GRADUATE ADMISSION PREDICTION

A PROJECT REPORT

Submitted by

Harini Priya B R (18C034)

Saranya S B (18C090)

Kannimalar K (18C042)

in partial fulfillment for the award of the degree

of

Bachelor of Engineering

IN

Computer Science and Engineering

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2021

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**Graduate Admission Prediction**” is the bonafide work of “**Harini Priya B R (18C034), Saranya S B (18C090), Kannimalar K (18C042)**” who carried out the project work under my supervision.

<<Signature of the Head of the Department>>	<<Signature of the Supervisor>>
SIGNATURE	**SIGNATURE**
Dr. P. Chitra	Dr. M .Nirmaladevi
HEAD OF THE DEPARTMENT	**SUPERVISOR**
	Assistant Professor
Computer Science and Engineering	Computer Science and Engineering
Thiagarajar College of Engineering	Thiagarajar College of Engineering
Madurai - 625 015	Madurai - 625 015

ABSTRACT:

Nowadays, the decision of whether getting admission in an institution is not predictable even sometimes the good guide may fail to predict. Hence in this paper, we apply a machine learning algorithm called logistic regression to predict whether a student will get a master degree admission in an institution or not. This will help the students to know whether they have a chance of admission in an institution and also the institution can also make decisions to whom they should give admission.

Keywords - logistic regression, master degree admission.

1.**INTRODUCTION**:

The world markets are developing at a faster rate.The industries are looking for best knowledge and skills among people.So the young talents mostly prefer to do higher degrees in order to stand persistently in the fast growing industries.Hence the number of students applying for a graduate degree has rapidly increased. Thus we made a machine learning model to predict the chance of admission which helps students to know their chance of admission in a particular college and helps colleges to predict the possibilities of accepting students every year. The dataset [1] presented in this paper is a graduate admission prediction dataset containing 500 rows and 7 columns. The whole process is done using Python.

2.**LITERATURE SURVEY**:

There are a number of predictors and classifiers for graduate prediction.One such significant work was observed in [2].This dataset contains three independent variables which includes gre,gpa and rank and an dependent target variable admit.The gre and gpa attributes are continuous and rank variable is discrete(0 or 1).

2.1 Pros -

1. Since the attributes are less in number,it makes the classification much easier.

2.2 Cons -

1. As the dataset contains only 3 attributes,it may not generalise well on all instances and increase the testing error.
2. The accuracy of the dataset is low.

3.**PARAMETERS**:

The features of the dataset are mentioned below:

3.1 GRE - Graduate Record Admission score. The score is out of 340.

3.2 TOEFL - Test Of English as a Foreign Language. The score is out of 120.

3.3 University Rating - This represents the rating of the college where the student has finished his/her bachelor degree. The rating is out of 5.

3.4 SOP - Statement Of Purpose. It is a document written by the student to showcase their life, ambitions, and motivations to choose this particular degree program in a particular university. The score is out of 5.

3.5 LOR - Letter Of Recommendation. This indicates the strength of the student depending on the letter. This score is out of 5.

3.6 CGPA - Cumulative Grade Points Average. This corresponds to the cgpa of the students bachelor degree. This score is out of 10.

3.7 Research - This field can have either 0 or 1. 1 represents that a student has worked as a research assistant with a university professor. 0 represents that the student has no experience in research.

The above listed variables are independent attributes through which we are going to predict the probability of dependent variable called chance of admit which is having range from 0 to 1.

4.ARCHITECTURE / SYSTEM DESIGN:

Data preprocessing is performed. Missing values and outliers are handled. After all the preprocessing steps the dataset was divided into training and testing sets.The testing set was 20 percent of the total data.The feature data X was divided to Xtrain and Xtest and Y was divided to Ytrain and Test.Since it is a binary classification, we have chosen logistic regression. This is implemented using the linear model present in sklearn. The accuracy score obtained is 89%.

5.EVALUATING MODELS:

5.1 Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

5.2 Precision is the number of correct positive results divided by the number of positive results predicted by the classifier.

5.3 Recall is the number of correct positive results divided by the number of all relevant samples

5.4 F1 score is the average of precision and recall.

5.5 ROC curve is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate. False Positive Rate.

5.6 Logistic Regression:

```
from sklearn import linear_model
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

from sklearn.metrics import roc_curve

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,

random_state=1) reg = linear_model.LogisticRegression()

reg.fit(X_train, y_train)

pred_prob1 = reg.predict(X_test)

variance score: 1 means perfect prediction

print("Logistic Regression:")

print('Accuracy: {}'.format(reg.score(X_test, y_test)*100))

print("Classification Report:")

print(classification_report(y_test,reg.predict(X_test))

```
Logistic Regression:
Accuracy: 89.0
Classification Report:
              precision    recall  f1-score   support

         0.0       0.75      0.40      0.52        15
         1.0       0.90      0.98      0.94        85

    accuracy                           0.89       100
   macro avg       0.83      0.69      0.73       100
weighted avg       0.88      0.89      0.88       100
```

Fig 1

5.7 **Decision Tree**:

from sklearn.tree import DecisionTreeClassifier

reg = DecisionTreeClassifier(random_state = 0)

fit the regressor with X and Y data

```
reg.fit(X_train, y_train)

pred_prob2= reg.predict(X_test)

print("DecisionTreeClassifier:")

print('Accuracy: {}'.format(reg.score(X_test, y_test)*100))

print("Classification Report:")

print(classification_report(y_test,reg.predict(X_test)))

fpr1,tpr1,thresholds1 = roc_curve(y_test, reg.predict(X_test))
```

```
DecisionTreeClassifier:
Accuracy: 86.0
Classification Report:
              precision    recall  f1-score   support

         0.0       0.53      0.53      0.53        15
         1.0       0.92      0.92      0.92        85

    accuracy                           0.86       100
   macro avg       0.73      0.73      0.73       100
weighted avg       0.86      0.86      0.86       100
```

Fig 2

5.8 **Random Forest**:

```
from sklearn.ensemble import RandomForestClassifier

reg = RandomForestClassifier(random_state=0,max_depth=1)

reg.fit(X_train, y_train)

pred_prob3 = reg.predict(X_test)
```

```
print("RandomForestClassifier:")

print('Accuracy score: {}'.format(reg.score(X_test, y_test)*100))

print("Classification Report:")

print(classification_report(y_test,reg.predict(X_test)))

fpr2,tpr2,thresholds2 = roc_curve(y_test, reg.predict(X_test))
```

```
RandomForestClassifier:
Accuracy score: 88.0
Classification Report:
              precision    recall  f1-score   support

         0.0       0.80      0.27      0.40        15
         1.0       0.88      0.99      0.93        85

    accuracy                           0.88       100
   macro avg       0.84      0.63      0.67       100
weighted avg       0.87      0.88      0.85       100
```

Fig 3

```
import matplotlib.pyplot as plt

plt.style.use('seaborn')

plt.title('ROC curve')

# x label

plt.xlabel('False Positive Rate')

# y label

plt.ylabel('True Positive rate')

plt.plot(fpr, tpr, linestyle='--',color='blue', label='Logistic Regression')

plt.plot(fpr1, tpr1, linestyle='--',color='orange',
```

label='DecisionTreeClassifier') plt.plot(fpr2, tpr2, linestyle='--',color='green', label='RandomForestClassifier')

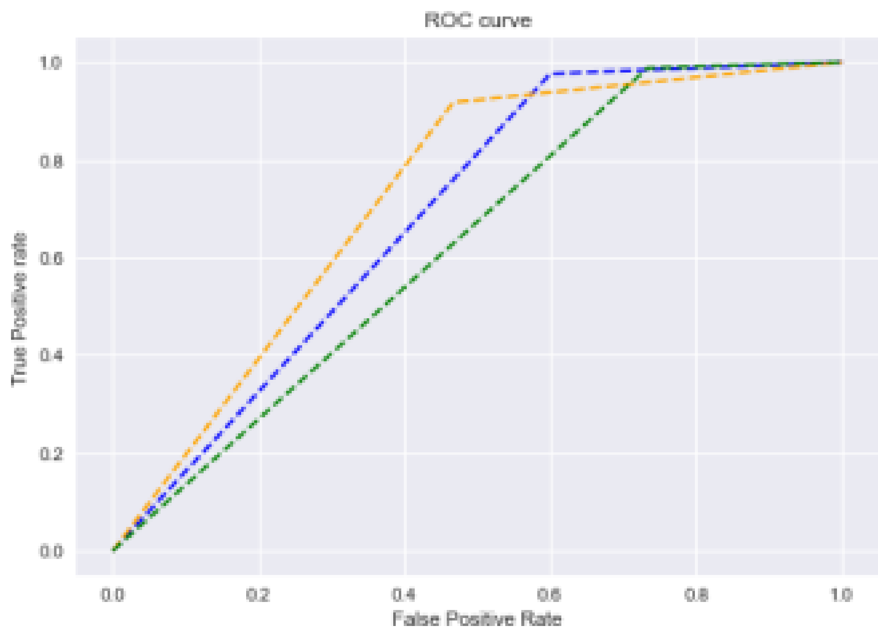


Fig 4

Blue-logistic regression

Orange-decision tree classification

green- random forest classification

The blue line has the highest true positive rate and this line indicates logistic regression. Hence logistic regression has the highest accuracy than decision tree and random forest regression.

6.**PROCEDURE**:

6.1.Understanding and visualizing data:

6.1.1 **Boxplot:**

We have taken research as one attribute and plotted against the target attribute called chance of admit.

Code:

```
import pandas as pd
df = pd.read_csv("dataset_lab.csv")
df.head()
df.boxplot(by ='Research', column =['Chance_of_Admit_'], grid = False)
```

Output:

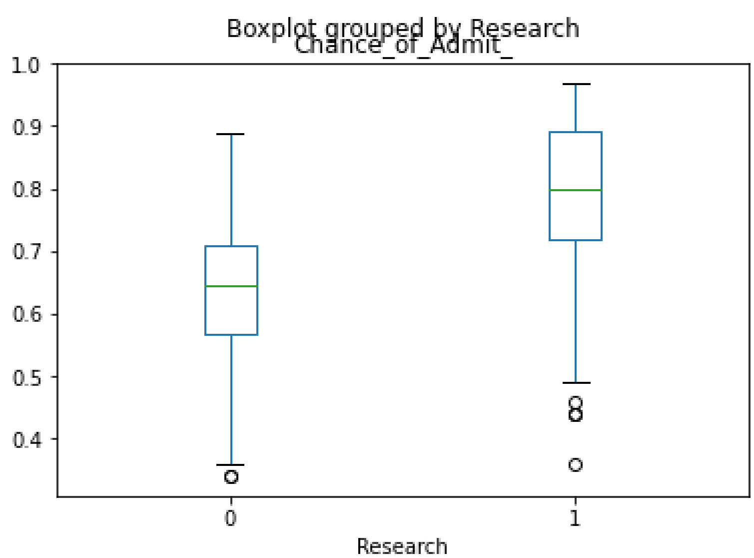


Fig 5

Inference:

This boxplot will tell in which interval of research attribute, chance of admission is high. Here it is a research area of 1 as higher probability.

6.1.2 **Crosstab:**

Crosstab is a method to quantitatively analyse the relationship between variables.Here we see the relationship between gre score and chance of admit.

Code:

```
import pandas as pd
my_data=pd.read_csv('Admission_Predict_Ver1.1.csv')
my_data.columns = [c.replace(' ', '_') for c in my_data.columns]
import numpy as np
my_data['Chance_of_Admit_'] = np.where((my_data.Chance_of_Admit_
>0.6),1,my_data.Chance_of_Admit_)
my_data['Chance_of_Admit_'] = np.where((my_data.Chance_of_Admit_
<=0.6),0,my_data.Chance_of_Admit_)

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df1=pd.crosstab(index=my_data['GRE_Score'],columns=my_data['Chance_of_Admit'])
df1
sns.heatmap(df1)
```

Output:

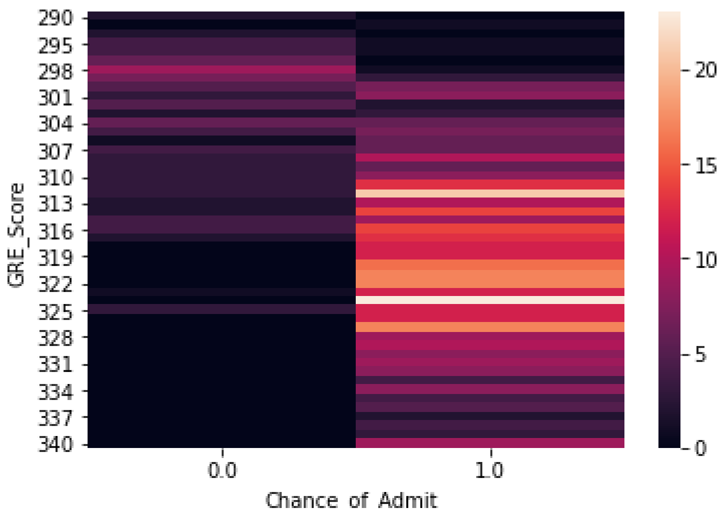


Fig 6

Inference:

- The inference after the visualization is when the marks are high, the black portion is more in “0” which means there are none who don't get a seat after getting a high mark.
- Similarly while seeing the lowest marks it is notable that the black portion is more in “1” which means students with less gre score dont get a seat in college.
- Hence this chart shows that GRE mark plays an important role in determining the chances of admit

6.1.3 **Histogram:**

Histogram can be used to visualise the frequency distribution.

Code:

plt.hist(xAxis)

plt.title('GRE SCORES')

plt.show()

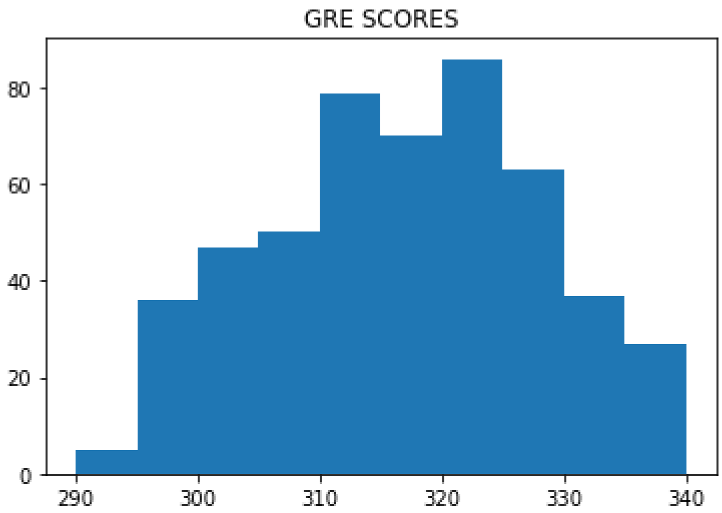


Fig 7

Code:

plt.title('CGPA RATE')

plt.hist(my_data['CGPA'])

plt.show()

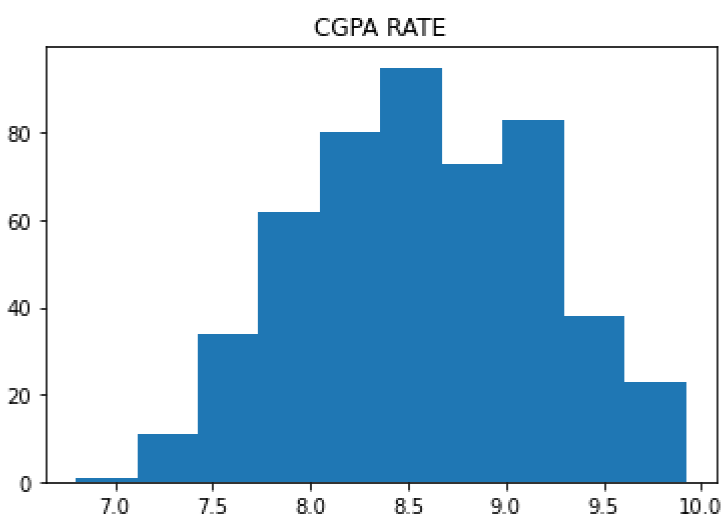


Fig 8

```
import matplotlib.pyplot as plt
plt.hist(my_data["TOEFL_Score"])
plt.title('TOEFL SCORES')
plt.show()
```

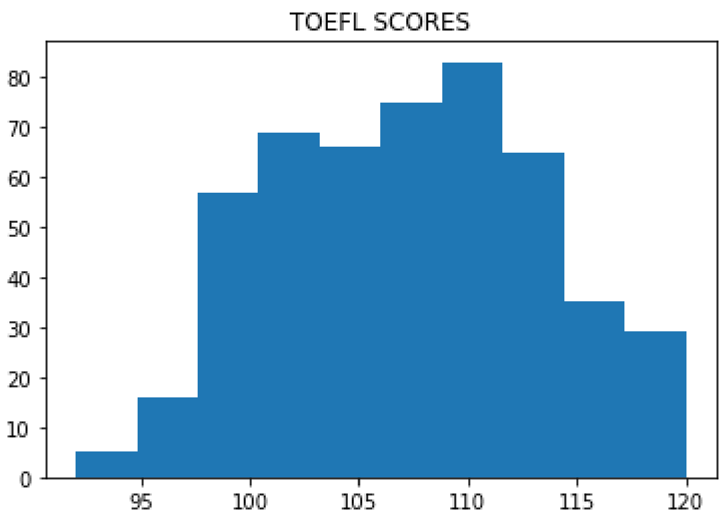


Fig 9

```
import matplotlib.pyplot as plt
plt.hist(my_data["University_Rating"])
plt.title('University Rating')
plt.show()
```

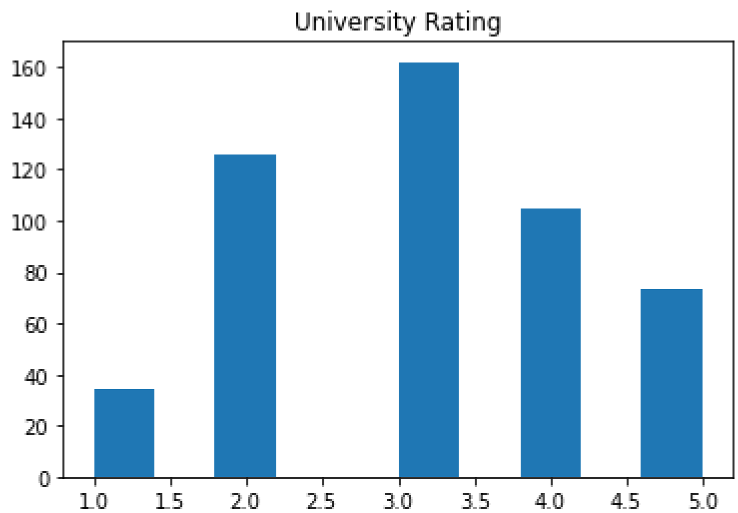
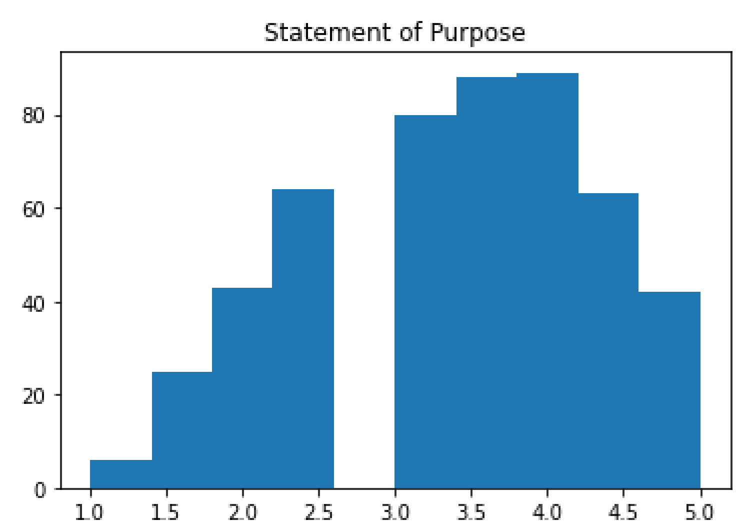


Fig 10

```
import matplotlib.pyplot as plt
plt.hist(my_data["SOP"])
plt.title('Statement of Purpose')
plt.show()
```




```
import matplotlib.pyplot as plt
```

```
plt.hist(my_data["LOR_"])
plt.title('Letter of Recommendation')
plt.show()
```

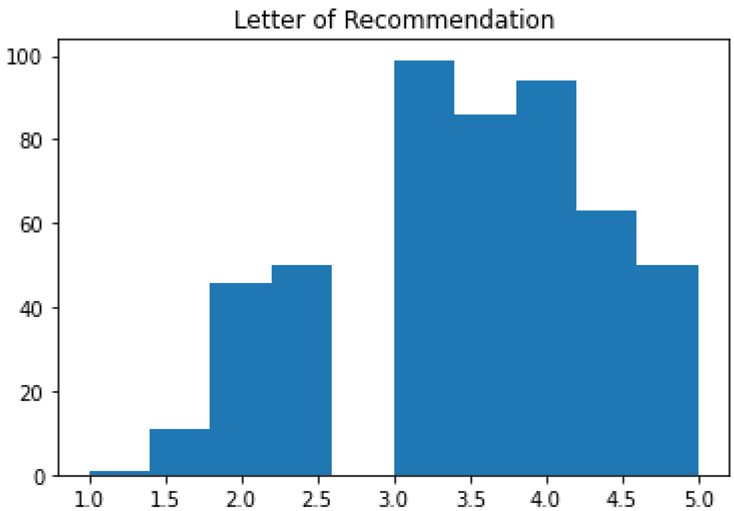


Fig 11

```
import matplotlib.pyplot as plt
plt.hist(my_data["Research"])
plt.title('Research')
plt.show()
```

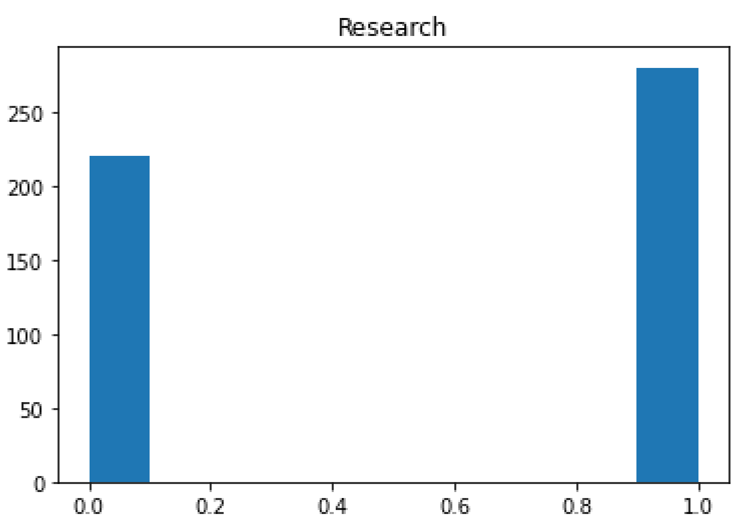


Fig 12

Inference:

- In our dataset we can visualise the score frequency from which we can get an idea of how many people scored a particular score.
- We can also use it to visualise the number of students getting a cgpa.

6.2 Understand the relationship between features:

6.2.1 **Chi squared test:**

Code:

```
import numpy as np
import pandas as pd
from scipy.stats import chi2_contingency
import seaborn as sns
df=pd.read_csv('data.csv')
df
df.columns = [c.replace(' ', '_') for c in df.columns]
data = [df['University_Rating']]
stat, p, dof, expected = chi2_contingency(data)
 alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
   print('Dependent (reject H0)')
else:
   print('Independent (H0 holds true)')
```

Output:

```
In [97]: runfile('C:/Users/baska/Desktop/2/d.py', wdir='C:/Users/baska/Desktop/2')
p value is 1.0
Independent (H0 holds true)
```

Fig 13

Inference:

Since University rating attribute is categorical values from 1 to 5, we have applied chi square test and we got the p value as 1.0 which is greater than 0.05 and so it is dependent on the target attribute called chance of admit so that null hypothesis is accepted.

6.2.2 Pearson Coefficient correlation test

Code:

```
import pandas as pd
import numpy as np
my_data=pd.read_csv('Admission_Predict_Ver1.1.csv')
my_data.columns = [c.replace(' ', '_') for c in my_data.columns]
xAxis = my_data["GRE_Score"]
x2 = my_data["TOEFL_Score"]
x4 = my_data["SOP"]
x5 = my_data["LOR_"]
x6 = my_data["CGPA"]
x7 = my_data["Research"]
yAxis =my_data["Chance_of_Admit_"]
#GRE SCORE VS CHANCE OF ADMIT
my_rho = np.corrcoef(xAxis, yAxis)
print(my_rho)
```

```
In [31]: my_rho = np.corrcoef(xAxis, yAxis)
    ...: print(my_rho)
[[1.         0.81035064]
 [0.81035064 1.        ]]
```

Fig 14

<u># TOEFL VS CHANCES OF ADMIT</u>

my_rho = np.corrcoef(x2, yAxis)

print(my_rho)

```
In [32]: my_rho = np.corrcoef(x2, yAxis)
    ...: print(my_rho)
[[1.         0.79222761]
 [0.79222761 1.        ]]
```

Fig 15

<u>#SOP VS CHANCES OF ADMIT</u>

my_rho = np.corrcoef(x4, yAxis)

print(my_rho)

```
In [34]: my_rho = np.corrcoef(x4, yAxis)
    ...: print(my_rho)
[[1.         0.68413652]
 [0.68413652 1.        ]]
```

Fig 16

<u>#LOR VS CHANCE OF ADMIT</u>

my_rho = np.corrcoef(x5, yAxis)

print(my_rho)

```
In [35]: my_rho = np.corrcoef(x5, yAxis)
    ...: print(my_rho)
[[1.         0.64536451]
 [0.64536451 1.        ]]
```

Fig 17

#CGPA VS CHANCE OF ADMIT

my_rho = np.corrcoef(x6, yAxis)

print(my_rho)

```
In [36]: my_rho = np.corrcoef(x6, yAxis)
    ...: print(my_rho)
[[1.         0.88241257]
 [0.88241257 1.        ]]
```

Fig 18

#RESEARCH VS CHANCE OF ADMIT

my_rho = np.corrcoef(x7, yAxis)

print(my_rho)

```
In [37]: my_rho = np.corrcoef(x7, yAxis)
    ...: print(my_rho)
[[1.         0.54587103]
 [0.54587103 1.        ]]
```

Fig 19

6.2.3 **Visualization:**

sb.heatmap(pearsoncorr,xticklabels=pearsoncorr.columns,yticklabels=pearsoncorr.columns,cmap='RdBu_r',annot=True,linewidth=0.5)

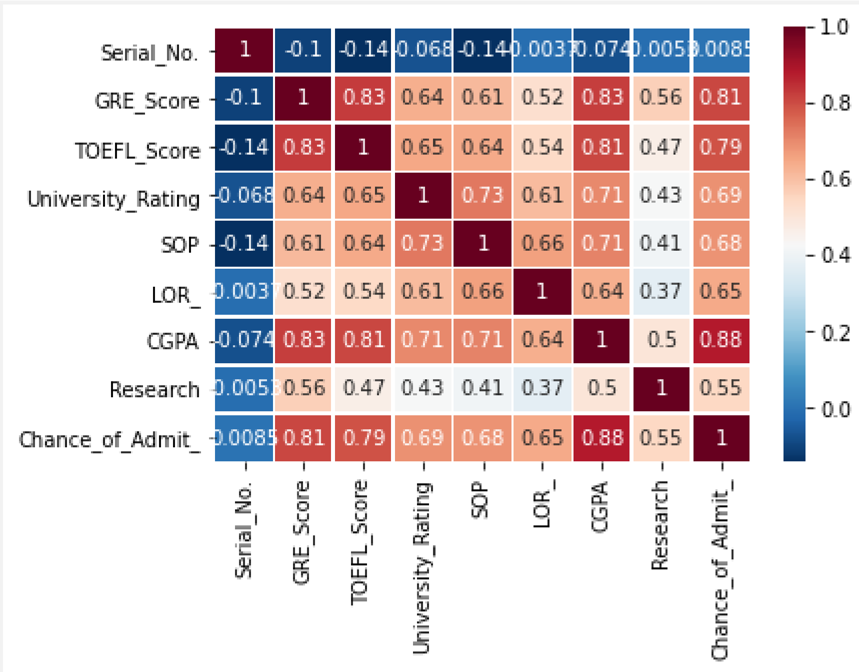


Fig 20

<u>Inference:</u>

Features such as GRE Score, TOEFL Score, CGPA, SOP, LOR and Research are having Pearson Co-efficient- Correlation Analysis as positive with the target attribute called chance of admit. So all these attributes are needed for predicting the chance of admission.The serial number has negative values so we can remove the serial number .

6.3 Inferential Statistics:

6.3.1 <u>Conditional Probability:</u>

- Conditional probability is the probability of one event occurring with some relationship to one or more other events.

- Probability of chance of admit is calculated given that probability of University rating is already calculated.

Code:

```
count=1
count1=1
for label,row in dp.iterrows():
  if(row['University Rating']==1):
    count=count+1
  if(row['University Rating']==1 and row['Chance of Admit ']>0.60):
    count1=count1+1
print("probability of chance of admit/University rating ",count1/count)
```

Output:

```
probability of chance of admit/University rating  0.2857142857142857
```

Fig 21

Inference:

It is seen that the probability of getting a seat with university rating 1 is less.

Code:

```
count=1C
count1=1
for label,row in dp.iterrows():
  if(row['University Rating']==5):
    count=count+1
  if(row['University Rating']==5 and row['Chance of Admit ']>0.60):
```

```
    count1=count1+1
print("probability of chance of admit/University rating ",count1/count)
```

Output:

```
probability of chance of admit/University rating  1.0
```

Fig 22

Inference:

It is seen that the probability of getting a seat with university rating 5 is high.

6.3.2 **Probability distribution function**:

- Normal distribution, also known as the Gaussian distribution, is a probability distribution that is symmetric about the mean, showing that data near the mean are more frequent in occurrence than data far from the mean. In graph form, normal distribution will appear as a bell curve.
- Skewness is a state of distribution where the distribution is highly biased towards the right or left side of the plot.

Probability distribution function:

Code:

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (11, 4)
plt.style.use('fivethirtyeight')

plt.xticks(rotation=30)
sns.distplot(my_data['Chance_of_Admit_'])
plt.title('Distribution of Target Column')
```

plt.show()

Output:

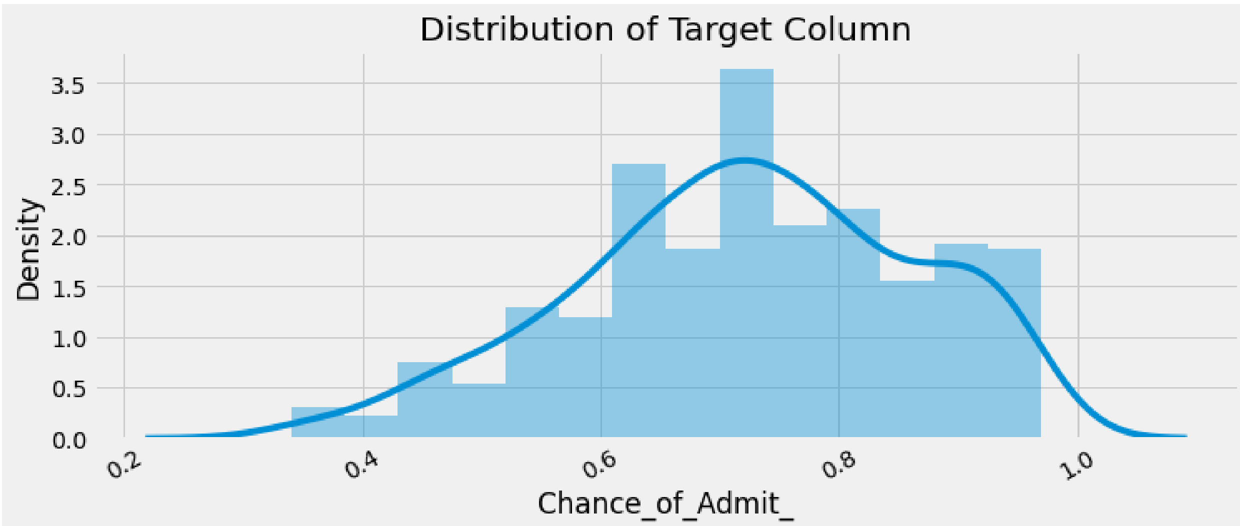


Fig 23

Inference:

The distribution of our target variable “Chances of Admit” does not seem to be a normal distribution.It is slightly skewed to the left. So the data has little outliers.

6.3.3 <u>Sample mean ,Population mean and Confidence Interval:</u>

- The subset of the population is the Sample data.
- We calculate the mean for both sample and population.
- The mean of both would be near or little far.
- Z-critical value is calculated which has been used to calculate the margin of error.
- Margin of error is used to calculate the confidence interval.
- We do this to ensure that we make necessary tests using sample data itself.
- Thus gaining an insight about the larger data using sample data.

Code:

np.random.seed(6)

```
sample_chance=np.random.choice(a= my_data['Chance_of_Admit_'], size=250)
print ("Sample mean:", sample_chance.mean() )
print("Population mean:", my_data['Chance_of_Admit_'].mean())
import scipy.stats as stats
import math
np.random.seed(10)

sample_size = 250
sample = np.random.choice(a= data['Chance_of_Admit_'],size = sample_size)
sample_mean = sample.mean()
z_critical = stats.norm.ppf(q = 0.95)
print("z-critical value: ",z_critical)
pop_stdev = data['Chance_of_Admit_'].std()
margin_of_error = z_critical * (pop_stdev/math.sqrt(sample_size))
confidence_interval = (sample_mean - margin_of_error,sample_mean +
margin_of_error)
print("Confidence interval:",end=" ")
print(confidence_interval)
print("True mean: {}".format(data['Chance_of_Admit_'].mean()))
```

Output:

```
In [74]: runfile('C:/Users/KMK/Desktop/Python folder/dslab.py', wdir='C:/Users/KMK/Desktop/Python folder')
Sample mean: 0.72108
Population mean: 0.7217399999999996
z-critical value:  1.6448536269514722
Confidence interval: (0.7003572092842292, 0.7297227907157711)
True mean: 0.7217399999999996
```

Fig 24

Inference:

The sample mean is usually not exactly the same as the population mean. This difference can be caused by many factors including poor survey design, biased sampling methods and the randomness inherent to drawing a sample from a population.

But we have nearly the same sample mean and population mean and also true mean is contained in our confidence interval. So in our data, there is no such poor survey design.

6.3.4 Hypothesis testing:

- In statistical hypothesis testing, the p-value or probability value is the probability of obtaining test results at least as extreme as the results actually observed during the test, assuming that the null hypothesis is correct.
- For the given attribute in a given condition the mean of that attribute and mean of the target attribute is compared and p value is calculated.
- If p value is less than 0.05 null hypothesis rejected else accepted.

Code:

```
from statsmodels.stats.weightstats import ztest
z_statistic, p_value = ztest(x1 = my_data[my_data['GRE_Score']
>=300]['Chance_of_Admit_'],
                    value = my_data['Chance_of_Admit_'].mean())
# lets print the Results
print('Z-statistic is :{}'.format(z_statistic))
print('P-value is :{:.50f}'.format(p_value))
```

Output:

```
Z-statistic is :3.05468337336045
P-value is :0.00225298237322186763534337394787598896073177456856
```

Fig 25

Inference:

The relationship taken is the mean of chances of admission of students whose gre score is more than 300 is same as that of mean of chances of admit of all students.

The obtained p value is <0.05.

Hence the null hypothesis is rejected.

Code:

```
from statsmodels.stats.weightstats import ztest


z_statistic, p_value = ztest(x1 =
my_data[my_data['GRE_Score']>=250]['Chance_of_Admit_'],
                             value = my_data['Chance_of_Admit_'].mean())


# lets print the Results
print('Z-statistic is :{}'.format(z_statistic))
print('P-value is :{:.50f}'.format(p_value))
```

Output:

```
Z-statistic is :7.03564417775483e-14
P-value is :0.99999999999994382271495396707905456423759460449219
```

Fig 26

Inference:

The relationship taken is the mean of chances of admission of students whose gre score is more than 250 is same as that of mean of chances of admit of all students.

The obtained p value is >0.05.

Hence the null hypothesis is accepted.

Hypothesis testing**:**

Code:

```
from statsmodels.stats.weightstats import ztest
z_statistic, p_value = ztest(x1 = my_data[my_data['GRE_Score']
>=300]['Chance_of_Admit_'],
                              value = my_data['Chance_of_Admit_'].mean())
# lets print the Results
print('Z-statistic is :{}'.format(z_statistic))
print('P-value is :{:.50f}'.format(p_value))
```

Output:

```
Z-statistic is :3.05468337336045
P-value is :0.00225298237322186763534337394787598896073177456856
```

Fig 27

Inference:

The relationship taken is the mean of chances of admission of students whose gre score is more than 300 is same as that of mean of chances of admit of all students.

The obtained p value is <0.05.

Hence the null hypothesis is rejected.

Code:

```
from statsmodels.stats.weightstats import ztest

z_statistic, p_value = ztest(x1 = my_data[my_data['GRE_Score']
>=250]['Chance_of_Admit_'], value = my_data['Chance_of_Admit_'].mean())

# lets print the Results
print('Z-statistic is :{}'.format(z_statistic))
print('P-value is :{:.50f}'.format(p_value))
```

Output:

```
Z-statistic is :7.03564417775483e-14
P-value is :0.99999999999994382271495396707905456423759460449219
```

Fig 28

Inference:

The relationship taken is the mean of chances of admission of students whose gre score is more than 250 is same as that of mean of chances of admit of all students.

The obtained p value is >0.05.

Hence the null hypothesis is accepted.

6.3.5 **Chi square test**:

- Chi square test is applied to university rating attributes since it is categorical.

Chi- Squared test:

Code:

```
import numpy as np
import pandas as pd
from scipy.stats import chi2_contingency
import seaborn as sns
df=pd.read_csv('data.csv')
df.columns = [c.replace(' ', '_') for c in df.columns]
data = [df['University_Rating']]
stat, p, dof, expected = chi2_contingency(data)
 alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```

Output:

```
In [97]: runfile('C:/Users/baska/Desktop/2/d.py', wdir='C:/Users/baska/Desktop/2')
p value is 1.0
Independent (H0 holds true)
```

Fig 29

Inference:

Since University rating attribute is categorical values from 1 to 5, we have applied chi square test and we got the p value as 1.0 which is greater than 0.05 and so it is dependent on the target attribute called chance of admit so that null hypothesis is accepted.

6.4 **Proposed Work**:

6.4.1Data preprocessing:

1)The dataset is understood by classifying the features according to its type of data.

University Rating - Ordinal

SOP,LOR,SOP,GRE,CGPA,TOEFL - continuous

Research - Binary

2)The dataset contains an independent variable called serial number. When a heat map with respect to the pearson coefficient is implemented it is seen that the dependent variable is not dependent on the serial number.Hence we remove serial number from the dataset. There are no missing values in any row of the dataset.

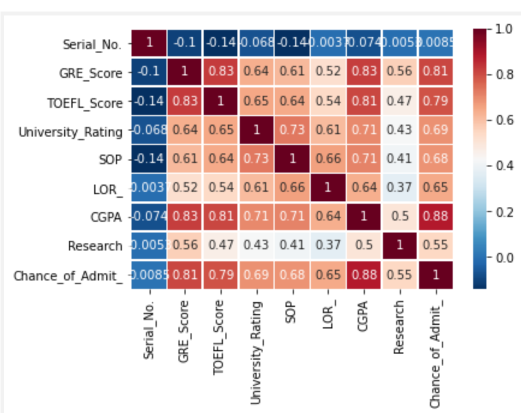


Fig 30

3)Outliers are the data values that differ greatly from the dataset. They are not needed for the prediction. This can be found using box plot. In the box plot, the middle part represents the first and third quartiles. The line near the middle of the box represents the median. The whiskers on either side of the IQR

represent the lowest and highest quartiles of the data. The ends of the whiskers represent the maximum and minimum of the data, and the individual circles beyond the whiskers represent outliers in the dataset.

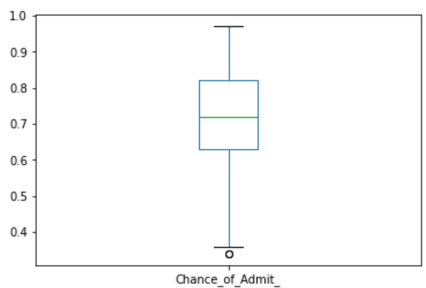


Fig 31

From the above figure, we can identify that some outliers are present. We can handle this in three ways:

i)Remove them from the dataset

ii)Replace them with mean value

iii)Replace them with median values. Here we handled outliers by replacing them with the median value.

4)The target variable containing continuous values were converted to categorical values. The values greater than 0.6 are converted as 1. And the values less than 0.6 are converted as 0.

6.4.2 Dataset Division:

The dataset was divided into training and testing sets.The testing set was 20 percent of the total data.The feature data X was divided to Xtrain and Xtest and Y was divided to Ytrain and Ytest.

6.4.3 Model selection:

Since it is a binary classification, we have chosen logistic regression. This is implemented using the linear model present in sklearn. The accuracy score obtained is 89%.

7.**MATHEMATICAL MODELLING**

- The original dataset contains independent variables and one dependent variable.This dependent variable was continuous.
- Inorder to improve accuracy, the dependent variable was changed to 0 and 1 to apply logistic regression.if value is>=0.6 it is 1,otherwise it is 0.
- Hence binary classification is applied,the dataset is divided into test and training dataset.
- 20 percent of the dataset was taken as a test set and the remaining as a train set.
- We have chosen logistic regression. This is implemented using the linear model present in sklearn. The accuracy score obtained is 89%.

8.**CODE SNIPPET**:

```
import pandas as pd
df=pd.read_csv('Admission_Predict_Ver1.1.csv',index_col=0)
df.columns = [c.replace(' ', '_') for c in df.columns]
import numpy as np
median = df.loc[df['Chance_of_Admit_']>=0.35,
'Chance_of_Admit_'].median()
df.loc[df.Chance_of_Admit_<= 0.35, 'Chance_of_Admit_'] = np.nan
df.fillna(median,inplace=True)
df['Chance_of_Admit_'] = np.where((df.Chance_of_Admit_
>0.6),1,df.Chance_of_Admit_)
df['Chance_of_Admit_'] = np.where((df.Chance_of_Admit_
<=0.6),0,df.Chance_of_Admit_)
from sklearn import linear_model
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=1)
reg = linear_model.LogisticRegression()
reg.fit(X_train, y_train)
pred_prob1 = reg.predict(X_test)
# variance score: 1 means perfect prediction
print("Logistic Regression:")
```

```
print('Accuracy: {}'.format(reg.score(X_test, y_test)*100))
print("Classification Report:")
print(classification_report(y_test,reg.predict(X_test)))
fpr,tpr,thresholds = roc_curve(y_test, reg.predict(X_test))
import statsmodels.api as sm
log_clf =sm.Logit(y_train,X_train)
classifier = log_clf.fit()
y_pred = classifier.predict(X_test)
print(classifier.summary2())
```

9.**RESULTS AND DISCUSSION:**

```
Logistic Regression:
Accuracy: 89.0
Classification Report:
              precision    recall  f1-score   support

         0.0       0.75      0.40      0.52        15
         1.0       0.90      0.98      0.94        85

    accuracy                           0.89       100
   macro avg       0.83      0.69      0.73       100
weighted avg       0.88      0.89      0.88       100
```

Fig 32

```
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.404
Dependent Variable: Chance_of_Admit_ AIC:              252.4047
Date:               2021-04-28 20:23 BIC:              280.3449
No. Observations:   400              Log-Likelihood:   -119.20
Df Model:           6                LL-Null:          -200.16
Df Residuals:       393              LLR p-value:      2.3246e-32
Converged:          1.0000           Scale:            1.0000
No. Iterations:     8.0000
-----------------------------------------------------------------
                   Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
-----------------------------------------------------------------
GRE_Score         -0.0985   0.0209 -4.7226 0.0000 -0.1394 -0.0576
TOEFL_Score        0.0343   0.0542  0.6334 0.5265 -0.0719  0.1406
University_Rating  0.3747   0.2326  1.6107 0.1073 -0.0813  0.8306
SOP                0.4576   0.2454  1.8645 0.0622 -0.0234  0.9387
LOR_               0.4342   0.2428  1.7882 0.0737 -0.0417  0.9102
CGPA               2.9520   0.5803  5.0868 0.0000  1.8146  4.0895
Research           1.0505   0.3626  2.8971 0.0038  0.3398  1.7611
=================================================================
```

Fig 33

The confidence interval for logistic regression is [0.025,0.975]. Hence from this, we can conclude that logistic regression provides 97.5% confidence interval which is a range of values that has 97.5% certain contains the true mean of the population.

```
In [218]: confusion_matrix(y_test, reg.predict(X_test))
Out[218]:
array([[ 6,  9],
       [ 2, 83]], dtype=int64)
```

Fig 34

True positive - 6 (actually yes, predicted also yes)
False positive - 9 (actually yes, predicted is no)

False negative - 2 (actually no, predicted is yes)
True negative - 83 (actually no, predicted also no)

```
In [219]: accuracy_score(y_test, reg.predict(X_test))
Out[219]: 0.89
```

Fig 35

After referring to many papers , it was clearly seen that the highest accuracy they got was 83%. We after taking the above measures got an accuracy of 89%

10. **CONCLUSION AND FUTURE WORK:**

In this paper, logistic regression is used through which the model is built to predict the opportunity of a student to get admitted to a master's program. This model can tell a student can or cannot get admission of up to 89 percent accuracy.

In future, more models will be trained on large datasets through which the model will be trained for more accuracy to give the best performance.

11.**REFERENCES**:

[1]https://www.kaggle.com/mohansacharya/graduate-admissions

[2]https://towardsdatascience.com/predicting-ms-admission-afbad9c5c599

[3]https://www.researchgate.net/publication/348433004_Graduate_Admission_Prediction_Using_Machine_Learning

[4]https://www.kaggle.com/vermsn01/verma-datamodeling-graduate

[5]https://dl.acm.org/doi/10.1145/3388818.3393716

[6]https://towardsdatascience.com/graduate-admission-prediction-using-machine-learning-8e09ba1af359

[7]https://tanwirkhan.medium.com/predicting-graduate-admissions-using-linear-regression-bab05b6988b5

[8]https://www.cs.utexas.edu/users/ai-lab/downloadPublication.php?filename=http://www.cs.utexas.edu/users/nn/downloads/papers/waters.iaai13.pdf&pubid=127269

[9]https://www.ijrte.org/wp-content/uploads/papers/v8i6/F9043038620.pdf

[10]https://www.kaggle.com/adepvenugopal/predicting-graduate-admissions-using-ml

[11]https://ieeexplore.ieee.org/abstract/document/8862140