ABSTRACT

Student admission problem is very important in educational institutions. This addresses machine learning models to predict the chance of a student to be admitted to a master’s program. This will assist students to know in advance if they have a chance to get accepted. The machine learning models are multiple linear regression, random forest, random forest with pca, multiple linear regression with pca.

Every year, the number of students wanting to pursue higher studies in abroad especially in US is more and they find difficult to search the best university. And thus, this helps on predicting the eligibility of Indian students getting admission in best university based on their Test attributes like GRE, TOEFL, CGPA. According to their scores the possibilities of chance of admit is calculated. This project uses machine learning technique for predicting the output which helps them to admit in the best university and also helps the student to find the possibility of admitting in other universities

1.INTRODUCTION

Machine learning mainly solves classification and regression problems, whichis an important research area and very useful in making decision. Both of classification and regression algorithms aim to evaluate unknow data by training labeled data sample. But they are still different. The output of the former is discrete, which means we separate data to a few classes by classification algorithm then evaluate new coming data belongs to which class. The output of the latter is continuous, which means we learn from old data then evaluate the probability of an event happening.

Prediction of USA graduate admission for Indian undergraduate appliers"based on regression algorithms. The main content is to show the process from analyzing data, making model to result evaluation. At the same time, the principles of the methods used in my project. In section we will explain my dataset and introduce the research background. We will give brief introduction about how I organize my project and the main algorithm (Linear Regression) used in my project, the completed work line will be provided including selecting base model and other models fitting, and the methods of data preprocessing, cross validation and best parameter search will also be mentioned. We will provide details about how to optimize model by stacking and analyze the results.

The dataset contains 500 students data and includes 7 features as predictors such as “GRE score”, “TOEFL score”, ”University Rating” (Indian undergraduate school), ”SOP” (Statement of Purpose), ”LOP” (Letter of recommendation strength),”CGPA” (out of 10), ”Research experience”, and 1 column as target ”Chance of Admission” (float datatype, ranging from 0 to 1). “Chance of admit” were evaluated by appliers. This dataset has various use. Many people have provided solutions to this problem. Most people take this problem as classification problem (The studentcan be admitted or not), generally, they get very high f1-score (up to 96%). Some people take it as admission line case. They analyze what is the minimal line for everyfeature to make the admission possibility over 90%. For example, if a student wantsto get 93.5% admission chance, the scores or rates must satisfy GRE > 332.8, TOEFL > 116.2, University Rating > 4.6, SOP > 4.5, LOR > 4.5, CGPA > 9.5,
Research = 1. Only few people take it as regression problem to predict the possibility
of admission. This report show that they built some basic models to get reasonable results and use cross validation to select the best feature

combination. The most common used algorithms are Linear Regression

,Random Forest ,Multiple Linear Regression using PCA, Random Forest using PCA. Linear Regression performs the best among them. projects . Importing numpy and pandas library to deal with data. Matplotlib and seaborn associate analysis by visualizing data. Sklearn is the strongest library to call popular algorithms package to train data and evaluate results. Although these tools have some limitation and not super flexible, but for basic usagethey are enough.

Objective:

400 applicants have been surveyed as potential students for UCLA. The universityweighs certain aspects of a student's education to determine their acceptance.

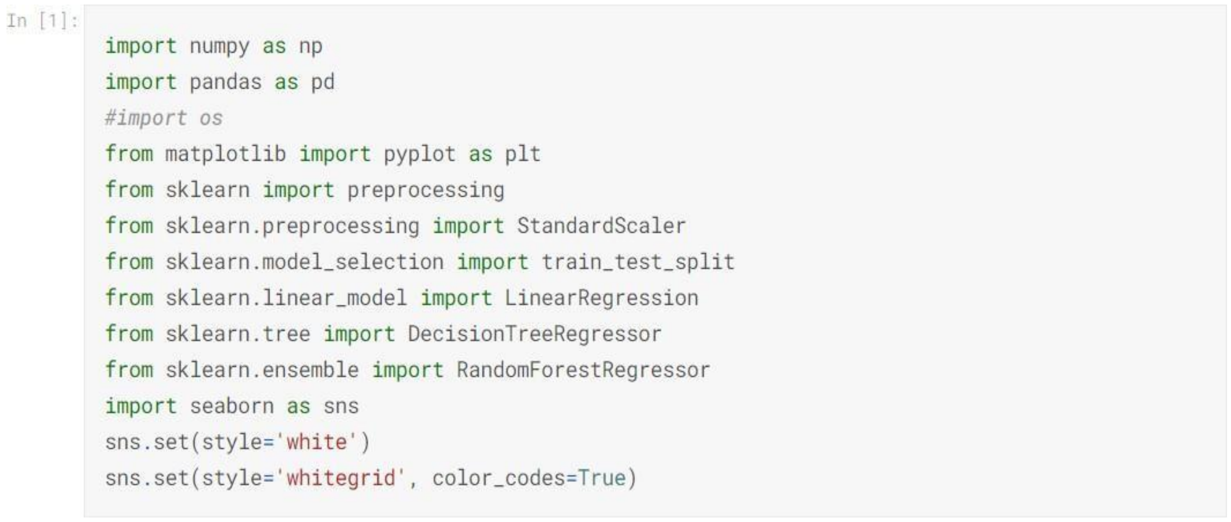
The objective is to explore what kind of data is provided, determine the most important factors that contribute to a student's chance of admission, and select themost accurate model to predict the probability of admission.

Data Description:

The dataset contains information about a student's:

- GRE Score
- TOEFL Score
- University Ratings
- Statement of Purpose Score
- Letter of Recomendation Score
- CGPA
- Whether the Student Has Done Any Research
- Chance of Admission (What We're Trying to Predict)

2.SOFTWARE LIBRARIES USED AND METHODOLOGY



```
In [1]:
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)
```

Problem Understanding:
Initially first we have to spend some time on what are the problems or concerns students having during their pre admission period and we should set the solutions to those problems as objectives of this research.
Data Understanding:
Data should be collected from multiple sources like yocket and also consider all the factors including which will play a tiny role in student admission process.
Data Preparation:
Data should be cleaned that is removing the noise in the data and filling the missing values or extreme values and finalising the attributes/factors which will have crucial importance in student admission process.
Building Models:
several ML models have to be developed using various machine learning algorithms for admission to a particular university and the user interface has to be developed to access those models.
Evaluation:
Developed models are evaluated according to their accuracy scores. Once the model is finalised that model will be merged with node red for final deployment.
Data Cleaning and Analysis:
• Inspecting feature values that help identify what needs to be done to clean or pre-process until you see the range or distribution of values typical of each attribute.
• You may find missing or noisy data, or anomalies such as the incorrect data

form used for a column, incorrect measuring units for a particular column, or that there are not enough examples of a specific class.

• You can know that without machine learning, the problem is actually solvable. The data cleaning process has several key benefits to it:

1. This eliminates major errors and inconsistencies which are unavoidable when dragging multiple data sources into one dataset.
2. Having data cleaning software will make everyone more effective as they will be able to get easily from the data what they need.
3. Fewer mistakes mean happy clients, and less unhappy workers.

• There are no missing values and outliers because we analysed the data, so for this data there is no need to fill the missing values and deal with outliers. If there are any missing values and outliers we can fill (or) drop using the fillna method and drop method and we can also standardize the data using the min-max scaler, if necessary.

Data Visualization:

• After analysing the data, we will be able to know what the features and labels are, so from the above data, the label we have to consider is Chance of Admission and then we have to consider the parameters that influence or play a major role in Chance of Admission

• We can get to know certain features that are more affected by the visualization (or) analysis or the use of feature importance method in decision tree

3.ALGORITHMS

➡ Linear Regression
➡Random Forest
➡Decision Tree

1.Linear Regression

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

Mathematically, we can represent a linear regression as:

$$y= a_0+a_1x+ \varepsilon$$

Here,

Y=Dependent Variable(Target Variable)
X=Independent Variable(predictorVariable)
a0= intercept of the line (Gives an additional degree of freedom)
a1 = Linear regression coefficient (scale factor to each input value).
ε = random error

The values for x and y variables are training datasets for Linear Regression model representation.

Types of Linear Regression

Linear regression can be further divided into two types of the algorithm:

- **Simple Linear Regression:**
  If a single independent variable is used to predict the value of a numerical

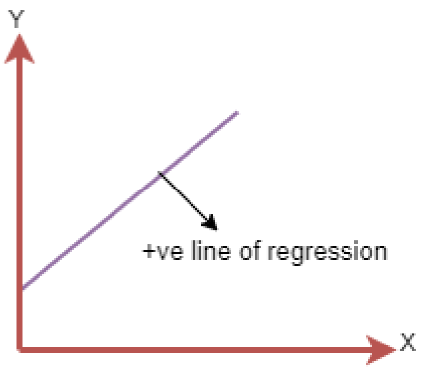
dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

- **Multiple Linear Regression:**
  If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

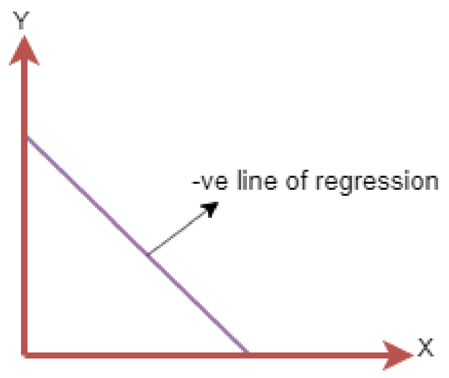
Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

- **Positive Linear Relationship:**
  If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.
- **Negative Linear Relationship:**
  If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.

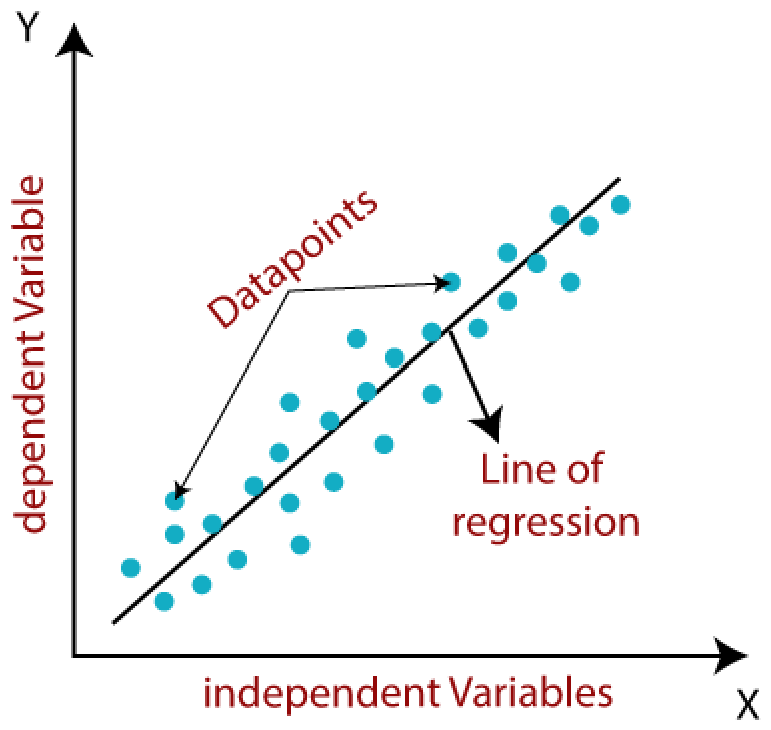


The line equation will be: **$Y= a_0+a_1x$**



The line of equation will be: **$Y = -a_0 + a_1x$**

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



Finding the best fit line:

When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

The different values for weights or the coefficient of lines (a_0, a_1) gives a different line of regression, so we need to calculate the best values for a_0 and a_1 to find the best fit line, so to calculate this we use cost function.

Model Performance:

The Goodness of fit determines how the line of regression fits the set of observations. The process of finding the best model out of various models is called **optimization**. It can be achieved by below method:

1. R-squared method:

- R-squared is a statistical method that determines the goodness of fit.
- It measures the strength of the relationship between the dependent and independent variables on a scale of 0-100%.
- The high value of R-square determines the less difference between the predicted values and actual values and hence represents a good model.
- It is also called a **coefficient of determination,** or **coefficient of multiple determination** for multiple regression.
- It can be calculated from the below formula:

$$\text{R-squared} = \frac{\text{Explained variation}}{\text{Total Variation}}$$

Assumptions of Linear Regression

Below are some important assumptions of Linear Regression. These are some formal checks while building a Linear Regression model, which ensures to get the best possible result from the given dataset.

- **Linear relationship between the features and target:** Linear regression assumes the linear relationship between the dependent and independent variables.
- **Small or no multicollinearity between the features:** Multicollinearity means high-correlation between the independent variables. Due to multicollinearity, it may difficult to find the true relationship between the predictors and target variables. Or we can say, it is difficult to determine which predictor variable is affecting the target variable and which is not. So, the model assumes either little or no multicollinearity between the features or independent variables.
- **Homoscedasticity Assumption:** Homoscedasticity is a situation when the error term is the same for all

the values of independent variables. With homoscedasticity, there should be no clear pattern distribution of data in the scatter plot.

- **Normal distribution of error terms:**
  Linear regression assumes that the error term should follow the normal distribution pattern. If error terms are not normally distributed, then confidence intervals will become either too wide or too narrow, which may cause difficulties in finding coefficients.
  It can be checked using the **q-q plot**. If the plot shows a straight line without any deviation, which means the error is normally distributed.
- **Noautocorrelations:**
  The linear regression model assumes no autocorrelation in error terms. If there will be any correlation in the error term, then it will drastically reduce the accuracy of the model. Autocorrelation usually occurs if there is a dependency between residual errors.

2.Random Forest

Random Forest is a popular machine learning algorithm that belongs to thesupervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based onthe majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy andprevents the problem of overfitting.

Implementation of Random Forest Algorithm

Implementation Steps are given below:

- Data Pre-processing step

- Fitting the Random forest algorithm to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.

Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below aretwo assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset sothat the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

<="" li="">

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

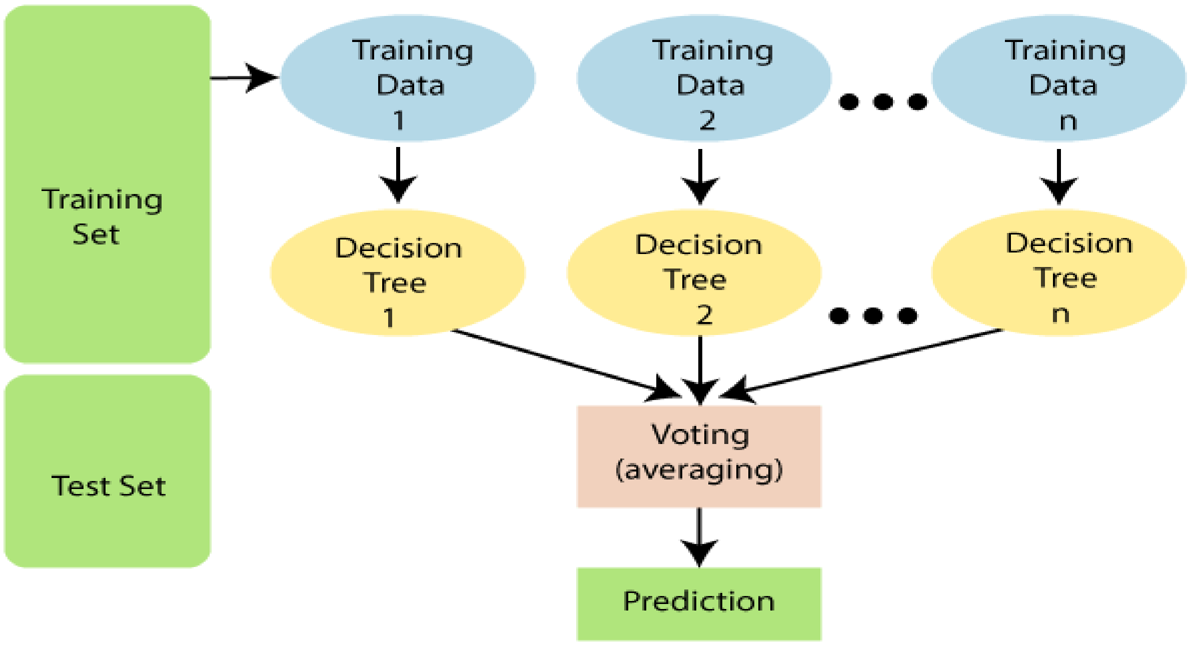
The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.



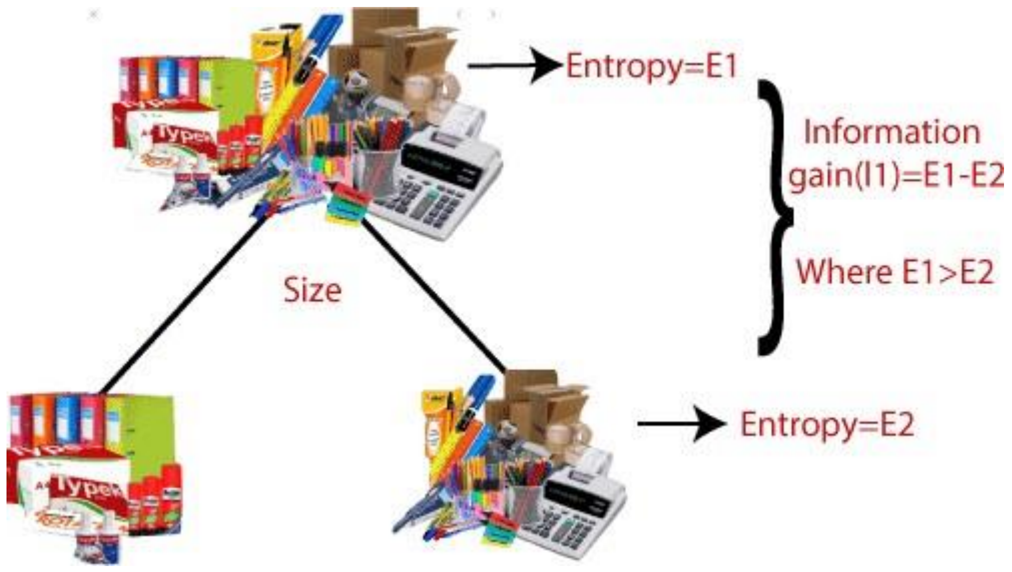
3.Decision Trees

Decision Tree is a supervised learning method used in data mining for classification and regression methods. It is a tree that helps us in decision-making purposes. The decision tree creates classification or regression models as a tree structure. It separates a data set into smaller subsets, and at the same time, the decision tree is steadily developed. The final tree is a tree with the decision nodes and leaf nodes. A decision node has at least two branches. The leaf nodes show a classification or decision. We can't accomplish more split on leaf nodes-The uppermost decision node in a tree that relates to the best predictor called the root node. Decision trees can deal with both categorical and numerical data.

Key factors:

Entropy:
Entropy refers to a common way to measure impurity. In the decision tree, it measures the randomness or impurity in data
Information Gain:
Information Gain refers to the decline in entropy after the dataset is split. It is

also called **Entropy Reduction**. Building a decision tree is all about discovering attributes that return the highest data gain.



In short, a decision tree is just like a flow chart diagram with the terminal nodes showing decisions. Starting with the dataset, we can measure the entropy to find a way to segment the set until the data belongs to the same class.

Why are decision trees useful?

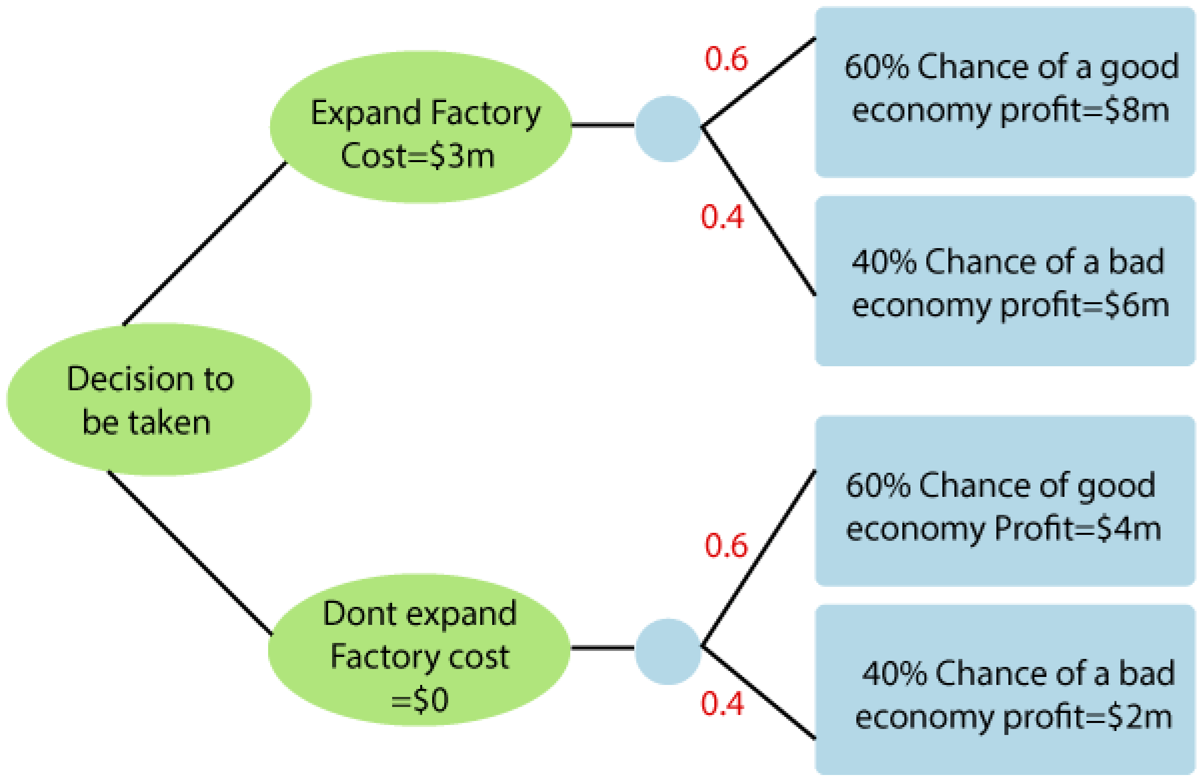
It enables us to analyze the possible consequences of a decision thoroughly.

It provides us a framework to measure the values of outcomes and the probability of accomplishing them.

It helps us to make the best decisions based on existing data and best speculations.

In other words, we can say that a decision tree is a hierarchical tree structure that can be used to split an extensive collection of records into smaller sets of the class by implementing a sequence of simple decision rules. A decision tree model comprises a set of rules for portioning a huge heterogeneous population into smaller, more homogeneous, or mutually exclusive classes. The attributes of the classes can be any variables from nominal, ordinal, binary, and quantitative values, in contrast, the classes must be a qualitative type, such as categorical or ordinal or binary. In brief, the given data of attributes together with its class, a decision tree creates a set of rules that can be used to identify the class. One rule is implemented after another, resulting in a hierarchy of segments within a segment. The hierarchy is known as the **tree**, and each segment is called a **node**. With each progressive division, the members from the subsequent sets become more and more similar to each other. Hence, the algorithm used to build a decision tree is referred to as recursive partitioning. The algorithm is known as **CART** (Classification and Regression Trees)

Consider the given example of a factory where



Decision to be taken
Expand Factory Cost=$3m
Dont expand Factory cost =$0
0.6
0.4
0.6
0.4
60% Chance of a good economy profit=$8m
40% Chance of a bad economy profit=$6m
60% Chance of good economy Profit=$4m
40% Chance of a bad economy profit=$2m

Expanding factor costs $3 million, the probability of a good economy is 0.6 (60%), which leads to $8 million profit, and the probability of a bad economy is 0.4 (40%), which leads to $6 million profit.

Not expanding factor with 0$ cost, the probability of a good economy is 0.6(60%), which leads to $4 million profit, and the probability of a bad economy is 0.4, which leads to $2 million profit.

The management teams need to take a data-driven decision to expand or not based on the given data.

Net Expand = (0.6 *8 + 0.4*6) - 3 = $4.2M
Net Not Expand = (0.6*4 + 0.4*2) - 0 = $3M
$4.2M > $3M,therefore the factory should be expanded.

Decision tree Algorithm:

The decision tree algorithm may appear long, but it is quite simply the basis algorithm techniques is as follows:

The **algorithm** is based on three parameters**: D, attribute_list, and Attribute _selection_method.**

Generally, we refer to **D** as a **data partition.**

Initially, **D** is the entire set of **training tuples** and their related **class levels** (input training data).

The parameter **attribute_list** is a set of **attributes** defining the tuples.

Attribute_selection_method specifies a **heuristic process** for choosing the attribute that "best" discriminates the given tuples according to **class**.

Attribute_selection_method process applies an **attribute selection measure**.

Advantages of using decision trees:

A decision tree does not need scaling of information.

Missing values in data also do not influence the process of building a choice tree to any considerable extent.

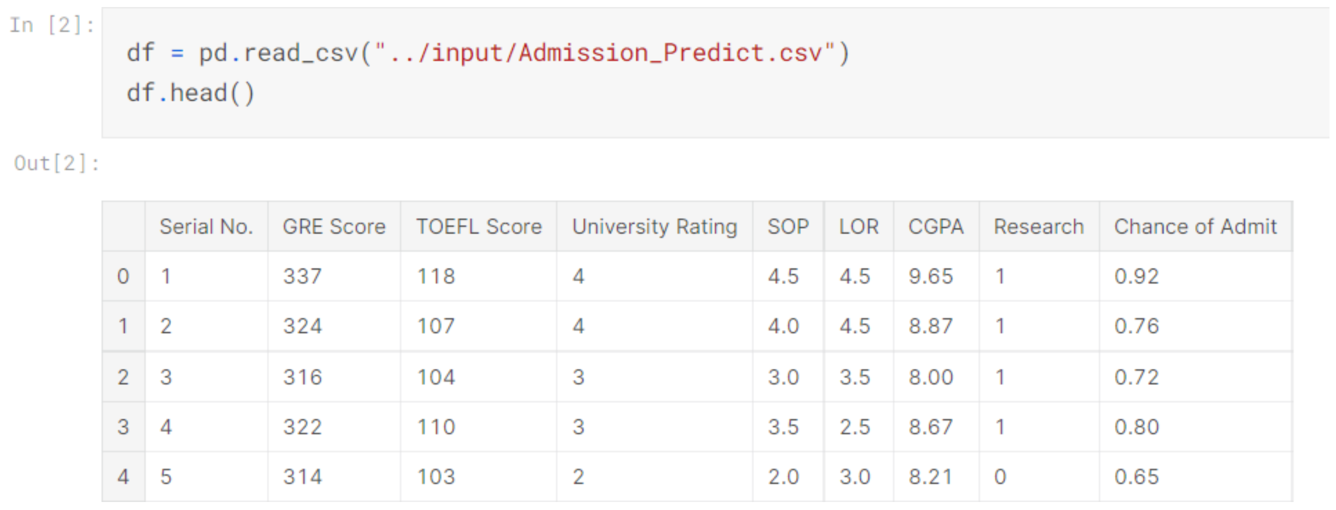
A decision tree model is automatic and simple to explain to the technical team as well as stakeholders.

Compared to other algorithms, decision trees need less exertion for data preparation during pre-processing.

A decision tree does not require a standardization of data.

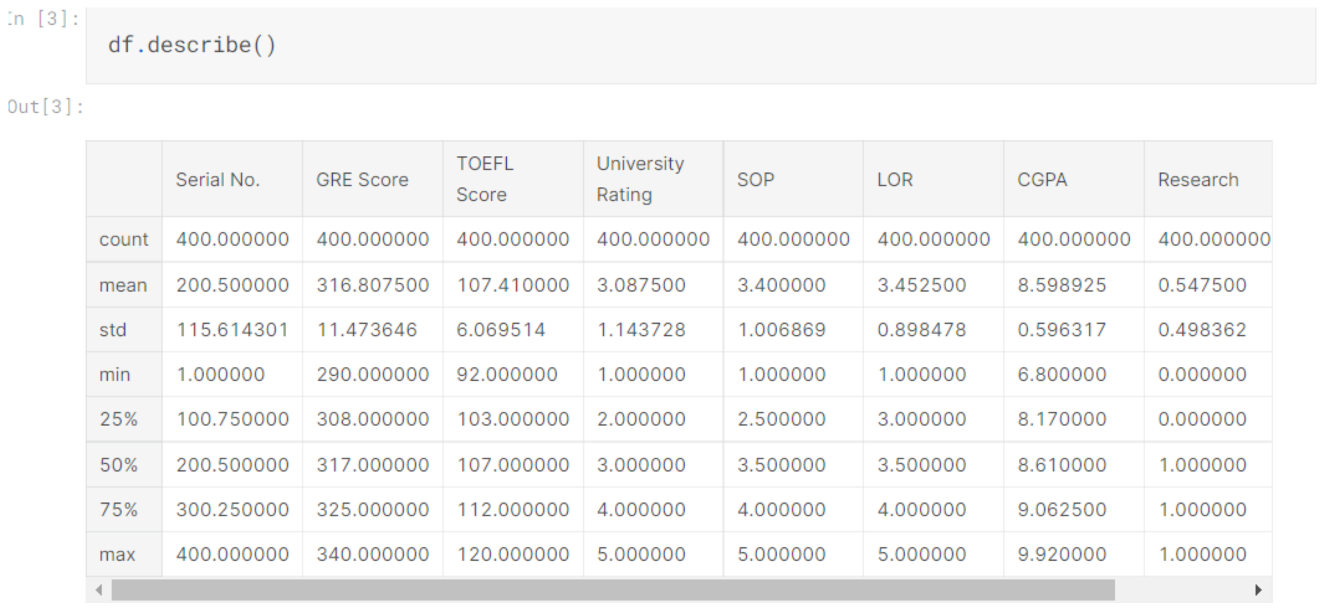
4. Implementation

Next, let's import our dataset and see what we're working with.



In [2]:
```
df = pd.read_csv("../input/Admission_Predict.csv")
df.head()
```
Out[2]:

	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



In [3]:
```
df.describe()
```
Out[3]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
count	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
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000

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

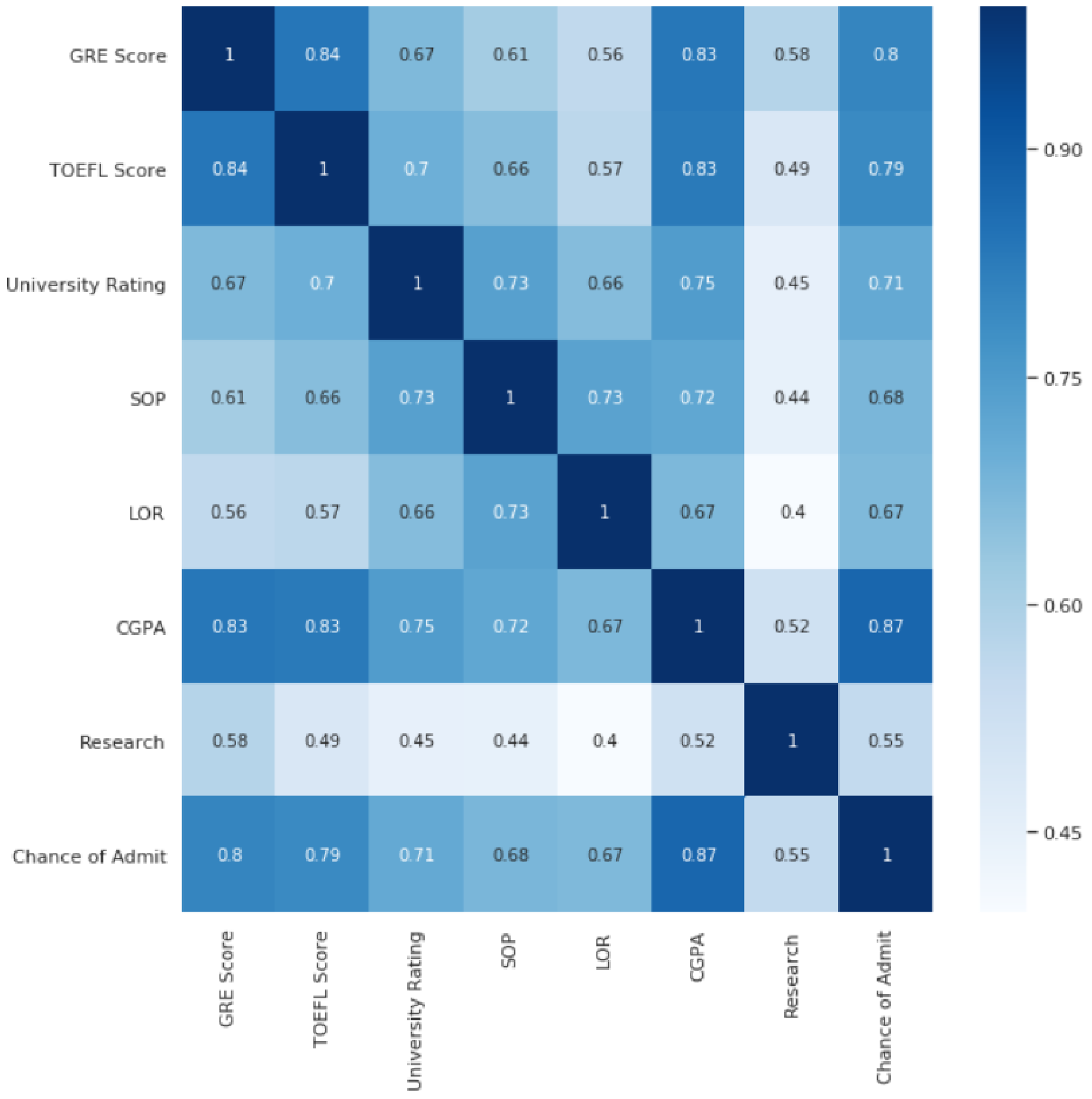
We are also removing the blank sapces.

In [4]:
```
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:

In [5]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(df.corr(), annot=True, cmap='Blues')
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8496c41438>
```



	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1	0.84	0.67	0.61	0.56	0.83	0.58	0.8
TOEFL Score	0.84	1	0.7	0.66	0.57	0.83	0.49	0.79
University Rating	0.67	0.7	1	0.73	0.66	0.75	0.45	0.71
SOP	0.61	0.66	0.73	1	0.73	0.72	0.44	0.68
LOR	0.56	0.57	0.66	0.73	1	0.67	0.4	0.67
CGPA	0.83	0.83	0.75	0.72	0.67	1	0.52	0.87
Research	0.58	0.49	0.45	0.44	0.4	0.52	1	0.55
Chance of Admit	0.8	0.79	0.71	0.68	0.67	0.87	0.55	1

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

1. CGPA
2. GRE Score
3. TOEFL Score

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

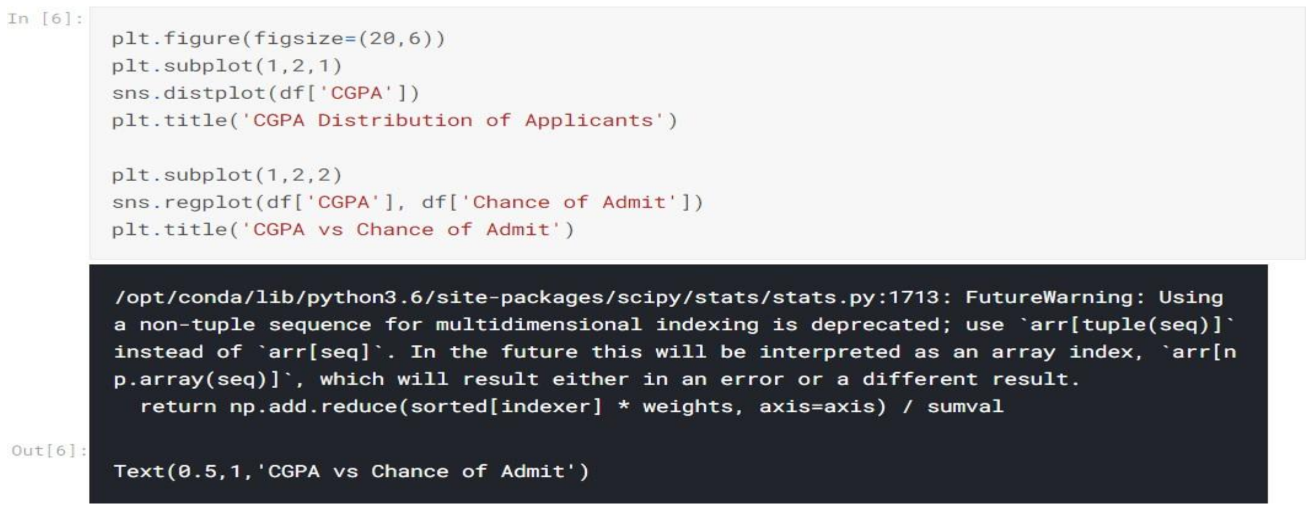
CGPA

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

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

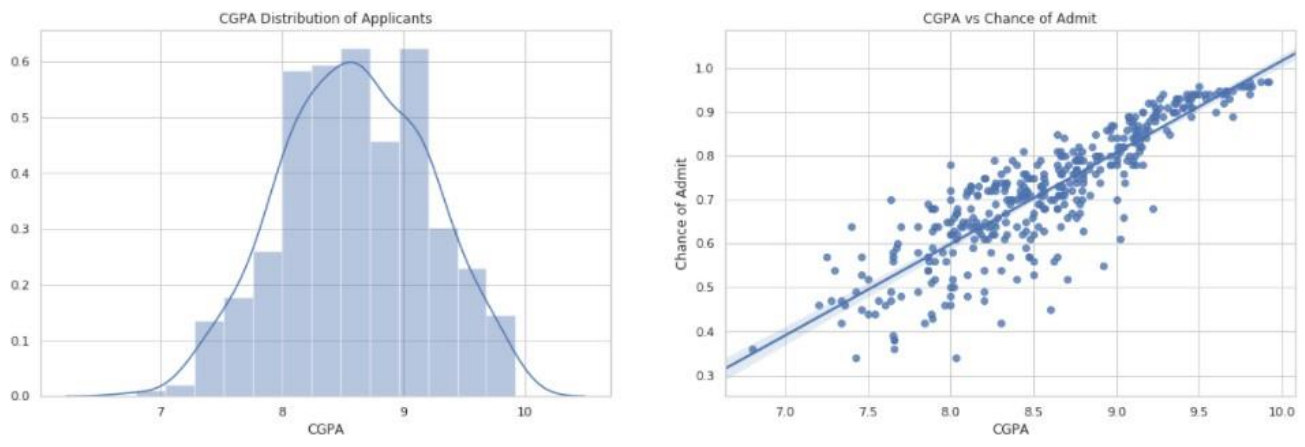


```
In [6]:
plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
sns.distplot(df['CGPA'])
plt.title('CGPA Distribution of Applicants')

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

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using
a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]`
instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[n
p.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval

Out[6]:
Text(0.5,1,'CGPA vs Chance of Admit')
```



GRE Score

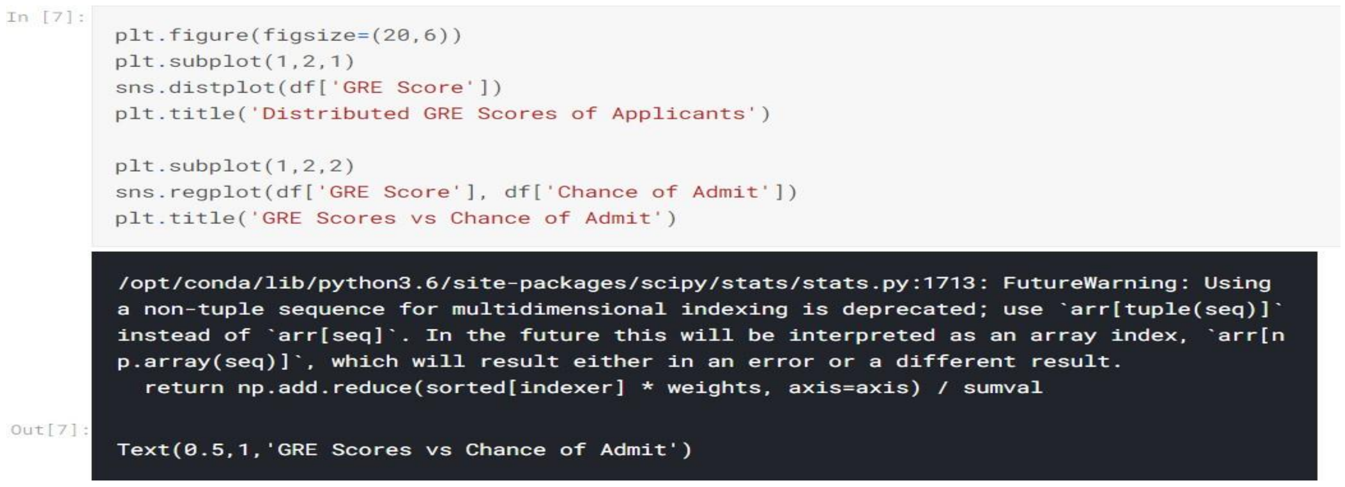
The Graduate Record Examination is a standarized exam, often required for admission to graduate and MBA programs globally. It's made up of three components:

1. Analytical Writing (Scored on a 0-6 scale in half-point increments)
2. Verbal Reasoning (Scored on a 130-170 scale)
3. 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.

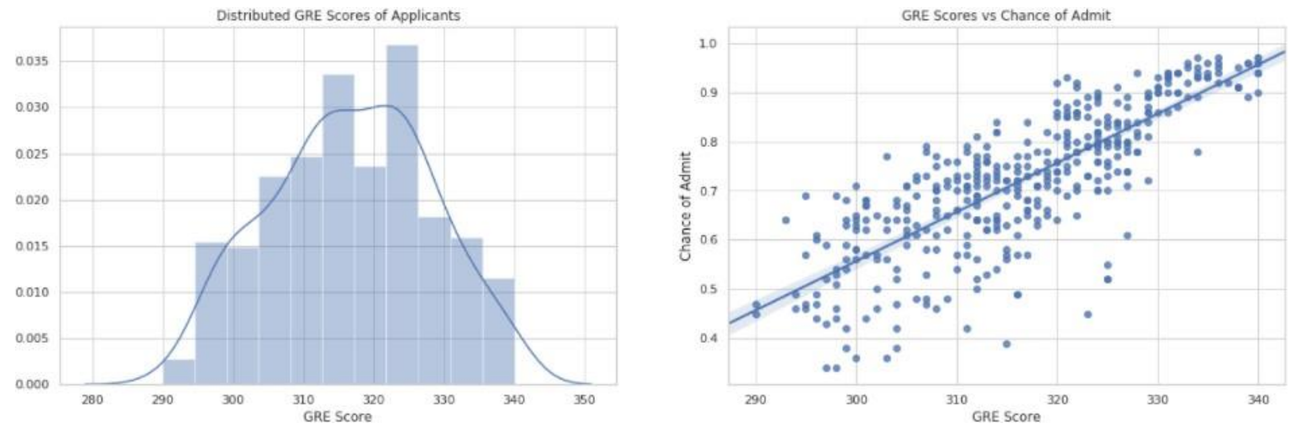


```
In [7]:
plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
sns.distplot(df['GRE Score'])
plt.title('Distributed GRE Scores of Applicants')

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

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using
a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]`
instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[n
p.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval

Out[7]:
Text(0.5,1,'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.

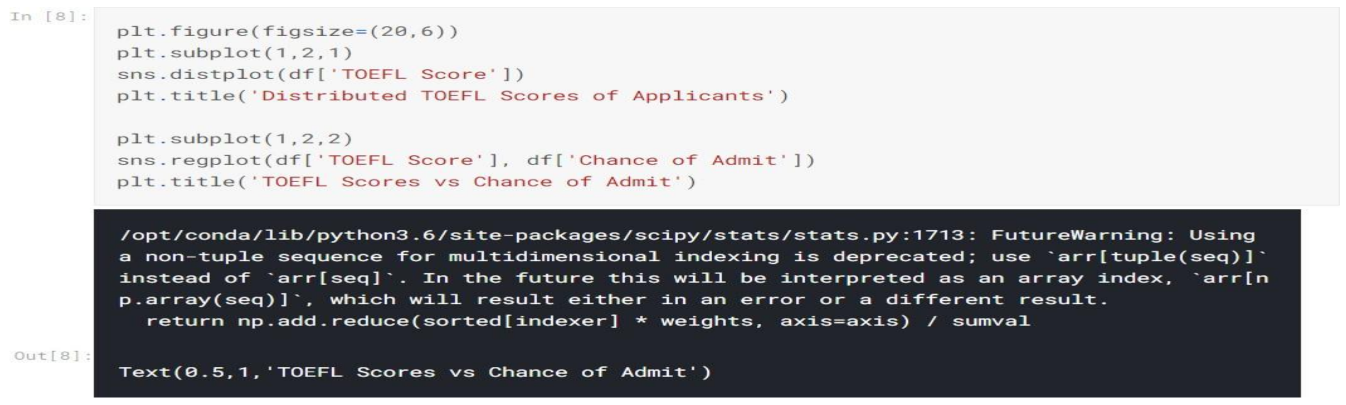
The test is split up into 4 sections:

1. Reading
2. Listening
3. Speaking
4. 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.

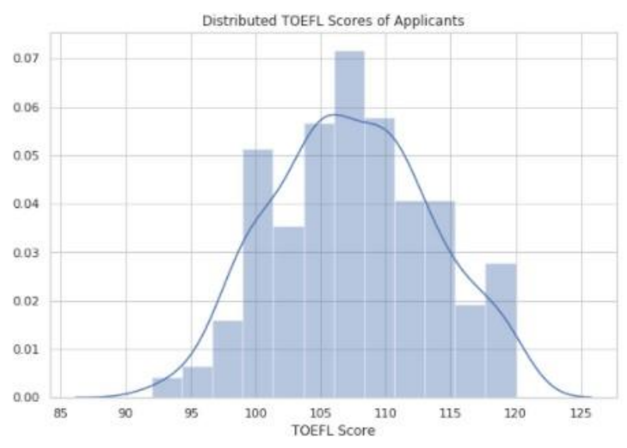


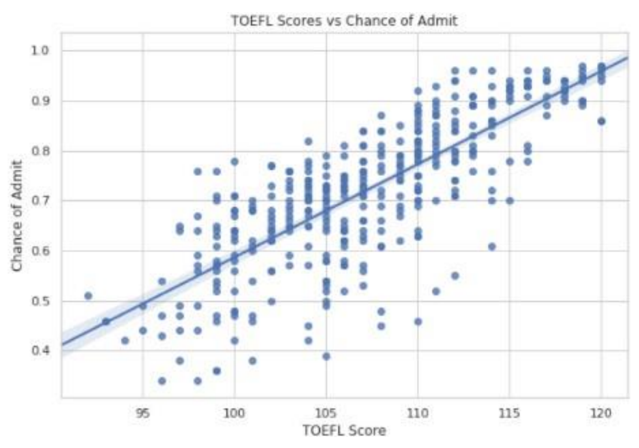
```
In [8]:
plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
sns.distplot(df['TOEFL Score'])
plt.title('Distributed TOEFL Scores of Applicants')

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

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using
a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]`
instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[n
p.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval

Out[8]:
Text(0.5,1,'TOEFL Scores vs Chance of Admit')
```





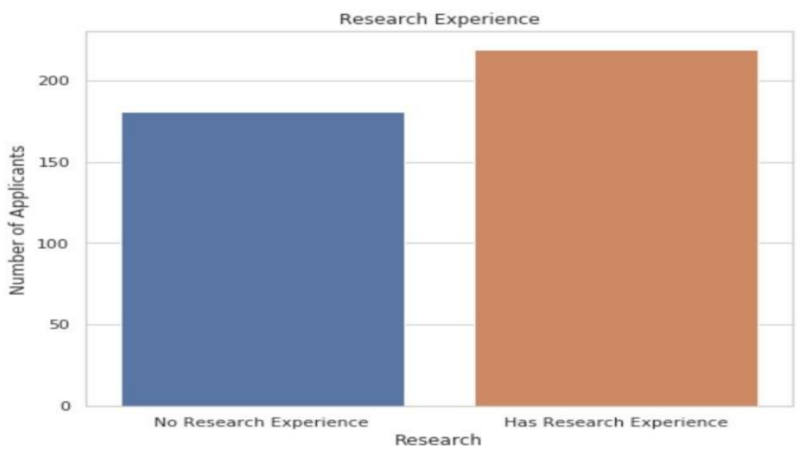
Research

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.

```
In [9]:  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'])
```

```
Out[9]:  [Text(0,0,'No Research Experience'), Text(0,0,'Has Research Experience')]
```



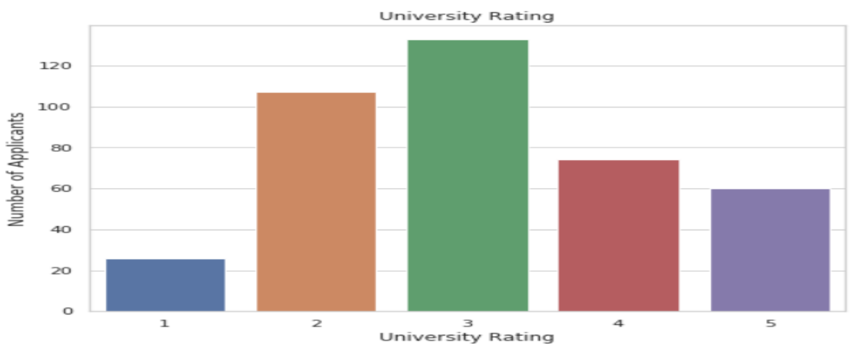
University Rating

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

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

```
In [10]:  fig, ax = plt.subplots(figsize=(8,6))
          sns.countplot(df['University Rating'])
          plt.title('University Rating')
          plt.ylabel('Number of Applicants')
```

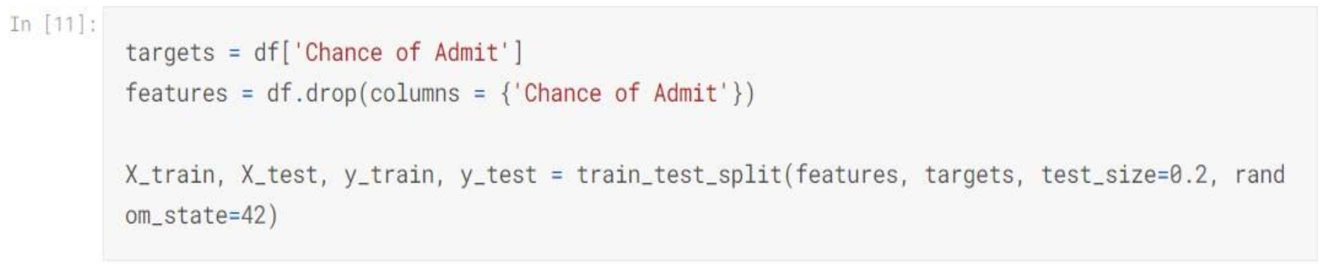
```
Out[10]:  Text(0,0.5,'Number of Applicants')
```



Preparing Data for Machine Learning

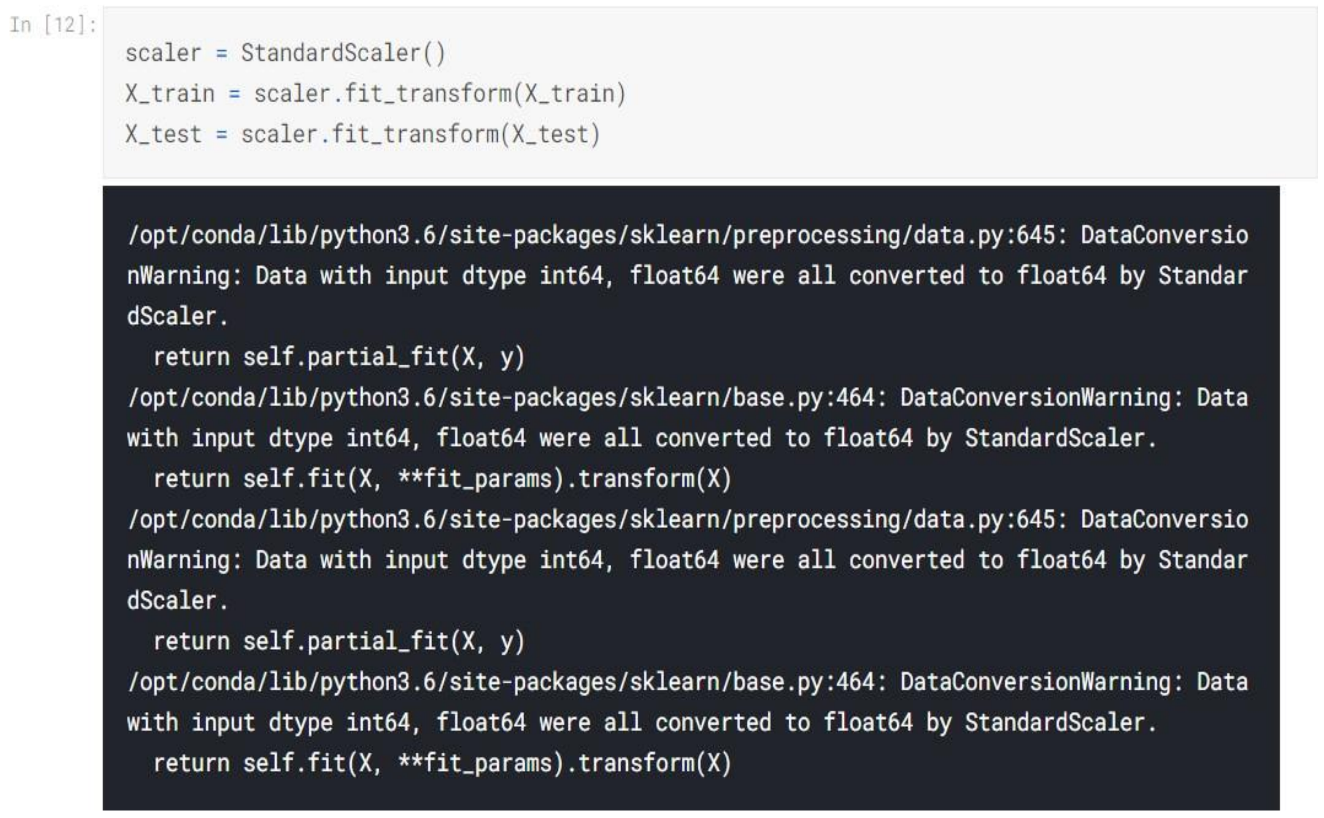
Now that we understand our dataset, it's time to implement machine learning methods to predict future applicant's chances of admission.

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.



```
In [11]:
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, rand
om_state=42)
```



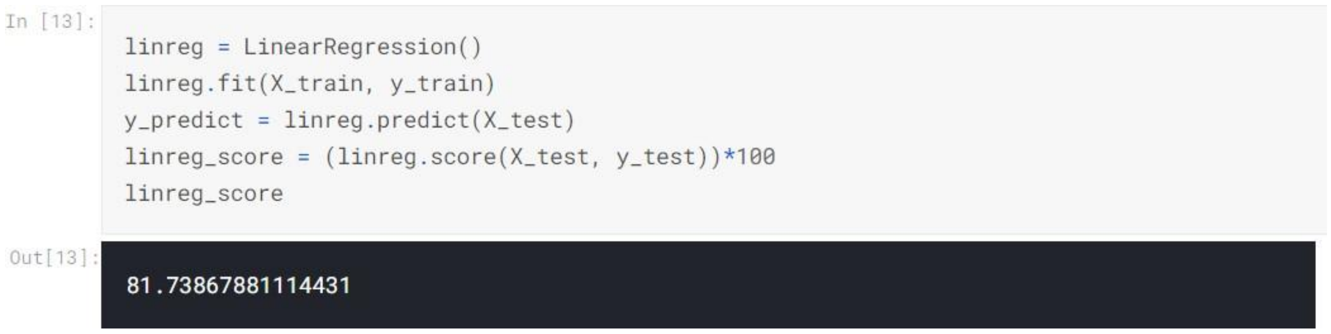
```
In [12]:
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/preprocessing/data.py:645: DataConversio
nWarning: Data with input dtype int64, float64 were all converted to float64 by Standar
dScaler.
  return self.partial_fit(X, y)
/opt/conda/lib/python3.6/site-packages/sklearn/base.py:464: DataConversionWarning: Data
with input dtype int64, float64 were all converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
/opt/conda/lib/python3.6/site-packages/sklearn/preprocessing/data.py:645: DataConversio
nWarning: Data with input dtype int64, float64 were all converted to float64 by Standar
dScaler.
  return self.partial_fit(X, y)
/opt/conda/lib/python3.6/site-packages/sklearn/base.py:464: DataConversionWarning: Data
with input dtype int64, float64 were all converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

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:

1. Linear Regression
2. Decision Trees
3. Random Forests

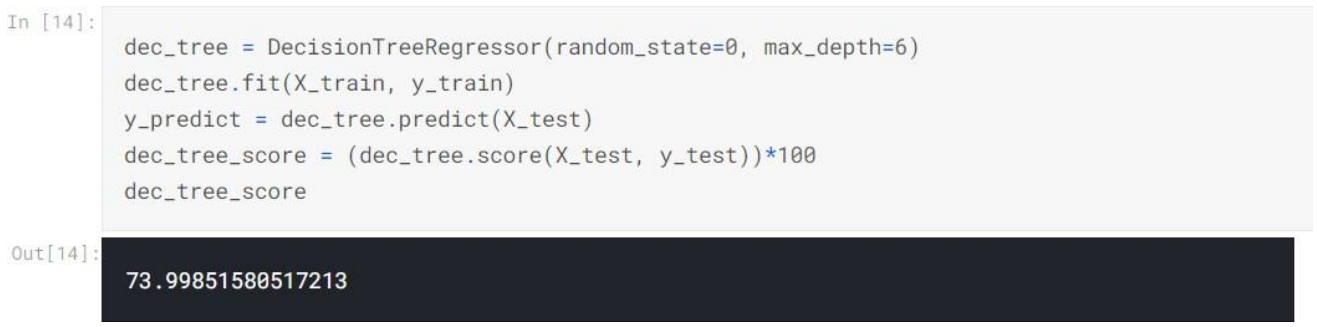
Linear Regression



```
In [13]:
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
```

```
Out[13]:
81.73867881114431
```

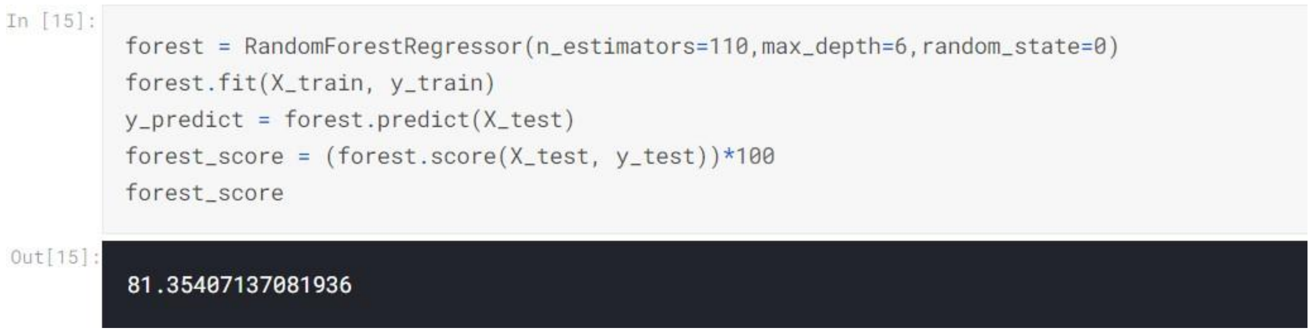
Random Forest



```
In [14]:
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
```

```
Out[14]:
73.99851580517213
```

Decision Tree



```
In [15]:
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
```

```
Out[15]:
81.35407137081936
```

5.Results

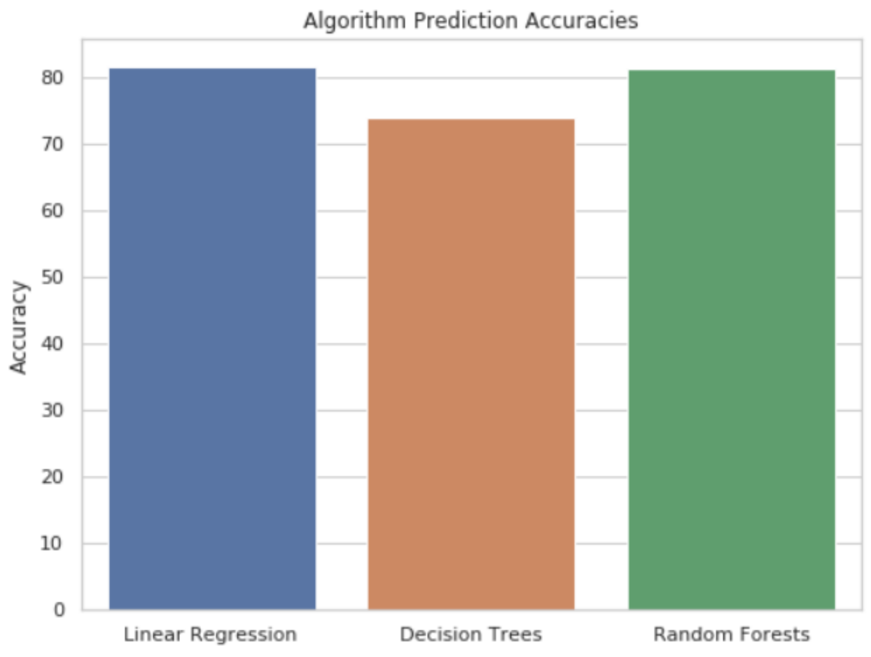
Comparing Scores

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

```
In [16]:
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')
```

```
Out[16]:
Text(0,0.5,'Accuracy')
```



Selecting the Best Algorithm

1. Linear Regression - 81.74%
2. Random Forests - 81.35%
3. 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.

6.Conclusion

My project mainly includes three parts: data preprocessing, base model selection, modeling and stacking. Because the original data is completed and clean without too many features, most energy I spent on my project is the base model selection and stacking. Selecting base model is important so as to I save more energy in result analysis and have base line to compare with following models. I chose Linear Regression as my first model, the truth proves that I do not need to set any hyperparameter but get a better model than average level. It is also a important stepto know the features of original data, which is benefited to choose following models and analysis metrics. Stacking is what I innovate in problem. Because my previous models are good, I want to take full advantages of them, stacking is ideal for this case. The a little bit of progress proves my guess but not too much. I think my final model can be optimized to a certain degree. There are a few factors affecting my modelling. Firstly, except Linear Regression, all regression contains many hyperparameters. Every hyperparameter has its own usage and meaning in Sklearn library. Under the situation that I do not know which algorithm suites my project best, so I only adjusted some hyperparameters. Exploring deeper about them can help improve single model. Secondly, the principle of stacking is simple but its usage is not so easy. How to choose the first level and second level models?What kind of regressors combination can achieve better result? What is the precondition of data and basic models before stacking? How many levels should bebuilt? Thirdly, the original data set is not accurate, this is an important reason why the accuracy can not be improved more. As we know, ”Graduation admission” is true or false problem. Where does the possibility infers from? After reading some materials, these possibility of chance of admission are from appliers. They guess the chance. This is not accurate, which may cause a lot noise in modeling. If we treat this set of data as a classification problem, the bias will be smaller although I think possibility of event happening is more meaningful.

7. References

[1] N. Datta. (2019) Graduate admissions. [Online]. Available:https://www.kaggle.com/nitindatta/graduate-admissions

[2] V. Garnepudi. (2019) Admission chances. [Online]. Available:https://www.kaggle.com/venky73/admission-chances

[3] A. Tiha. (2019) Admission prediction (mean squared error - 0.0013). [Online]. Available:https://www.kaggle.com/anjanatiha/admissionprediction-mean-squared-error-0-0013

[4] suhas maddali. (2018) Chance of admission (deep neural networks). [Online]. Available:https://www.kaggle.com/suhasmaddali007/chanceof-admission-deep-neural-networks

[5] M. Peixeiro. (2018) Linear regression understanding the theory. [Online]. Available: https://towardsdatascience.com/linear-regressionunderstanding-the-theory-7e53ac2831b5

[6] F. Nerdy. What is r squared and negative r squared. [Online]. Available:http://www.fairlynerdy.com/what-is-r-squared/

[7] S. RAY. (2015) 7 regression techniques you should know! [Online]. Available:https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guideregression/

[8] Sklearn. Mlpregressor homepage in sklearn. [Online]. Available: https://scikit-learn.org/stable/modules/neural-networks-supervised.html

[9] A. S. V. (2017) Understanding activation functions in neural networks. [Online]. Available:https://medium.com/the-theory-of-everything/understandingactivation-functions-in-neural- networks-9491262884e0

[10] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[11] S. Singh. (2018) Understanding the bias-variance tradeoff. [Online]. Available: https://towardsdatascience.com/understanding-thebias-variance-tradeoff-165e6942b229