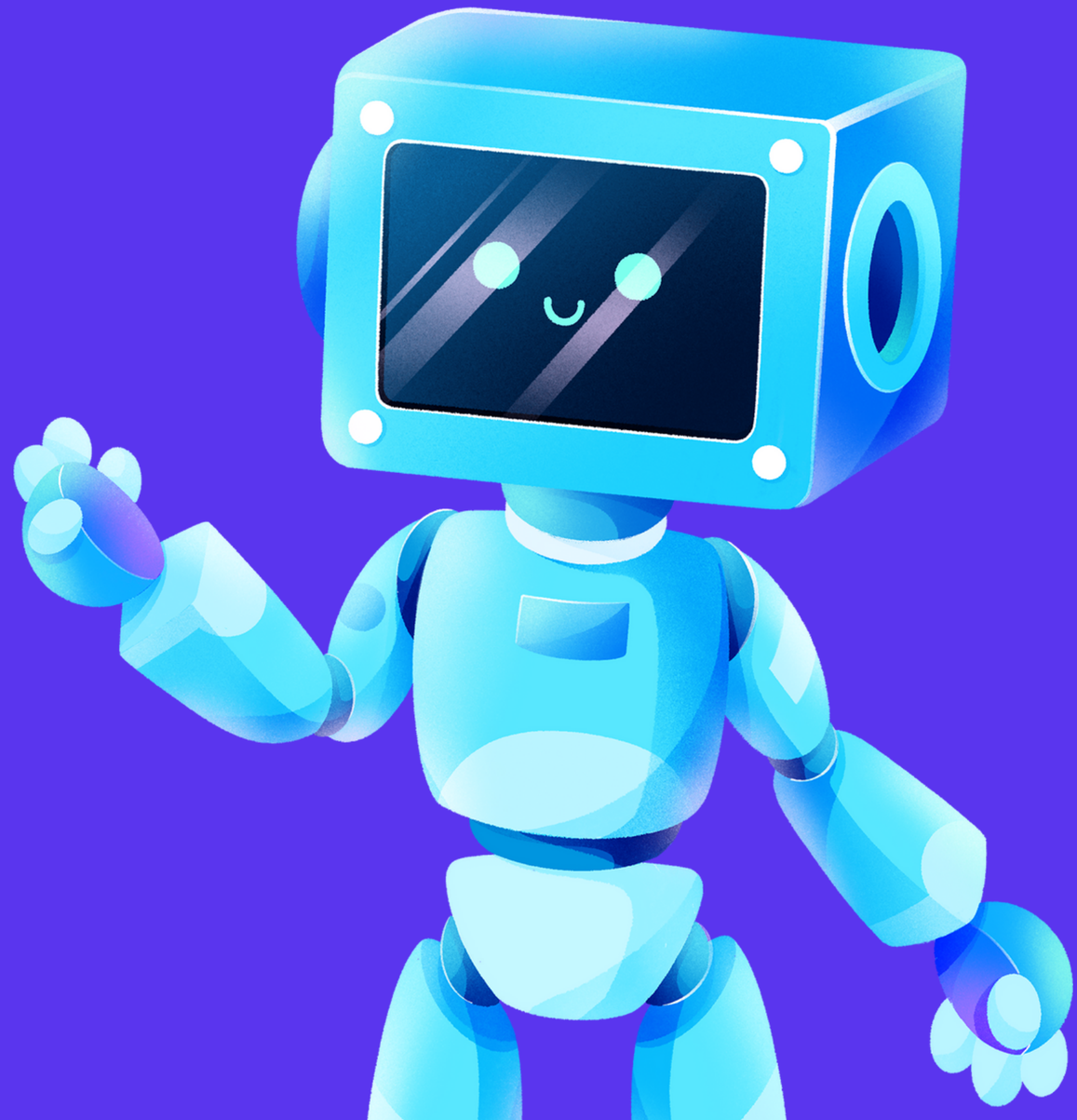
GRADUATE ADMISSIONS PREDICTION

By Myo Pone Pone Swe

2016-MIIT-CSE-023



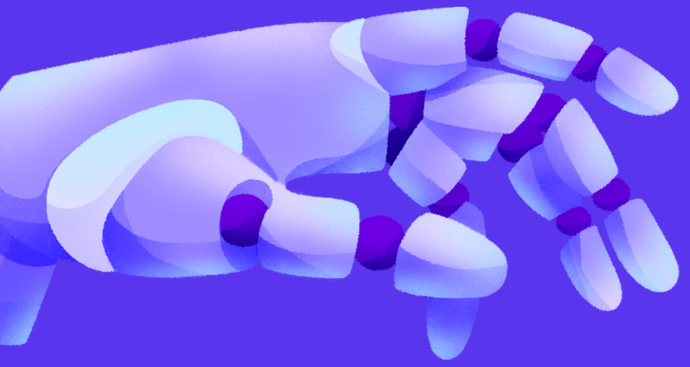
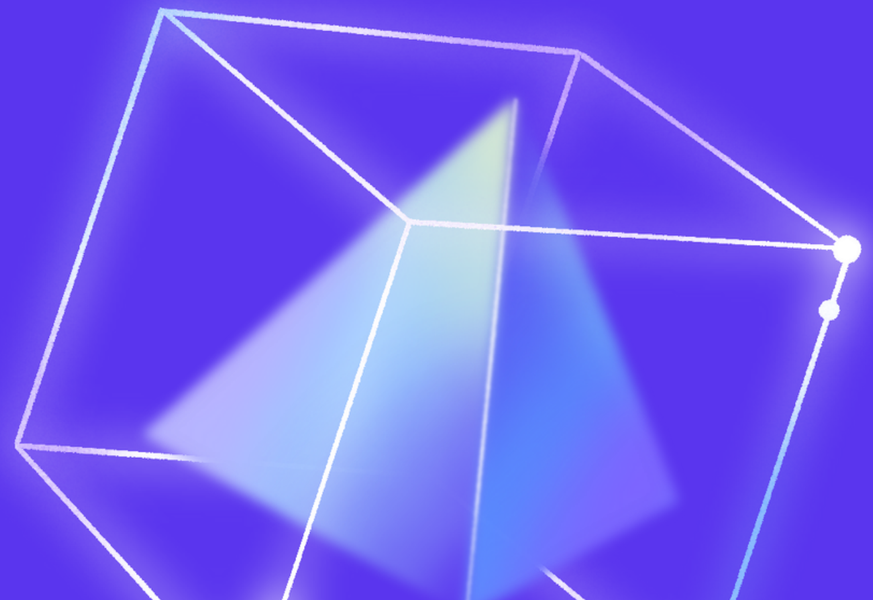


TABLE OF CONTENTS



INTRODUCTION

This project involves the analysis and prediction of graduation admission using the graduate admission dataset from Kaggle. We aim to develop a predictive model that estimates the likelihood of a student's admission based on various attributes such as GRE scores, TOEFL scores, university ranking, and more. The analysis will provide valuable insights into the admissions process, aiding both applicants and universities in their decision-making.

PROJECT OBJECTIVES

OBJECTIVE 01

To find the probability for a student to get an admit in the university before applying

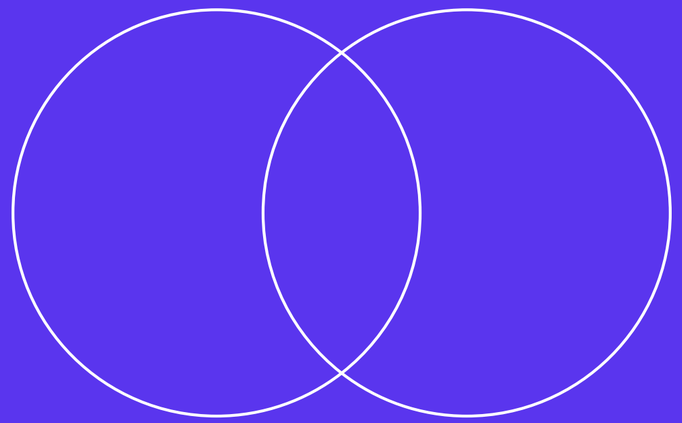
OBJECTIVE 02

To develop a model based on previous years data of the student who got admits or rejects in a particular university

OBJECTIVE 03

Help students access their chances and assist universities in making fair admission choice

Motivation of the project



The motivation behind this project lies in the desire to address the following:

·The uncertainty faced by applicants during the graduate admission process.

·The need for universities to efficiently evaluate and admit candidates.

·The potential for data-driven insights to improve transparency and fairness in admissions.

DATASET DESCRIPTION

- SERIAL NUMBER
- GRE SCORES (OUT OF 340)
- TOEFL SCORES (OUT OF 120)
- UNIVERSITY RATING(OUT OF 5)
- RESEARCH EXPERIENCE (EITHER 0 OR 1)
- LETTER OF RECOMMENDATION STRENGTH (OUT OF 5)
- STATEMENT OF PURPOSE STRENGTH (OUT OF 5)
- UNDERGRADUATE GPA (OUT OF 10)
- CHANCE OF ADMIT (RANGING FROM 0 TO 1)

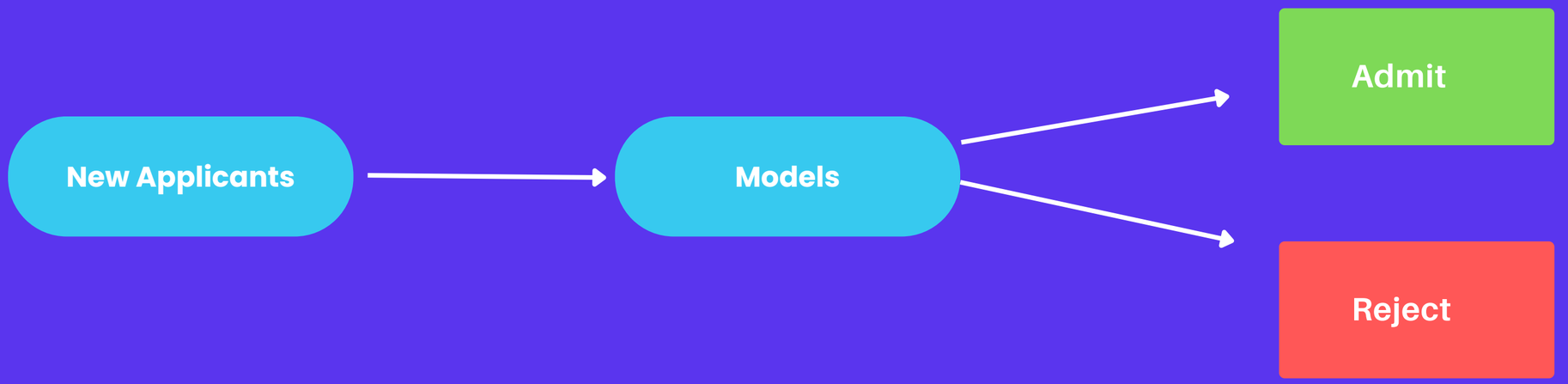
STUDENT SELECTION

Past years data

Preprocessing Techniques

Machine Learning Models

Predictions



PROCESSES

01

Data Cleaning

- handle missing data and outliers
- encode categorical variables
- perform normalization

02

Exploratory Data Analysis

- visualize data distributions,correlations and patterns
- identify relationships between features and the target variable

03

Model Development

- divide the dataset into training and testing sets for model development and evaluation
- build and train a predictive model using machine-learning techniques
- evaluate the model's accuracy using appropriate metrics

04

Conclusion

- aim to predict graduate admission chances using different models and compare the accuracy .

IMPORT THE LIBRARIES

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  %matplotlib inline
5  import seaborn as sns
6  import plotly.express as px
```

IMPORT THE DATA TO DATA FRAME

```
df = pd.read_csv("Admission_Predict.csv")
```

CHECK INFORMATION OF THE DATASET

```
1  #shape of the dataset
2  df.shape
```

✓ 0.0s

```
(396, 8)
```

```
1  #colums we have in dataset
2  df.columns
```

✓ 0.0s

```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

```
1  #check data types
2  df.dtypes
```

✓ 0.0s

```
Serial No.            int64
GRE Score             int64
TOEFL Score           int64
University Rating     int64
SOP                 float64
LOR                 float64
CGPA                float64
Research              int64
Chance of Admit     float64
dtype: object
```

```
1  df.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         400 non-null    int64
 1   GRE Score          400 non-null    int64
 2   TOEFL Score        400 non-null    int64
 3   University Rating  400 non-null    int64
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64
 8   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 28.3 KB
```

```
1  #statistical summary of the data
2  df.describe()
```

✓ 0.0s

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

Data preprocessing

```
#Check if there duplicate
df.duplicated().any()
```

✓ 0.0s

```
False
```

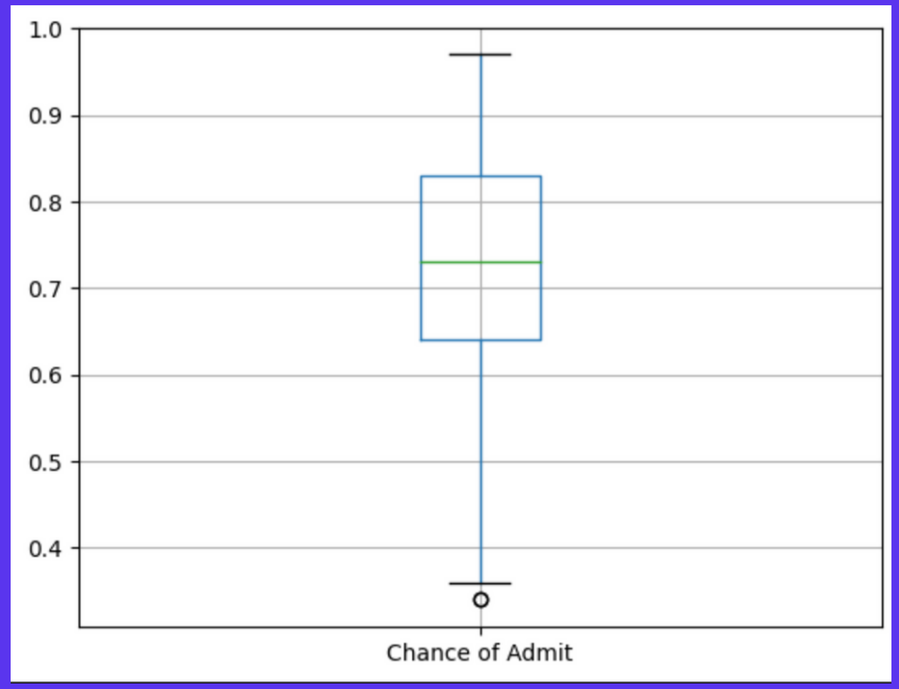
```
#check null values
df.isnull().sum()
```

```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

Observation: No duplicated and missing values found.

Identifying and checking outliers

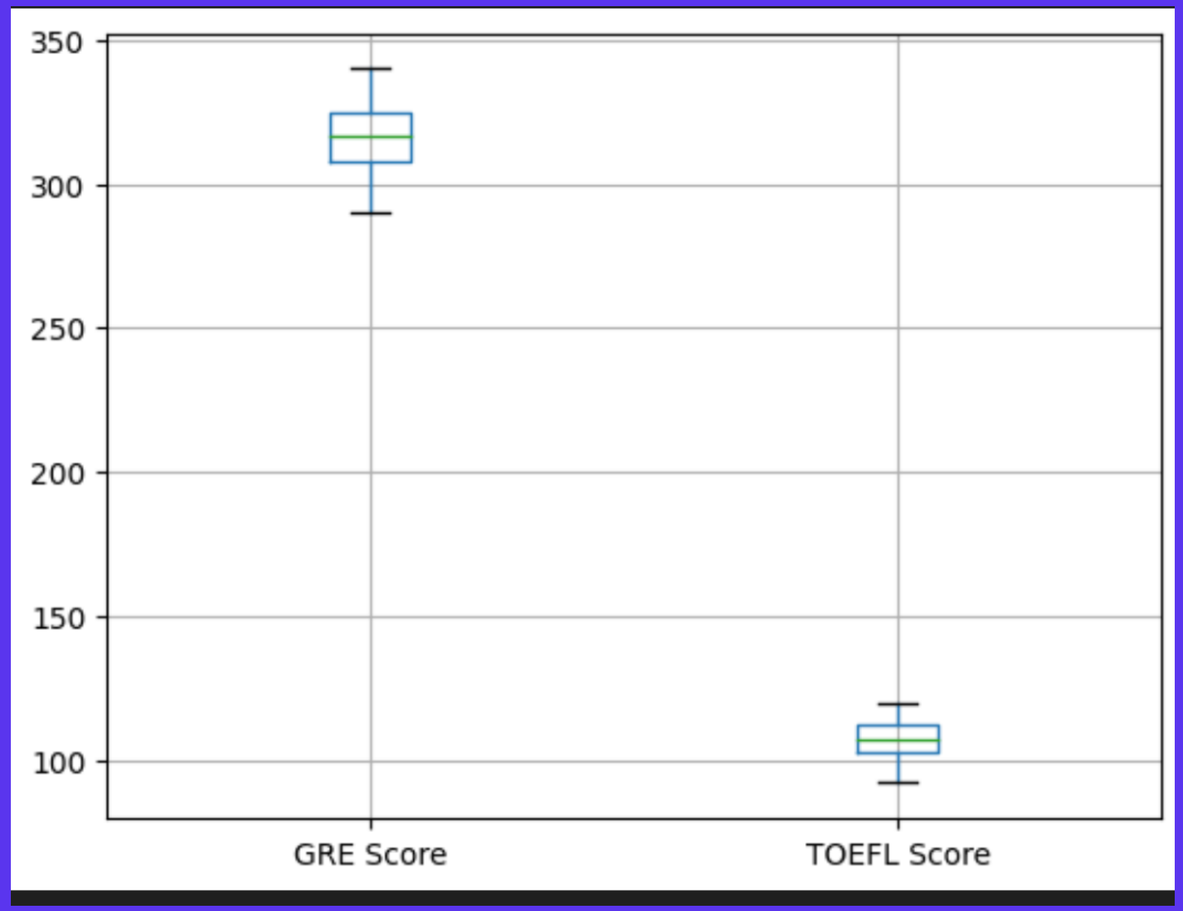
```
df.boxplot(column=['Chance of Admit'])
```



Observations:

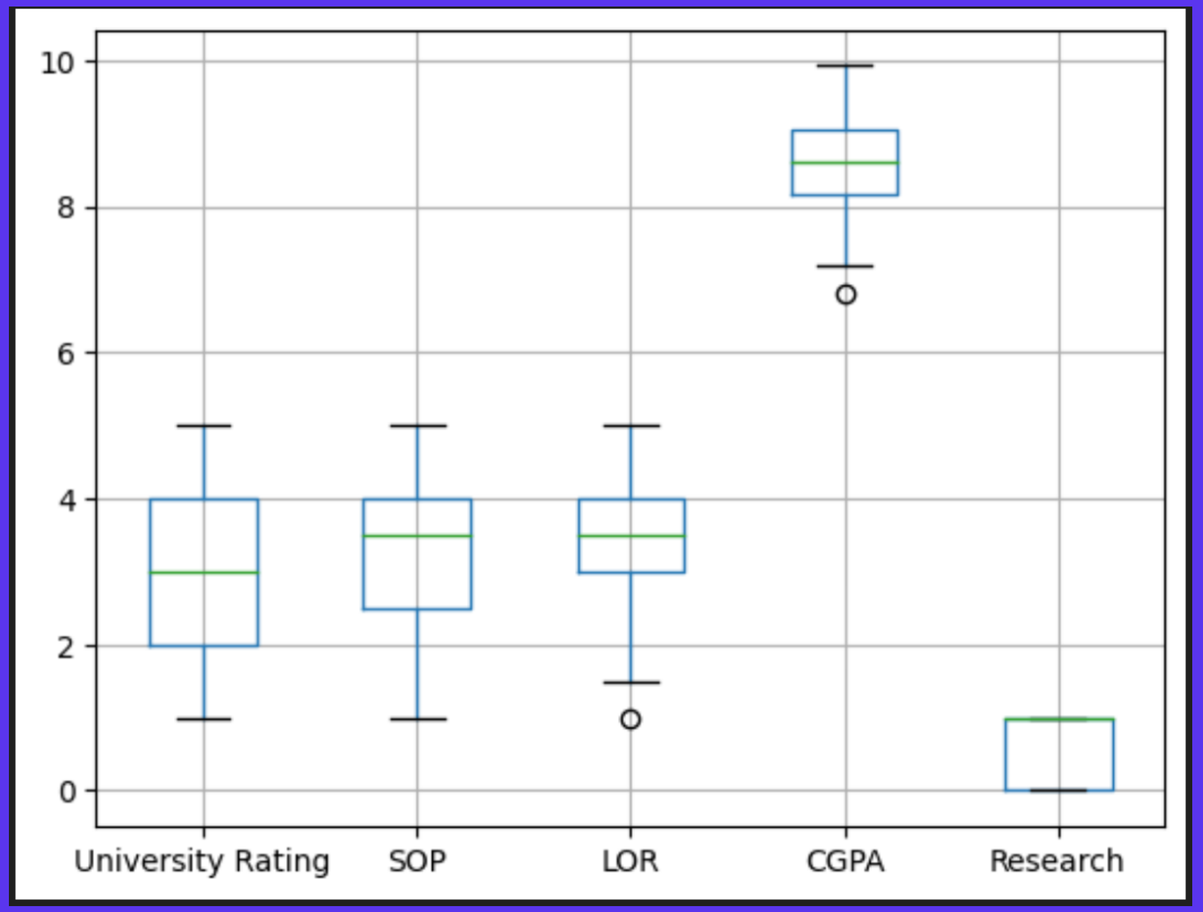
- Chance of admit have the outlier values because the circle value is lower than the min

```
1  df.boxplot(column=['GRE Score','TOEFL Score'])
2
```



Observation: No outliers found.

```
df.boxplot(column=['University Rating','SOP','LOR','CGPA','Research'])
```



Observation:

- LOR and CGPA have outliers

Removing Outliers using IQR method

```
#Calculating Quartiles of each features
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR =Q3 - Q1
print(IQR)
```

```
GRE Score            17.0000
TOEFL Score           9.0000
University Rating     2.0000
SOP                   1.5000
LOR                   1.0000
CGPA                  0.8925
Research              1.0000
Chance of Admit       0.1900
dtype: float64
```

Checking outliers data

```
outlier = df[((df<(Q1-1.5*IQR))|(df>(Q3+1.5*IQR))).any(axis=1)]
outlier.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
58	300	99	1	3.0	2.0	6.80	1	0.36
92	298	98	2	4.0	3.0	8.03	0	0.34
347	299	94	1	1.0	1.0	7.34	0	0.42
376	297	96	2	2.5	2.0	7.43	0	0.34

```
not_outlier = df[~((df<(Q1-1.5*IQR))|(df>(Q3+1.5*IQR))).any(axis=1)]
df= not_outlier.copy()
```

```
df.info()
```

Observation :

4 outliers data are removed from the dataset.

```
<class 'pandas.core.frame.DataFrame'>
Index: 396 entries, 0 to 399
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   GRE Score          396 non-null    int64  
 1   TOEFL Score        396 non-null    int64  
 2   University Rating  396 non-null    int64  
 3   SOP                396 non-null    float64
 4   LOR                396 non-null    float64
 5   CGPA               396 non-null    float64
 6   Research           396 non-null    int64  
 7   Chance of Admit    396 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 27.8 KB
```

Average requirements of all features to get admission for all universities on the basis of their Ratings.

```
# Groupby the data by "University rating".
df.groupby("University Rating").mean()
```

✓ 0.0s

University Rating	GRE Score	TOEFL Score	SOP	LOR	CGPA	Research	Chance of Admit
1	303.153846	99.076923	1.884615	2.211538	7.745769	0.192308	0.548077
2	309.177570	103.523364	2.705607	2.925234	8.183738	0.299065	0.625981
3	315.954887	106.887218	3.364662	3.402256	8.552256	0.533835	0.711880
4	324.824324	111.824324	4.108108	4.006757	9.021622	0.797297	0.818108
5	328.333333	113.666667	4.500000	4.358333	9.291167	0.866667	0.888167

Minimum requirements for more than 60% chance to get admission.

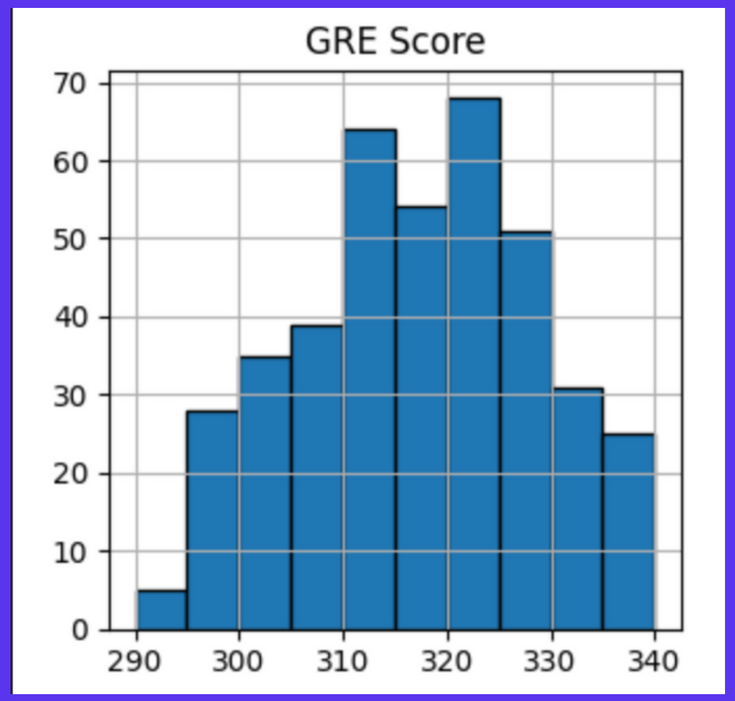
```
1  df[(df['Chance of Admit']>0.6)].min()
```

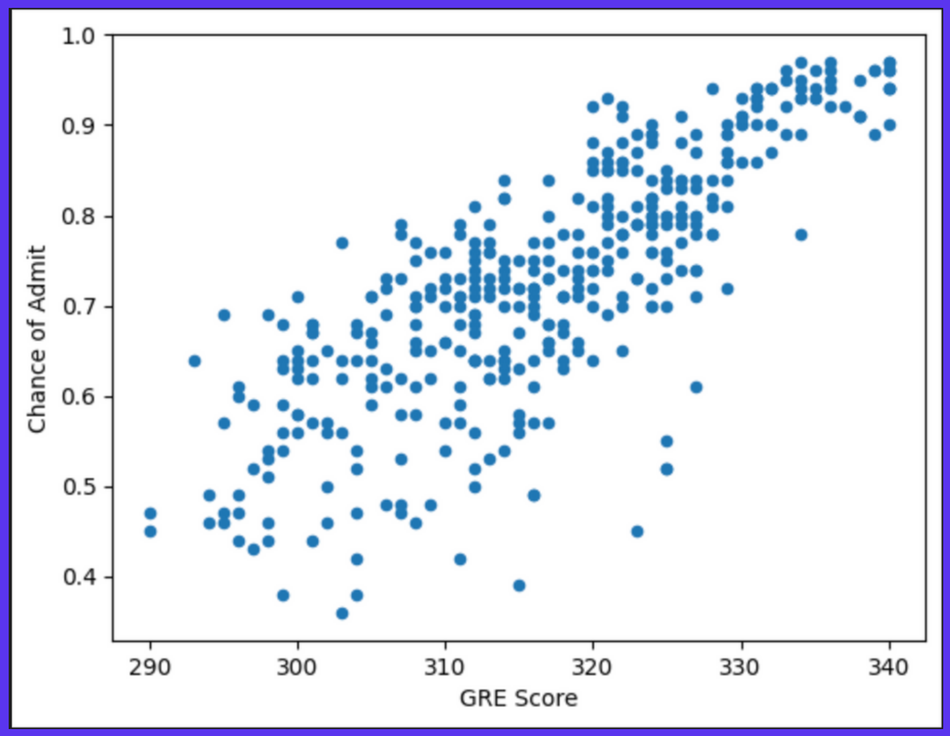
✓ 0.0s Pyt

```
GRE Score           293.00
TOEFL Score          97.00
University Rating     1.00
SOP                   1.50
LOR                   1.50
CGPA                  7.40
Research              0.00
Chance of Admit       0.61
dtype: float64
```

Exploratory Data Analysis

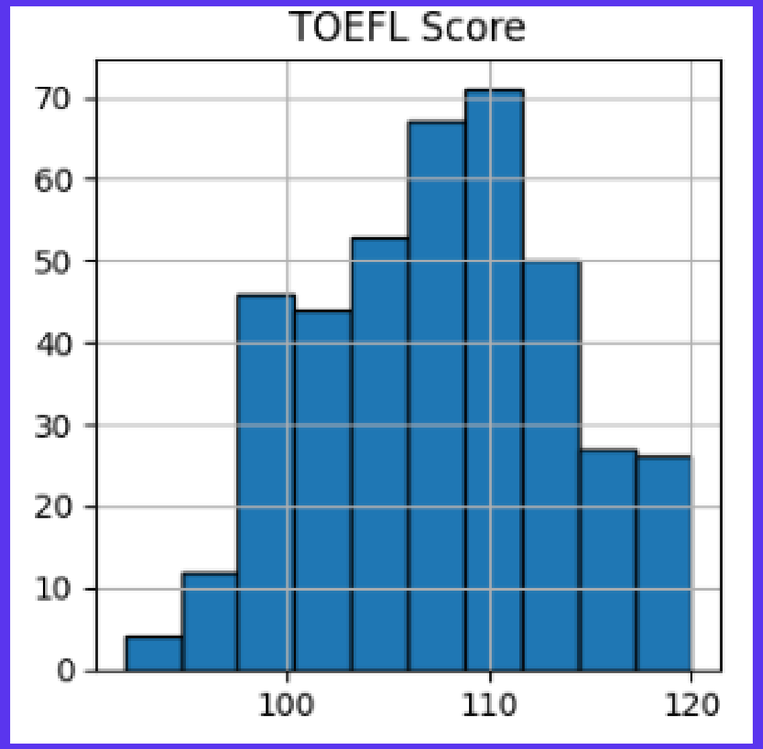
The Graduate Record Examinations (GRE) is a standardized test that is an admissions requirement for many graduate schools in the United States and Canada. It aims to measure verbal reasoning, quantitative reasoning, analytical writing, and critical thinking skills.

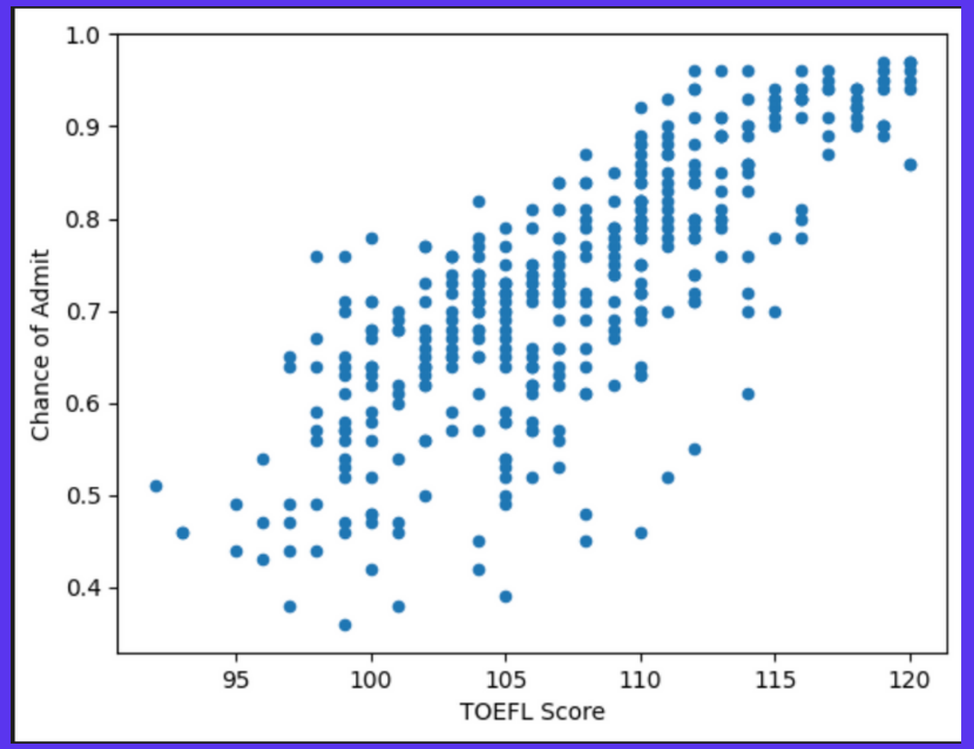




Observation :
Maximum students get between 320 and 325 scores.
Students who have high GRE Score have higher chances of getting an admit.

The Test of English as a Foreign Language, or TOEFL, is a test which measures people's English language skills to see if they are good enough to take a course at university or graduate school in English-speaking countries. It measures how well a person uses listening, reading, speaking and writing skills to perform academic tasks. The TOEFL iBT test is scored on a scale of 0 to 120 points.

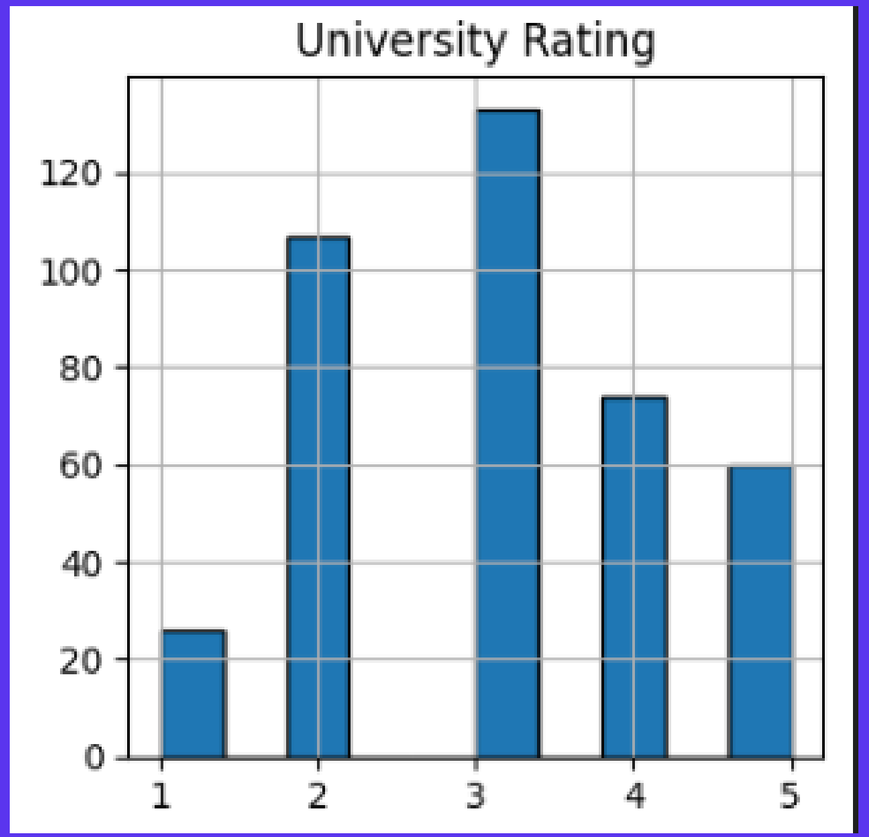




Observation :
Students who have high TOEFL scores have higher chances of getting an admit.

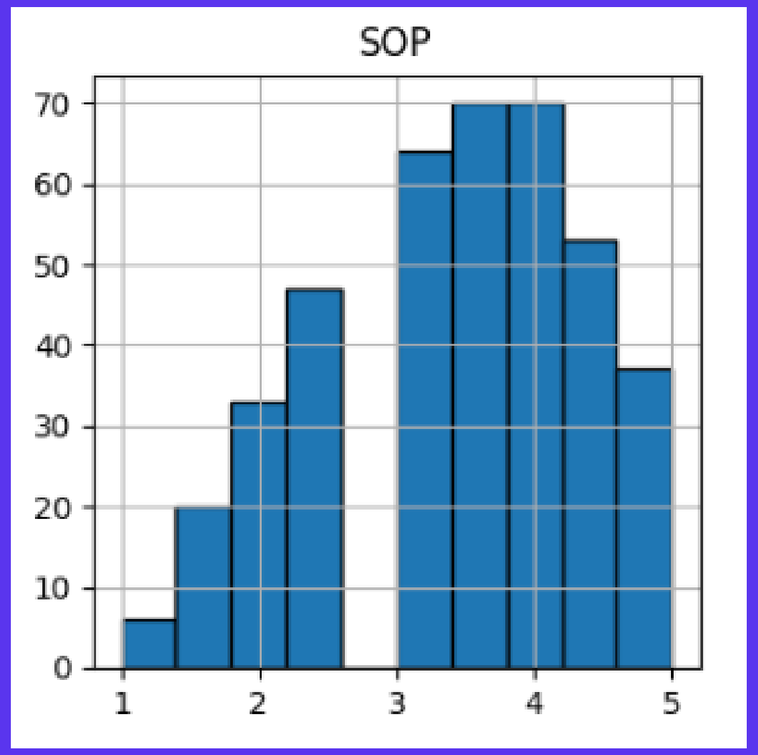
University Rating

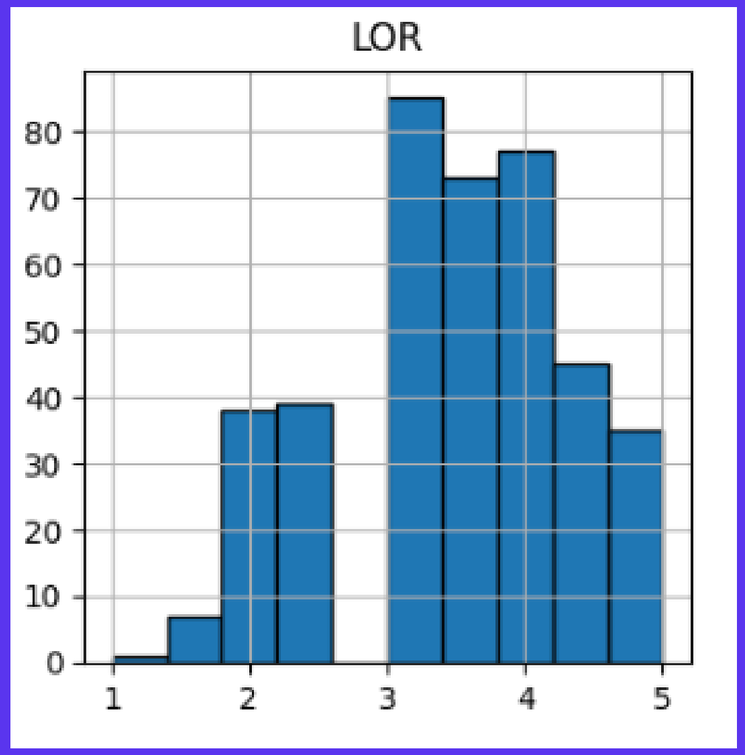
It's a value between 1 and 5 which represents the prestigious of the universities (1 min - 5 max).



Statement of Purpose and Letter of Recommendation Strength

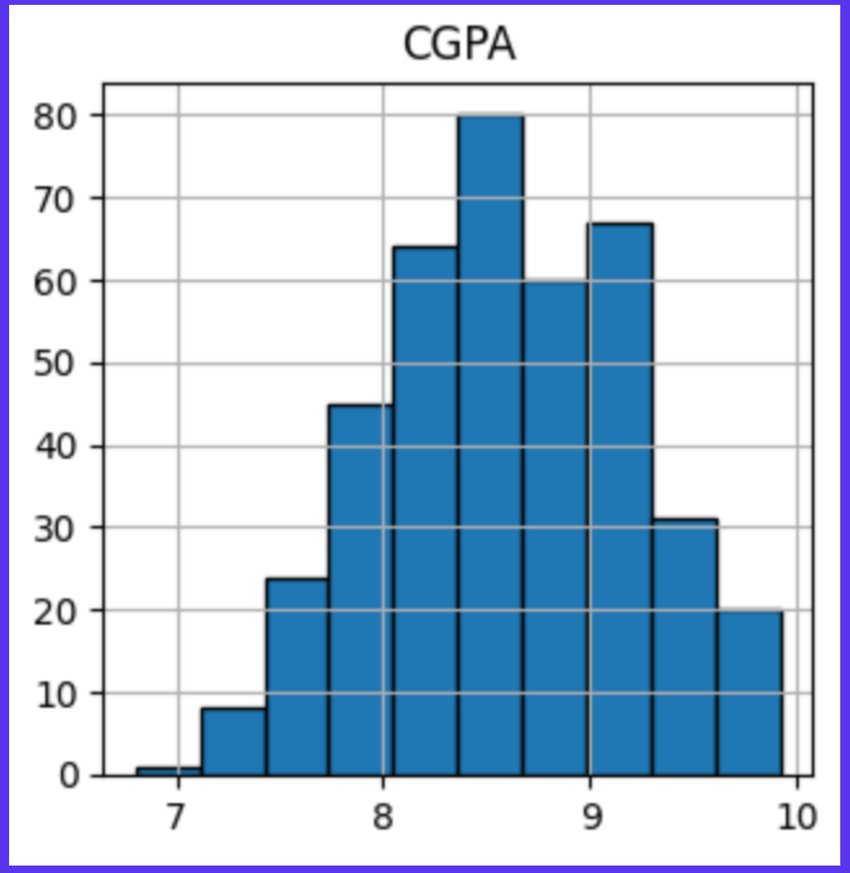
They are two different metrics which measure the quality of the Statement of Purpose and of the Letter of Recommendation Strength. Both range beween 1 and 5.
The Statement of Purpose (SOP) is an essay or other written statement written by an applicant, often a prospective student applying to some college, university, or graduate school.
A Letter of Recommendation (LOR) or recommendation letter, also known as a letter of reference, reference letter or simply reference, is a document in which the writer assesses the qualities, characteristics, and capabilities of the person being recommended in terms of that individual's ability to perform a particular task or function.

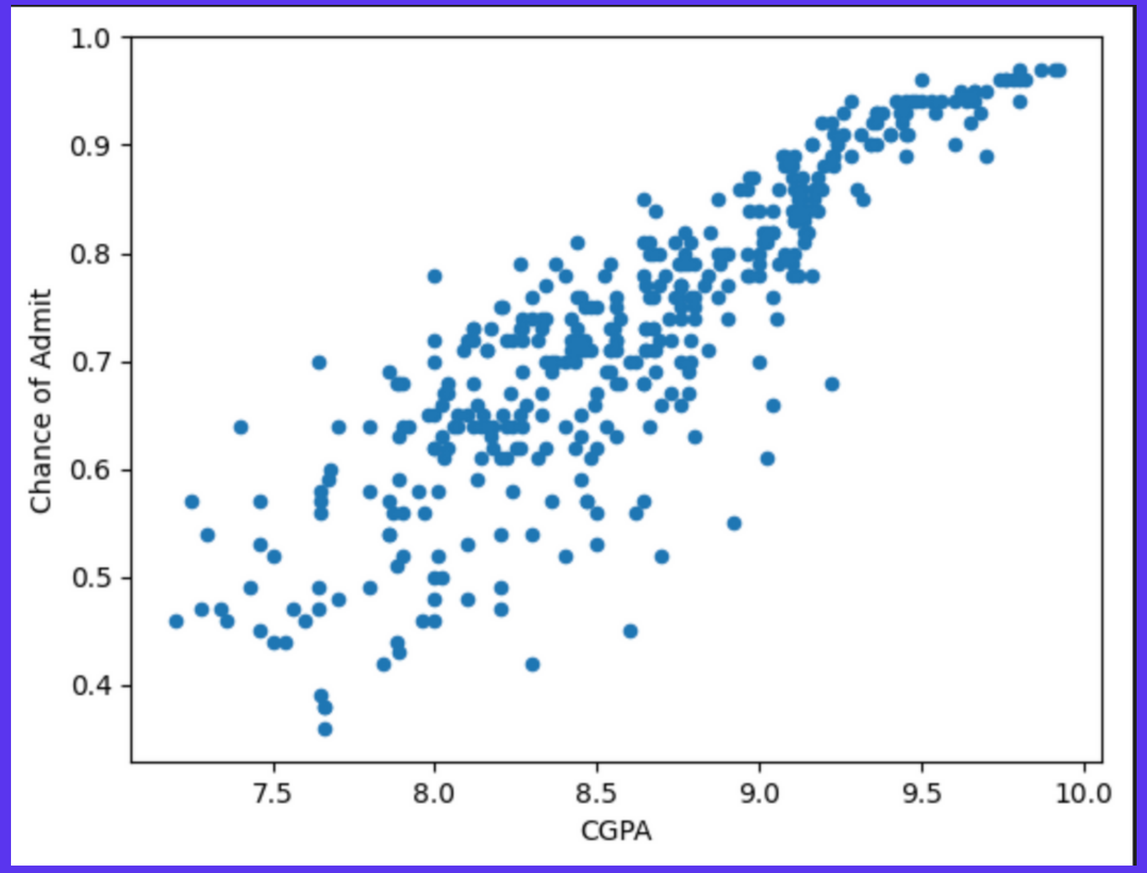




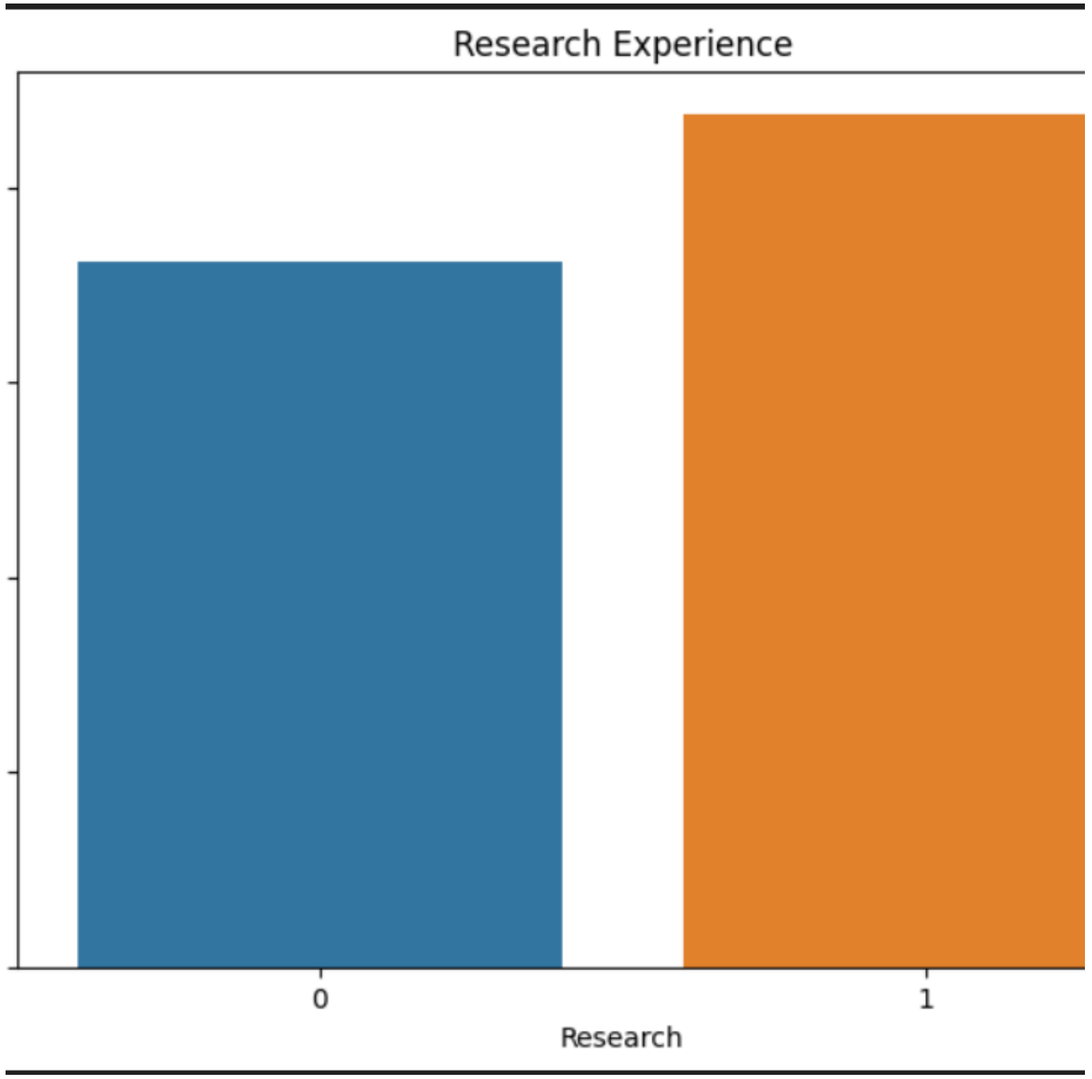
Undergraduate CGPA

CGPA is a grade pointing system used in the educational sector. Cumulative Grade Points is the average of grade points obtained in all the subjects (Note:- excluding the sixth additional subject). In our dataset, it ranges between 6.80 and 9.92.



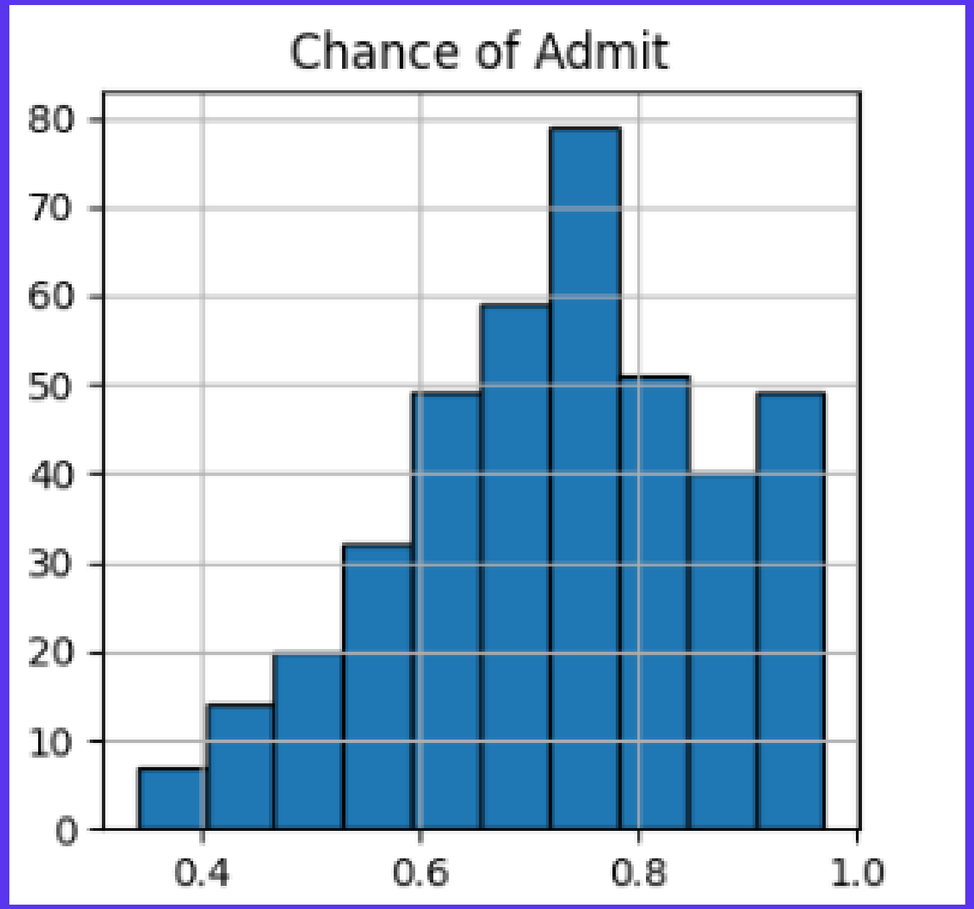


Observation :
Higher CGPA have higher chances of getting an admit.



Research Experience

Student research is self-directed work in which students from all areas of study work individually or as part of a team to explore issues of interest to them. It is an optional activity so students can decide if do it or not.



Chance of Admit

The Chance of Admit is a percentage which represents the probability of being admitted.

Correlation Between the data features
We can check how each features are reated with others using corr() function. The correlation value ranges between -1 to 1.When it is close to 1, it means that there is a strong positive correlation. When the coefficient is close to –1, it means that there is a strong negative correlation. Finally, coefficients close to zero mean that there is no linear correlation. We can observe the detail information using correlation matrix

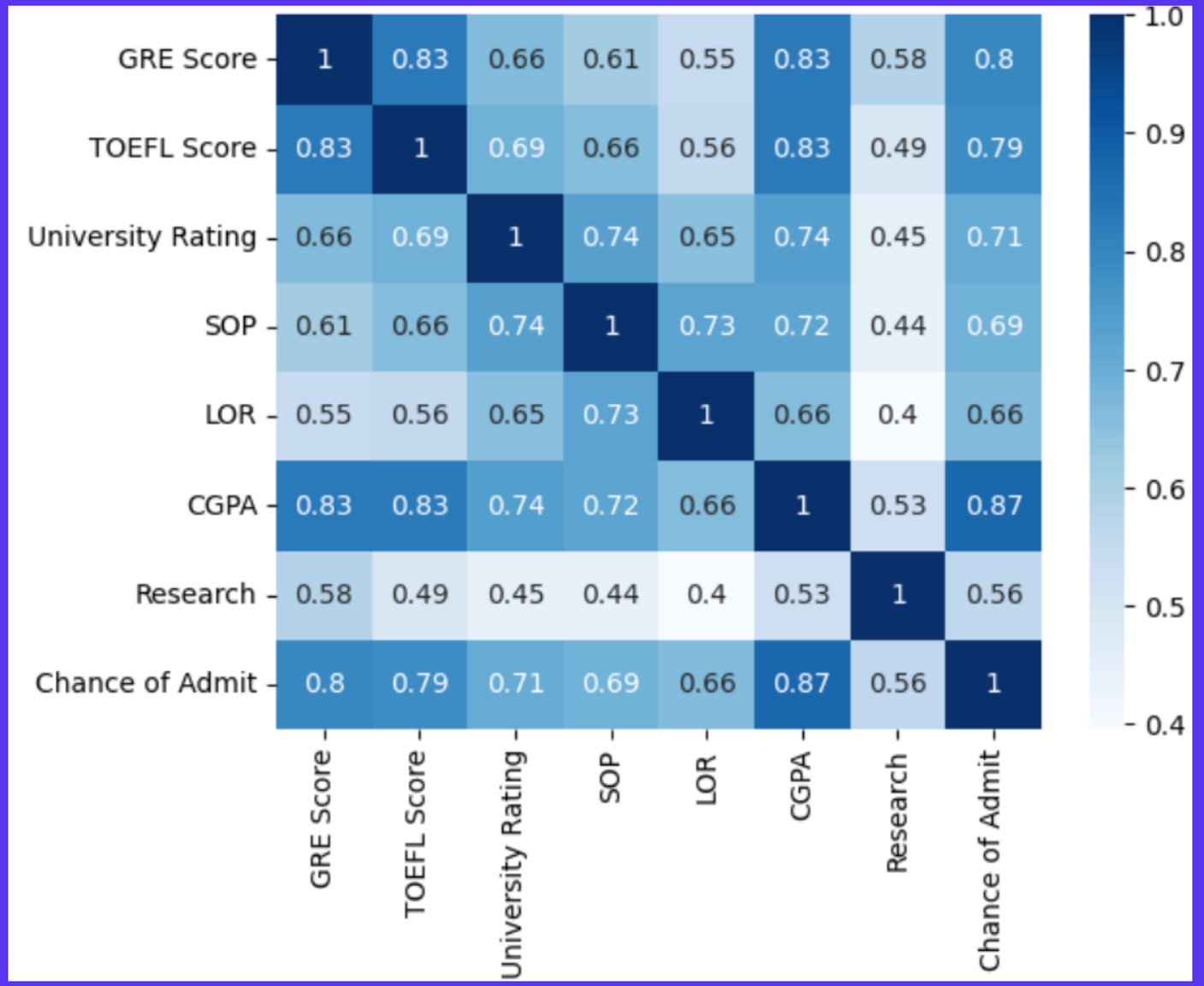
Observations:

01

Students who have High GRE scores tend to also have high TOEFL scores and CGPA. That means they are positively correlated

02

CGPA and Chance of Admit are also highly correlated which suggests that CGPA is a very important factor



MACHINE LEARNING MODELS

Predict Graduate University Admissions using

Linear Regression

Decision Trees

Random Forests

Create Training and Testing Split

```
X =df.drop(['Chance of Admit'],axis=1)
Y = df['Chance of Admit']
X.shape,Y.shape
```

```
1  #import train test split function
2  from sklearn.model_selection import train_test_split
3  train_x,test_x,train_y,test_y = train_test_split(X,Y,random_state=56)
```

✓ 0.4s Python

Linear Regression

```
1  from sklearn.linear_model import LinearRegression
2  from sklearn.metrics import mean_absolute_error as mae
```

✓ 0.1s

```
1  linreg = LinearRegression()
2  #input to model is train_x and output is train_y
3  linreg.fit(train_x, train_y)
4
```

✓ 0.0s Python

```
▾ LinearRegression
LinearRegression()
```

Prediction from the model

```
2  train_predict = linreg.predict(train_x)
3  k = mae(train_predict,train_y)
4  print('Test MAE for training data',k)
```

✓ 0.0s

```
1  test_predict = linreg.predict(test_x)
2  k = mae(test_predict,test_y)
3  print("Test MAE for testing data",k)
```

✓ 0.0s

Observation :
MAE for training data is 0.042322.
MAE for testing data 0.04759567870445306

Comparing predict and actual data points

```
predictions = pd.DataFrame(test_predict,test_y).reset_index()
predictions.columns = ['Predictions','Actual']
predictions.head()
```

✓ 0.0s

	Predictions	Actual
0	0.47	0.526520
1	0.76	0.715711
2	0.76	0.703535
3	0.63	0.571284
4	0.74	0.767416

Observation :
It predicts almost accurately

Check the accuracy of the model

```
linreg_score = linreg.score(test_x,test_y) * 100
linreg_score
```

Observation : 77.60029591127446

Decision Trees

```
1  X = df.iloc[:,:-1]
2  Y = df['Chance of Admit'] >= 0.65
3
4  from sklearn.tree import DecisionTreeClassifier
5  from sklearn.metrics import accuracy_score
6  x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state =
7
8  dt = DecisionTreeClassifier(max_depth=2, ccp_alpha=0.01, criterion='gini
9  dt.fit(x_train, y_train)
10
11
```

✓ 0.1s Python

```
▼ DecisionTreeClassifier
DecisionTreeClassifier(ccp_alpha=0.01, max_depth=2)
```

assume as
- greater than 65%: 1 = admission likely
- less than 65%: 0 = admission unlikely

The remaining columns will be used as predictors.

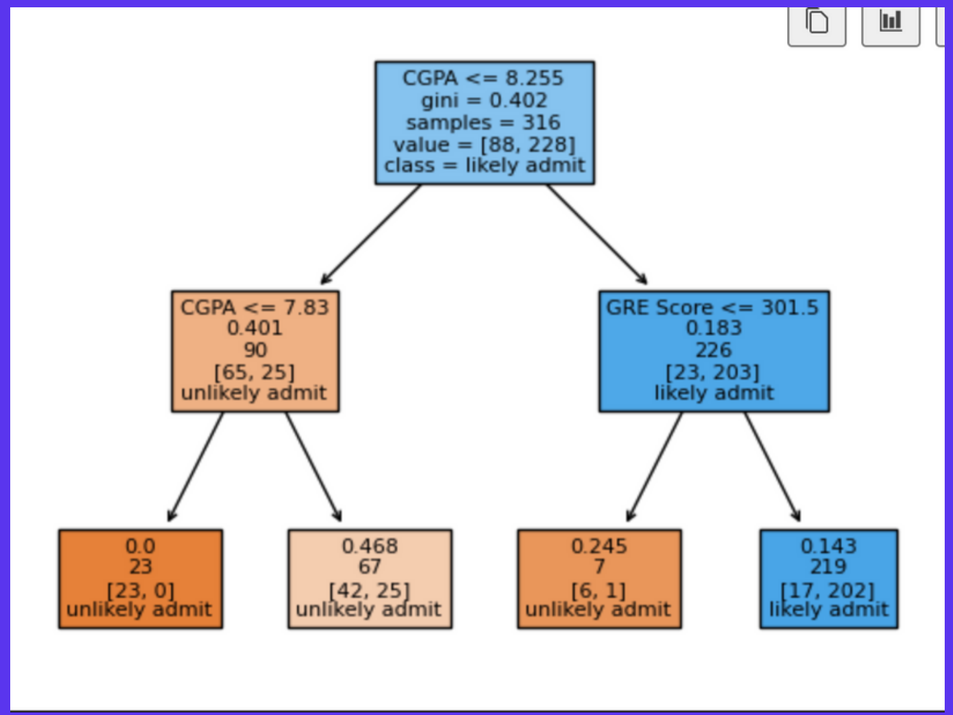
```python
from sklearn import tree
tree.plot_tree(dt,
               feature_names = train_x.columns,
               max_depth = 2,
               class_names = ['unlikely admit', 'likely admit'],
               label = 'root',
               filled = True)


```

✓ 0.0s Python

```python
y_pred = dt.predict(x_test)
dec_tree_score=accuracy_score(y_test, y_pred)*100

print('Accuracy score: ' +str(dec_tree_score))
```

✓ 0.0s

Observation:



Observation:

The accuracy of the decision tree model is 83.75.

Random Forests

```
1  from sklearn.ensemble import RandomForestRegressor
2  forest = RandomForestRegressor(n_estimators=110,max_depth=6,random_state=0)
3  forest.fit(train_x, train_y)
4  y_predict = forest.predict(test_x)
5  forest_score = (forest.score(test_x, test_y))*100
6  forest_score
```

✓ 0.1s

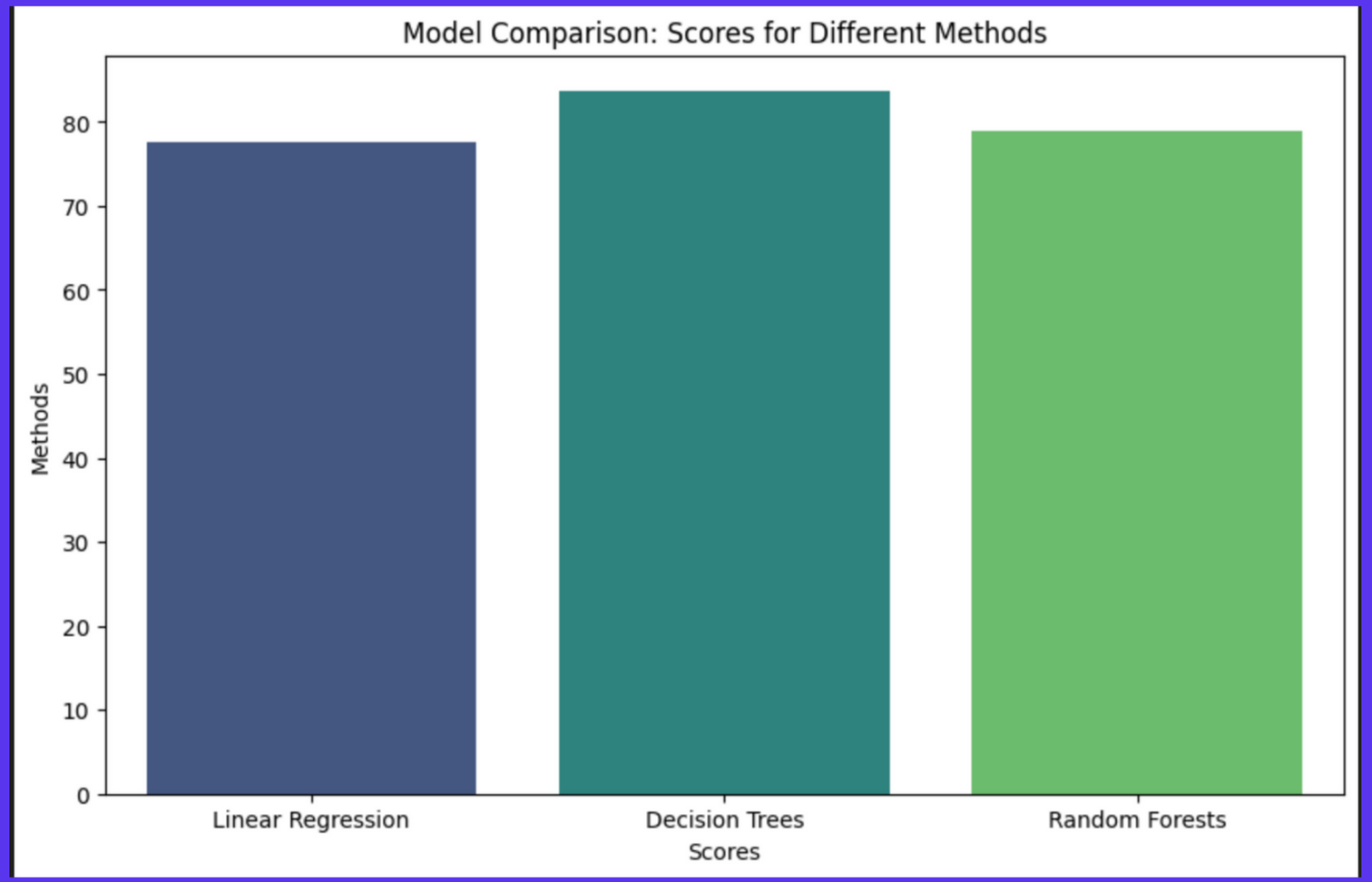
Observation :

the accuracy of the model is 78.90510037237351

COMPARING THE SCORES OF THE MODELS

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Define the methods and scores
Methods = ['Linear Regression', 'Decision Trees', 'Random Forests']
Scores = [linreg_score, dec_tree_score, forest_score]

# Create a bar plot with methods on the y-axis
plt.figure(figsize=(10, 6))
sns.barplot(y=Scores, x=Methods, palette='viridis')

# Add labels and a title
plt.xlabel('Scores')
plt.ylabel('Methods')
plt.title('Model Comparison: Scores for Different Methods')

# Display the plot
plt.show()
```

✓ 0.0s


Model Comparison: Scores for Different Methods
Methods
0
10
20
30
40
50
60
70
80
Linear Regression
Decision Trees
Random Forests
Scores

RESULTS AND ACHIEVEMENTS

Decision Trees: With the highest accuracy at 83.75%, Decision Trees excel in predicting graduate admission. They offer precise insights.

Random Forests: With a score of 78.9%, Random Forests offer a good balance between accuracy and being easy to understand, making them a versatile choice.

Linear Regression: While not the highest scorer at 77.6%, Linear Regression remains reliable and easy to understand, ensuring transparent predictions.

THANK YOU!