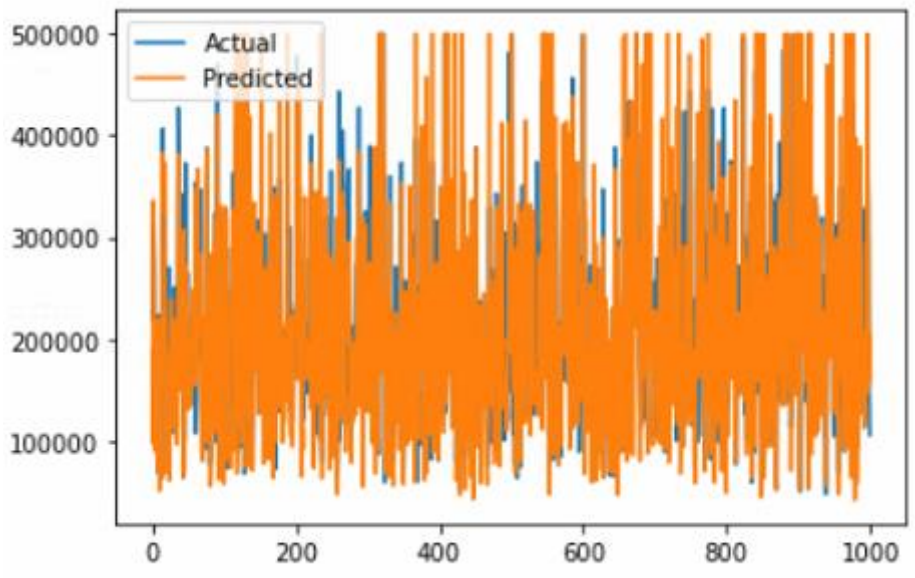
```
print(sqrt(mean_squared_error(Y_test, predict)))
print((r2_score(Y_test, predict)))
```

```
48700.52257000837
0.8162846245389843
```

Slika 19 Srednja kvadratna greška

5.4 Grafički prikaz stvarnih i predviđenih izlaza

```
gr = pd.DataFrame({'Predicted':predict,'Actual':Y_test})
gr = gr.reset_index()
gr = gr.drop(['index'],axis=1)
plt.plot(gr[:1000])
plt.legend(['Actual','Predicted'])
#gr.plot.bar();
```



Slika 20 Grafički prikaz stvarnih i predviđenih izlaza

6 Support Vector Regression

6.1 Kreiranje i treniranje modela

```
svr=SVR()
svr.fit(X_train,Y_train)
```

6.2 Testiranje modela

```
predict = svr.predict(X_test)
print('Predicted Value :',predict[3])
print('Actual Value :',Y_test.values[3])
```

```
Predicted Value : 180578.01638398194
Actual Value : 139900.0
```

Slika 21 Predviđena i stvarna cena

6.3 Srednja kvadratna greška

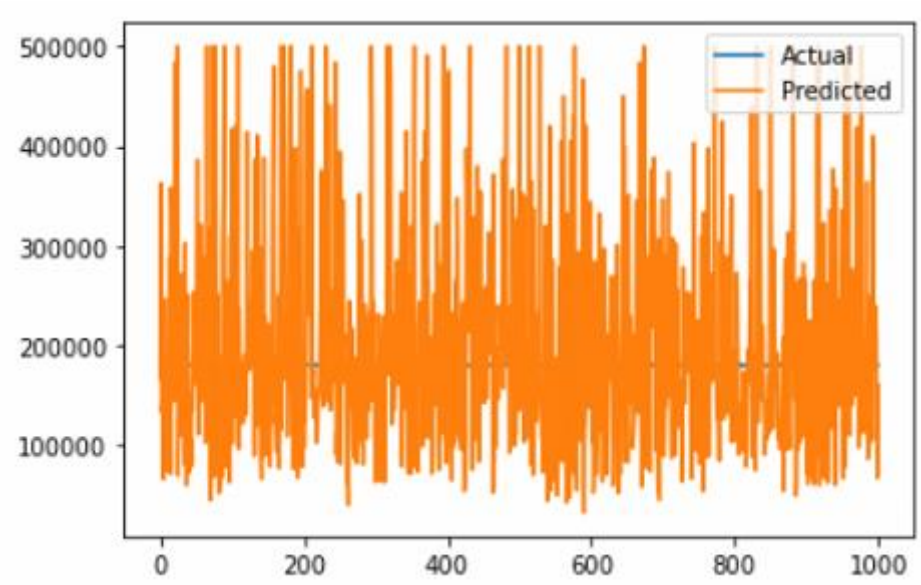
```
print(sqrt(mean_squared_error(Y_test, predict)))
print((r2_score(Y_test, predict)))
```

```
117278.6798810293
-0.0495335772101575
```

Slika 22 Srednja kvadratna greška

6.4 Grafički prikaz stvarnih i predviđenih izlaza

```
gr = pd.DataFrame({'Predicted':predict,'Actual':Y_test})
gr = gr.reset_index()
gr = gr.drop(['index'],axis=1)
plt.plot(gr[:1000])
plt.legend(['Actual','Predicted'])
#gr.plot.bar();
```



Slika 23 Grafički prikaz stvarnih i predviđenih izlaza

7 Literatura

[1] Kaggle - California Housing Data, link: https://www.kaggle.com/anjutyagi/california-housing-data#Data-preparation-for-Machine-Learning-algorithms, 13.12.2021(22:25)

[2] W3schools - Multiple Regression, link: https://www.w3schools.com/python/python_ml_multiple_regression.asp, 13.12.2021(22:26)

[3] Medium - Make your own model to predict house prices in Python, link: https://medium.com/@kumar.bits009/make-your-own-model-to-predict-house-prices-in-python-ad843aee1e2, 13.12.2021(22:26)