Project on Crime Rate Prediction using Machine Learning Models

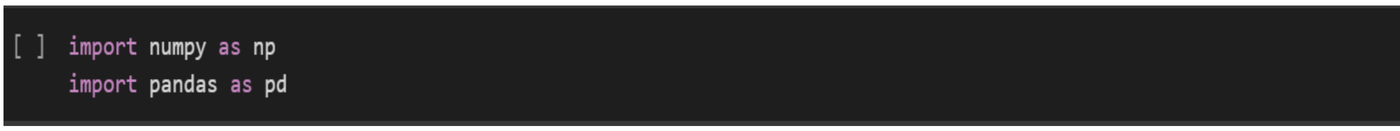
- Introduction: This is a Data Science project, where we'll be predicting the crime rate using machine learning models with the help of csv-dataset provided which contains of 99 standard metropolitan areas in the US. The data set provides information on 10 variables for each area for the period 1976-1977. The areas have been divided into 4 geographic regions: 1=North-East, 2=North-Central, 3=South, 4=West.
  In this project I have used different kinds of model for having the best accuracy based on the situations given.

- The project is sub-divided into the following sections. These are:

  - Loading necessary libraries such as 'numpy', 'pandas', 'sklearn.model' etc.
  - Loading Dataset as a CSV file.
  - Splitting the data set into independent & dependent sets.
  - Visualization of data using Tableau.
  - Training the model with the help of fit ().
  - Predicting the train model for output which will be compared with y_test.
  - Finally, will check the accuracy of the model for predicting the chances of crime rate in a particular area.

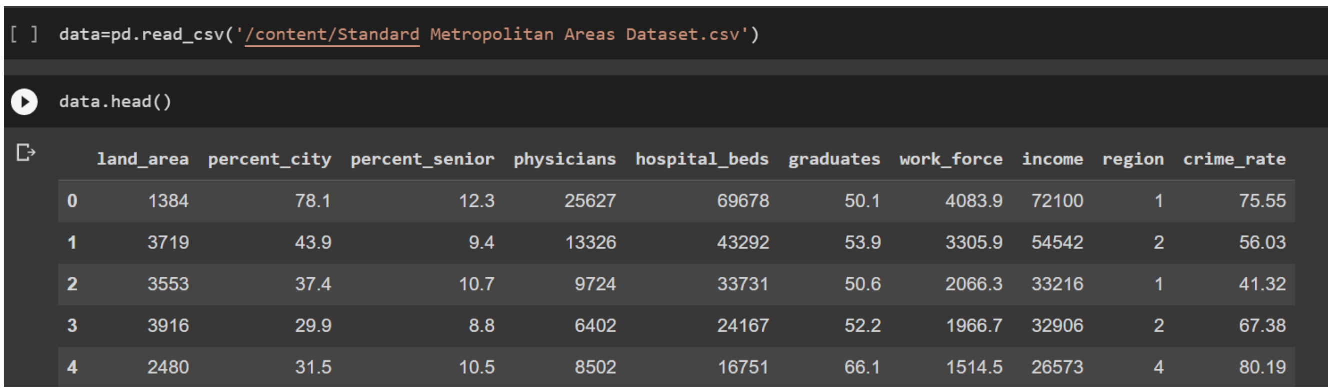
- Steps for creating a model:

- Step-1: Importing numpy as np & pandas as pd for loading and reading the data-set.



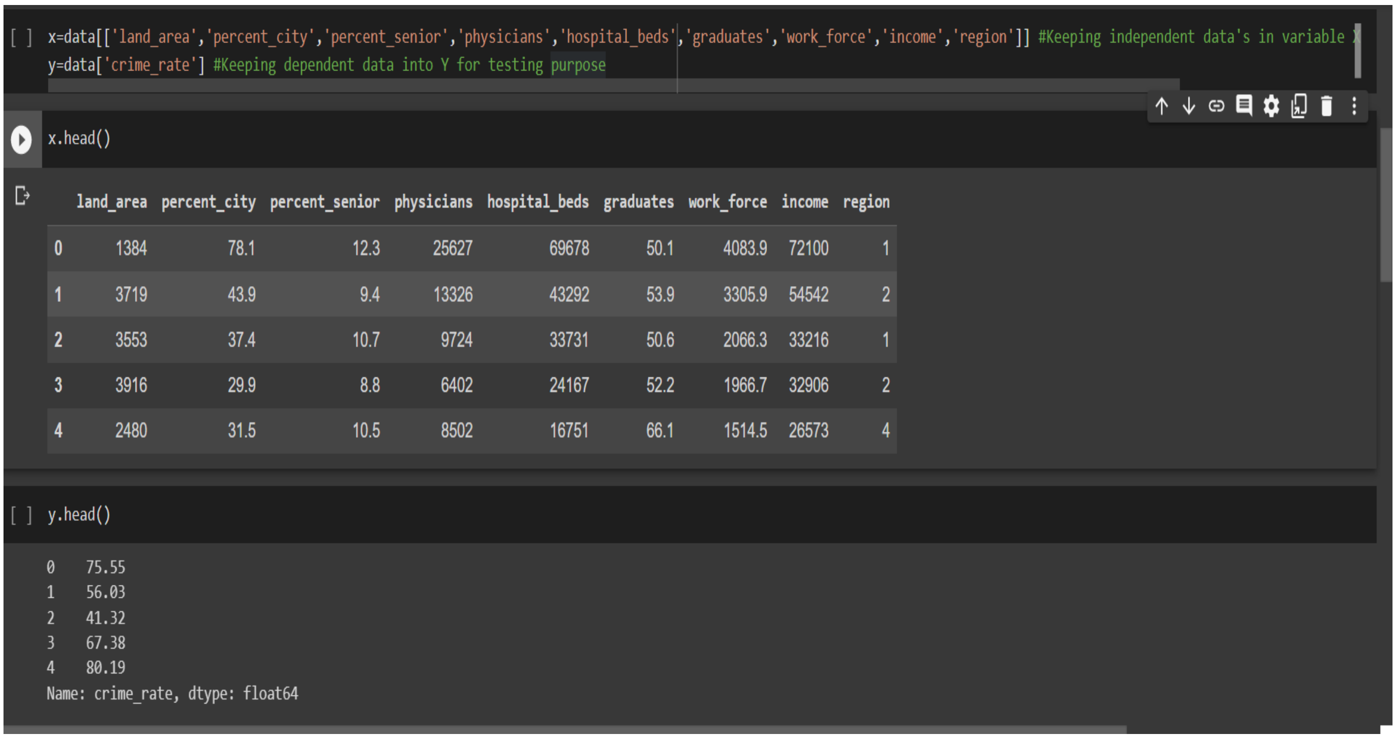
```
[ ] import numpy as np
    import pandas as pd
```

- Step-2: Loading the csv-dataset in the variable name 'data' for analysing with the help of data.head()function.



```
[ ] data=pd.read_csv('/content/Standard Metropolitan Areas Dataset.csv')

    data.head()

     land_area  percent_city  percent_senior  physicians  hospital_beds  graduates  work_force  income  region  crime_rate
  0       1384          78.1            12.3       25627          69678       50.1      4083.9   72100       1       75.55
  1       3719          43.9             9.4       13326          43292       53.9      3305.9   54542       2       56.03
  2       3553          37.4            10.7        9724          33731       50.6      2066.3   33216       1       41.32
  3       3916          29.9             8.8        6402          24167       52.2      1966.7   32906       2       67.38
  4       2480          31.5            10.5        8502          16751       66.1      1514.5   26573       4       80.19
```

➢ <u>Step-3</u>: Splitting the data into dependent & independent data.



```
[ ] x=data[['land_area','percent_city','percent_senior','physicians','hospital_beds','graduates','work_force','income','region']] #Keeping independent data's in variable
    y=data['crime_rate'] #Keeping dependent data into Y for testing purpose
```

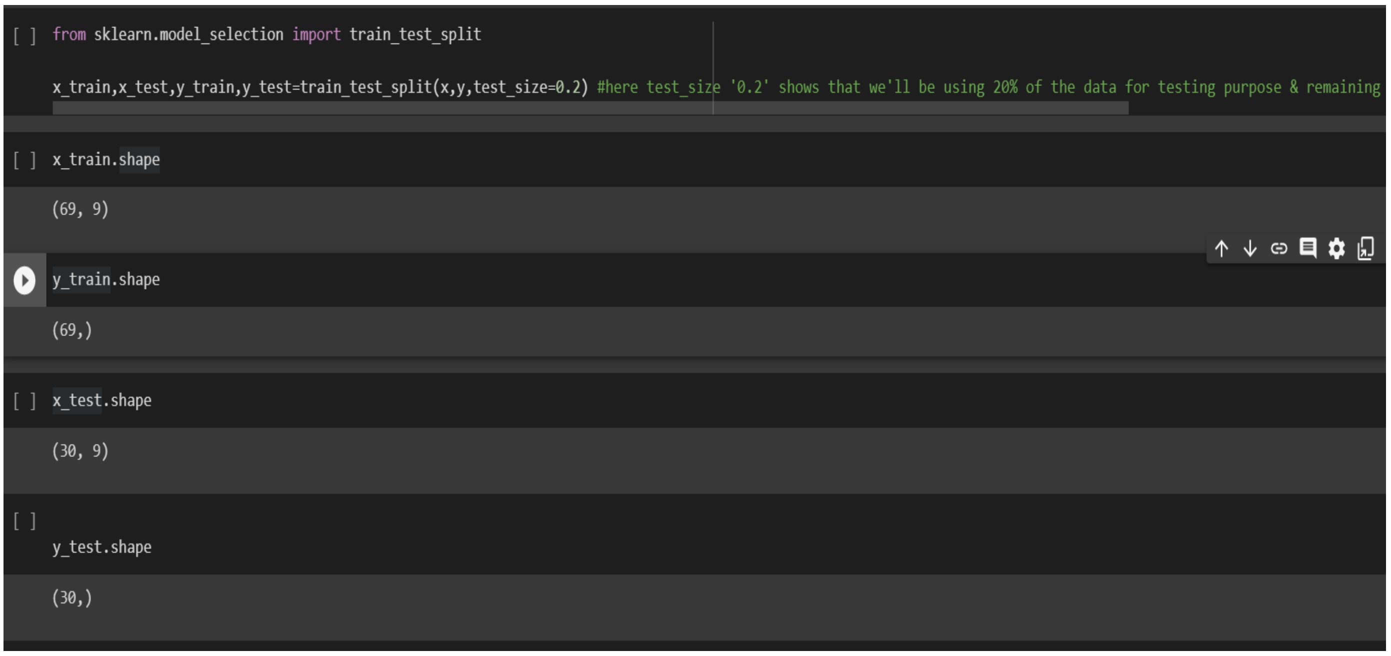
```
x.head()
```

```
   land_area  percent_city  percent_senior  physicians  hospital_beds  graduates  work_force  income  region
0       1384          78.1            12.3       25627          69678       50.1      4083.9   72100       1
1       3719          43.9             9.4       13326          43292       53.9      3305.9   54542       2
2       3553          37.4            10.7        9724          33731       50.6      2066.3   33216       1
3       3916          29.9             8.8        6402          24167       52.2      1966.7   32906       2
4       2480          31.5            10.5        8502          16751       66.1      1514.5   26573       4
```

```
[ ] y.head()
```

```
0    75.55
1    56.03
2    41.32
3    67.38
4    80.19
Name: crime_rate, dtype: float64
```

➢ <u>Step-4</u>: Importing train_test_split from sklearn.model library for splitting the data into train and test sets.



```
[ ] from sklearn.model_selection import train_test_split

    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2) #here test_size '0.2' shows that we'll be using 20% of the data for testing purpose & remaining
```

```
[ ] x_train.shape
```

```
(69, 9)
```

```
y_train.shape
```

```
(69,)
```

```
[ ] x_test.shape
```

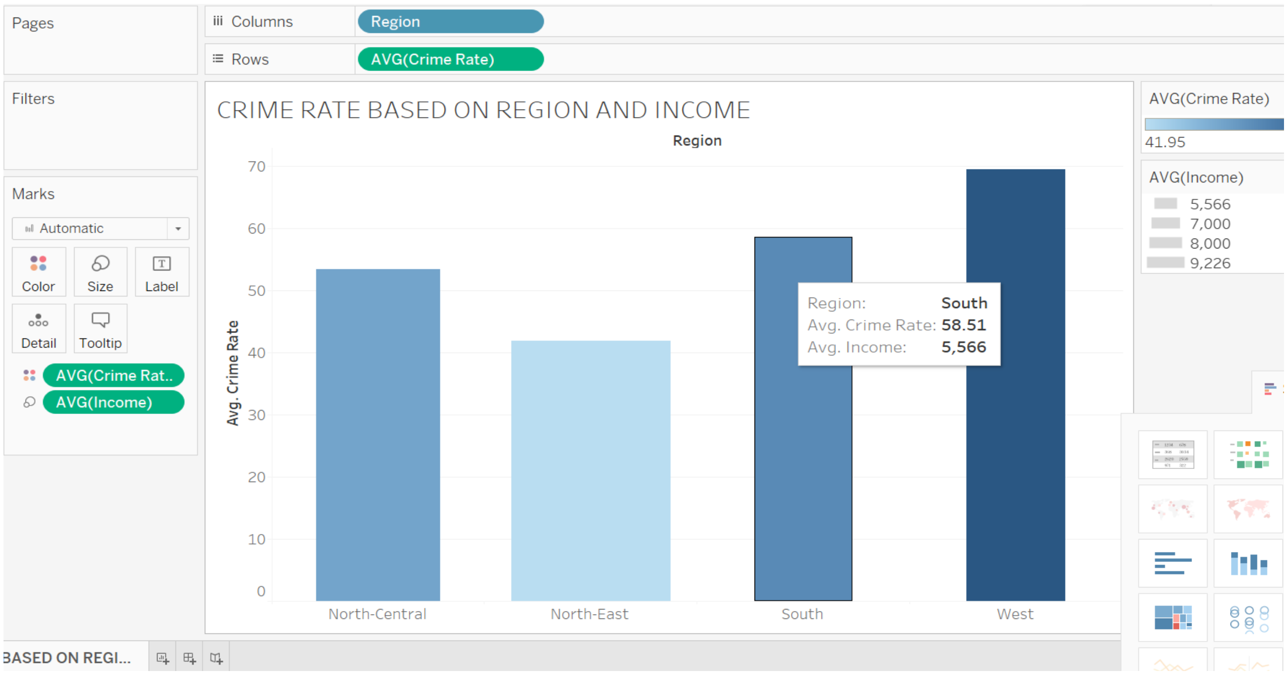
```
(30, 9)
```

```
[ ]
    y_test.shape
```

```
(30,)
```

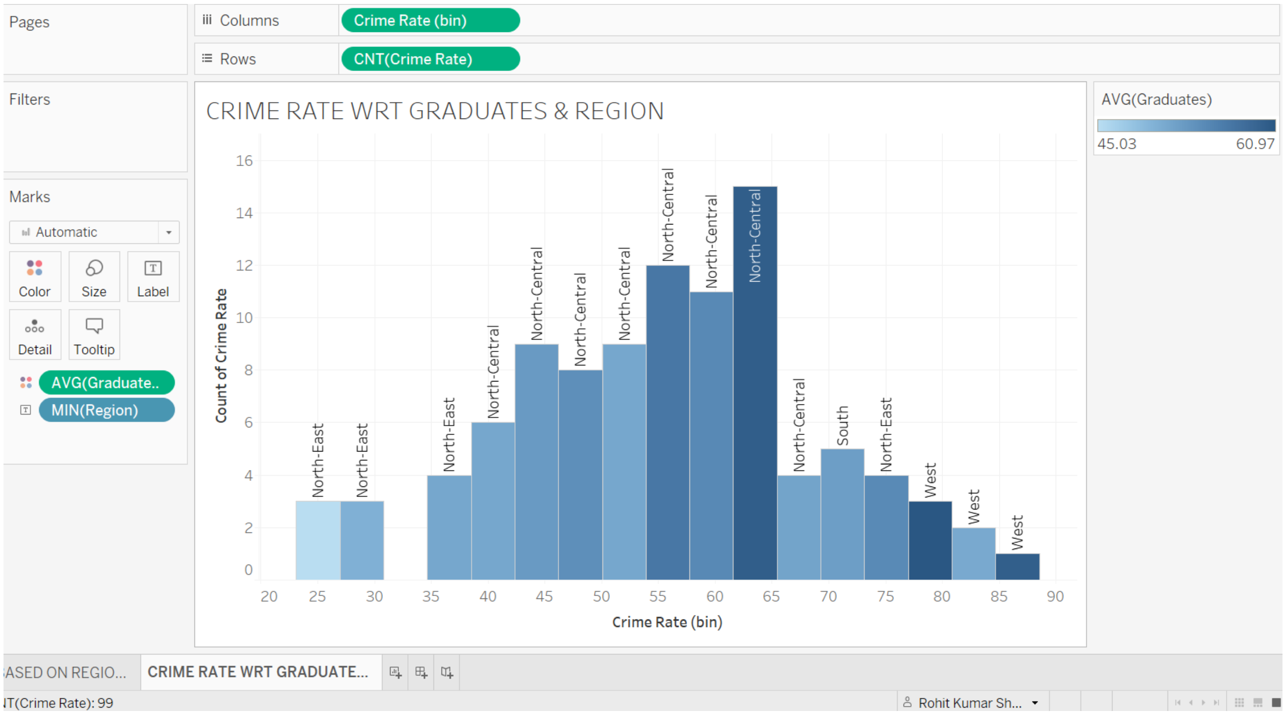
➢

➢ Step-5: Visualizing the crime rate prediction using Tableau on different aspects.



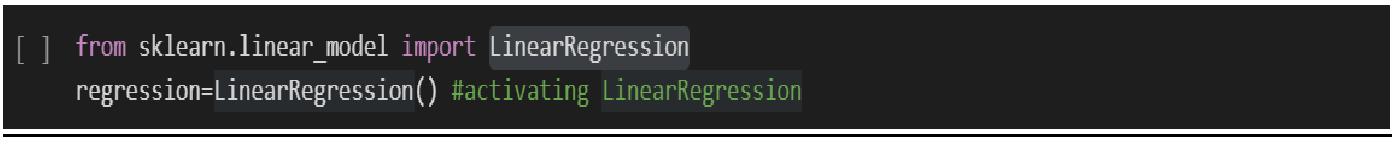
Region: South
Avg. Crime Rate: 58.51
Avg. Income: 5,566

-In the above Visualization we can see that person's whose avg. income is between $5000-$7000 are more into crimes & those who earns more than $7000 are comparatively lesser into crimes.



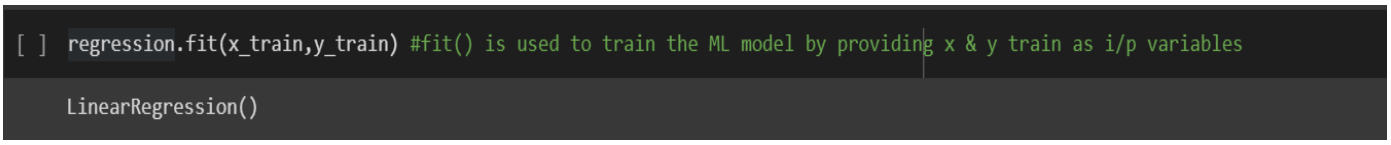
-In the above Visualization we can see that irrespective of region the more the people are graduate the more they are into crimes.

- Step-6: Importing LinearRegression from sklearn.model library & then activating it by storing into the variable name-regression.



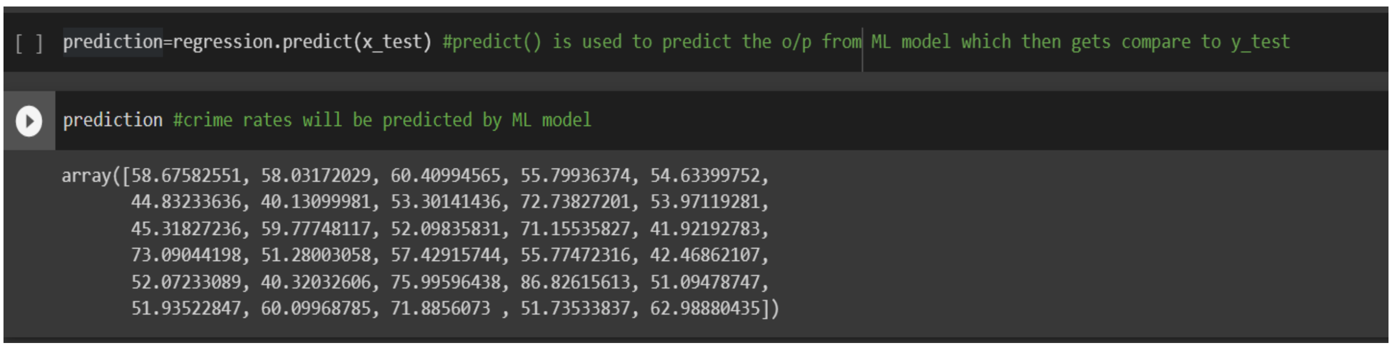
```
[ ] from sklearn.linear_model import LinearRegression
    regression=LinearRegression() #activating LinearRegression
```

- Step-7: Training the model using fit() by providing x & y as a input variables.



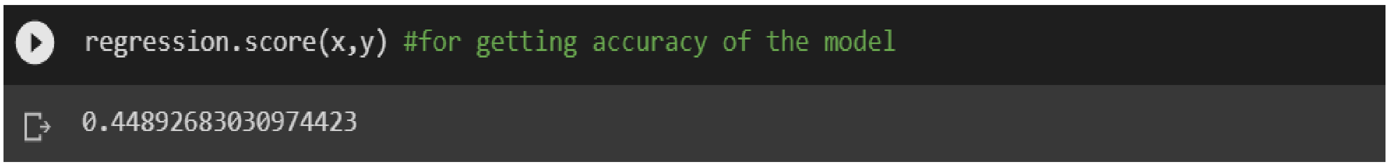
```
[ ] regression.fit(x_train,y_train) #fit() is used to train the ML model by providing x & y train as i/p variables

    LinearRegression()
```

- Step-8: Used predict() for predict the output which will gets compared to y_test. Then predicted the crime rates with the help of model.



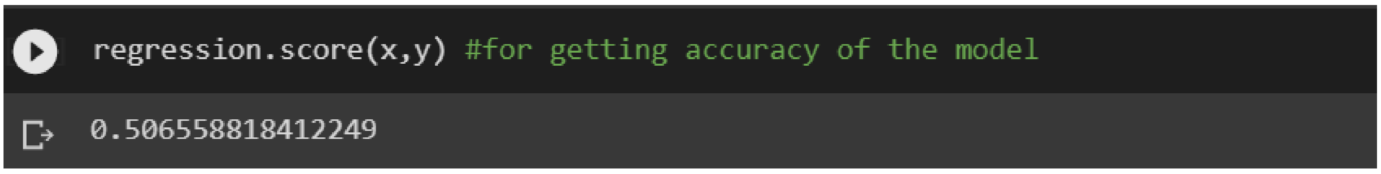
```
[ ] prediction=regression.predict(x_test) #predict() is used to predict the o/p from ML model which then gets compare to y_test

    prediction #crime rates will be predicted by ML model

    array([58.67582551, 58.03172029, 60.40994565, 55.79936374, 54.63399752,
           44.83233636, 40.13099981, 53.30141436, 72.73827201, 53.97119281,
           45.31827236, 59.77748117, 52.09835831, 71.15535827, 41.92192783,
           73.09044198, 51.28003058, 57.42915744, 55.77472316, 42.46862107,
           52.07233089, 40.32032606, 75.99596438, 86.82615613, 51.09478747,
           51.93522847, 60.09968785, 71.8856073 , 51.73533837, 62.98880435])
```

- Step-9: Checking the accuracy of the model using '.score()'



```
regression.score(x,y) #for getting accuracy of the model

0.44892683030974423
```

-Above accuracy was obtained when the test_size was 20%

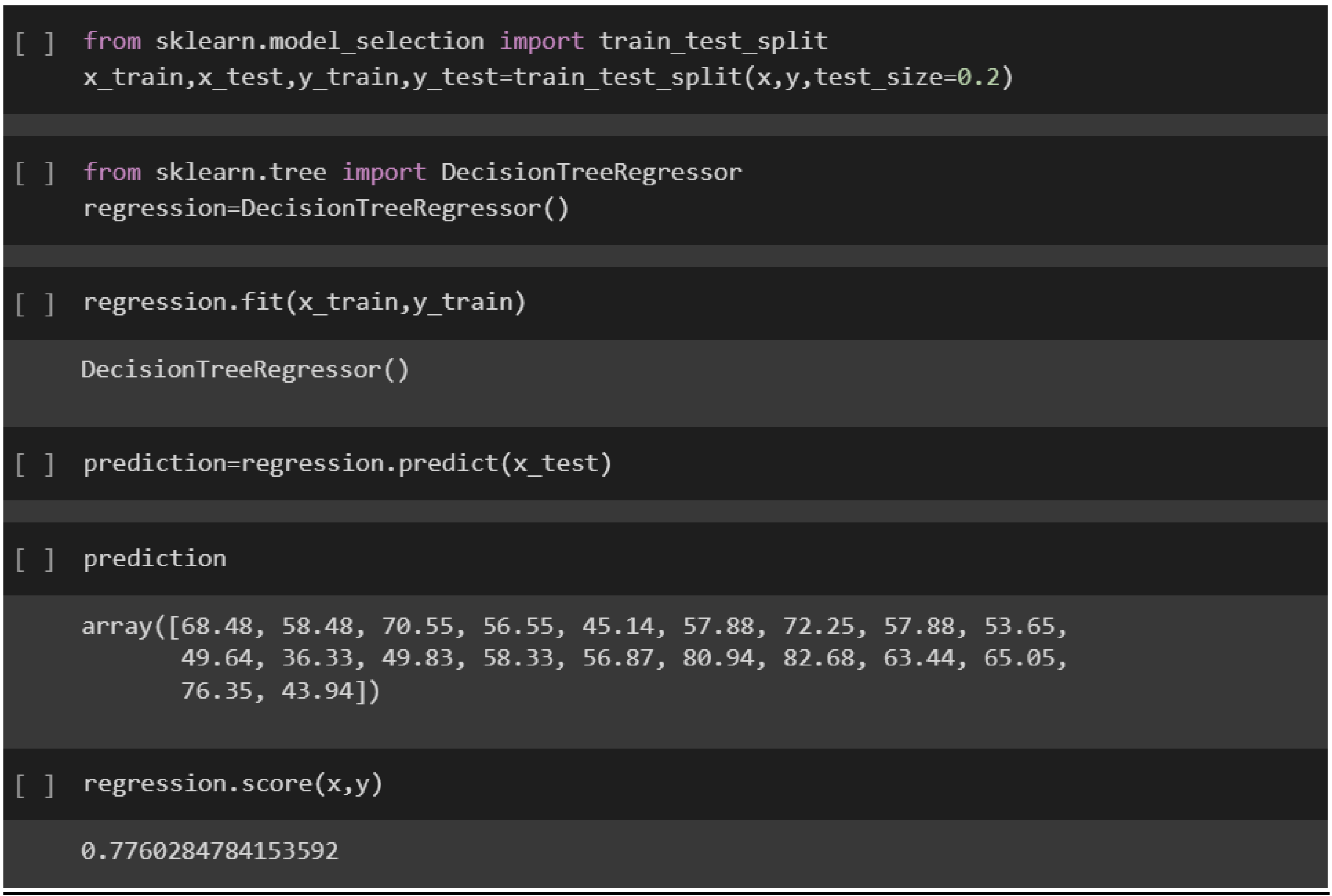


```
regression.score(x,y) #for getting accuracy of the model

0.506558818412249
```

-Above accuracy was obtained when the test_size was 10%

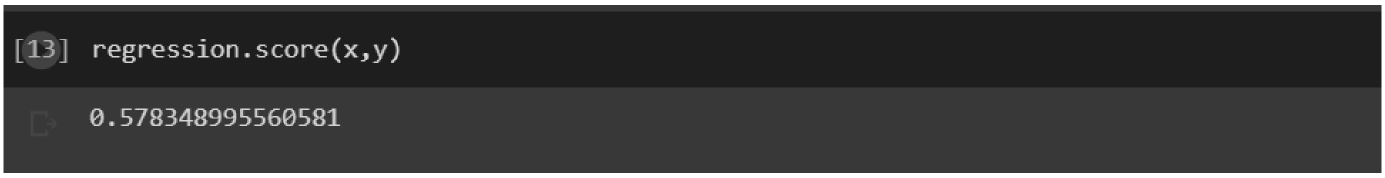
➢ As we can see in the above model the accuracies were not up to the mark or we can say it is quite not good. So, I have tried different kinds of model to get the higher accuracy supporting my model.

➢ Model using Decision Tree-



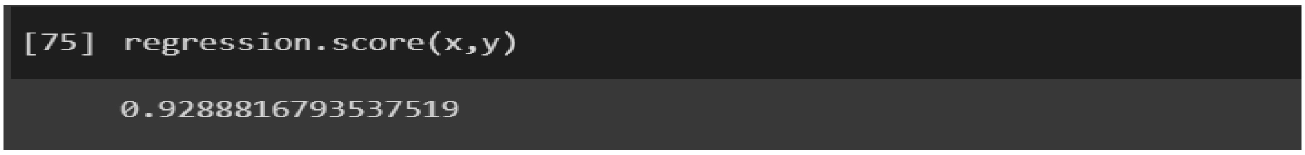
```
[ ]  from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

[ ]  from sklearn.tree import DecisionTreeRegressor
     regression=DecisionTreeRegressor()

[ ]  regression.fit(x_train,y_train)

     DecisionTreeRegressor()

[ ]  prediction=regression.predict(x_test)

[ ]  prediction

     array([68.48, 58.48, 70.55, 56.55, 45.14, 57.88, 72.25, 57.88, 53.65,
            49.64, 36.33, 49.83, 58.33, 56.87, 80.94, 82.68, 63.44, 65.05,
            76.35, 43.94])

[ ]  regression.score(x,y)

     0.7760284784153592
```

- Accuracy obtained 77% when the test_size was 20%



```
[13]  regression.score(x,y)

      0.578348995560581
```

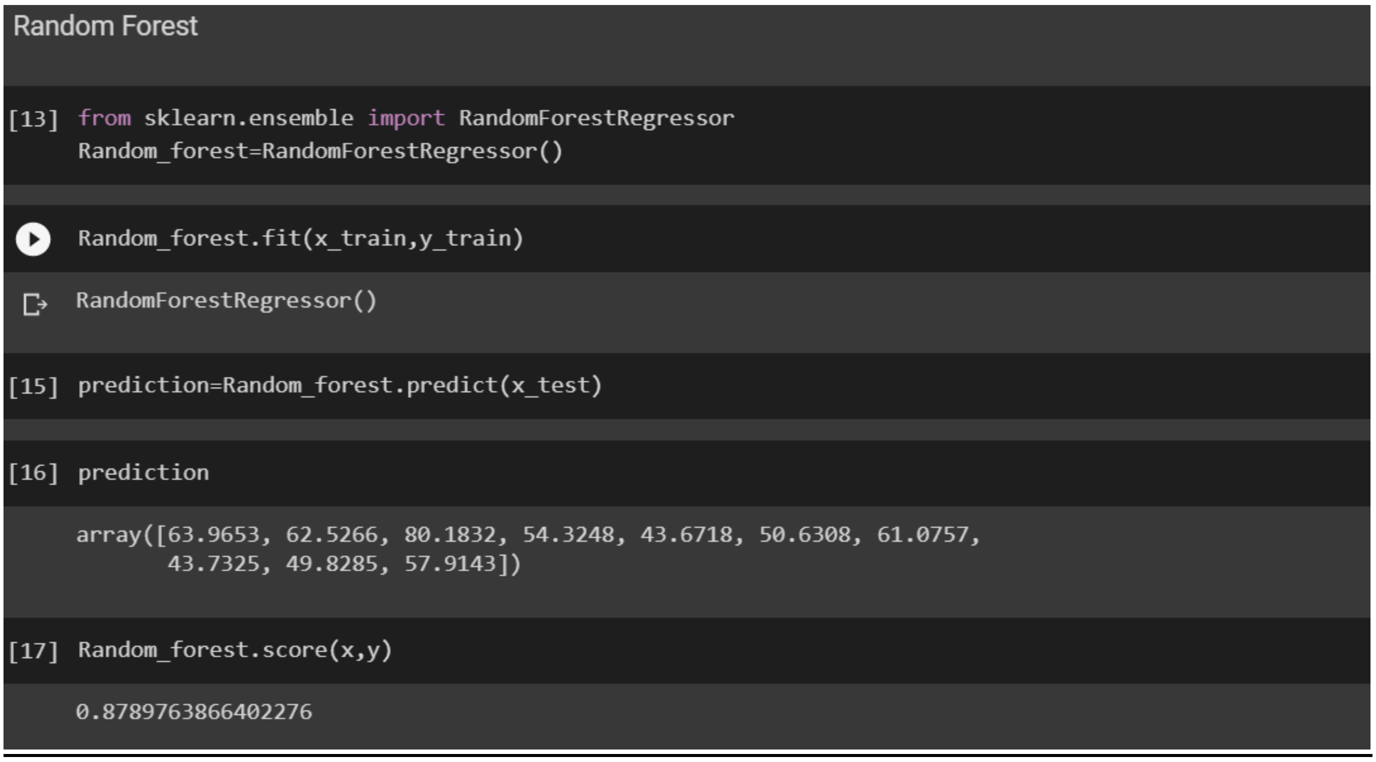
- Accuracy obtained 57% when the test_size was 30%



```
[75]  regression.score(x,y)

      0.9288816793537519
```

- Accuracy obtained 92% when the test_size was 10%

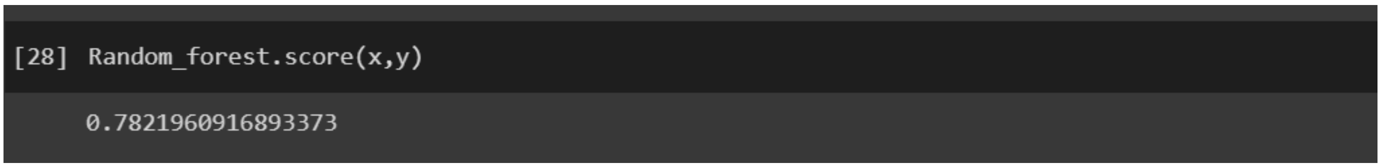
➢ <u>Model using Random Forest-</u>



Random Forest

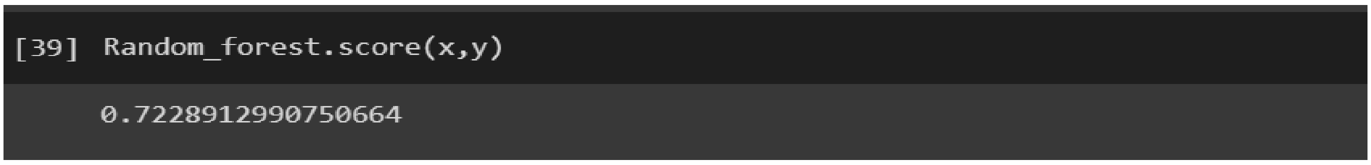
```
[13] from sklearn.ensemble import RandomForestRegressor
     Random_forest=RandomForestRegressor()

     Random_forest.fit(x_train,y_train)

     RandomForestRegressor()

[15] prediction=Random_forest.predict(x_test)

[16] prediction

     array([63.9653, 62.5266, 80.1832, 54.3248, 43.6718, 50.6308, 61.0757,
            43.7325, 49.8285, 57.9143])

[17] Random_forest.score(x,y)

     0.8789763866402276
```

- Accuracy obtained 87% when the test_size was 10%



```
[28] Random_forest.score(x,y)

     0.7821960916893373
```

- Accuracy obtained 78% when the test_size was 20%



```
[39] Random_forest.score(x,y)

     0.7228912990750664
```

- Accuracy obtained 72% when the test_size was 30%

➢ <u>Conclusion:</u> From the above models we can see that the accuracy found in <u>RandomForest</u> was more accurate than the other models. Also, accuracy found using <u>LinearRegression</u> was much lesser than any other model.