1.INTRODUCTION

Our project 'GRADUATE ADMISSIONS' is a standard Machine Learning approach for predicting the chance of admission for an individual aspiring for a Masters Degree specifically in the United States.Machine Learning is an application of Artificial Intelligence(AI) that provides the capability to a system to think and make decisions just like a human mind without being explicitly programmed.Machine Learning algorithms and methodologies are specifically designed to understand the data provided and apply it to learn on it's own.The people who are aspiring for a Masters Degree will make sure that they are applying for a good university with limitless opportunities for both professional and career orientation.This project aims at providing the individual's an insight of their admission chances by taking in the data of their GRE, TOEFL, CGPA scores and the recommendations like LOR and SOP eventually drawing inferences by providing a rating out of 1(for higher chances) and 0 (for critical chances).

The goal is to develop a model that can predict the score 'Chance Of Admit', that determines an individual's chances of getting an admission in the university he desires to pursue his/her education**.**

The tasks involved are :-

- Download and preprocess the Graduate Admissions data.
- Train a BenchMark Model and record it's performance.
- Then three supervised learning models were trained using the training data and a comparison is done based on the performance metric and decided which among the three is the best model.
- The best model thus selected is optimized using GridSearchCV technique.
- The Optimized model is then compared with the Benchmark model and deciding which is the best for the given data.
- Then the best model is validated against unseen data and documenting the intuition.

The final model can be applied to determine the Chance's for an individual of getting an admission into a specific university of his desire.

2. TOOLS AND TECHNOLOGIES USED

The entire project is built upon Python programming language. The Algorithms and methodologies used in the respective model building are facilitated by the implementation of “SCIKIT- LEARN” framework specifically built for applying Machine Learning to build practical systems that have the capability to make decisions without any external human interference. The statistical part of the project is implemented by the use of “NUMPY” module in Python. The Data Munging part is handled by the implementation of “PANDAS” module and the Data Visualization part is implemented by the aid of “MATPLOTLIB” and “SEABORN ” libraries available in Python. All the algorithms and techniques used in the project for building the model including training,testing,analyzing it’s performance, pre-processing,evaluation and validation are performed by the methodologies available in the “SCIKIT-LEARN” framework of Python.

2.1 INTRODUCTION TO PYTHON

Python is an interpreted, high-level, general-purpose programming language created by Guido van Rossum and first released in 1991. Python is meant to be an easily readable language. It’s formatting is visually clear, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional.Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).Python can be implemented in a procedural way, an object-oriented way or a functional way.

Applications of Python :-

- Building Web Applications
- Developing interactive UI applications
- Software Development
- Gaming Development
- Research and Scientific Purposes

To code, edit and debug Python programs, we need an interactive environment to run the scripts. Hence for this project, entire implementation is performed by the aid of “GOOGLE COLAB”, an exclusive environment built specifically for performing Data Analytics and Model Building.

2.2 SCIKIT-LEARN

Scikit-learn is an open source Machine Learning Framework specifically built for implementing Data Mining and Data Analytics processes efficiently. The automated tools and methodologies available in the scikit-learn facilitate in easy and practical implementation of Algorithms thus meant for building models that fit in the real-time data with less complexity and aid in deployable tier.

Scikit-Learn covers the following implementations upon real time data feed :-

- Classification (Identifying to which class an object belongs to)
- Regression (Predicting a continuous-valued attribute associated with an object)
- Clustering (Grouping of similar objects into sets automatically)
- Preprocessing (Cleaning, processing and filtering out anomalies in the data)
- Dimensionality Reduction (Limiting the number of variables to consider)
- Model Selection (Evaluating, Validating, and Exploring the best model for the data)

2.2.1 SCIKIT-LEARN MODEL

The following architecture clearly describes the tiers and components present in the Scikit-Learn Framework

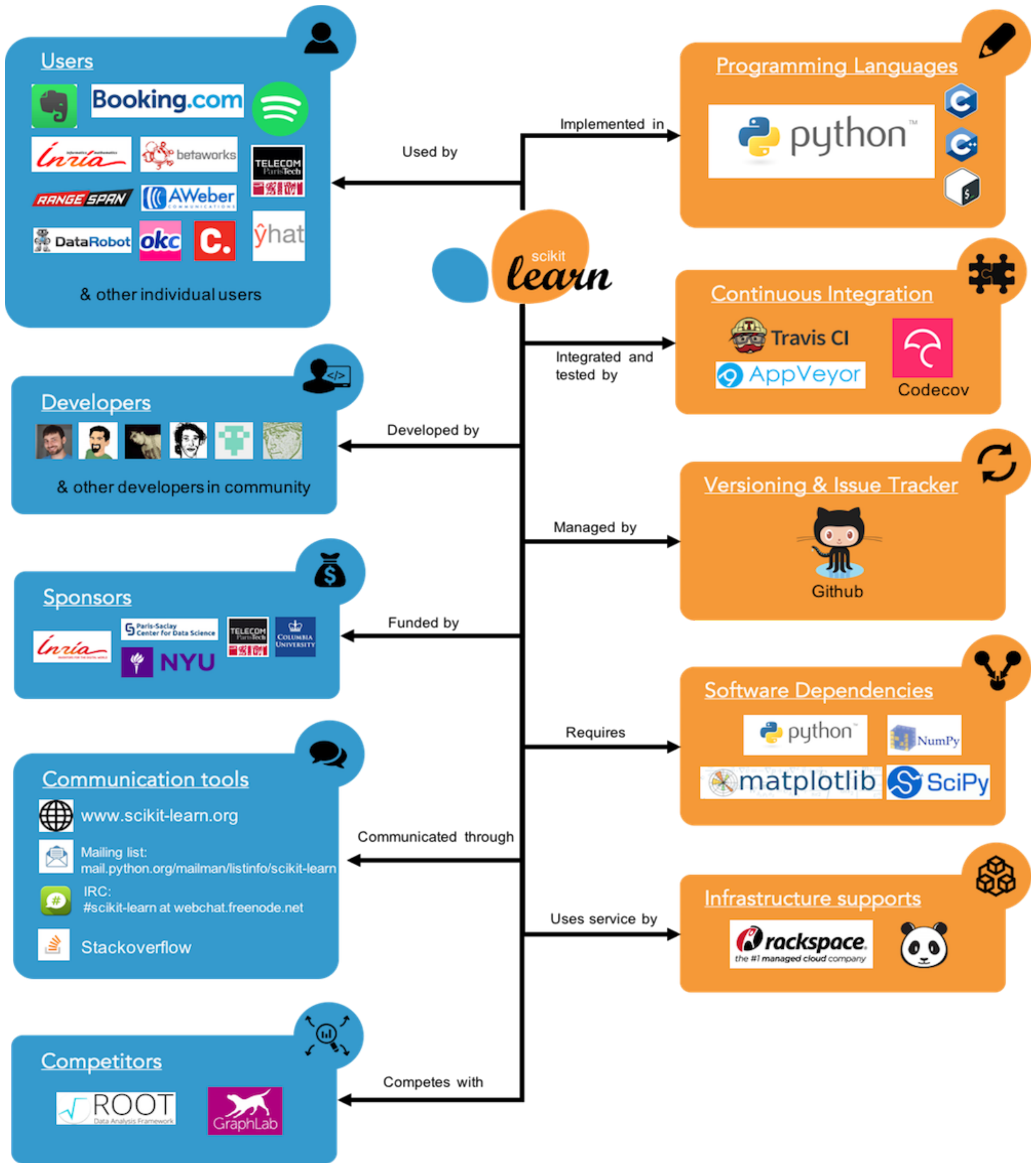


Figure 2.1. SCIKIT-LEARN STRUCTURE MODEL

2.3 NUMPY

Numpy is a dynamic package available in Python specifically built for scientific and statistical computation. NumPy provides automated scientific computations by enriching Python Language with powerful data structures, implementing multi-dimensional arrays and matrices. These data structures guarantee efficient calculations with matrices and arrays.

Advantages of using Numpy with Python:

- array oriented computing
- efficiently implemented multi-dimensional arrays
- designed for scientific computation

2.4 PANDAS

Pandas is an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Pandas is specifically built for performing Data Analysis and Data Modeling, enabling the user or programmer to carry out the entire data analysis workflow in Python.

Salient Features in Pandas Library:

- An interactive DataFrame object for fast and efficient data manipulation process.
- Tools and Methodologies for reading and writing data in different formats such as CSV, MS EXCEL, text files and other different formats available in terms of real time data feed.
- Flexible reshaping.
- Integrated Handling of Missing Data.
- Interactive merging and joining of datasets.
- Widely used in Academic and Commercial domains like Economics, Statistics, Finance etc.
- Provides inline visualizations for the dataframes implementation.

2.5 MATPLOTLIB

Matplotlib is an interactive plotting library available in Python, specifically designed for visualizing the trends in the dataset, thereby documenting certain inferences that could potentially aid in understanding the dataset in a visual scale and is based on the parental "NUMPY" library available in Python.

Types of plots frequently used by the implementation of Matplotlib in Python:-

- Histograms ("hist()" function automatically generates histograms and returns bin counts).
- Bar Charts ("bar()" function is used to make barcharts).
- Pie Charts("pie()" function is used to make pie charts).
- Scatterplots("scatter()" function is used to make scatterplots for visualizing the correlations between the parameters present in the dataset)

2.6 SEABORN

Seaborn is a interactive and dynamic data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Types of plots available and frequently used by the implementation of Seaborn in Python:-

- Histograms
- Kernel Density Estimation (KDE)
- Pairplot
- Barplot etc.

2.7 GOOGLE COLAB

Google Colab is an interactive free cloud service for Data Analysis and Model Building applications which supports free GPU (Nvidia Tesla K80).It is supposedly developed for:-

- improving or providing room for people to code and implement statistical python code.
- developing deep learning applications with the implementation of libraries like TensorFlow, Keras, OpenCV etc.

The present project is built upon this environment where the dataset is to be uploaded from the native computer and the entire process is similar to how users work in the "JUPYTER NOTEBOOK" of the Anaconda Distribution.

2.8 DATA SET DESCRIPTION

The current dataset is extracted from Kaggle published by Mohan S.Acharya, and the dataset is described by the following steps:-

1. Data Acquisition
2. Data Exploration
3. Feature Set Exploration

2.8.1 DATA ACQUISITION

Data Acquisition is one of the key steps in Machine Learning.Since, it is more prerogative to understand the data or just to feed the data as an input to the specific Machine Learning Model is achievable only in a 'Tabular Sense' i.e., if the input dataset is converted to a tabular data, then it is more flexible for the model to build accurate predictions out of the perfected data format.

Code for Data Acquisition

```
# importing the necessary libraries for the project

import numpy as np

import time

import pandas as pd
```

import matplotlib.pyplot as plt

import seaborn as sns

from IPython.display import display #facilitates the use of display() for Data frames

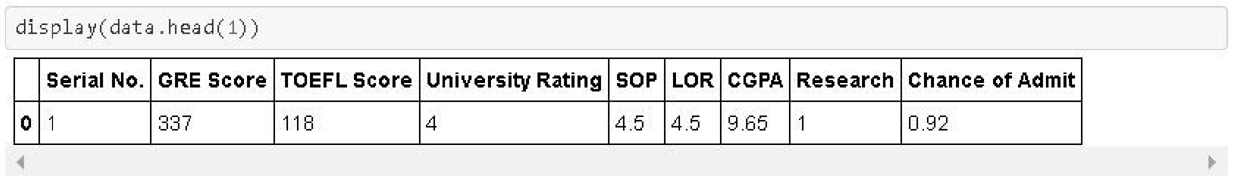
# loading the dataset intended to work out :)

data = pd.read_csv('Admission_Predict.csv')

print("Graduate Admissions dataset has {} data points with {} variables each.".format(*data.shape))

2.8.2 DATA EXPLORATION

- Data Exploration is a crucial step in the process of Machine Learning.It helps us to understand the patterns and available features in a data set from which we can determine the sort of actions that we can perform for further analysis.
- Data Exploration gives an intuition that a cursory investigation of the data-set is necessary for familiarizing Yourself with the data through an explorative process and is a fundamental practice to help you better understand and justify your results.
- Since, the main goal of this project is construct a working model that has the capability of predicting the 'ChanceOfAdmit' scores, we will need to separate the dataset into features and the target variable.



```
display(data.head(1))
```

	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

Figure 2.2. Code sample for Data Exploration

The Intuition obtained from the above cell is described below :-

- We can observe that the features, 'GRE Score' , 'TOEFL Score', 'University Rating' , 'SOP' , 'LOR' , 'CGPA' , 'Research' give us with the quantitative information about each data point.
- The target variable, 'Chance of Admit', will be the variable we intend to predict. These are stored in the variables , features and scores respectively.

Code for Data Exploration Phase

```
# Extracting information by separating the features and the target variable

# Here 'scores -> target variable.

# 'features' -> fundamental elements of the dataset

# the below for loop is to navigate to the last column of the dataset where we will have 'Chance Of Admit' data

for i in data.columns:

   k = i

scores = data[k] #target variable

features = data.drop(k, axis = 1) #input features
```

The following illustrates description and code sample for the necessary statistics implemented:-

- For the initial coding implementation, calculating descriptive statistics about the Chance of Admit Scores is important.These statistics will be extremely important later on to analyze various prediction results from the constructed model.

Code for Summary Statistics in Data Exploration phase

```
# the below for loop is to navigate to the last column of the dataset where we will have
'Chance Of Admit' data

for i in data.columns:

    k = i

# The total number of records in the data

n_records = len(data)

# Minimum score of the data

minimum_score = np.min(data[k])

# Maximum score of the data

maximum_score = np.max(data[k])

# Mean score of the data

mean_score = np.mean(data[k])

# Median score of the data
```

```
median_score = np.median(data[k])

# Standard deviation of scores of the data

std_score = np.std(data[k])

# Showing the calculated statistics

print("Statistics for Graduate Admissions Data Set:\n")

print("Total Number of records : {}".format(n_records))

print("Minimum score: {:,.2f}".format(minimum_score))

print("Maximum score: {:,.2f}".format(maximum_score))

print("Mean score: {:,.2f}".format(mean_score))

print("Median score: {:,.2f}".format(median_score))

print("Standard deviation of scores: {:,.2f}".format(std_score))

#print features
```

2.8.3 FEATURE SET EXPLORATION

- Serial Number -> it is just to show the number of students from which the data was collected on.

- GRE(Graduate Recruitment Exam) Scores (out of 340) is a score that enables an individual to get admission in top universities-> continuous
- TOEFL(Test For English As Foreign Language) Scores (out of 120) is a score that determines the individuals knowledge and speaking English-> continuous
- University Rating (out of 5) is a score that states the ranking of a university based on the facilities and other factors-> continuous
- Statement of Purpose (out of 5) is a long essay that is often asked by the universities abroad-> continuous
- Letter of Recommendation and Strength 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.(out of 5) -> continuous
- Research Experience (either 0 or 1) is a discrete score that determines whether an individual has done a research priorly or not.-> discrete
- Chance of Admit (ranging from 0 to 1) is a continuous score that predicts the probability of an individual for being selected in a particular university -> continuous

Feature Observation :-

- 'Serial No.' is just to show the number of students from which the data was collected on and potentially shows no impact on the target variable i.e., it has no meaning and can be no harm if we remove it from the data set.
- 'GRE Scores', are one of the predominant factors in the determination of an individual's chances for getting admission in a specific university. Since, GRE Scores determine the individuals capability to understand and manage the course curriculum or load. Hence a low or high score of the GRE scores can significantly affect the individual’s chances for getting an admission.
- 'TOEFL Scores' , are also one of the significant factors the determination of an individual's chances for getting admission in a specific university.Since,TOEFL

Scores determine the english speaking capability of a non-native individual. Hence, a low or high score may affect the individual's chances of admission .

- 'University Rating', score that states the ranking of a university based on the facilities and other factors.It plays an important role in the individuals aspirations because the more the rating the higher the standard of education.
- 'Statement Of Purpose ' ,is a long essay that is often asked by the universities abroad,seeks to understand the candidate's life, the motivations for the chosen career path and his/her goals. It is an important part of the application process but it is not that deterministic in our current scenario, since, we only describe the zeal of the individuals who are aspiring for a specific university.
- 'Letter Of Recommendation' , 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.It is crucial because it plays an important role in the admission to institutions of higher education, or scholarship eligibility.
- 'Research' , that determines whether an individual has done a research priorly or not.It shows a good effect on the career chances for an individual.
- 'Chance Of Admit' , is target variable and that we are intended to find out. It determines the Probability of an individual being selected for the university of his desire and it depends on certain important features that we are gonna find out later part of the project.

3. DATASET VISUALIZATION FOR UNDERSTANDING THE DATA

3.1 EXPLORATORY VISUALIZATION

Exploratory Visualization can be defined as an approach for analyzing data sets to summarize their important characteristics, often by the application of visual methods.The Primary theme of Exploratory Visualization is for observing what the data can give us an intuition far beyond the conventional modeling or hypothesis testing tasks.

The Following session of Data Visualization will provide the intuition of how each feature is 'Correlated' with the target variable 'Chance of Admit'.

1. Correlation of 'GRE Score' with respect to 'Chance of Admit'

```
# the below for loop is to navigate to the last column of the dataset where we will have 'Chance Of Admit' data
for i in data.columns:
   k = i
plt.scatter(data['GRE Score'],data[k])
plt.xlabel('GRE Scores')
plt.ylabel('Chance Of Admit')
plt.show()
```

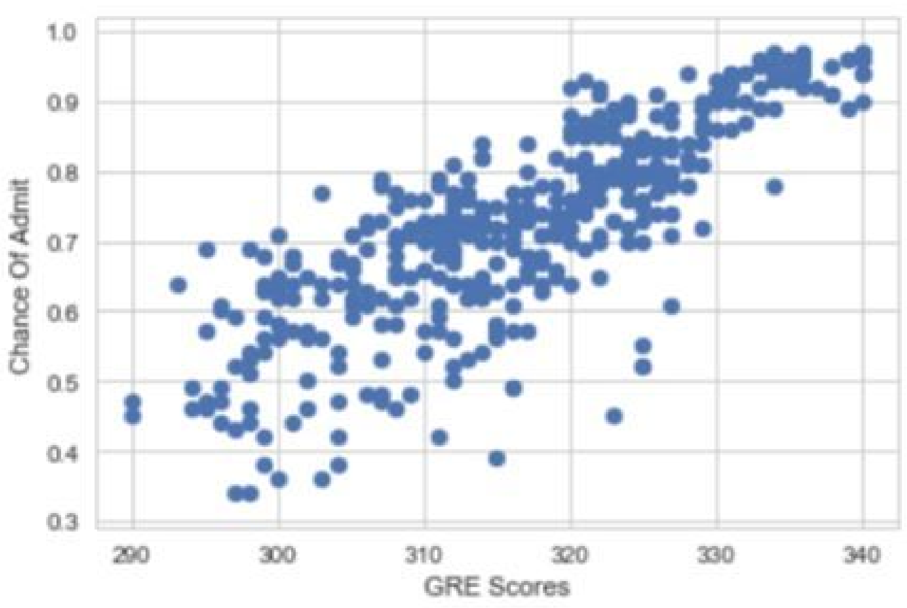


Figure 3.1. Code Sample for establishing correlation between 'GRE Score' and 'Chance of Admit'

Explanation :-

- It is quite evident that the scores of 'GRE Scores' and 'Chance of Admit' are highly correlated i.e., Chances of Admit are high as the GRE Score is trending to nice scores.
- Hence it is an important factor in the determination of the chances of admission to a specific university in the perspective of an individual.

2. Correlation of 'TOEFL Score' with respect to 'Chance of Admit'

Code Implementation

```
plt.scatter(data['TOEFL Score'],data[k])
plt.xlabel('TOEFL Score')
plt.ylabel('Chance Of Admit')
plt.show()
```

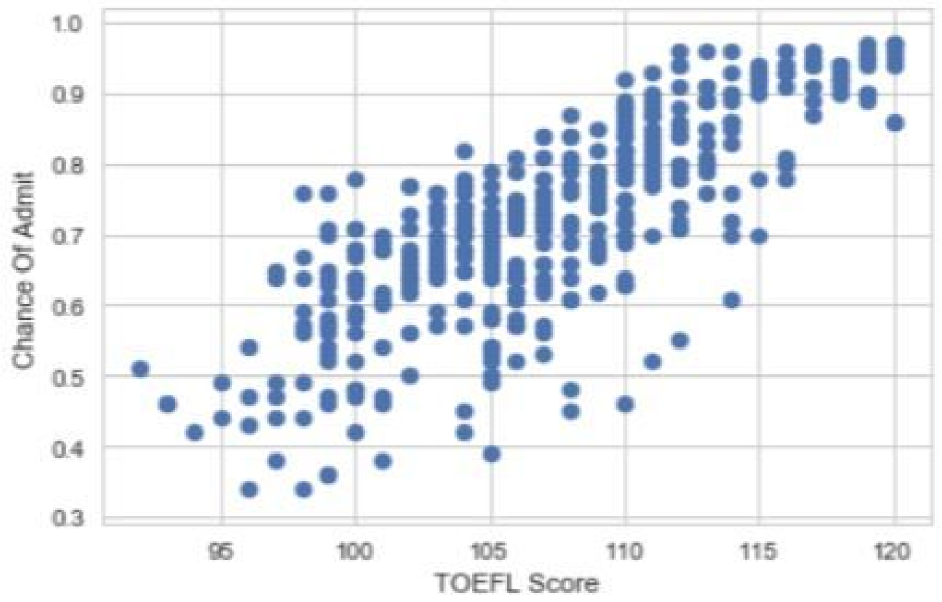


Figure 3.2. Code Sample for establishing correlation between 'TOEFL Score' and 'Chance of Admit'

Explanation

- It is quite evident that the scores of 'TOEFL Scores' and 'Chance of Admit' are highly correlated i.e., Chances of Admit are high as the GRE Score is trending to nice scores.
- Hence it is an important factor in the determination of the chances of admission to a specific university in the perspective of an individual.

3. Correlation of 'University Rating' with respect to 'Chance of Admit'

Code Implementation

```
sns.set(style = 'whitegrid')
plt.scatter(data['University Rating'],data[k])
plt.xlabel('University Rating')
plt.ylabel('Chance Of Admit')
plt.show()
```

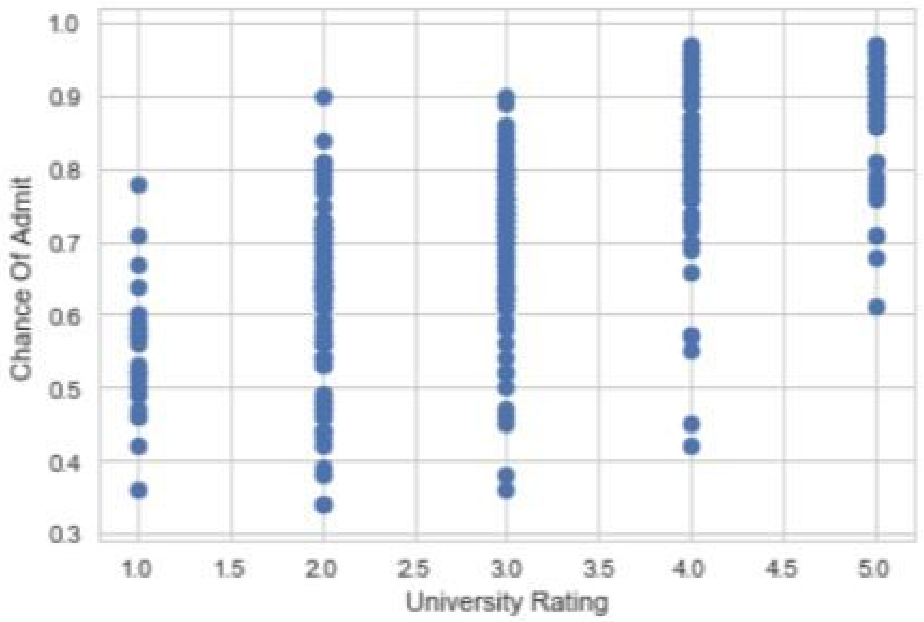


Figure 3.3. Code Sample for establishing correlation between 'University Rating' and 'Chance of Admit'

Explanation

- It intuitively shows the intimate relationship between the 'University Rating' and the 'Chance of Admit'.
- It seems that most of the people who are admitted for a certain university is predominantly concentrated between University Rating '2.0' and '4.0'.

4. Correlation of 'Statement of Purpose' with respect to 'Chance of Admit'

Code Implementation

```
plt.scatter(data['SOP'],data[k])
plt.xlabel('Statement of Purpose')
plt.ylabel('Chance Of Admit')
plt.show()
```

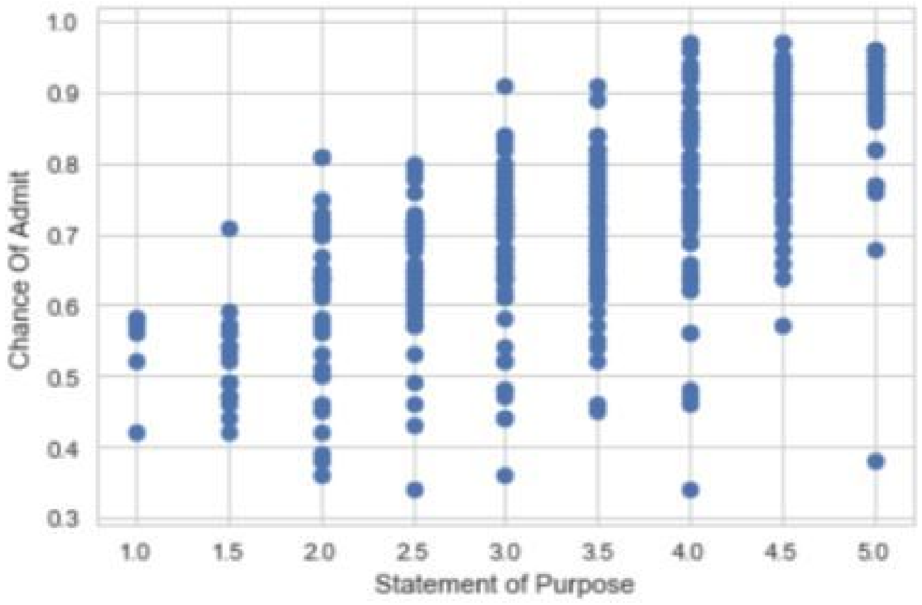


Figure 3.4. Code Sample for establishing correlation between 'SOP' and 'Chance of Admit'

Explanation

- It describes the skill of an individual to write effectively to an abroad university regarding his life style, motivation for chosen career path and his/her goals.

5. Correlation of 'LOR' with respect to 'Chance of Admit'

Code Implementation

```
# below for loop is to obtain the index of 'LOR' in the dataset
j = 0
for i in data.columns:
    if j == 5:
        n = i
    j += 1
#print data[n]
# visualizations
```

```
plt.scatter(data[n],data[k])
plt.xlabel('Letter of Recommendation')
plt.ylabel('Chance Of Admit')
plt.show()
```

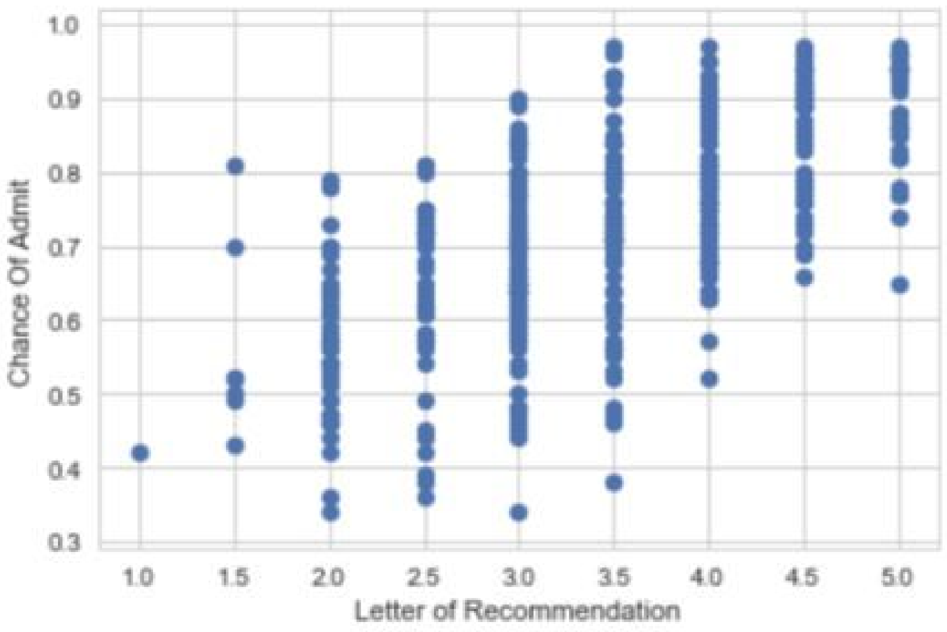


Figure 3.5. Code Sample for establishing correlation between 'LOR' and 'Chance of Admit'

Explanation

- It describes a sheer intuition that the Letter Of Recommendation shows a striking factor in influencing an individual's overall chances for getting admission in a specific university.

6. Correlation of 'CGPA' with respect to 'Chance of Admit'

Code Implementation

```
plt.scatter(data['CGPA'],data[k])
plt.xlabel('CGPA')
plt.ylabel('Chance Of Admit')
plt.show()
```

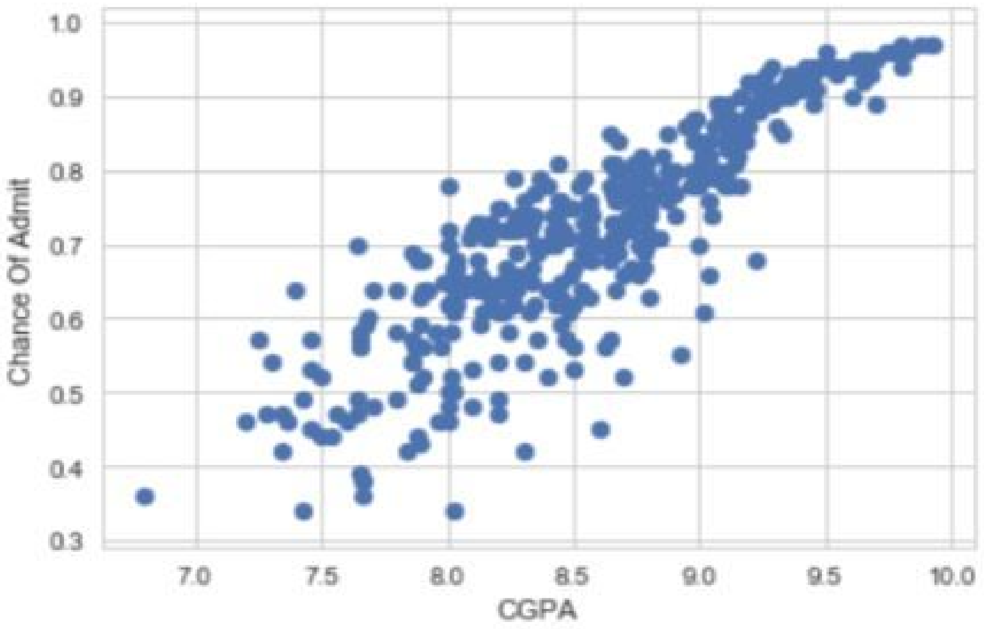


Figure 3.6. Code Sample for establishing correlation between 'CGPA' and 'Chance of Admit'

Explanation

- It clearly shows that the individuals who got higher scores in their Under Graduation, have higher scope of getting admission in a desired university.

7. Correlation of 'Research' with respect to 'Chance of Admit'

Code Implementation

```
plt.scatter(data['Research'],data[k])
plt.xlabel('Research')
plt.ylabel('Chance Of Admit')
```

plt.show()

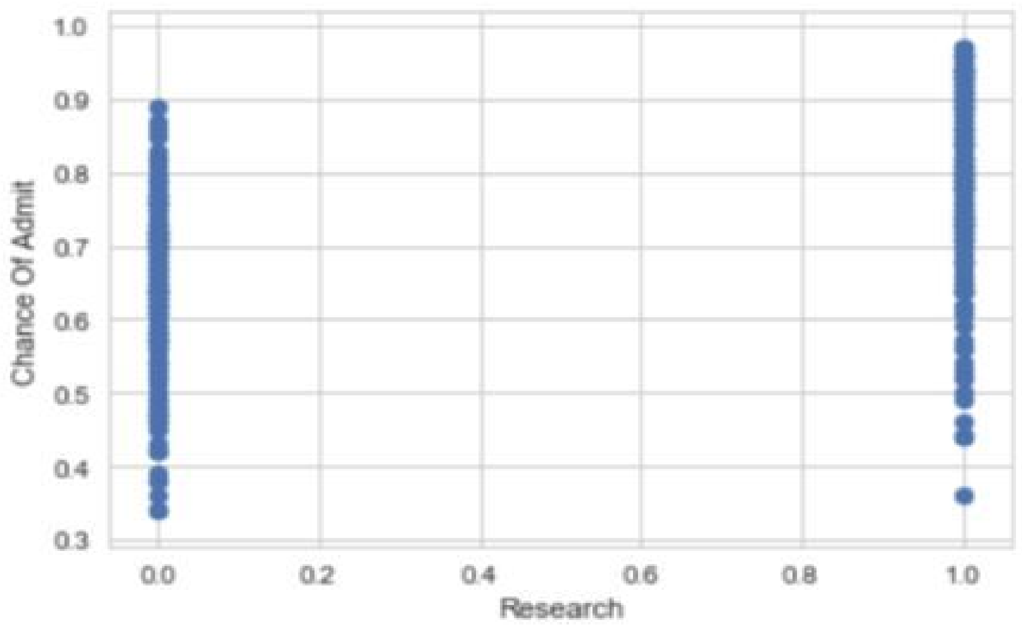


Figure 3.7. Code Sample for establishing correlation between 'Research' and 'Chance of Admit'

Explanation

- From this we can understand that the students who don't pursue a Research in their career before going to a Masters Program may also possess a good chance of getting admitted to a specific university of their desire.
- And it also states how the chances predominantly increased with the pursuing of Research in their career before going to a Masters Program.

Intuition

From these scatter plots, we can understand the exceptional relationships between the features and the target variable.Hence we can state that the data is consistent and show's striking relationships that could make the model more accurate and reliable for predicting the data values that are new and novel to the model.

4. METHODS OF PREPROCESSING APPLIED TO THE DATASET

4.1 DATA PREPROCESSING

- Data Preprocessing is one of the key procedures in Machine Learning that involves transforming the raw input data into an understandable format.
- This step is quite crucial because the Real-World data is often incomplete and inconsistent, or lacking in certain behaviours or trends, and is most probably consists of many errors.
- Data Preprocessing is a proven procedure for solving this problems and it prepares raw data for further processing.
- In the current problem an irrelevant feature is present that is 'Serial No',which actually plays no role in the determination of the 'Chance Of Admit' rate.Hence, it is quite valid to remove it from the dataset.

4.1.1 Criteria for Data Preprocessing

A typical data preprocessing procedure has the following criteria :-

- Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data.
- Noisy: containing errors or outliers.
- Inconsistent: containing discrepancies in codes or names.

4.1.2 Tasks in Data Preprocessing

- Data cleaning: fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.

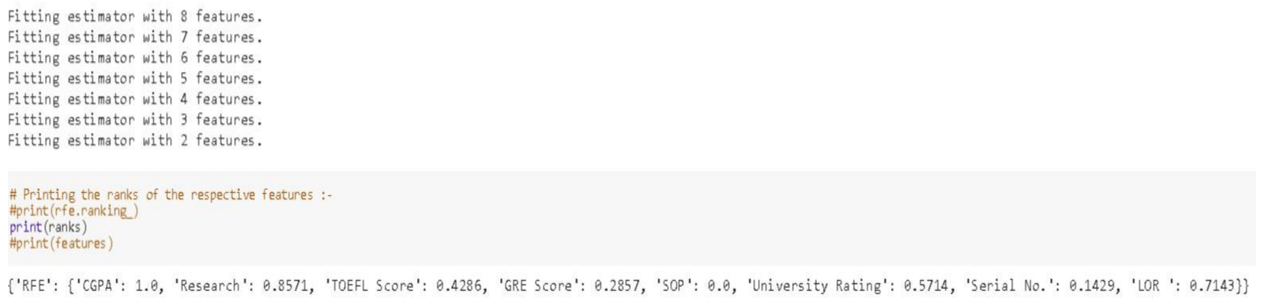
- Data discretization: part of data reduction, replacing numerical attributes with nominal ones.
- Data integration: using multiple databases, data cubes, or files.
- Data transformation: normalization and aggregation.
- Data reduction: reducing the volume but producing the same or similar analytical results.

4.2 FEATURE SELECTION

- Feature Selection is an important concept in the Data Preprocessing phase which involves the removal of irrelevant attributes and retaining only those features that have show better performance.
- I implemented Feature Selection using a Linear Regressor, the benchmark model that i used to prune the features that actually have a lower rank of performance.
- Recursive Feature Elimination(RFE) uses a model (here Linear Regressor) to select either the best or worst performing feature, and then simples prunes the feature. After this the entire process is iterated until all the features in the data set are used up(limit).
- Sklearn has an inbuilt RFE function i.e., sklearn.feature_selection and I'm using this along with my Linear Regressor Model.

Code Implementation

```
# storing the associated ranks of the features.
from sklearn.preprocessing import MinMaxScaler
ranks = {}
# Creating a function which stores the feature rankings to the ranks dictionary
def ranking(ranks, names, order=1):
    min_max = MinMaxScaler()
    ranks = min_max.fit_transform(order*np.array([ranks]).T).T[0]
    ranks = map(lambda m: round(m,4), ranks)
    return dict(zip(names, ranks))
# implementing feature selection using RFE
f= features
v = scores
# import RFE from sklearn.feature_selection
from sklearn.feature_selection import RFE
# using a linear regressor
lr = LinearRegression(normalize=True)
# stop the search when only the last feature is left
rfe = RFE(lr, n_features_to_select=1,verbose=3,step=1)
rfe.fit(f,v)
#print(rfe.ranking_)
ranks["RFE"] = ranking(list(map(float, rfe.ranking_)), data.columns, order=-1)
```



```
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
```

```
# Printing the ranks of the respective features :-
#print(rfe.ranking_)
print(ranks)
#print(features)
```

```
{'RFE': {'CGPA': 1.0, 'Research': 0.8571, 'TOEFL Score': 0.4286, 'GRE Score': 0.2857, 'SOP': 0.0, 'University Rating': 0.5714, 'Serial No.': 0.1429, 'LOR ': 0.7143}}
```

Figure 4.1. Code sample for Feature Selection Implementation

Observation

- Hence, by this intuition I came to the conclusion that the features with worst ranking i.e., the features having the rank with highest values are literally pruned from the dataset.
- Here, we observe that the feature obviously 'Serial No.' although having a good rank, it potentially adds nothing to the value of determining the score of 'Chance of Admit'.
- Although the features such as 'CGPA' , 'Research' 'SOP' are given high scores, since they are key factors in determining an individual's personal qualification for getting a fair admission and the universities in abroad have a different set of factors to consider from an individual before providing the admission. As the 'CGPA' ,'SOP' and 'Research' are the key elements in providing an individual scholarships and other benefits, it worth not to ignore them.
- Hence, I decided to remove 'Serial No.' attribute from the dataset.

4.2.1 Removing Irrelevant Attributes

1. **Removing Irrelevant Attribute 'Serial No.' from the data set**
   Code Implementation

```
# Removing 'Serial No.' from the data set.
features = features.drop('Serial No.',axis=1)
# Fresh features after the irrelevant attribute removal
print(features)
```

5. MACHINE LEARNING ALGORITHMS APPLIED

5.1 ALGORITHMS AND TECHNIQUES

By observing the problem, it is quite evident that it is a 'Regression' Problem. It is important to understand the intuition behind the consideration of specific model, since it has to generate an optimal possibility of results that can improve the model's performance on a sample of new data. Hence, taking the performance of the model into consideration, I chose three Supervised Machine Learning Algorithms that can be better compatible for the data being available.

They are :-

- Decision Trees
- Ensemble Methods - Random Forests
- Support Vector Machines (SVM)

5.1.1 DECISION TREES

- Decision Trees are very flexible,easy to understand and easy to debug.They usually work on Classification and Regression Problems i.e.,for categorical problems having [green,red,blue..etc.] and continuous inputs like [2.9,3.8..etc].They usually cover both the perspectives.
- It divides the dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches,

each representing values for the attribute tested. Leaf node represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

- As the given data is comprised of continuous features,Decision Trees perform well in regression tasks.

5.1.2 ENSEMBLE METHODS - RANDOM FORESTS

- Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes in case of the classification tasks or mean prediction for the regression tasks of the individual trees.
- The Random forests have a stroke of brilliance when a performance optimization happens to enhance precision of the model, or vice versa.Tuning down the fraction of features that is considered at any given node can let you easily work on datasets with thousands of features.
- Since Random Forests perform well on almost every machine learning problem and they also show less overfit behavior when compared to Decision Trees. Since our problem is composed of a lot of continuous features for which Random Forests serve a better choice.

5.1.3 SUPPORT VECTOR MACHINES

- SVM's are simple,accurate and perform well on smaller and cleaner datasets.It can be more efficient as it uses subset of training points.
- The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. Initially as the output is a real

number and continuous it becomes very difficult to predict the information at hand, which has infinite possibilities.

- In the case of regression, the factor margin of tolerance (epsilon) is set in approximation to the SVM.
- The main theme for SVM is always to minimize error, particularize the hyperplane which maximizes the margin, keeping in mind that part of the error is convinced.
- Since For the current regression problem consists of a lot of continuous features the application of SVM's can serve a better purpose.

5.2. OBSERVATION

By the next approach, I will choose the best of these of three models, that can be an optimized model for the data set.I will then use the performance metric (r2_score) and compare the three potential based on their scores, the model which has the best r2_score will be eventually considered for further analysis.Eventually I'll optimize the selected model by 'GridSearchCV' and evaluate the model by comparing the final r2_score of the optimized model and the benchmark model.

5.3 METRICS

- The current problem is a Regression task, since it takes certain features as inputs and attempts to find a score that helps an individual to get an idea about the chances of getting an admission in a specific university.
- Hence, Coefficient Of Determination is considered as the performance metric that can be applied to compare the performances of the scores obtained from the BenchMark and the Optimal Model considered.
- The CoEfficient Of Determination(R^2) is the key output of the Regression Analysis.It can be defined as the proportion of the variance in the dependent variable that is predictable from the independent variable.

- It's values ranges from 0 to 1, and the results are given intuition by, if:-
    1. The value of R^2 -> 0 , indicates that the model is a worst fit to the given data.
    2. The value of R^2 -> 1 , indicates that the model is the best fit to the given data.
    3. The value of R^2 in between 0 and 1 -> indicates that the respective variability exhibited by the target variable.
- The formula for CoEfficient Of Determination(R^2) is given by :-

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

- From the above formula SSreg is called Sum of Squares of Residuals, also called the Residual Sum of Squares
- And the SStot is called the Total Number of Squares.

5.3.1 IMPLEMENTATION OF THE COEFFICIENT OF DETERMINATION

```
# Import 'r2_score'
from sklearn.metrics import r2_score
def performance_metric(y_true, y_predict):
    """ Calculates and returns the performance score between
        true and predicted values based on the metric chosen. """
    # TODO: Calculate the performance score between 'y_true' and 'y_predict'
    score = r2_score(y_true,y_predict)

    # Return the score
    return score
```

5.3.2 SHUFFLING AND SPLITTING OF DATA

```
# Import 'train_test_split'
from sklearn.cross_validation import train_test_split
# Shuffle and split the data into training and testing subsets
X_train,X_test,y_train,y_test=
train_test_split(features,scores,test_size=0.2,train_size=0.8,random_state=10)
# Successful shuffle and split of the data
print("Training and testing split was successful.")
```

5.4 BENCHMARK MODEL

- A Benchmark Model can be defined as a standard model that already shows a better performance on a given data.The factors on which our results or the solution is tested, are mostly going to be the amount of training/testing data, and then we compare your solution with that of the benchmarked solution obviously based on a performance metric(here r2_score).
- The main theme here is to understand which model works delivers the best solution than their existing solution.So, it can be achieved by sheer analysis, implementing standard algorithms and observance and coming to the conclusion that the model shows good solutions or results than the benchmark model's solution.
- Since, the problem is a 'Regression' task , I'm implementing a 'Linear Regression' model as my BenchMark Model.

Code Implementation

```
# importing Linear Regression model library from sklearn
from sklearn.linear_model import LinearRegression
# Create a linear regression object
reg = LinearRegression()
# Fitting the model using the training sets
reg.fit(X_train, y_train)
# Making predictions using the testing set
y_pred = reg.predict(X_test)
# Calculating the performance of the model
score  = performance_metric(y_test,y_pred)
print("Model has a coefficient of determination, R^2, of {:.3f}.".format(score))
```

Explanation

- It is evident that the Linear Regression Model (BenchMark Model) shows a striking performance of 0.757 which means that the target variable('Chance Of Admit') captured 75.7% of variance.

6. IMPLEMENTATION OF MACHINE LEARNING ALGORITHMS

6.1 IMPLEMENTATION

In the further section of the project, I'll intuitively select the best out of the three models that I considered for the current problem by using the performance metric(r2_score), based on the results generated, I'll decide best of the three models, which is optimal for the given problem.

6.1.1 INITIAL MODEL EVALUATION

In this section, I'll clearly show the coding implementation of the three supervised learning models

- Import the necessary libraries and initialize the models and store them in respective variables.
- And finally comparing the r2_scores of the three learning models and decide which one is the best.

1. Decision Tree Regressor

   Code Implementation

```
# import necessary library

from sklearn.tree import DecisionTreeRegressor

# Create a Decision Tree Regressor Object

dec_reg = DecisionTreeRegressor(random_state=42)
```

```
# Fitting the model using training sets

dec_reg.fit(X_train,y_train)

# Making predictions using the testing set

dec_pred =  dec_reg.predict(X_test)

# Calculating the performance of the decision tree regressor model

dec_score = performance_metric(y_test,dec_pred)

print("Model has a coefficient of determination, R^2, of {:.2f}.".format(dec_score))
```

2. Support Vector Regressor

Code Implementation

```
# import necessary library

from sklearn.svm import SVR

# Create a Support Vector Regressor Object

svr_reg = SVR()

# Fitting the model using training sets

svr_reg.fit(X_train,y_train)

# Making predictions using the testing set

svr_pred =  svr_reg.predict(X_test)

# Calculating the performance of the decision tree regressor model

svr_score = performance_metric(y_test,svr_pred)

print("Model has a coefficient of determination, R^2, of {:.2f}.".format(svr_score))
```

3. Random Forest Regressor

Code Implementation

```
# import necessary library

from sklearn.ensemble import RandomForestRegressor

# Create a Random Forest Regressor Object

rfr_reg = RandomForestRegressor(random_state=42)

# Fitting the model using training sets

rfr_reg.fit(X_train,y_train)

# Making predictions using the testing set

rfr_pred =  rfr_reg.predict(X_test)

# Calculating the performance of the decision tree regressor model

rfr_score = performance_metric(y_test,rfr_pred)

#plt.scatter(y_test,y_pred)

#plt.show()

print("Model has a coefficient of determination, R^2, of {:.2f}.".format(rfr_score))
```

6.1.2 CHOOSING THE BEST MODEL

Since, it is obvious that the model which has best high r2_score when compared to the other models can be termed as the best optimal model for the current problem, as the fact that if :-

- r2_score is 0 -> it indicates that the model is a worst fit to the given data.
- r2_score is 1 -> it indicates that the model is the best fit to the given data.

- r2_score in between 0 and 1 -> indicates that the respective variability exhibited by the target variable.

The values can be tabulated as follows :-

Results:

Metric	Decision Tree Regressor Model	Support Vector Model	Random Forest Regressor Model
r2_score	0.54	0.57	0.71

Intuition

- The major analysis that can be obtained from the above tabulated data is that the 'Random Forest Regressor Model' is the best model among the three, since it exhibits a high score of 0.71i.e, the target variable accounted for about 71% of the variance.
- And Support Vector Regressor shows a score of '0.57' which indicates that 57% is accounted for the target variable..
- Decision Tree Regressor shows a decent score of '0.54' which indicates that 54% is accounted for the target variable.

6.2. REFINEMENT

In this section of the project, the model('Random Forest Regressor Model') is optimized by the application of 'GridSearchCV' technique for fine tuning the parameters for the final model thus chosen and later calculating the performance metric(r2_score) of the optimized model.

6.2.1 MODEL TUNING

Here, I will find the implementation of GridSearchCV by initially importing the libraries sklearn.grid_search.GridSearchCV and sklearn.metrics.make_scorer.

1. Initialize the regressor('Random Forest Regressor Model') and store it in the variable 'lgr_grid'.

2. Creating a dictionary of parameters, in the variable parameters.

3. Using make_scorer to create a r2_score scoring object. -> scorer

4. Perform Grid Search on the Regressor lgr_grid using the 'scorer', and store it in grid_obj.

5. Fit the Grid Search Object to the training data (X_train, y_train) and store it in the grid_fit.

Code Implementation

```
# importing necessary libraries
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import make_scorer
from sklearn.grid_search import GridSearchCV
def fit_model(X_train,y_train):
   # Initialize the Regressor
   lgr_grid = RandomForestRegressor(random_state=42)
   # Create the parameters list to tune.
```

```
params = {"n_estimators"      : [105,205,305],
          "max_features"      : ["auto", "sqrt", "log2"],
          "min_samples_split" : [2,4,8],
          "bootstrap": [True, False]}
# Make an r2_scorer scoring object using make_scorer()
scorer = make_scorer(performance_metric)
# Perform grid search on the classifier using 'scorer' as the scoring method using GridSearchCV()
grid_obj = GridSearchCV(lgr_grid,params,scoring=scorer)
# Fit the grid search object to the training data and find the optimal parameters using fit()
grid_fit = grid_obj.fit(X_train,y_train)
# calculate the estimator
best_clf = grid_fit.best_estimator_
# Return the optimal model after fitting the data
best_predictions = best_clf.predict(X_test)
return grid_fit.best_estimator_
```

Observation :-

- It is quite obvious that the 'Random Forest Regressor' Model showed a best improvement upon Model Tuning using the GridSearchCV technique.

We can observe the trend of the improvement by the values tabulated as follows :-

Metric	**Random Forest Regressor(Before Tuning)**	**Random Forest Regressor Model(After Tuning)**
r2 score	0.71	0.761

It is quite evident from the table that the Random Forest Regressor showed a good improvement upon optimization by the target variable accounting for about 76.1% of the variance.

6.3. FINAL MODEL EVALUATION

In this part of the project I'll demonstrate the comparison of the performances between the BenchMark Model('Linear Regressor') and the Optimal Model('Random Tree Regressor') based on their performance metrics(r2_score) in a tabular form.

Results:

Metric	**Linear Regressor Model(BenchMark Model)**	**Random Forest Regressor Model(Optimal Model)**
r2 score	0.757	0.761

Observation :-

Although the performances of the BenchMark Model and the Optimal Model thus considered are quite figurative and are literally showing nearly similar performances, it is quite evident that the Random Forest Regressor Model(Optimal Model) shows an awesome performance than the Linear Regressor Model(BenchMark Model).

And it is quite obvious that Random Forest Regressor Model i.e., Optimal Model shows a better performance on the input dataset.

6.4. MODEL VALIDATION

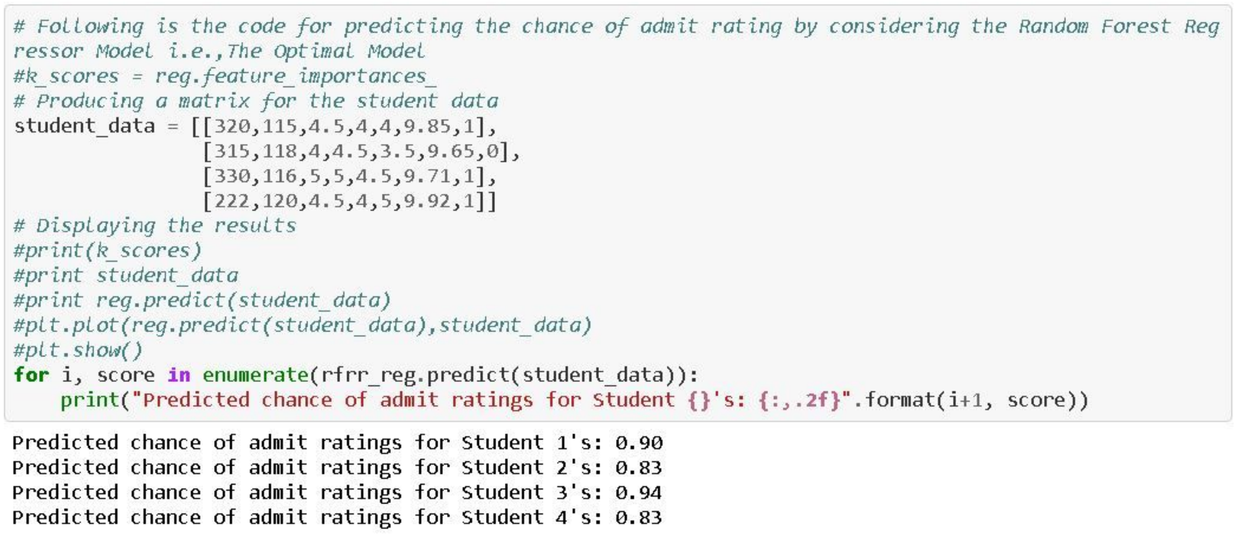
In this part of the project, I'll demonstrate the performance of the Best Model for the given regression task i.e., The Optimal Model against unseen data.

6.4.1 Task - Predicting The Ratings of Chance Of Admit

Imagine that there are four students who want to pursue Masters Program in the United States Of America and for that they have already written and arranged all the prerequisites like GRE Scores, TOEFL Scores etc. they now want to know the chances of being admitted to the university of a specific rating for which they are willing to study.

Reckon that the data of the three students are as follows :-

Feature	Student 1	Student 2	Student 3	Student 4
GRE SCORE	320	315	330	222
TOEFL SCORE	115	118	116	120
University Rating	4.5	4	5	4.5
SOP	4	4.5	5	4
LOR	4	3.5	4.5	5
CGPA	9.85	9.65	9.71	9.92
Research	1	0	1	1



```
# Following is the code for predicting the chance of admit rating by considering the Random Forest Reg
ressor Model i.e.,The Optimal Model
#k_scores = reg.feature_importances_
# Producing a matrix for the student data
student_data = [[320,115,4.5,4,4,9.85,1],
                [315,118,4,4.5,3.5,9.65,0],
                [330,116,5,5,4.5,9.71,1],
                [222,120,4.5,4,5,9.92,1]]
# Displaying the results
#print(k_scores)
#print student_data
#print reg.predict(student_data)
#plt.plot(reg.predict(student_data),student_data)
#plt.show()
for i, score in enumerate(rfrr_reg.predict(student_data)):
    print("Predicted chance of admit ratings for Student {}'s: {:,.2f}".format(i+1, score))
```

```
Predicted chance of admit ratings for Student 1's: 0.90
Predicted chance of admit ratings for Student 2's: 0.83
Predicted chance of admit ratings for Student 3's: 0.94
Predicted chance of admit ratings for Student 4's: 0.83
```

Figure 6.1. Code Sample for Model Validation Implementation

Intuition

- For Student-1 the predicted score is quite accurate because has a significant good scores in GRE, TOEFL,and CGPA, and also has research experience and the university that he is willing to get admitted is a high rated one i.e, 4.5 it expects the students with reasonably high scores than good scores, so chances for getting an admission is possible pretty easily of his performance and hence the admit rate is also very high.
- For Student-2 the predicted score is accurate because he has a considerably good scores in GRE, TOEFL and CGPA but he doesn't have any research experience and also has the aspiration for getting an admission in a high rated university. These factors might hindered his admitting chance rate to an above average one.
- For Student-3 the predicted score is accurate because he has a consistent good scores in GRE, TOEFL, CGPA and also has a research experience. Since the scores are pretty high and also has the aspiration for getting an admission in a

high rated university which can be possible pretty easily of his performance and hence the admit rate is also very high.

- For Student-4 the the predicted score is accurate because although he has good TOEFL and CGPA scores and indeed having a research experience, it is quite evident that he has pretty low GRE Score and also has the aspiration for getting an admission in a high rated university, but due to low score in GRE, his chances of admission were dropped to above average rating.

7. CONCLUSION

By observing the validation results above, it is quite evident that the model is performing well on the given data.

When compared to the BenchMark Model, the Optimal Model('Random Forest Regressor Model') indeed shows a striking performance as shown in the Table-4 of the 'Final Model Evaluation' section.

- Although the performance scores are so close and similar, it is likely to understand that the Optimal Model('Random Forest Regressor Model') holds a tight control in generalizing the input data.
- The performance of the BenchMark Model is 75.7 % and that of the Optimal Model is 76.1%.

It is also important to note that the individual’s chances of getting an admission in their dream university are predominantly dependent on his/her performance in the prerequisite tests and will to succeed in their respective careers.

The model can be applied or is useful for students who are aspiring for a certain university of their desire to pursue their education and build their career.

In summary the application of the model is quite useful only in the domain of 'Education' and that the data should be very consistent that if any discrepancies in the features may lead to bad predictions i.e., a good student may get a bad prediction and a student with ill IQ can gain advantage of it.

8. IMPROVEMENT

Potentially the 'Data Preprocessing' phase is the crucial part of any Machine Learning Problem, Since during this phase we can potentially identify the flaws in the data set that could actually mess the results and performance of the model thus considered.

- Hence, removing irrelevant data during this phase can predominantly increase the model's performance and can benefit in generalized results.
- But I personally feel that the feature selection that I used is good but there are other techniques that are used for the application of Feature Selection.
- Since, this is a regression task, the Wrapper Method implementation i.e., RFE may not be the best one.
- There is also room for trying Embedded method such as LASSO and Elastic Net and Ridge Regression that don't need the external implementation of the feature selection techniques , since these methods have embedded feature selection and regularization built in. It's worth a trial, since there is always scope for improving the model performance agaInst the given dataset.
- Furthermore, some ensemble methods such as 'XGBOOST' should also take care of larger data dimensions and these methods can themselves be used for feature selection.

There are lot of other possibilitIes that can make the feature selection more intuitive and literally I am new to this and the 'Feature Selection' implementation gave me a hard core challenging of applying it in real time.

9. REFERENCES

9.1 Web References

1. https://www.kaggle.com/mohansacharya/graduate-admissions#Admission_Predict.csv
2. https://en.wikipedia.org/wiki/Residual_sum_of_squares
3. https://en.wikipedia.org/wiki/Total_sum_of_squares
4. http://scholarstrategy.com/importance-of-gpa-and-gre-scores-in-your-ms-applications/
5. https://studyabroad.careers360.com/what-toefl-test
6. https://en.wikipedia.org/wiki/College_and_university_rankings
7. https://studyabroad.shiksha.com/sop-statement-of-purpose-applycontent1701
8. https://en.wikipedia.org/wiki/Letter_of_recommendation
9. https://www.phdstudent.com/Getting-Involved-in-Research/the-importance-of-research-to-grad-school-admission