Student Chance of Admission Prediction in Higher Studies

Predict the chances of admission of a student to a University fro Graduate program based on different parameters such as:

GRE Scores (290 to 340) TOEFL Scores (92 to 120) University Rating (1 to 5) Statement of Purpose (1 to 5) Letter of Recommendation Strength (1 to 5) Undergraduate CGPA (6.8 to 9.92) Research Experience (0 or 1) Chance of Admit (0.34 to 0.97)

Dataset : https://github.com/ybifoundation/Dataset/raw/main/Admission%20Chance.csv

```
#Importing Libraries and Data
#Importing libraries and setting the default style in Seaborn.
```

```
import numpy as np
import pandas as pd
#import os
from matplotlib import pyplot as plt
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
import seaborn as sns
sns.set(style='white')
sns.set(style='whitegrid', color_codes=True)
```

```
# import dataset
df = pd.read_csv("/content/Admission_Predict.csv")
df.head()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

```
df.describe()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Res
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.0
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.5
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.4
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.0
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.0
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.0
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.0
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.0

Exploratory Analysis

From these charts it looks like we have no missing values!

It seems as though Serial No. is just an index for students, which we can take out.

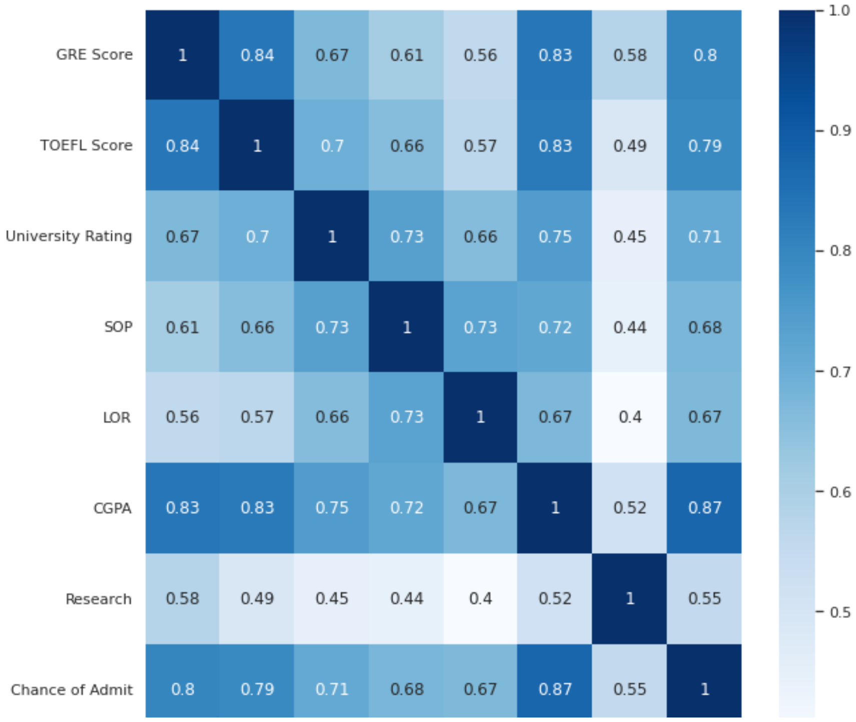
Two columns also have an added space in the label which we'll take out

We are also removing the blank sapces.

```
df.rename(columns = {'Chance of Admit ':'Chance of Admit', 'LOR ':'LOR'}, inplace=True)
df.drop(labels='Serial No.', axis=1, inplace=True)
```

```
#Let's plot a heatmap to see the correlation of all the features compared to Chance to Admit:
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(df.corr(), annot=True, cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f88bc1993d0>
```



The top three features that affect the Chance to Admit are:

CGPA

GRE Score

TOEFL Score

Let's explore these three features to get a better understanding.

CGPA

The Cumulative Grade Point Average is a 10 point grading system.

From the data shown below, it appears the submissions are normally distributed. With a mean of 8.6 and standard deviation of 0.6.

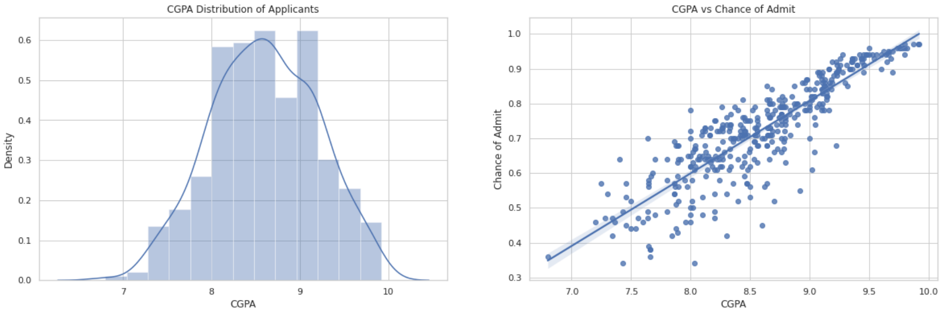
CGPA vs Chance of Admit

appears as applicant's CGPA has a strong correlation with their chance of admission.

```
plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
sns.distplot(df['CGPA'])
plt.title('CGPA Distribution of Applicants')

plt.subplot(1,2,2)
sns.regplot(df['CGPA'], df['Chance of Admit'])
plt.title('CGPA vs Chance of Admit')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distpl
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fo
  warnings.warn(
Text(0.5, 1.0, 'CGPA vs Chance of Admit')
```



GRE Score

The Graduate Record Examination is a standarized exam, often required for admission to graduate and MBA programs globally. It's made up of three components:

Analytical Writing (Scored on a 0-6 scale in half-point increments)

Verbal Reasoning (Scored on a 130-170 scale)

Quantitative Reasoning (Scored on a 130-170 scale)

In this dataset, the GRE Score is based on a maximum of 340 points. The mean is 317 with a standard deviation of 11.5.

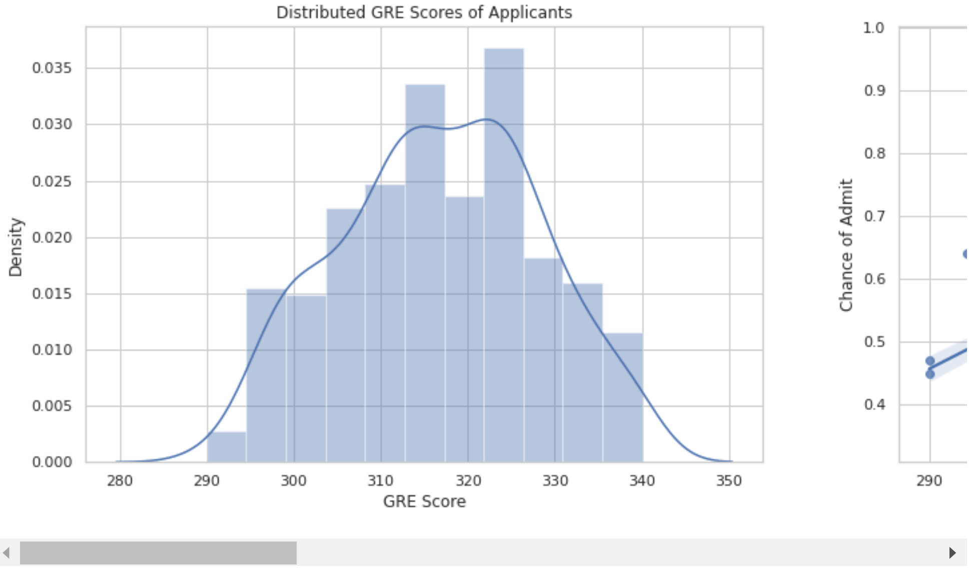
GRE Score vs Chance of Admit

GRE scores have a strong correlation with the chance of admission however not as strong as one's CGPA.

```
plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
sns.distplot(df['GRE Score'])
plt.title('Distributed GRE Scores of Applicants')

plt.subplot(1,2,2)
sns.regplot(df['GRE Score'], df['Chance of Admit'])
plt.title('GRE Scores vs Chance of Admit')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot`
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the foll
  warnings.warn(
Text(0.5, 1.0, 'GRE Scores vs Chance of Admit')
```



TOEFL Score

The Test of English as a Foreign Language is a standarized test for non-native English speakers that are choosing to enroll in English-speaking universities.

The test is split up into 4 sections:

Reading

Listening

Speaking

Writing

All sections are scored out of 30, giving the exam a total score of 120 marks. In this dataset, the TOEFL scores have a mean of 107 and a standard deviation of 6.

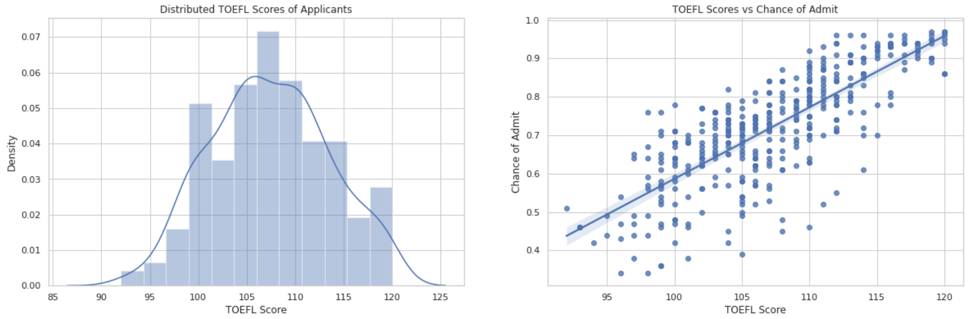
TOEFL Score vs Chance of Admit

Like GRE scores, the scores received for the TOEFL strongly correlate to an applicants chance of admission.

```
plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
sns.distplot(df['TOEFL Score'])
plt.title('Distributed TOEFL Scores of Applicants')

plt.subplot(1,2,2)
sns.regplot(df['TOEFL Score'], df['Chance of Admit'])
plt.title('TOEFL Scores vs Chance of Admit')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot`
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the foll
  warnings.warn(
Text(0.5, 1.0, 'TOEFL Scores vs Chance of Admit')
```



For my curiosity, I want to explore the data a little bit further regarding research and university rankings. Even though they hold a lower importance in the chance of admission, it would be nice to understand their characteristics in the dataset.

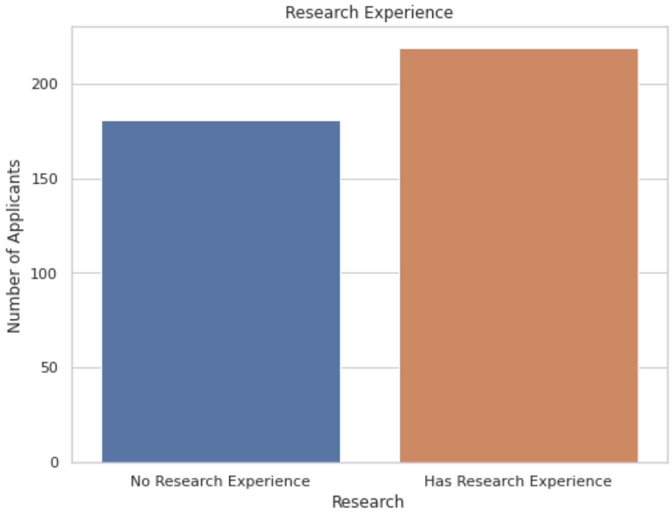
Research

Let's explore how many applicants have research experience.

It seems the majority of applicants have research experience. However, this is the least important feature, so it doesn't matter all too much if an applicant has the experience or not.

```
fig, ax = plt.subplots(figsize=(8,6))
sns.countplot(df['Research'])
plt.title('Research Experience')
plt.ylabel('Number of Applicants')
ax.set_xticklabels(['No Research Experience', 'Has Research Experience'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the foll
  warnings.warn(
[Text(0, 0, 'No Research Experience'), Text(0, 0, 'Has Research Experience')]
```



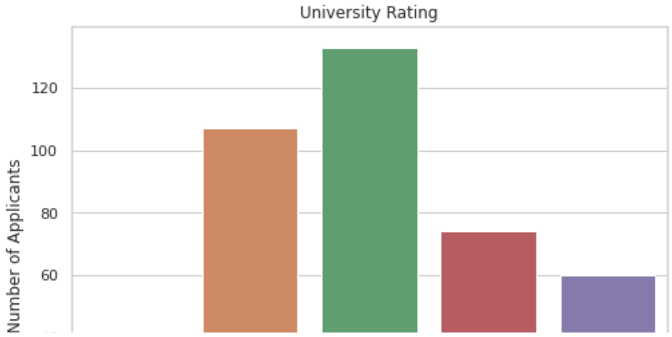
University Rating

Let's see the distribution of applicants coming from each kind of university.

Most applicants come from a tier 3 and tier 2 university.

```
fig, ax = plt.subplots(figsize=(8,6))
sns.countplot(df['University Rating'])
plt.title('University Rating')
plt.ylabel('Number of Applicants')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fo
  warnings.warn(
Text(0, 0.5, 'Number of Applicants')
```



Preparing Data for Machine Learning

Now that we understand our dataset, it's time to implement machine learning methods to predict future applicant's chances of admission.

First we have to prepare our data, by splitting it into training and testing data. We'll also scale our data, from 0 to 1, to receive more accurate predictions

```
targets = df['Chance of Admit']
features = df.drop(columns = {'Chance of Admit'})

X_train, X_test, y_train, y_test = train_test_split(features, targets, test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

Machine Learning

Now we'll implement machine learning algorithms to predict the chance of admission. We'll use multiple techniques and eventually select the method with the best score. The methods used will be:

Linear Regression

Decision Trees

Random Forests

```
# Liner Regression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
y_predict = linreg.predict(X_test)
linreg_score = (linreg.score(X_test, y_test))*100
linreg_score
```

```
81.7386788111443
```

```
#decision trees
dec_tree = DecisionTreeRegressor(random_state=0, max_depth=6)
dec_tree.fit(X_train, y_train)
y_predict = dec_tree.predict(X_test)
dec_tree_score = (dec_tree.score(X_test, y_test))*100
dec_tree_score
```

```
73.99851580517213
```

```
#Random Forests
forest = RandomForestRegressor(n_estimators=110,max_depth=6,random_state=0)
forest.fit(X_train, y_train)
y_predict = forest.predict(X_test)
forest_score = (forest.score(X_test, y_test))*100
forest_score
```

```
81.34052472373693
```

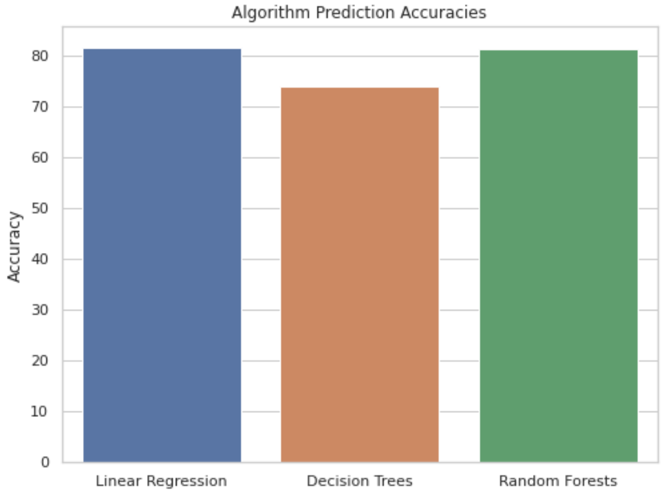
Comparing Scores

Let's put all the scores in a table and display their scores side-by-side.

```
Methods = ['Linear Regression', 'Decision Trees', 'Random Forests']
Scores = np.array([linreg_score, dec_tree_score, forest_score])

fig, ax = plt.subplots(figsize=(8,6))
sns.barplot(Methods, Scores)
plt.title('Algorithm Prediction Accuracies')
plt.ylabel('Accuracy')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the fo
  warnings.warn(
Text(0, 0.5, 'Accuracy')
```



```
Selecting the Best Algorithm

Linear Regression - 81.74%

Random Forests - 81.35%

Decision Trees - 73.99%

It seems that Linear Regression is the most accurate of the 3 methods and w
be used to predict the future applicant's chances of admission.

Conclusion

This was a great way to get started on Kaggle and for my first project outs
of coursework. It gave me some practice some exploratory analysis and simpl
machine learning techniques.


It's great to see what specific variables contribute to the chance of admis
and how they are weighted against eachother.
```

Selecting the Best Algorithm

Linear Regression - 81.74%

Random Forests - 81.35%

Decision Trees - 73.99%

It seems that Linear Regression is the most accurate of the 3 methods and will be used to predict the future applicant's chances of admission.

Conclusion

This was a great way to get started on Kaggle and for my first project outside of coursework. It gave me some practice some exploratory analysis and simple machine learning techniques.

It's great to see what specific variables contribute to the chance of admission and how they are weighted against eachother.

✓ 0s completed at 9:53 AM