Machine Learning Term Project

Efe Mert Çırpar - 040190247

Electronics and Communication Engineering - Faculty of Electrical and Electronics

A Regression Study on Predicting Chances of University Application Acceptance

Abstract

Knowing your acceptance rate before applying to universities is extremely important for a variety of reasons.

First and foremost, it can help you set realistic expectations and goals for your college search. This is particularly important if you are considering applying highly competitive school.

On the other hand, understanding the acceptance rate can also be helpful when it comes to financial planning. Especially, as a student from Turkey, most of the colleges have a fee for application around 150-200 €. If we consider 1 € = 19.50 Turkish Liras, applying to universities is going to be very expensive. Thus, getting accepted to one of the universities is quite crucial.

In conclusion, knowing the acceptance rate of a college before applying can be incredibly important for a variety of reasons. It can give you a sense of your chances of being accepted, help you to gauge the level of competitiveness of a particular college, and assist with financial planning. As such, it is worth considering when deciding on colleges to apply to.

In particular, I am also considering applying for master's programs abroad. Therefore, when our lecturer led us to choose any topic we would like, this algorithm was the one that I wanted to work with. I collected the data from kaggle.com, this data shows the acceptance rate of students with the same accomplishments in 400 different scenarios [1].

1. Entrance

In this data, we have:

- GRE exam score
- TOEFL exam score
- University's rating of over 5
- Statement of Purpose (SOP) score
- Letter of Recommendation (LOR) score
- Student's CGPA over 10
- Student's research paper number

to consider.

We are going to use these variables as a multiple-variable regression and this operation is going to lead us to an estimated result of an acceptance rate for that specific situation. In this study, we are using regression algorithms, since we need to predict a probability based on our given dataset.

This study with a better dataset can actually make an application that is able to help many students.

2. Procedures, Measurements, and Graph for Single Variables.

To calculate the relation acceptance rate and every single input (variable) we are going to use a single variable regression algorithm and plot the results. Since our calculation is going to be based on multiple variable regression, these results are not going to be as accurate as they can be.

However, it is important to see every relationship.

I used “sklearn” documentation in order to create my algorithm. The following code structure line is the one I used for calculating every single variable linear regression.

```
from sklearn.linear_model import
LinearRegression
from sklearn.model_selection import
train_test_split

plt.scatter()
plt.xlabel()
plt.ylabel()
x_train, x_test, y_train, y_test =
train_test_split()

Regressor = LinearRegression()
Regressor=LinearRegression().fit(x_tr
ain, y_train)
prediction =Regressor.predict(x_test)
plt.plot()
```

a) GRE exam score

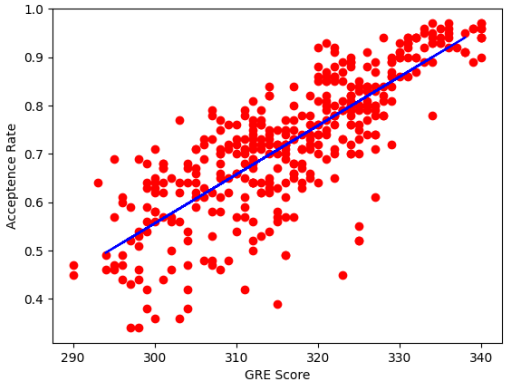

Figure.1

From Figure.1 the importance of the GRE score can be seen for all data points.

- The red dots are our data for GRE scores and their acceptance rates.
- The blue line is the result of regression fitted according to our dataset.

b) TOEFL exam score

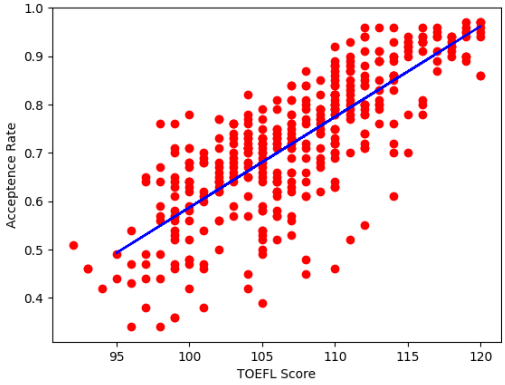

Figure.2

From Figure.2 the importance of the TOEFL score can be seen for all data points.

- The red dots are our data for TOEFL scores and their acceptance rates.
- The blue line is the result of regression fitted according to our dataset.

c) University rating

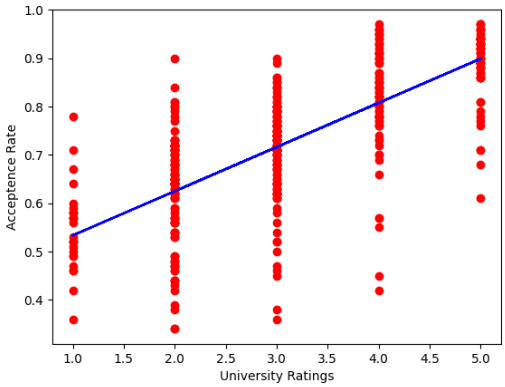

Figure.3

From Figure.3 it is clearly can be seen that higher-rated universities accepted students with higher rates.

- The red dots are our data for University Ratings and their acceptance rates.
- The blue line is the result of regression fitted according to our dataset.

d) SOP

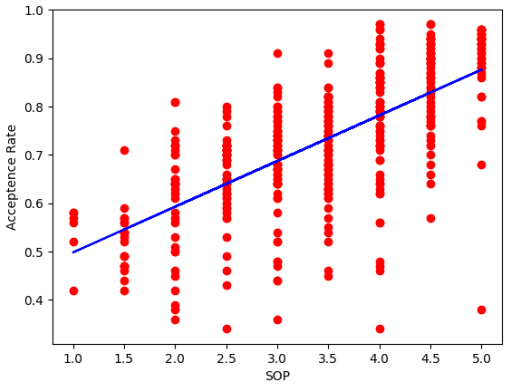

Figure.4

From Figure.4 it is clearly can be seen that students with higher graded SOPs accepted with higher rates.

- The red dots are our data for SOPs and their acceptance rates.
- The blue line is the result of regression fitted according to our dataset.

e) LOR

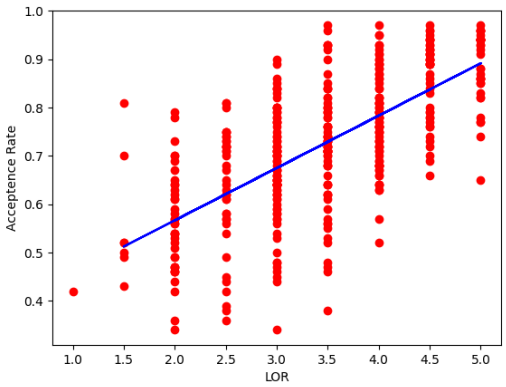

Figure.5

From Figure.5 it is clearly can be seen that students with higher graded LORs accepted with higher rates.

- The red dots are our data for LORs and their acceptance rates.
- The blue line is the result of regression fitted according to our dataset.

f) CGPA

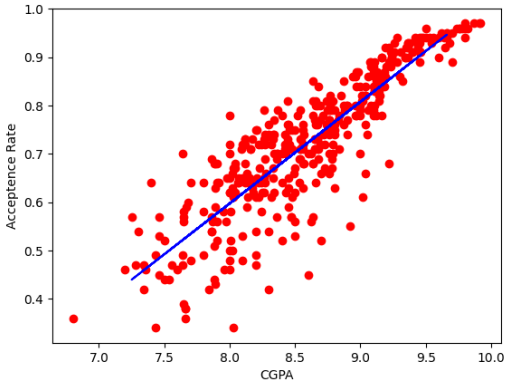

Figure.6

From Figure.6 we can make a command that is implying that students with higher CGPA are students who are much more

likely to accept the universities to which they applied.

- The red dots are our data for students' CGPAs and their acceptance rates.
- The blue line is the result of regression fitted according to our dataset.

3. Testing and Training for Multiple-Variable Regression Algorithm

After several tries of different rates for training and testing the code, the 0.2 test/train ratio was the one running better than every other rate. Like the single input linear regression, we used "sklearn" algorithms and multiple-variable linear regression algorithms to run the code. The following code structure is used in this Machine Learning project.

```
x_train, x_test, y_train, y_test =
train_test_split(input_students,
Acceptence_Rate, test_size=0.2,
random_state=0)


students_Linear_Regressor =
LinearRegression()

students_Linear_Regressor=
LinearRegression().fit(x_train,
y_train)

students_ar_prediction =
students_Linear_Regressor.predict(x_t
est)


print(y_test)

print(students_ar_prediction)
```

With these lines the following we get the following estimations as the resulting output (randomly 10 estimations have been chosen).

Real Acceptance Rate	Estimated Acceptance Rate
0.71	0.69791327
0.70	0.69343926
0.79	0.77882728
0.73	0.71577391
0.72	0.72281999
0.61	0.63445628
0.69	0.66728309
0.62	0.64518748
0.93	0.90584935
0.43	0.47460138

Table.1

From Table.1 it can be seen that even though there are small errors with estimated outputs, the algorithm works quite accurately overall for our dataset.

4. Working with Given Inputs

Finally, we need to take input values from our users in order to work with real-life individuals. This problem's code line structure is given below.

```
gre_input = int(input("Please enter
your GRE Score: "))

toefl_input = int(input("Please enter
your TOEFL Score: "))
```

```
unirate_input = float(input("Please
enter University's Rating: "))

sop_input = float(input("Please enter
your SOP Score: "))

lor_input = float(input("Please enter
your LOR Score: "))

cpga_input = float(input("Please
enter your CPGA: "))

researchpaper_input =
int(input("Please enter your Research
Paper number: "))


input_data = [[gre_input,
toefl_input, unirate_input,
sop_input, lor_input, cpga_input,
researchpaper_input]]

input_data_dataframe =
pd.DataFrame(input_data, columns =
['GRE Score', 'TOEFL
Score','University Rating', 'SOP',
'LOR ', 'CGPA', 'Research'])

x_train, x_test, y_train, y_test =
train_test_split(input_students,
Acceptence_Rate, test_size=0.2,
random_state=0)

predict_Linear_Regressor =
LinearRegression()

predict_Linear_Regressor=
LinearRegression().fit(x_train,
y_train)

predict_ar_prediction =
predict_Linear_Regressor.predict(inpu
t_data_dataframe)

print(predict_ar_prediction)
```

These code lines take every value from the user and put them in a “DataFrame” format. From this Data Frame, our machine takes the user’s accomplishments and tests them with the ones our machine has learned.

At the end of this paper, I am going to publish our code as a whole with functions and variables which I used for users to calculate their own acceptance probability in a specific situation. I used “VirtualStudio” platform to perform everything.

5. References

[1] A. D. Khare, “Data For Admission in The University,” Kaggle, 27-Oct-2022. [Online]. Available: https://www.kaggle.com/datasets/akshaydattatraykhare/data-for-admission-in-the-university.

6. Python Code

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

        students = pd.read_csv('adm_data.csv')
        #print(students)
        input_students = students.iloc[:,1:8]
        Acceptence_Rate = students.iloc[:,8:]
        #print(input_students)
        #print(Acceptence_Rate)

        GRE_Score = students.iloc[:,1:2]
        TOEFL_Score = students.iloc[:,2:3]
        University_Ratings = students.iloc[:,3:4]
        SOP = students.iloc[:,4:5]
        LOR = students.iloc[:,5:6]
        CGPA = students.iloc[:,6:7]
        Research = students.iloc[:,7:8]

        from sklearn.linear_model import LinearRegression
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler

        ######################################

        plt.scatter(GRE_Score, Acceptence_Rate, color = 'red')
        plt.xlabel("GRE Score")
        plt.ylabel("Acceptence Rate")

        x_train, x_test, y_train, y_test = train_test_split(GRE_Score,
    Acceptence_Rate, test_size=0.2, random_state=0)

        gre_Linear_Regressor = LinearRegression()
        gre_Linear_Regressor= LinearRegression().fit(x_train, y_train)
        gre_ar_prediction = gre_Linear_Regressor.predict(x_test)
        plt.plot(x_test, gre_ar_prediction, color = 'blue')

        plt.show()

        ######################################

        plt.scatter(TOEFL_Score, Acceptence_Rate, color = 'red')
        plt.xlabel("TOEFL Score")
        plt.ylabel("Acceptence Rate")
```

```
50.
51.       x_train, x_test, y_train, y_test = train_test_split(TOEFL_Score,
   Acceptence_Rate, test_size=0.2, random_state=0)
52.
53.       toefl_Linear_Regressor = LinearRegression()
54.       toefl_Linear_Regressor= LinearRegression().fit(x_train, y_train)
55.       toefl_ar_prediction = toefl_Linear_Regressor.predict(x_test)
56.       plt.plot(x_test, toefl_ar_prediction, color = 'blue')
57.
58.       plt.show()
59.
60.       ####################################
61.
62.       plt.scatter(University_Ratings, Acceptence_Rate, color = 'red')
63.       plt.xlabel("University Ratings")
64.       plt.ylabel("Acceptence Rate")
65.
66.       x_train, x_test, y_train, y_test =
   train_test_split(University_Ratings, Acceptence_Rate, test_size=0.2,
   random_state=0)
67.
68.       unirate_Linear_Regressor = LinearRegression()
69.       unirate_Linear_Regressor= LinearRegression().fit(x_train, y_train)
70.       unirate_ar_prediction = unirate_Linear_Regressor.predict(x_test)
71.       plt.plot(x_test, unirate_ar_prediction, color = 'blue')
72.
73.       plt.show()
74.
75.       ####################################
76.
77.       plt.scatter(SOP, Acceptence_Rate, color = 'red')
78.       plt.xlabel("SOP")
79.       plt.ylabel("Acceptence Rate")
80.
81.       x_train, x_test, y_train, y_test = train_test_split(SOP,
   Acceptence_Rate, test_size=0.2, random_state=0)
82.
83.       sop_Linear_Regressor = LinearRegression()
84.       sop_Linear_Regressor= LinearRegression().fit(x_train, y_train)
85.       sop_ar_prediction = sop_Linear_Regressor.predict(x_test)
86.       plt.plot(x_test, sop_ar_prediction, color = 'blue')
87.
88.       plt.show()
89.
90.       ####################################
```

```
91.
92.       plt.scatter(LOR, Acceptence_Rate, color = 'red')
93.       plt.xlabel("LOR")
94.       plt.ylabel("Acceptence Rate")
95.
96.       x_train, x_test, y_train, y_test = train_test_split(LOR,
   Acceptence_Rate, test_size=0.2, random_state=0)
97.
98.       lor_Linear_Regressor = LinearRegression()
99.       lor_Linear_Regressor= LinearRegression().fit(x_train, y_train)
100.      lor_ar_prediction = lor_Linear_Regressor.predict(x_test)
101.      plt.plot(x_test, lor_ar_prediction, color = 'blue')
102.
103.      plt.show()
104.
105.      ######################################
106.
107.      plt.scatter(CGPA, Acceptence_Rate, color = 'red')
108.      plt.xlabel("CGPA")
109.      plt.ylabel("Acceptence Rate")
110.
111.      x_train, x_test, y_train, y_test = train_test_split(CGPA,
   Acceptence_Rate, test_size=0.2, random_state=0)
112.
113.      cpga_Linear_Regressor = LinearRegression()
114.      cpga_Linear_Regressor= LinearRegression().fit(x_train, y_train)
115.      cpga_ar_prediction = cpga_Linear_Regressor.predict(x_test)
116.      plt.plot(x_test, cpga_ar_prediction, color = 'blue')
117.
118.      plt.show()
119.
120.      ######################################
121.

122.      x_train, x_test, y_train, y_test = train_test_split(input_students,
   Acceptence_Rate, test_size=0.2, random_state=0)
123.
124.      students_Linear_Regressor = LinearRegression()
125.      students_Linear_Regressor= LinearRegression().fit(x_train, y_train)
126.      students_ar_prediction = students_Linear_Regressor.predict(x_test)
127.
128.      print(y_test)
129.      print(students_ar_prediction)
130.
```

```
131.      gre_input = int(input("Please enter your GRE Score: "))
132.      toefl_input = int(input("Please enter your TOEFL Score: "))
133.      unirate_input = float(input("Please enter University's Rating: "))
134.      sop_input = float(input("Please enter your SOP Score: "))
135.      lor_input = float(input("Please enter your LOR Score: "))
136.      cpga_input = float(input("Please enter your CPGA: "))
137.      researchpaper_input = int(input("Please enter your Research Paper
   number: "))
138.
139.      input_data = [[gre_input, toefl_input, unirate_input, sop_input,
   lor_input, cpga_input, researchpaper_input]]
140.      input_data_dataframe = pd.DataFrame(input_data, columns = ['GRE
   Score', 'TOEFL Score','University Rating', 'SOP', 'LOR ', 'CGPA',
   'Research'])
141.      x_train, x_test, y_train, y_test = train_test_split(input_students,
   Acceptence_Rate, test_size=0.2, random_state=0)
142.      predict_Linear_Regressor = LinearRegression()
143.      predict_Linear_Regressor= LinearRegression().fit(x_train, y_train)
144.      predict_ar_prediction =
   predict_Linear_Regressor.predict(input_data_dataframe)
145.      print(predict_ar_prediction)
```