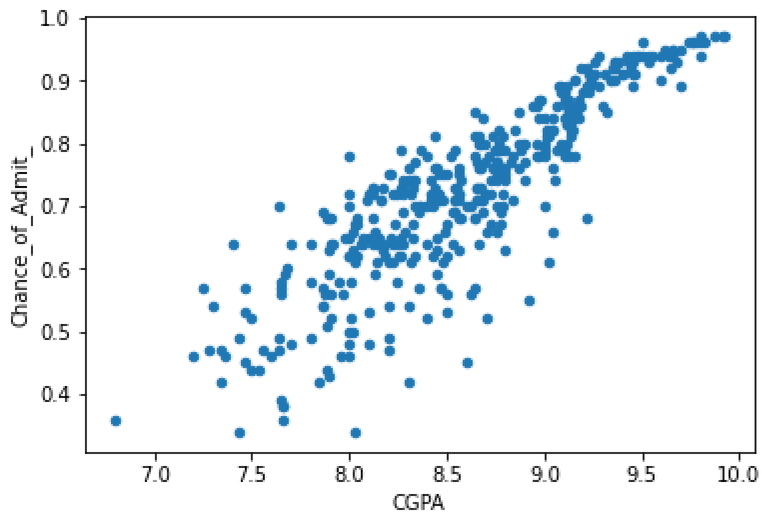
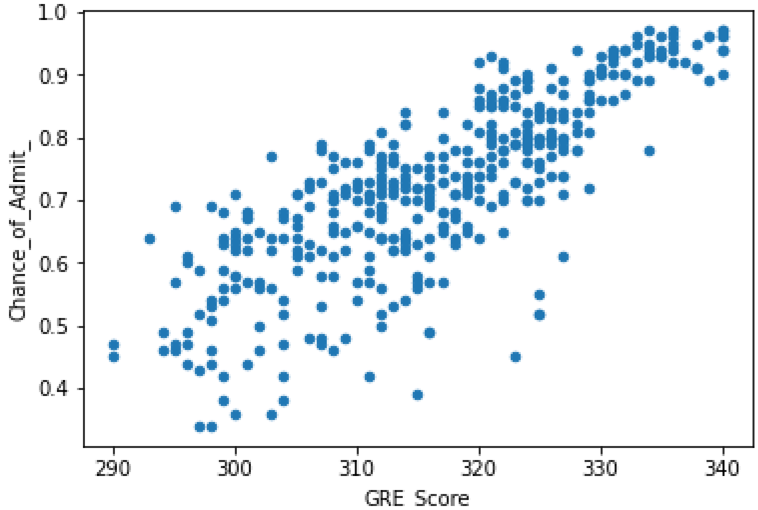
ASSIGNMENT-4: Introduction To Machine Learning

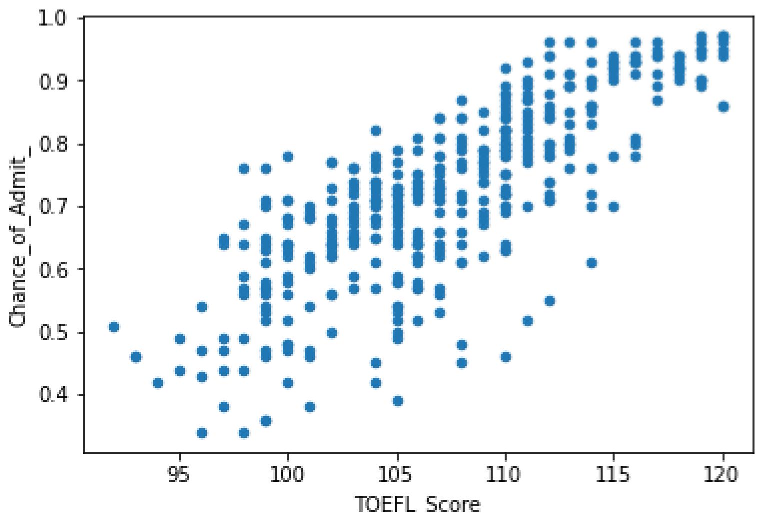
Abhishek Pai Angle, Aum Jain, Dev Desai, Madhumitha S, Tanay Sharma

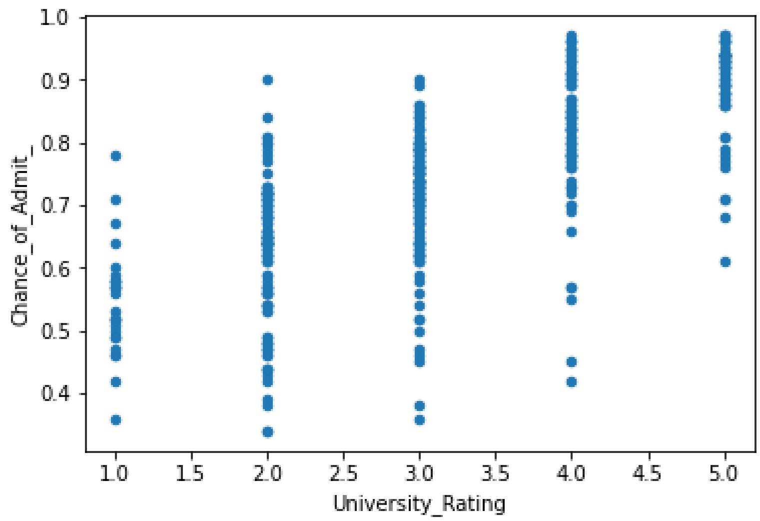
April 17, 2020

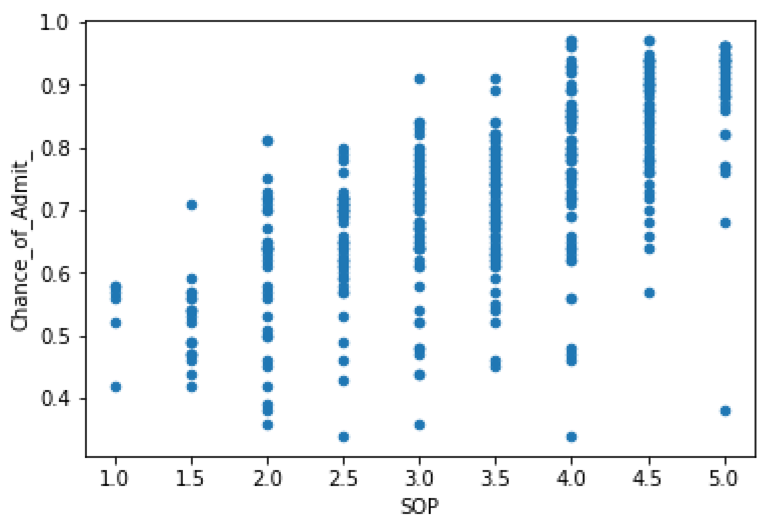
1. **Part 1(Predicting Chances of Admission from the CGPA acquired):**
    a. **Coming up with a model:** We had to decide as to what model could fit the data best, so we plotted the *Chance of getting Admitted* vs *CGPA* acquired. We observed that chances of getting admission increase with increase in CGPA, and that it was possible to approximate the distribution as a linear distribution.

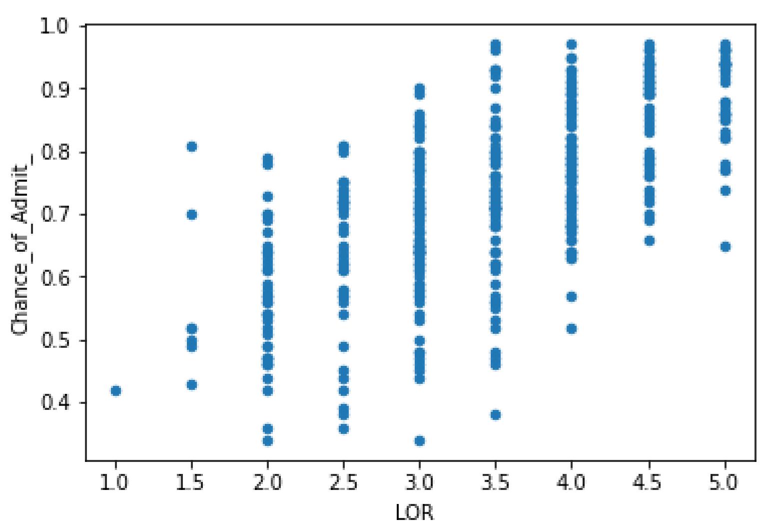


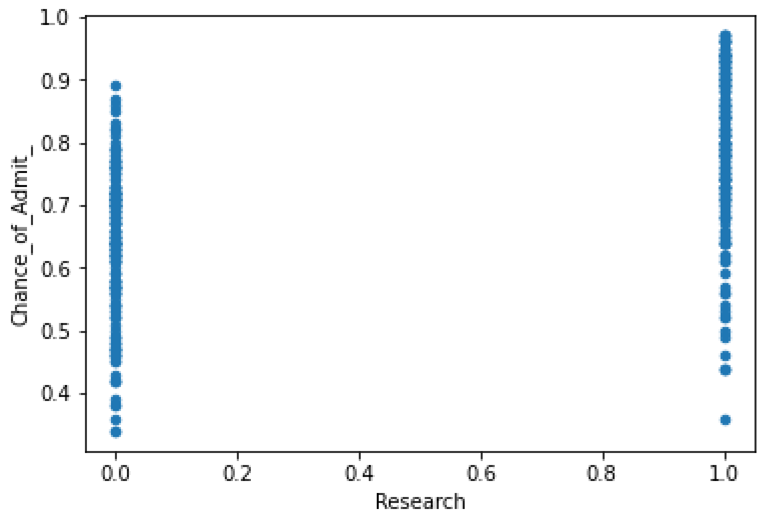

Chance_of_Admit_
1.0
0.9
0.8
0.7
0.6
0.5
0.4
290
300
310
320
330
340
GRE_Score


Chance_of_Admit_
1.0
0.9
0.8
0.7
0.6
0.5
0.4
95
100
105
110
115
120
TOEFL_Score


Chance_of_Admit_
University_Rating


Chance_of_Admit_
SOP


Chance_of_Admit_
LOR_



We also observe that plotting Chance of Admission v/s any other parameter does not yield much important insights, except for the fact that there is a general increase in chance of admission with the test scores. So we have an intuition that CGPA may be the most significant factor in getting admitted.

b. Building the model: We came up with a multilinear regression model to predict the chance of admission given other parameters.

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


dataset= pd.read_csv('/content/drive/My Drive/Intelligent
Agents/Assignment 4/Admission_Predict.csv')
x= dataset.iloc[:,1:8].values
y=dataset.iloc[:,-1].values
print(dataset)

import statsmodels.api as sm
x=np.append(arr=np.ones((400,1)).astype(int),values=x,axis=1)
#adding a column of ones since we use equation y=b0 +b1x1
+b2x2 +... in which x0 whose coefficient is b0 is always 1



x_opt=x[:,[0,1,2,3,4,5,6,7]]
```

```
regressor=sm.OLS(y,x_opt).fit()
print(regressor.summary())
print(x_opt)
#we check the p value. if p<0.05 than the variable(predictor) is significant.
```

```
            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.803
Model:                            OLS   Adj. R-squared:                  0.800
Method:                 Least Squares   F-statistic:                     228.9
Date:                Thu, 16 Apr 2020   Prob (F-statistic):          3.12e-134
Time:                        15:04:19   Log-Likelihood:                 537.37
No. Observations:                 400   AIC:                            -1059.
Df Residuals:                     392   BIC:                            -1027.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.2594      0.125    -10.097      0.000      -1.505      -1.014
x1             0.0017      0.001      2.906      0.004       0.001       0.003
x2             0.0029      0.001      2.680      0.008       0.001       0.005
x3             0.0057      0.005      1.198      0.232      -0.004       0.015
x4            -0.0033      0.006     -0.594      0.553      -0.014       0.008
x5             0.0224      0.006      4.034      0.000       0.011       0.033
x6             0.1189      0.012      9.734      0.000       0.095       0.143
x7             0.0245      0.008      3.081      0.002       0.009       0.040
==============================================================================
Omnibus:                       87.895   Durbin-Watson:                   0.759
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              181.191
Skew:                          -1.159   Prob(JB):                     4.52e-40
Kurtosis:                       5.344   Cond. No.                     1.31e+04
==============================================================================
```

```
x_opt=x[:,[0,1,2,5,6,7]]
regressor=sm.OLS(y,x_opt).fit()
print(regressor.summary())
```

```
print(x_opt)
#only significant variables
```

```
                    OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.803
Model:                            OLS   Adj. R-squared:                  0.800
Method:                 Least Squares   F-statistic:                     320.6
Date:                Thu, 16 Apr 2020   Prob (F-statistic):          2.04e-136
Time:                        15:04:31   Log-Likelihood:                 536.61
No. Observations:                 400   AIC:                            -1061.
Df Residuals:                     394   BIC:                            -1037.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.2985      0.117    -11.070      0.000      -1.529      -1.068
x1             0.0018      0.001      2.992      0.003       0.001       0.003
x2             0.0030      0.001      2.847      0.005       0.001       0.005
x3             0.0228      0.005      4.741      0.000       0.013       0.032
x4             0.1210      0.012     10.312      0.000       0.098       0.144
x5             0.0246      0.008      3.103      0.002       0.009       0.040
==============================================================================
Omnibus:                       87.489   Durbin-Watson:                   0.750
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              179.337
Skew:                          -1.157   Prob(JB):                     1.14e-39
Kurtosis:                       5.325   Cond. No.                     1.23e+04
==============================================================================
```

We observe that CGPA is indeed the most important factor in predicting chance of Admission

c. **Adding additional data from the other team:** We received 20 entries of the data from the other team. We added the data to our initial train data and tried to fit the initial multilinear model on the total data i.e. 420 entries.

Observations- We recieved additional 20 entries from the other team. We received an additional 20 entries from team Los Angeles. We included the entries in our train data and tried to fit another similar linear regression model on the net data.

```
dataset_2= pd.read_csv('/content/drive/My Drive/Intelligent Agents/Assignment 4/Admission_Predict (1).csv')
x_2= dataset_2.iloc[:,1:8].values
y_2= dataset_2.iloc[:,-1].values
print(dataset_2)
import statsmodels.api as sm
x_2=np.append(arr=np.ones((420,1)).astype(int),values=x_2,axis=1)


x_2_opt=x_2[:,[0,1,2,3,4,5,6,7]]
regressor_2=sm.OLS(y_2,x_2_opt).fit()
print(regressor_2.summary())
```

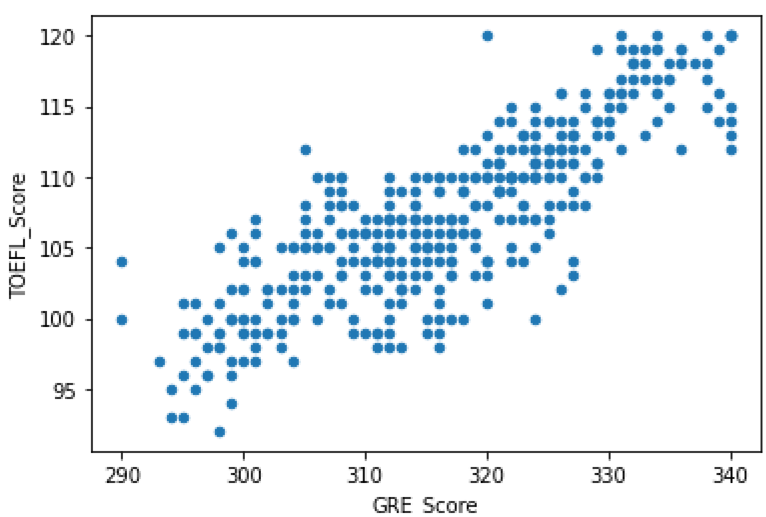
```
        OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.695
Model:                            OLS   Adj. R-squared:                  0.691
Method:                 Least Squares   F-statistic:                     156.9
Date:                Thu, 16 Apr 2020   Prob (F-statistic):          3.11e-103
Time:                        15:19:19   Log-Likelihood:                 460.80
No. Observations:                 420   AIC:                            -907.6
Df Residuals:                     413   BIC:                            -879.3
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.6882      0.068    -10.192      0.000      -0.821      -0.555
x1            -0.6882      0.068    -10.192      0.000      -0.821      -0.555
x2             0.0029      0.001      4.197      0.000       0.002       0.004
x3             0.0037      0.001      2.767      0.006       0.001       0.006
x4             0.0018      0.006      0.318      0.750      -0.009       0.013
x5             0.0140      0.006      2.375      0.018       0.002       0.026
x6             0.0229      0.006      3.776      0.000       0.011       0.035
```

x7 0.0772 0.011 6.820 0.000 0.055 0.099
==
Omnibus: 108.293 Durbin-Watson: 1.174
Prob(Omnibus): 0.000 Jarque-Bera (JB): 682.777
Skew: -0.933 Prob(JB): 5.46e-149
Kurtosis: 8.961 Cond. No. 1.96e+17
==

Observations: Training on the original data led to 2 columns being insignificant, whereas training on the net data we found only one of the columns insignificant. As we can see from the results the net data had a decreased error.

Conclusions- The additional data (la.csv) was a better fit and had a decreased spread. It's distribution was represented by the model much better.

2. **Part 2(Predicting the TOEFL Score given the GRE Score):**
   a. **Coming up with a model:** We had to decide as to what model could fit the data best so we plotted the *TOEFL Score* against the *GRE Score* . We observed that it was a fairly linear distribution, and the TOEFL Score generally increased with the increase in GRE Score.



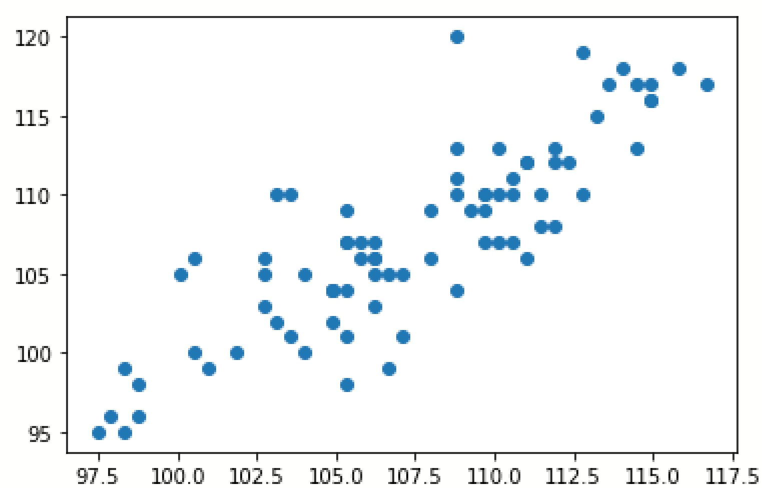
b. **Building the model:** We came up with a simple linear regression to predict the TOEFL Score of a person given the GRE Score.

```
from sklearn import linear_model
from sklearn import preprocessing
import numpy as np
#the GRE scores are our X features9used to predict the y
values)
feature_x = np.array(df['GRE_Score'])
feature_x = feature_x.reshape(-1, 1)
#WE ARE NOT NORMALIZING THE FEATURE AS THERE IS ONLY ONE X
FEATURE SO IT IS NOT REQUIRED

#feature y is the TOEFL score
feature_y = np.array(df['TOEFL_Score'])
```

Splitting data into train and test parts

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(feature_x,
feature_y, test_size=0.2, random_state=0)
print(X_train.shape, y_train.shape, X_test.shape,
y_test.shape)
model = linear_model.LinearRegression()
model.fit(X_train, y_train)
predicted_y=model.predict(X_test)
plt.scatter(predicted_y, y_test)
```




```
# print the score of our simple linear regression model
```

```
print(model.score(X_test, y_test))
```

We have 0.71 accuracy from our basic linear regression model

c. **Adding additional entries:** We received an additional 20 entries from team Los Angeles. We included the entries in our train data and tried to fit another similar linear regression model on the net data.
Observations:

```
#We shall train our Linear Regression Model on our total
initial data
feature_x = np.array(df['GRE_Score'])
feature_x = feature_x.reshape(-1, 1)
feature_y = np.array(df['TOEFL_Score'])

model_ = linear_model.LinearRegression()
#fitting the model on train data
model_.fit(feature_x,feature_y)

print("The coefficient of the regression is",model_.coef_)
The coefficient of the regression is [0.44222845]

print("The intercept of the regression is",model_.intercept_)
The intercept of the regression is -32.691289309982665
```

Test the model on the same train data so as to see the spread of the data

```
from sklearn.metrics import mean_squared_error
y_predicted=model_.predict(feature_x)
print("The Score of the model is",
model_.score(feature_x,feature_y))
print("The mean Squared Errror in the model is
",mean_squared_error(y_predicted,feature_y))

The Score of the model is 0.6988572151781711
```

```
The mean Squared Error in the model is  11.066063799569266
```

Training on the net data

```
#net_data is the concatenated data i.e. the initial and the recieved data
net_x=np.array(net_data['GRE_Score']).reshape(-1,1)
net_y=np.array(net_data.TOEFL_Score,)
model_net=linear_model.LinearRegression()
#fitting the model on net train data
model_net.fit(net_x, net_y)
print("The coefficient of the model is",model_net.coef_)
The coefficient of the model is [0.44243213]

print("The intercept of the model is",model_net.intercept_)
The intercept of the model is -32.75630813920847
```

Testing on the same net data to see the spread of the data

```
y_net_predicted=model_net.predict(net_x)
print("The Score of the model is",
model_net.score(net_x,net_y))
print("The mean Squared Errror in the model is
",mean_squared_error(y_net_predicted,net_y))

The Score of the model is 0.714541721259486
The mean Squared Errror in the model is  10.542964599344035
```

Testing the new model only on the 20 entries

```
la_x = np.array(df_la['GRE_Score']).reshape(-1,1)
la_y=np.array(df_la['TOEFL_Score'])
```

```
print("Score of the net model on recieved data
is",model_net.score(la_x,la_y))
```

```
Score of the net model on recieved data is 0.9979803490700061
```

```
print("Mean Squared error on the new data
is",mean_squared_error(model_net.predict(la_x),la_y))
```

```
Mean Squared error on the new data is 0.0808668232369561
```

i) The slope of line increased slightly
ii) The intercept of the model decreased slightly
iii) The mean squared error decreases
We now test the data on only the new data (la.csv)
i) The score of the model increases insanely and nearly touches 100%
ii) The mean squared error decreases to a negligible value

Conclusions:

The data received fits the model extremely well. It is approximately an ideal distribution for the model.

Bugs Faced:

1. It was difficult to work with Columns having whitespaces in their names, and we received numerous errors: That's when we decided to eliminate all the white spaces from column names.
2. For Task 2 we tried to normalize the input column, i.e. GRE score column.
   We thought it was a better practice to normalize columns in Linear Regression. But the values were quite close by, i.e the difference in the values was incomparable to the values themselves thus our model was unable to work as all the input values were coming equal. We then decided not to normalize it as there was only one input feature But a better way would be to consider GRE Score-min(GRE Score) as a feature and normalize that.

Bibliography:

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=15&cad=rja&uact=8&ved=2ahUKEwics_Ch7-7oAhV46nMBHS9qCWcQFjAOegQIARAB&url=https%3A%2F%2Fmachinelearningmastery.com%2Flinear-regression-for-machine-learning%2F&usg=AOvVaw0MCmkUUX9B46GXu-Ht-U1S

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwiAouqy7-7oAhVCjOYKHTwTB6IQFjAAegQIAhAB&url=http%3A%2F%2Fscikit-learn.org%2Fstable%2Fmodules%2Fgenerated%2Fsklearn.model_selection.train_test_split.html&usg=AOvVaw0VvM_hdQkS03aadJz9AKKA

https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-with-scikit-learn-83a8f7ae2b4f

https://www.geeksforgeeks.org/python-implementation-of-polynomial-regression/