

UNIVERSITY OF ASIA PACIFIC

Department of Computer Science & Engineering



Artificial Intelligence Lab Course Code: CSE 404

Project - 2

Student Name : Md. Sohanuzzaman Soad
Student ID : 18101064
Section : B1
Date of Submission : 10.11.2021
Submitted To : Dr. Nasima Begum,
Assistant Professor
CSE, UAP

INDEX

Content	Page Number
Problem Statement	2
Introduction	2
Algorithm	2
Datasets	3
Implementation & Visualization	4
Result Analysis	7
Conclusion	7

Problem Statement :

Multi-variable Linear Regression for predicting the chance of admit of Graduate Admission using Open-Source Dataset.

Dataset: [Graduate Admission 2](#)

Introduction :

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression is to model the linear relationship between the explanatory (independent) variables and response (dependent) variables. In essence, multiple regression is the extension of ordinary least-squares (OLS) regression because it involves more than one explanatory variable.

Algorithm:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n + \epsilon$$

Here,

y = predicted value of the dependent variable

θ_0 = y-intercept or value of y with all parameters set to 0

$\theta_1 x_1$ = regression coefficient of the first independent variable

$\theta_n x_n$ = regression coefficient of the last independent variable

ϵ = model error or level of variation

Cost Function:

$$J(\theta) = 1/2m \sum_{i=1}^m (h(\theta)^{(i)} - y^{(i)})^2$$

Gradient Decent:

$$\frac{\partial J(\theta)}{\partial \theta_j} = 1/m \sum_{i=1}^m \left(h(\theta^{(i)}) - y^{(i)} \right) \cdot X_j^{(i)}$$

Datasets:

Context:

This dataset is created for prediction of Graduate Admissions from an Indian perspective.

Content:

The dataset contains several parameters which are considered important during the application for Masters Programs.

The parameters included are :

1. GRE Scores (out of 340)
2. TOEFL Scores (out of 120)
3. University Rating (out of 5)
4. Statement of Purpose and Letter of Recommendation Strength (out of 5)
5. Undergraduate GPA (out of 10)
6. Research Experience (either 0 or 1)
7. Chance of Admit (ranging from 0 to 1)

Implementation:

Programming Language: Python

Tools: Jupyter Notebook, Pandas, Matplotlib

1) Without SK-Learn:

Multiple Linear Regression without SK-Learn

Dataset: [Graduate Admission 2](#)

Import Necessary Libraries

```
In [1]: 1 import matplotlib.pyplot as plt # plotting
2 import numpy as np # linear algebra
3 import os # accessing directory structure
4 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
In [2]: 1 import warnings
2 warnings.filterwarnings('ignore')
3 print(os.listdir('./')) #Check File Directory For Identify DataSets
```

['.ipynb_checkpoints', 'Admission_Predict.csv', 'Admission_Predict_Ver1.1.csv', 'archive.zip', 'Multiple Linear Regression with SK-Learn.ipynb', 'Multiple Linear Regression without SK-Learn.ipynb', 'Others']

Data Loader & Pre-Processing

```
In [3]: 1 df = pd.read_csv('Admission_Predict_Ver1.1.csv', delimiter=',', ) #Read From Dataset File
2 df.dataframeName = 'Admission_Predict_Ver1.1.csv' #Set DataFrame Name
3 nRow, nCol = df.shape #Extract Row and Column from DataFrame Shape
4 df=df.drop(columns=['Serial No.']) #Drop 'Serial No.' Column From DataFrame
5 print(f'There are {nRow} rows and {nCol} columns')
```

There are 500 rows and 9 columns

```
In [6]: 1 df.head() #Check Some Rows from Top
```

Out[6]:

	GRE	TOEFL	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

```
In [7]: 1 #calculating some statistical data like percentile,
2 #mean and std of the numerical values of the DataFrame
3 df.describe()
```

Out[7]:

	GRE	TOEFL	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000
mean	316.472000	107.192000	3.114000	3.374000	3.484000	8.576440	0.560000	0.72174
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	308.000000	103.000000	2.000000	2.500000	3.000000	8.127500	0.000000	0.630000
50%	317.000000	107.000000	3.000000	3.500000	3.500000	8.560000	1.000000	0.720000
75%	325.000000	112.000000	4.000000	4.000000	4.000000	9.040000	1.000000	0.820000
max	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

```

In [12]: 1 cost_ls=[]
2 for step in range(epoch):
3     t0,t1,t2,t3,t4,t5,t6,t7=(theta) #Extract Values from List of Theta
4     x1,x2,x3,x4,x5,x6,x7=(x) #Extract Values from List of X
5     #Hypothesis function
6     df['pred']=t0+t1*x1+t2*x2+t3*x3+t4*x4+t5*x5+t6*x6+t7*x7
7
8     #Squared error/difference
9     df['diff']=(df.pred-df.Chance_of_Admit)**2
10
11     m = len(y) # Number of datapoints
12
13     cost = (df['diff'].sum()/(2*m)) # Cost function
14     cost_ls.append(cost) #Append Value of Every Epoch
15     print('\n\n')
16     print('For step : ', step+1)
17     print('Cost is : {}'.format(cost))
18
19     step_size = 0.000001 #initializing Learning rate
20
21     # Gradient decent optimization of theta values
22
23     theta[0] = theta[0] - (step_size/m)*(np.sqrt(df['diff']).sum())
24     theta[1] = theta[1] - (step_size/m)*((np.sqrt(df['diff'])*x1).sum())
25     theta[2] = theta[2] - (step_size/m)*((np.sqrt(df['diff'])*x2).sum())
26     theta[3] = theta[3] - (step_size/m)*((np.sqrt(df['diff'])*x3).sum())
27     theta[4] = theta[4] - (step_size/m)*((np.sqrt(df['diff'])*x4).sum())
28     theta[5] = theta[5] - (step_size/m)*((np.sqrt(df['diff'])*x5).sum())
29     theta[6] = theta[6] - (step_size/m)*((np.sqrt(df['diff'])*x6).sum())
30     theta[7] = theta[7] - (step_size/m)*((np.sqrt(df['diff'])*x7).sum())
31
32     print('Gradient Decent Optimized Thetas: ', theta)
33

```

```

For step : 1
Cost is : 600385920.734564
Gradient Decent Optimized Thetas: [316.43736983827824, 96.21859601954671, -0.6043270709781994, 3.2652565878086803, 3.366399
4358779, 8.455180467867198, 0.26237724954940134, 0.70200125392232]

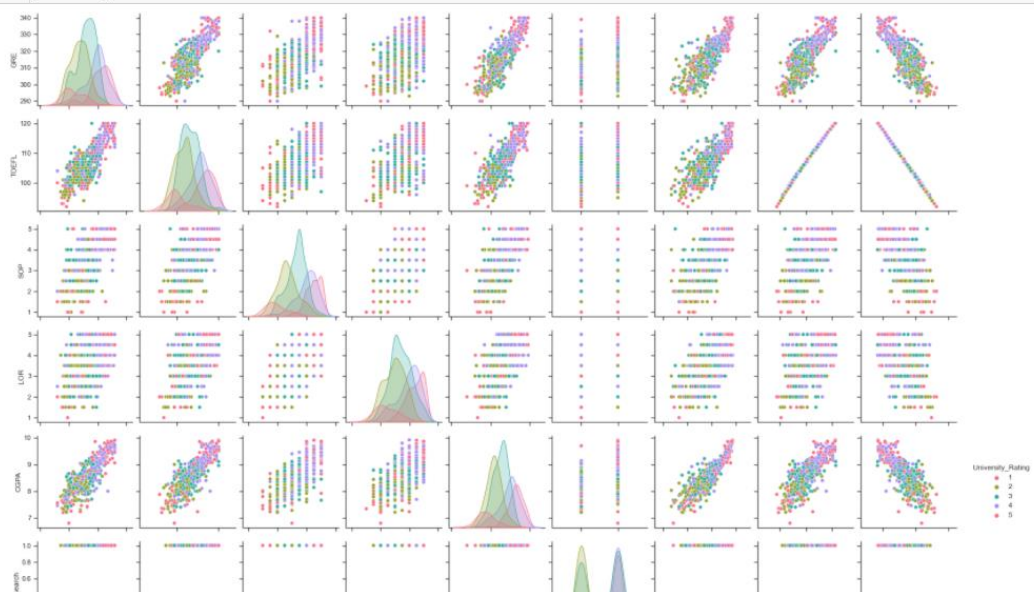
For step : 2
Cost is : 473532354.80532783
Gradient Decent Optimized Thetas: [316.4066147826834, 86.47316678619075, -3.906511958023119, 3.16868736679928, 3.2619635078
99899, 8.34749353474166, -0.001937759525835425, 0.6844725532785806]

```

```

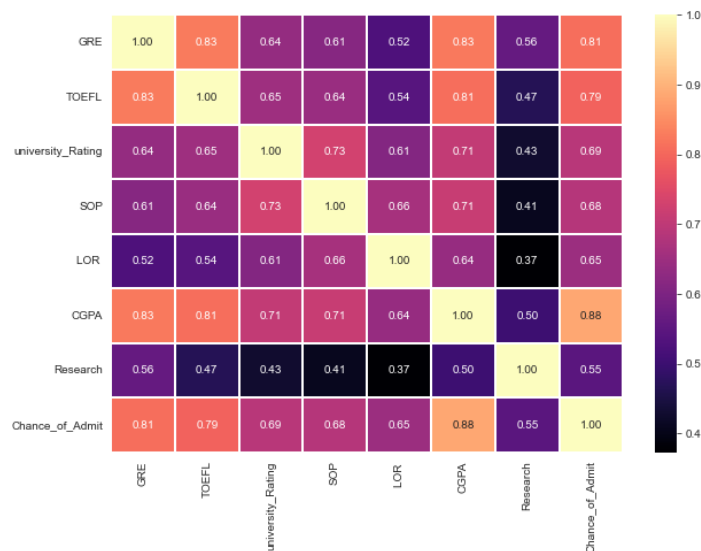
In [18]: 1 sns.set(style="ticks", color_codes=True)
2 sns.pairplot(df, hue="University_Rating", palette="husl")
3 plt.show()

```

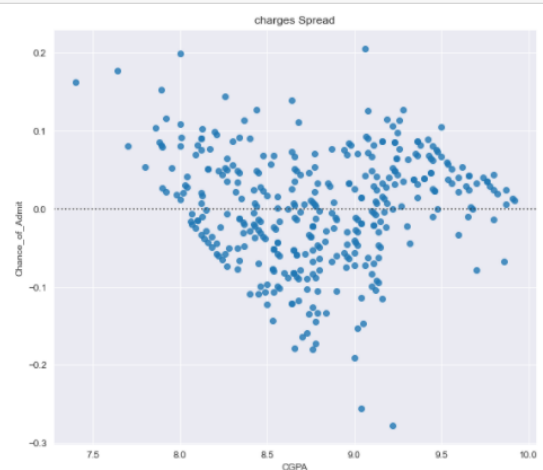
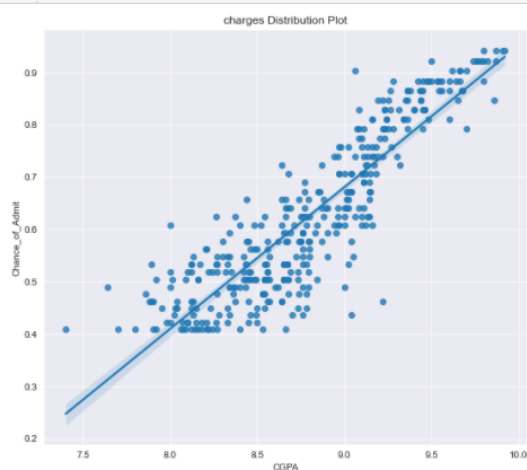
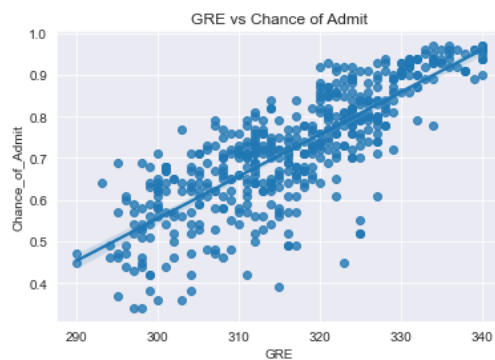


2) With SK-Learn:

```
In [12]: 1 #Heat-Map For Compute pairwise correlation of columns
2 plt.figure(figsize=(10,7))
3 cr = data.corr()
4 sb.heatmap(cr, annot=True, linewidths=0.1, fmt= '.2f',cmap="magma")
5 plt.show()
```

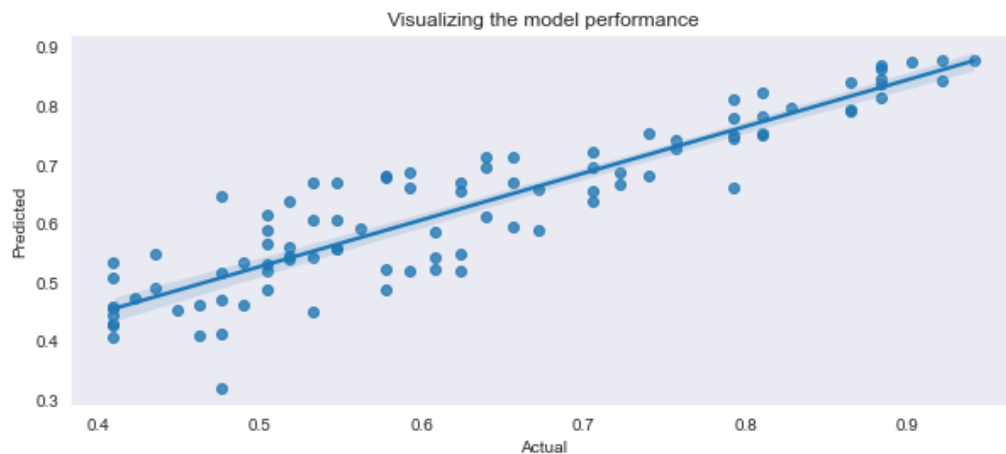


```
In [13]: 1 #RegPlot for GRE & Chance_Of_Admit
2
3 fig = sb.regplot(data.GRE,data.Chance_of_Admit)
4 plt.title("GRE vs Chance of Admit")
5 plt.show()
```



Result Analysis:

```
5 plt.ylabel('Predicted')
6 plt.title('Visualizing the model performance')
7 plt.grid()
8
```



Test Linear Regression Model

```
In [32]: 1 # our model is ready to predict y.
          2 y_predict = Lin_regressor.predict(X_test)
```

```
In [33]: 1 # our model prediction
          2 y_predict[:10]
```

```
Out[33]: array([0.5545905, 0.45969253, 0.5398055, 0.67006317, 0.31890853,
                0.58834413, 0.44172199, 0.45338511, 0.86326694, 0.52002521])
```

```
In [34]: 1 # y test
          2 y_test[:10]
```

```
Out[34]: array([0.5476, 0.4624, 0.5184, 0.6241, 0.4761, 0.6724, 0.4096, 0.4096,
                0.8836, 0.5776])
```

```
In [35]: 1 from sklearn import metrics
          2 #Check the Accuracy of the Model
          3 metrics.mean_absolute_error(y_test,y_predict)
```

```
Out[35]: 0.05313501697385051
```

5% Error Means our Model is 95% correct

Conclusion:

By doing this project I gained knowledge about Multi-variable Linear Regression and also learned the way to apply this on any real-life dataset.