

GRADUATE ADMISSION PREDICTION

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

SHRUTI SAWARN [RA2011033010042]

GAYATRI MALLADI[RA2011033010047]

RISHI RANJAN[RA2011033010052]

Under the guidance of

Dr. M.FERNI UKRIT

Associate Professor, Department of Computational Intelligence

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled “**GRADUATE ADMISSION PREDICTION**” is the bonafide work of **SHRUTI SAWARN(RA2011033010042),GAYATRI MALLADI (RA2011033010047)** and **RISHI RANJAN(RA2011033010052)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. M.Ferni Ukrit

GUIDE

Associate Professor

Department of Computational Intelligence

SIGNATURE

Dr. R.Annie Uthra

HEAD OF THE DEPARTMENT

Professor & Head

Department of Computational Intelligence

ABSTRACT

The future of the traditional higher education model is an important topic of discussion. These issues are of course highly dependent on the type of academic institution being considered, such as public colleges and universities. Every year, the number of students wanting to pursue higher studies abroad, especially in the US, is more and they find it difficult to search for the best university. Most Indian students face difficulties in enrolling in abroad universities especially US based university and thus Graduate admission is useful in bridging the gap between Indian students and US based university where student can find better university they needed, without any inconvenience. The number of Indian students willing to enroll in the US crossed 1M for the first time in 2015-16 as per an IIE report backed by the State department. The biggest drawback in hiring a CAC is expensive as service can cost thousands of Rupees, thus through our project we are attempting to predict the eligibility of Indian students getting admission in the best university based on their test attributes like GRE, TOEFL, IELTS, SOP, LOR, CGPA and research papers published. According to their scores the possibilities of chance of admit is calculated. This project uses machine learning techniques, specifically the decision tree algorithm(version 0.22.1) 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 based on their scores. The objective of our project is to build a web application to find their corresponding universities based on that we will be creating a data set containing the above mentioned attributes for the accurate prediction so that they can apply to those universities which they are eligible for and which they are interested in.

TABLE OF CONTENTS

ABSTRACT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	v
LIST OF TABLES	vi
ABBREVIATIONS	vii
1 INTRODUCTION	8
2 LITERATURE SURVEY	11
3 SYSTEM ARCHITECTURE AND DESIGN	13
3.1 Architecture diagram of proposed Admission Prediction project	14
3.2 Description of Module and components	15
4 METHODOLOGY	17
4.1 SVM	17
4.2 Random Forest	18
4.3 Logistic Regression	19
4.4 Decision Tree	19
5 CODING AND TESTING	21
6 SREENSHOTS AND RESULTS	38
6.1 Exploratory Data Analysis	38
6.2 Heatmap	39
6.3 Confusion Matrix	40
6.4 Results comparison table	43
6.5 Graph plot for comparison of accuracy of models	44
6.6 Sample input and output for prediction	45
7 CONCLUSION AND FUTURE ENHANCEMENT	46
7.1 Conclusion	
7.2 Future Enhancement	
REFERENCES	48

LIST OF FIGURES

FIGURE NUMBER	FIGURE NAME	PAGE NUMBER
3.1	Data issues in the unsampled dataset	13
3.2	Data pre-processing and feature engineering	13
3.3	Architecture diagram of the proposed model	14
3.4	Architecture of the proposed model	14
4.1	Working of SVM	17
4.2	Working of Random Forest	18
4.3	Working of Logistic Regression	19
4.4	Working of Decision Tree	20
6.1	Regression plot: GRE score vs COA	38
6.2	Box plot: Research vs COA	38
6.3	Regression plot: CGPA vs COA	38
6.4	Box plot: UR vs COA	38
6.5	Regression plot: LOR vs COA	38
6.6	Box plot: SOP vs COA	38
6.7	Heatmap of all the parameters	39
6.8	Confusion matrix of SVM	40
6.9	Confusion matrix of Logistic Regression	41
6.10	Confusion matrix of Random Forest	41
6.11	Comparison of model vs accuracy with all parameters taken into consideration	44
6.12	Comparison of model vs accuracy with 3 parameters taken into consideration	44
6.13	Comparison of other models	45

LIST OF TABLES

TABLE NUMBER	TABLE NAME	PAGE NUMBER
2.1	Literature survey of different papers	12
6.2	Metrics comparison of models	43

ABBREVIATIONS

SVM	Support Vector Machine
UR	University Rating
SOP	Statement of Purpose
LOR	Letter of recommendation
CGPA	Cumulative Grade Point Average
COA	Chance of Admit
EDA	Exploratory Data Analysis

CHAPTER 1

INTRODUCTION

The world business sectors are growing quickly and constantly searching generally advantageous information and experience among individuals. Youthful specialists who need to hang out in their positions are continually searching for Higher degrees that can help them in working on their abilities and information. Thus, the quantity of understudies applying for Graduate examinations has expanded in the last decade. One of the principle concerns is getting conceded to their fantasy University. It's seen that understudies actually decide to get their schooling from universities that are known Universally. What's more, with regards to international alumni, the United States of America is the primary inclination of most of them. With most incredibly famous universities, Wide assortment of courses accessible in each order, exceptionally authorize instruction and educating programs, understudy grants are accessible for international understudies.

Applying to universities abroad is often difficult. Students find it hard to estimate exactly where they stand. It involves many steps and procedures to follow. Researching the universities and programs is itself an arduous and lengthy task. Choosing the right universities to apply to is definitely a hurdle students have to face. Many students apply for the universities in which they have a slim chance of acceptance. University applications alone can cost hundreds of dollars. This can take a toll on the finances of students coming from a poor economic background. Students have to throw away lots of hard-earned money for nothing if they get rejected by these universities.

The USA has been the favourite hub for higher education in the world most importantly for Graduate education. Every year thousands and thousands of students come to the USA for studies. The number of international students in the United States has surpassed one million for the third consecutive year, increasing by 1.5 percent to reach a new high of 1,094,792, according to the survey done in 2018. The admission procedure as long as it can be essentially split into various phrases where the last one being the decision to Admit or Reject. It sometimes becomes stressful and also involves a large sum of money in applying to a number of universities. So, we plan to use machine learning to expedite the process to get to know the chance of getting an admit from the university that an applicant is applying.

As per gauges, there are in excess of 10 million international understudies enlisted in more than 4200. Universities and Colleges including both private and public across the United States. Generally, number of understudies concentrating in America are from Asian nations like India, Pakistan, Sri Lanka, Japan and China. They are picking America as well as UK, Germany, Italy, Australia and Canada. The quantity of individuals seeking after higher investigations in these nations are quickly expanding.

The foundation justification the understudies going to abroad Colleges for Masters is the quantity of open positions present are low and number of individuals for those positions are exceptionally high in their separate nations. This moves numerous understudies in their calling to seek after Postgraduate investigations. It is seen that there is a significant huge number of understudies from Universities in the USA seeking after Masters in the field of Computer Science, the accentuation of this exploration will be on these understudies. Numerous schools in the U.S. follow comparative prerequisites for understudy affirmation. Schools consider various variables, for example, the positioning on fitness appraisal and scholastic record audit. The order over the English language is determined based on their exhibition in the English abilities test, for example, TOEFL and IELTS. The entrance advisory board of universities takes the choice to endorse or reject a particular up-and-comer based on the general profile of the candidate application. The dataset taken in this undertaking is identified with instructive area. In this project, we will be using the admission_predict dataset in CSV format to predict the universities that a student might get into based on several academic performance measurements.

Parameters that help predict the universities:

1. GRE Scores (out of 340)

2. TOEFL Scores (out of 120)

3. University Rating (out of 5) that demonstrates the Bachelor University positioning among different colleges. The score will be out of 5.

4.1 Statement of Purpose which is a record written to show the applicant's life, driven and the inspirations for the picked degree/college. The score will be out of 5 focuses.

4.2. Letter of Recommendation Strength (out of 5) which confirms the applicant proficient experience, fabricates validity, supports certainty and guarantees your ability. The score is out of 5 focuses.

5. Undergraduate GPA (out of 10): The GPA attained by the applicant in respective bachelors degree.

6. Research Experience (either 0 or 1) that can uphold the application, like distributing research papers in gatherings, filling in as examination right hand with college teacher.

7. Chance of Admit (ranging from 0 to 1) Acknowledgments This dataset is inspired by the UCLA Graduate Dataset. The test scores and GPA are in the older format.

The dataset is owned by Mohan S Acharya. Inspiration This dataset was built with the purpose of helping students in shortlisting universities with their profiles. The predicted output gives them a fair idea about their chances for a particular university. One word variable can be anticipated which is possibility of affirmation, that is as per the input given will be going from 0 to 1.

In this project, we are focused on modelling graduate admission to Computer Science in United states. The project provides answers to some of the questions for students and University: • To which university should I go to for computer science in USA? • What are my Top 5 colleges for computer science to apply to get an admit in USA? • Which candidate should be selected from vast range of students? Our goal for the models is to solve the issue faced by students and Universities by developing machine learning models using the previous year Admits/Rejects of the students.

- Assisting Student Model: The model addressed the problems faced by students by developing a machine learning approach which make an applicant make informed decision by evaluation their chances of admission using a probability-based method across 29 Universities.

- Assisting University Model: The model is built on Northeastern university data set to help make better choices of applicants from diverse group of student's profiles.

CHAPTER 2

LITERATURE SURVEY

A substantial number of research programs have been carried out on subjects related to university admissions. Each one has used datasets from various sources and is specific to a certain course or university. Hence, the prediction may not be accurate for every student. Previous studies done in this area evaluated the chance of student admission into respective universities primarily based on a single parameter – the GRE score. The downside is that they only considered the GRE score and left out all the other factors that might contribute to admission prediction. Our base research paper has considered all these parameters, but the machine learning model has low accuracy. Another con is that the dataset used is specific to Indian students and is not generalized. The main drawback of all the previous research carried out is that it only predicts the chance of admission of a student. A better system would predict the universities that a student might get into, rather than the chance of admission. In this way, a student would be able to apply to universities where they have a better chance at getting admission.

Some research papers give a conclusion that the success rate of hybrid systems with the implementations of more than one algorithm concurrently is higher with low error rates, as the work done by each algorithm in any hybrid system relies on each of them only for the specific task and hence later collaboration ensures higher accuracy.

2.1 LIMITATIONS:

- Major drawback of all the previous research carried out is that it only predicts the chance of admission of a student. A better system would predict the universities that a student might get into, rather than the chance of admission. [3]
- The system is built on a limited data set, this could affect the accuracy of the predictions as a whole. The system cannot guarantee that our predictions will be a 100% guarantee of admissions because a lot other factors such as the Personal Interview also plays a major role in the admissions procedure.[4]
- Admissions also depend on the individual university's policy regarding the intake of foreign students and is not modelled by our system.[1]

Table 2.1 Literature survey of different papers

PAPER NAME	YEAR	JOURNAL NAME	INFERENCE
A Comparative Study on University Admission Predictions Using Machine Learning Techniques	March-April-2021	International Journal of Scientific Research in Computer Science, Engineering and Information Technology	If we go through the research done till date, the success rate of hybrid systems with the implementations of more than one algorithm concurrently is higher with low error rates, as the work done by each algorithm in any hybrid system relies on each of them only for the specific task and hence later collaboration ensures higher accuracy.
Post Graduation Admission Prediction System	June-2022	International Research Journal of Engineering and Technology	Flask language is used to build a web interface and pickle library is used to integrate both model and web page.
University Admission Prediction using Machine Learning	October-2021	Global Journal of Research and Review	Comparative study of algorithms proved logistic regression to be more accurate.
Prediction Student University Admission Using Logistic Regression	June-2020	European Journal of Computer Science and Information Technology	Comparative study of algorithms proved logistic regression to be more accurate.

CHAPTER 3

3.1 SYSTEM ARCHITECTURE AND DESIGN

3.1.1 DATA ISSUES:

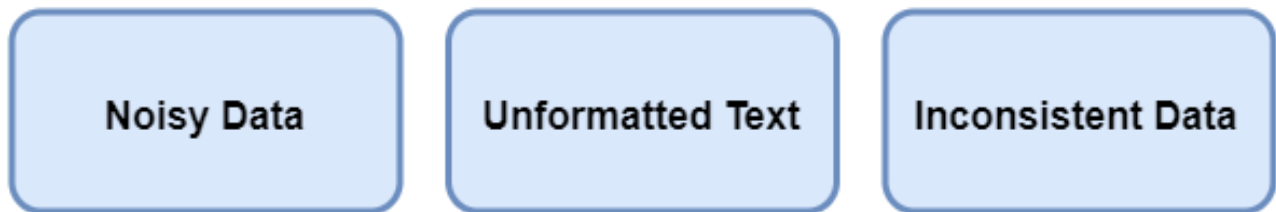


Fig 3.1. Data issues in the unsampled dataset

3.1.2 DATA PREPROCESSING AND FEATURE ENGINEERING:

- Data Pre-processing and Feature Engineering were carried out to prepare cleaned dataset, by following the below approach:

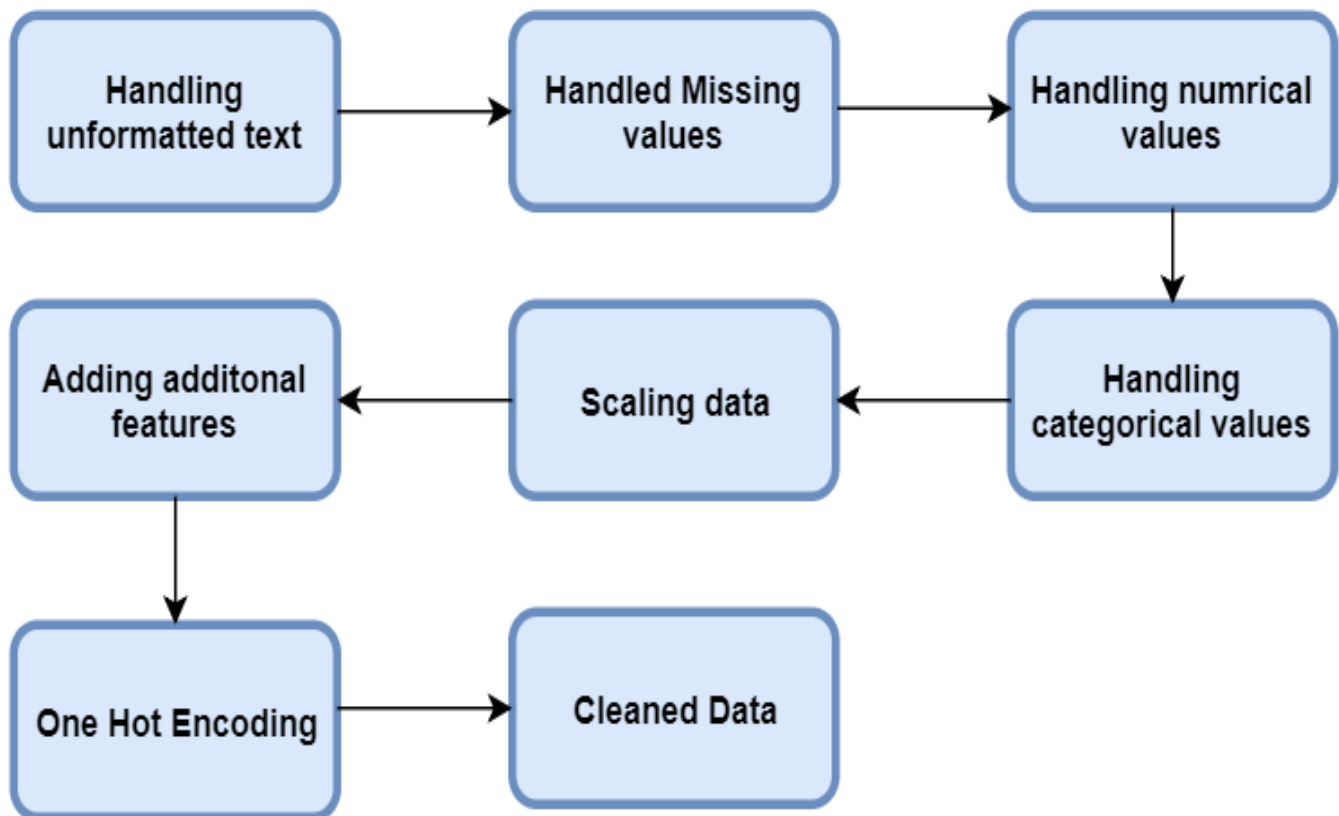


Fig 3.2 Data pre-processing and feature engineering

3.1.3 PROPOSED SYSTEM ARCHITECTURE:

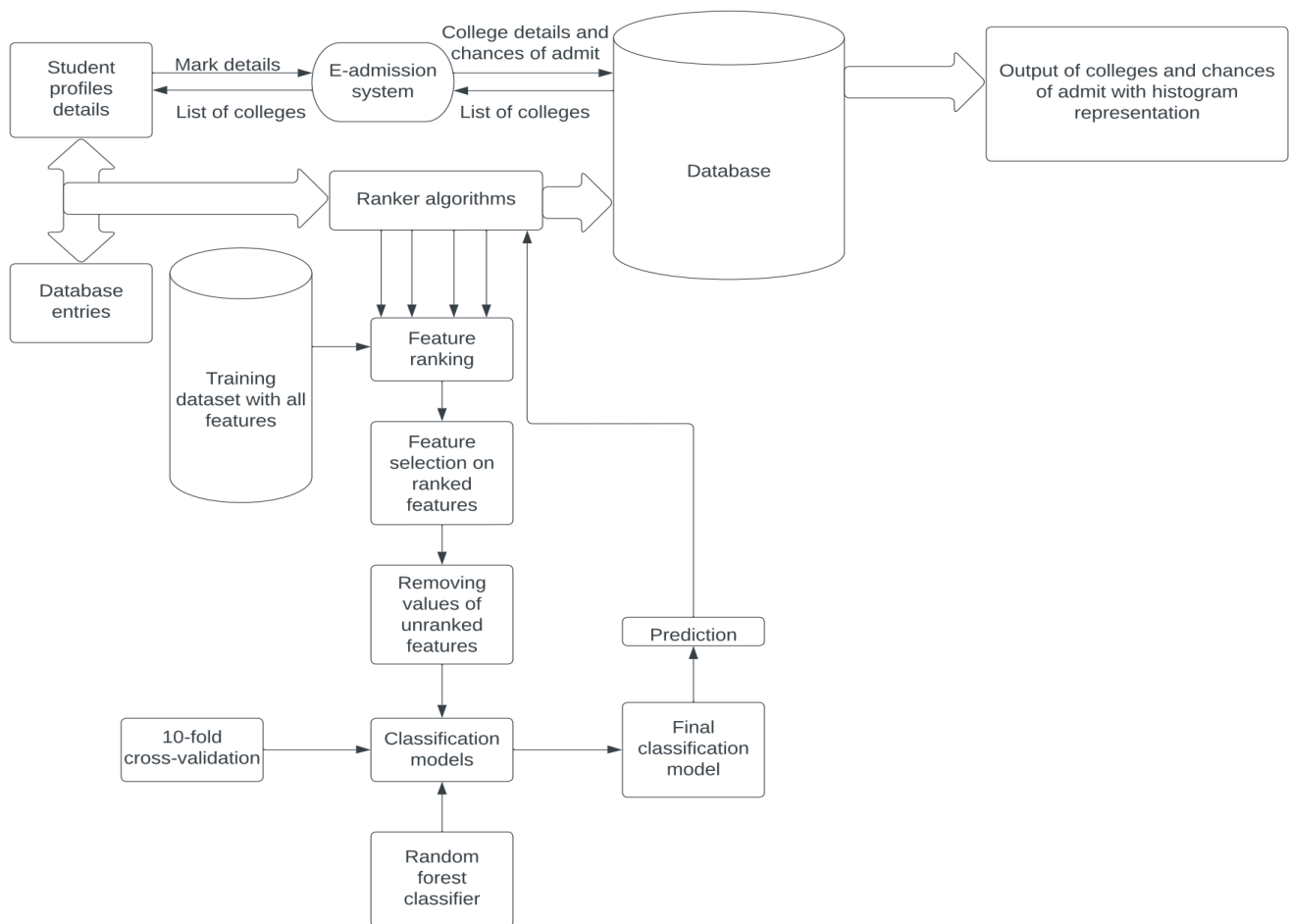


Fig 3.3 Architecture diagram of proposed model

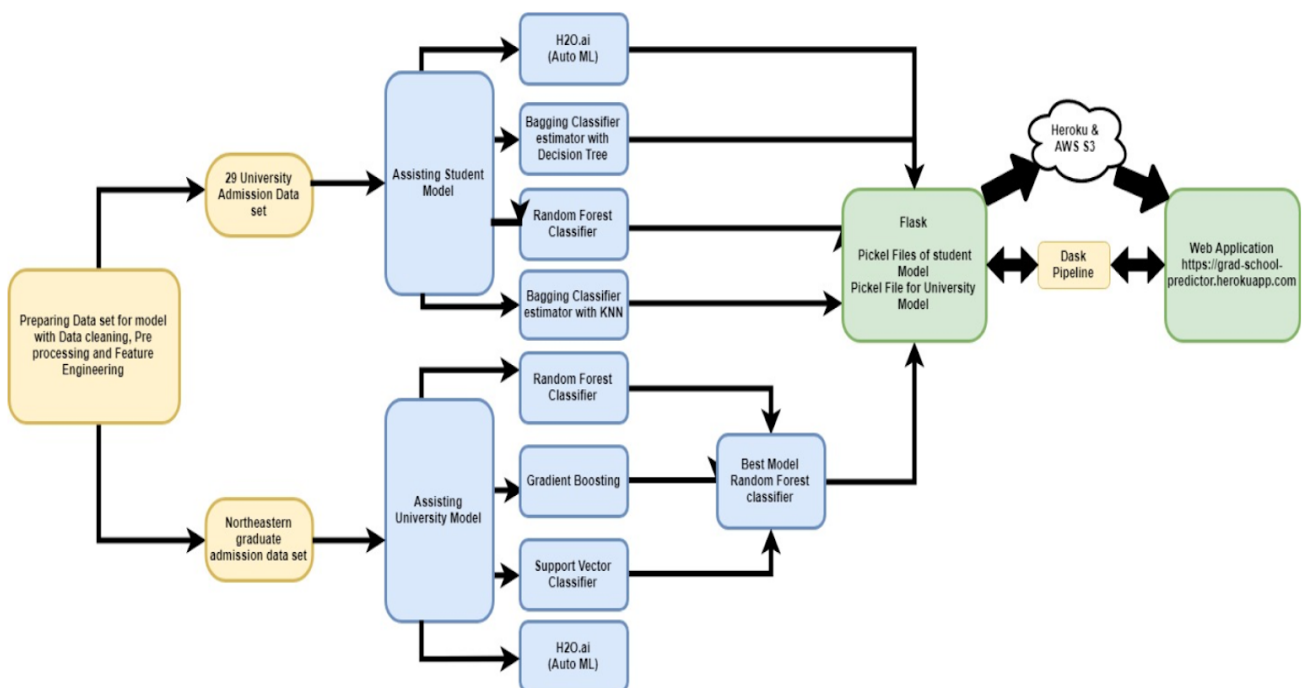


Fig 3.4 Architecture diagram of proposed model

3.2 MODULE DESCRIPTION AND COMPONENTS

3.2.1 DATA LOADING AND ANALYSIS OF VARIABLE

- Load the libraries
- Load the dataset in csv file format and check the Column names to check the Space in the names in the data.
- Check shape of data and whether there exists missing values.

3.2.2 EXPLORATORY DATA ANALYSIS MODULE

- Basic statistics of the data.
- Understand skewness of data.
- Understand how spread out the data is through standard deviation.
- Analyse how rating affects the prediction.
- Kurtosis
- Module index
- Outliers treatment

3.2.3 DATA VISUALIZATION

- Plot the data in histogram format as follows:
- Line plot
- Scatter plot
- Numeric variables

3.2.4 DATA PREPARATION

- Data will be splitted using train_test_split module of scikitlearn library where splitting ratio is chosen as 20% for test data(10-cross-validation)
- Data normalization

3.2.5 CLASSIFICATION MODEL MODULE

- Fitting random forest model.
- Printing accuracy score and confusion matrix.
- Feature importance.

3.2.6 MODEL EVALUATION AND HYPERPARAMETER TUNING MODULE

- Model accuracy in predicting chances of admit will be the output.
- RME and MAE evaluation

CHAPTER 4

METHODOLOGY

4.1 SUPPORT VECTOR MACHINE

Support Vector Machines (SVMs) are a family of supervised learning methods that can be used to solve regression and classification tasks. It is most often used for sorting, but it can also be very useful for regression. A SVM classifier constructs a maximum-margin hyperplane in a transformed input space and divides the sample classes thereby minimizing the distance to the closest cleanly separated instances. This hyper-plane is nothing other than a line in two dimensions. Each data object in the dataset is plotted in an N-dimensional space in SVM, where N represents the number of characteristics in the data. Find the best hyperplane to isolate the results. As a result, SVM can only do classification task (i.e., choose between two classes). Even so, there are several methods for dealing with multi-class problems.

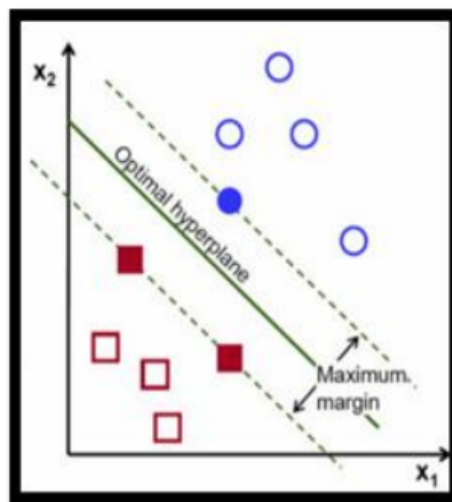


Fig 4.1 Working of SVM

4.2. RANDOM FOREST

A Random Forest is an ensemble learning method that can execute both regression and classification tasks by combining several decision trees and a technique known as Bootstrap and Aggregation, also known as bagging. The basic principle is to use several decision trees to determine the final production rather than relying on individual decision trees. Random Forest's foundation learning structures are multiple decision trees.

The randomly select rows and features from the dataset to create sample datasets for each model. This section is known as Bootstrap. Each individual tree in the random forest produces a class prediction, and

the class with the most votes becomes the prediction of a model. The main point is the low correlation between models.

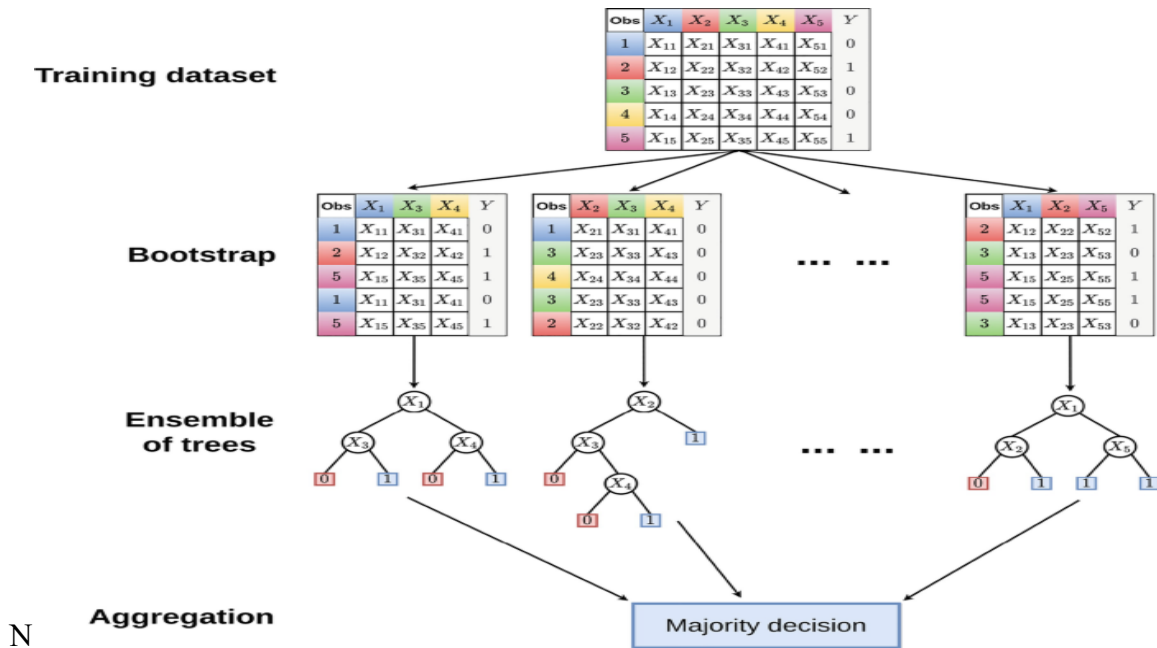


Fig 4.2 Working of Random Forest

Each individual tree in the random forest produces a class prediction, and the class with the most votes becomes the prediction of a model. The main point is the low correlation between models. Uncorrelated models can generate ensemble predictions that are more reliable than individual predictions. The explanation for this great result is that the trees shield each other from their individual reaching a necroed mimic the electrical stimulation of a neuron and, as a result, information flow within the network or brain. The input data to a processing unit, in, are amplified by a connection weight, $W(n,m)$, which imitates neural pathway increasing in the brain. Learning is emulated in ANNs by adjusting the link strength or weights.

4.3 LOGISTIC REGRESSION

Logistic regression is similar to linear regression but instead of a linear response given by the linear regression, the logistic regression has a binomial response variable. In the logistic regression we can have more than 2 continuous explanatory variables

and it's easier to handle those variables

simultaneously. In the case of more than one explanatory variable, logistic regression is used to calculate the odds ratio. With the fact that the outcome variable is binomial, the technique is very similar to multiple

linear regression. The effect of each vector on the odds ratio of the observe occurrence of interest is the result. The biggest benefit is that when considering the relationship of all factors together, misleading consequences area voided. A machine learning model created using logistic regression models the chance of all the outcomes based on the characteristics or features provided to the model through data feeding.

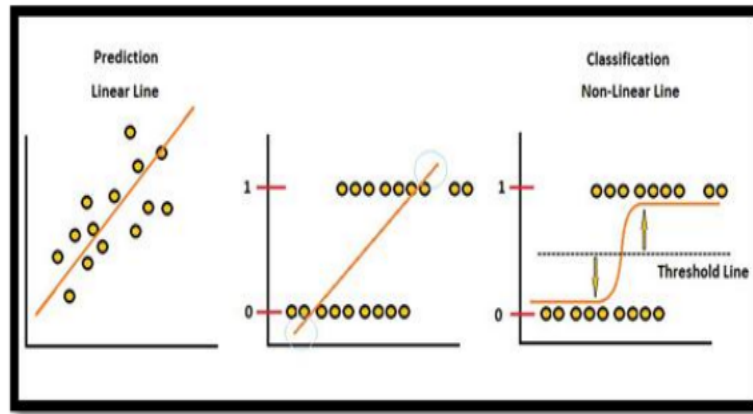


Fig 4.3 Working of Logistic Regression

4.4 DECISION TREE

The decision tree is the most effective and widely used classification and prediction method. A Decision tree is a tree structure that looks like a flow chart, with each internal node representing a test on an attribute, each branch representing the test's result, and each leaf node (terminal node) holding a class name. Decision trees are fundamentally a basic form of classifier, which is one of their advantages[18]. In its most basic form, we ask yes-or-no questions, and each internal node has a 'yes' and a 'no' child. Following the model from the topmost node, the root, to a node without children, a leaf, an item is sorted into a class depending on the responses that refer to the item in review.

Since they combine basic questions about the data in an intuitive manner, decision trees are often more easy to interpret than other classifiers such as neural networks and support vector machines.

Methods for removing decision rules from decision trees have also shown to be effective. Unfortunately, minor differences in input data will also result in significant changes in the tree's structure. Decision trees are adaptable enough to deal with objects that have a combination of real-valued and categorical features, as well as items that lack any features[18]. Thus decision trees can produce rules that are easy to understand. Decision trees are used to build models without involving a lot of calculation and can accommodate factors that are both constant and categorical clearly showing which areas are most relevant for forecasting or classification.

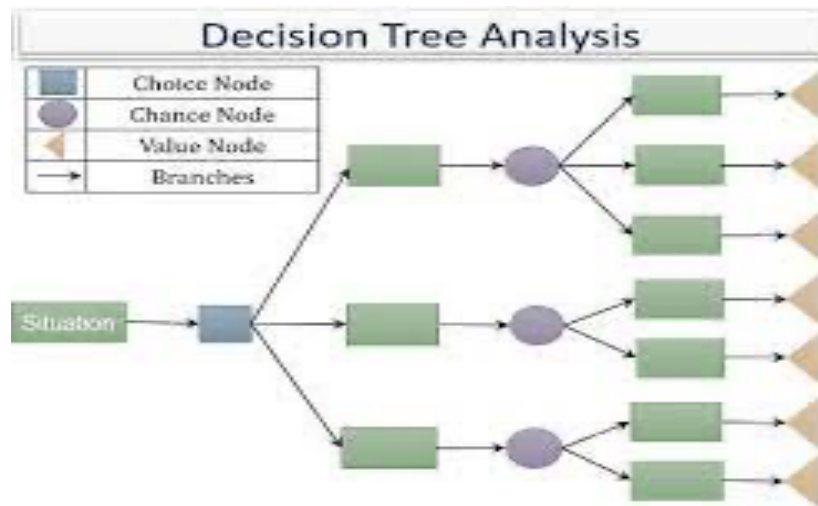


Fig 4.4 Working of Decision Tree

CHAPTER 5

CODING AND TESTING

5.1 LOAD THE DATASET

```
import numpy as np

import pandas as pd

from google.colab import files

data_to_load=files.upload()

import io

df = pd.read_csv(io.BytesIO(data_to_load['Admission_Predict_Ver1.1.csv']))
```

This dataset is taken from Kaggle. The following information is directly copied from this [Kaggle](#) page. This dataset is created for prediction of Graduate Admissions from an Indian perspective. This dataset is inspired by the UCLA Graduate Dataset. The test scores and GPA are in the older format. The dataset is owned by Mohan S Acharya.

5.2 DATA WRANGLING

```
# Importing the required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
%matplotlib inline

sns.set()
warnings.simplefilter('ignore')

# Read the Data and check the Column Names to check the Space in the Names
df=pd.read_csv("Admission_Predict_Ver1.1.csv")
col_names=df.columns.tolist()
print("Column names:")
print(col_names)
```

```
print("\nSample Data:")
print(df.head())
df=df.copy()
df.tail(20) df.drop('Serial No.', axis=1, inplace=True)
df.head()
```

```
df.isnull().sum()
df.shape
df.describe()
```

```
df.columns = df.columns.str.strip()
df.columns
```

5.3 EXPLORATORY DATA ANALYSIS

- Perform EDA to gain insights from the data
- Response Variable: Chance of Admit
- Individual columns' correlation with response variable

5.3.1 Column: GRE Score(out of 340)

Checking the correlation coefficient of GRE Score with Chance of Admit

- `df[['GRE Score', 'Chance of Admit']].corr()`

```
plt.figure(figsize=(10, 6))
sns.regplot(x='GRE Score', y='Chance of Admit', data=df)
from scipy import stats
p_coeff, p_value = stats.pearsonr(df['GRE Score'], df['Chance of Admit'])
print('Pearson Coefficient:', p_coeff)
print('P Value:          ', p_value)
```

5.3.2 Column: TOEFL Score(out of 120)

Checking the correlation-coefficient

```
df[['TOEFL Score', 'Chance of Admit']].corr()
```

```
plt.figure(figsize=(10, 6))
sns.regplot(data=df, x='TOEFL Score', y='Chance of Admit')
p_coeff, p_value = stats.pearsonr(df['TOEFL Score'], df['Chance of Admit'])
print('Pearson Coefficient:', p_coeff)
print('Pearson Value:    ', p_value)
```

5.3.3 Column: University Ranking

```
plt.figure(figsize=(10, 6))
sns.regplot(x=df['University Rating'], y=df['Chance of Admit'])
plt.figure(figsize=(10, 6))
```

```
sns.boxplot(data=df, x='University Rating', y='Chance of Admit')
df[['University Rating', 'Chance of Admit']].corr()
coef, pvalue = stats.pearsonr(df['University Rating'], df['Chance of Admit'])
coef, pvalue
```

```
df_rating_grp = df[['University Rating', 'Chance of Admit']].groupby(['University Rating'])
f, pvalue = stats.f_oneway(df_rating_grp.get_group(1)['Chance of Admit'],
                           df_rating_grp.get_group(2)['Chance of Admit'],
                           df_rating_grp.get_group(3)['Chance of Admit'],
                           df_rating_grp.get_group(4)['Chance of Admit'],
                           df_rating_grp.get_group(5)['Chance of Admit'])
```

```
print('f oneway:', f, '\nP Value:', pvalue)
```

5.3.4 Column: SOP or Statement of Purpose (out of 5)

```
plt.figure(figsize=(10, 6))
sns.regplot(data=df, x='SOP', y='Chance of Admit')
df[['SOP', 'Chance of Admit']].corr()
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='SOP', y='Chance of Admit')
p_coeff, pvalue = stats.pearsonr(df.SOP, df['Chance of Admit'])
```

```
print('Pearson Coefficient: ', p_coeff)
```

```
print('P Value: ', pvalue)
```

```
df_sop_grp = df[['SOP', 'Chance of Admit']].groupby(['SOP'])
f, pvalue = stats.f_oneway(df_sop_grp.get_group(1.0)['Chance of Admit'],
                           df_sop_grp.get_group(1.5)['Chance of Admit'],
                           df_sop_grp.get_group(2.0)['Chance of Admit'],
                           df_sop_grp.get_group(2.5)['Chance of Admit'],
                           df_sop_grp.get_group(3.0)['Chance of Admit'],
                           df_sop_grp.get_group(3.5)['Chance of Admit'],
                           df_sop_grp.get_group(4.0)['Chance of Admit'],
                           df_sop_grp.get_group(4.5)['Chance of Admit'],
                           df_sop_grp.get_group(5.0)['Chance of Admit'])
f, pvalue
```

5.3.5 Column: LOR(Letter of Recommendation)

```
plt.figure(figsize=(10, 6))
```

```
sns.regplot(x=df.LOR, y=df['Chance of Admit'])
```

```
df[['LOR', 'Chance of Admit']].corr()
```

```
p_coeff, pvalue = stats.pearsonr(df.LOR, df['Chance of Admit'])
```

```
p_coeff, pvalue
```

```
df_lor_grp = df[['LOR', 'Chance of Admit']].groupby('LOR')
```

```
f, pvalue = stats.f_oneway(df_lor_grp.get_group(1.0)['Chance of Admit'],
                           df_lor_grp.get_group(1.5)['Chance of Admit'],
                           df_lor_grp.get_group(2.0)['Chance of Admit'],
                           df_lor_grp.get_group(2.5)['Chance of Admit'],
                           df_lor_grp.get_group(3.0)['Chance of Admit'],
                           df_lor_grp.get_group(3.5)['Chance of Admit'],
                           df_lor_grp.get_group(4.0)['Chance of Admit'],
                           df_lor_grp.get_group(4.5)['Chance of Admit'],
                           df_lor_grp.get_group(5.0)['Chance of Admit'])
```


f, pvalue

5.3.6 Column: CGPA

```
plt.figure(figsize=(10, 6))
sns.regplot(x=df.LOR, y=df['Chance of Admit'])
p_coeff, pvalue = stats.pearsonr(df.CGPA, df['Chance of Admit'])
```

p_coeff, pvalue

5.3.7 Column: Research

```
plt.figure(figsize=(10, 6))
sns.regplot(x=df.Research, y=df['Chance of Admit'])
df[['Research', 'Chance of Admit']].corr()
plt.figure(figsize=(10, 6))
sns.boxplot(x=df.Research, y=df['Chance of Admit'])
```

```
f, pvalue = stats.f_oneway(df_res_grp.get_group(1)['Chance of Admit'],
                           df_res_grp.get_group(0)['Chance of Admit'])
f, pvalue
```

5.4 IMPLEMENTING ALGORITHMS

5.4.1 Support Vector Machine:

```
# import statements
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

```

from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    classifier.fit(X, y)
    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)

    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha=0.8,
                    c=colors[idx],
                    marker=markers[idx],
                    label=cl,
                    edgecolor='black')

    # highlight test examples
    if test_idx:
        # plot all examples

```

```
X_test, y_test = X[test_idx, :], y[test_idx]
```

```
plt.scatter(X_test[:, 0],  
            X_test[:, 1],  
  
            c="",  
            edgecolor='black',  
            alpha=1.0,  
            linewidth=1,  
            marker='o',  
            s=100,  
            label='test set')
```

```
#read the data into a pandas dataframe
```

```
df = pd.read_csv('Admission_Predict_Ver1.1.csv')
```

```
df.dropna(inplace=True)
```

```
df.columns = ["No", "GRE", "TOEFL", "UR", "SOP", "LOR", "CGPA", "RES", "CoA"]
```

```
#print(df.describe())
```

```
X = df[['GRE', 'TOEFL', 'CGPA']]
```

```
y = df['CoA']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=1)
```

```
# Standardizing the features:
```

```
sc = StandardScaler()
```

```
sc.fit(X_train)
```

```
X_train_std = sc.transform(X_train)
```

```
X_test_std = sc.transform(X_test)
```

```
# chose 0.82 because it is the 3rd quartile for chance of admit
```

```
ty_train=[1 if CoA > 0.82 else 0 for CoA in y_train] # learned from internet
```

```
ty_train=np.array(ty_train)
```

```

ty_test=[1 if CoA > 0.82 else 0 for CoA in y_test] #learned from internet
ty_test=np.array(ty_test)

svm = SVC(kernel='rbf', C=10.0, random_state=1)
svm.fit(X_train_std, ty_train)
svc_pred = svm.predict(X_test_std)

X_combined_std = np.vstack((X_train_std[:, 1:], X_test_std[:, 1:]))
y_combined = np.hstack((ty_train, ty_test))
plot_decision_regions(X=X_combined_std, y=y_combined, classifier=SVC(kernel='rbf', C=10.0,
random_state=1))
plt.savefig("svm.png")
plt.show()

print("SVM Accuracy: %.3f" % accuracy_score(ty_test, svc_pred))
print("SVM F1-Score: %.3f" % f1_score(ty_test, svc_pred))
print("SVM Precision: %.3f" % precision_score(ty_test, svc_pred))
print("SVM Recall: %.3f" % recall_score(ty_test, svc_pred))

```

5.4.2 Logistic regression:

```

# import statements
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

```

```
def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):
```

```
    # setup marker generator and color map
```

```
    markers = ('s', 'x', 'o', '^', 'v')
```

```
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
```

```
    cmap = ListedColormap(colors[:len(np.unique(y))])
```

```
    classifier.fit(X, y)
```

```
    # plot the decision surface
```

```
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
```

```
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
```

```
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),  
                           np.arange(x2_min, x2_max, resolution))
```

```
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
```

```
    Z = Z.reshape(xx1.shape)
```

```
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
```

```
    plt.xlim(xx1.min(), xx1.max())
```

```
    plt.ylim(xx2.min(), xx2.max())
```

```
    for idx, cl in enumerate(np.unique(y)):
```

```
        plt.scatter(x=X[y == cl, 0],  
                    y=X[y == cl, 1],  
                    alpha=0.8,  
                    c=colors[idx],  
                    marker=markers[idx],  
                    label=cl,  
                    edgecolor='black')
```

```
    # highlight test examples
```

```
    if test_idx:
```

```
        # plot all examples
```

```
        X_test, y_test = X[test_idx, :], y[test_idx]
```

```
        plt.scatter(X_test[:, 0],
```

```
                    X_test[:, 1],
```

```
                    c="",
```

```
edgecolor='black',  
alpha=1.0,  
linewidth=1,  
marker='o',  
s=100,  
label='test set')
```

```
#read the data into a pandas dataframe
```

```
df = pd.read_csv('Admission_Predict_Ver1.1.csv')  
df.dropna(inplace=True)
```

```
df.columns = ["No", "GRE", "TOEFL", "UR", "SOP", "LOR", "CGPA", "RES", "CoA"]
```

```
#print(df.describe())  
X = df[['GRE', 'TOEFL', 'CGPA']]  
y = df['CoA']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=1)
```

```
# Standardizing the features:
```

```
sc = StandardScaler()  
sc.fit(X_train)  
X_train_std = sc.transform(X_train)  
X_test_std = sc.transform(X_test)  
y_test = np.array(y_test)
```

```
log_reg = LogisticRegression(C=100, random_state=1, solver='lbfgs', multi_class='ovr')  
log_reg.fit(X_train_std, y_train)  
log_pred = log_reg.predict(X_test_std)
```

```
print("Logistic Regression Accuracy: %.3f" % accuracy_score(y_test, log_pred))  
print("Logistic Regression F1-Score: %.3f" % f1_score(y_test, log_pred))  
print("Logistic Regression Precision: %.3f" % precision_score(y_test, log_pred))  
print("Logistic Regression Recall: %.3f" % recall_score(y_test, log_pred))  
# Logistic Regression Accuracy: 0.958
```

```
# Logistic Regression F1-Score: 0.941
```

```
# Logistic Regression Precision: 0.960
```

```
# Logistic Regression Recall: 0.923
```

```
X_combined_std = np.vstack((X_train_std[:, 1:], X_test_std[:, 1:]))
```

```
y_combined = np.hstack((ty_train, ty_test))
```

```
plot_decision_regions(X=X_combined_std, y=y_combined, classifier=LogisticRegression(C=100,  
random_state=1, solver='lbfgs', multi_class='ovr'))
```

```
plt.savefig("log_reg.png")
```

```
plt.show()
```

```
ransform(X_train)
```

```
X_test_std = sc.transform(X_test)
```

```
# chose 0.82 because it is the 3rd quartile for chance of admit
```

```
ty_train=[1 if CoA > 0.82 else 0 for CoA in y_train] # learned from internet
```

```
ty_train=np.array(ty_train)
```

5.4.3 Random Forest:

```
# import statements
```

```
from sklearn.model_selection import train_test_split
```

```
import pandas as pd
```

```
import numpy as np
```

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

```
from matplotlib.colors import ListedColormap
```

```
import tensorflow as tf
```

```
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score
```

```
from sklearn.metrics import f1_score
```

```
from sklearn.metrics import precision_score
```

```
from sklearn.metrics import recall_score
```

```
def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):
```

```

# setup marker generator and color map
markers = ('s', 'x', 'o', '^', 'v')
colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
cmap = ListedColormap(colors[:len(np.unique(y))])
classifier.fit(X, y)
# plot the decision surface
x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                        np.arange(x2_min, x2_max, resolution))
Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
Z = Z.reshape(xx1.shape)

```

```

plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
plt.xlim(xx1.min(), xx1.max())
plt.ylim(xx2.min(), xx2.max())

```

```

for idx, cl in enumerate(np.unique(y)):

```

```

    plt.scatter(x=X[y == cl, 0],
               y=X[y == cl, 1],
               alpha=0.8,
               c=colors[idx],
               marker=markers[idx],
               label=cl,
               edgecolor='black')

```

```

# highlight test examples

```

```

if test_idx:

```

```

    # plot all examples

```

```

    X_test, y_test = X[test_idx, :], y[test_idx]

```

```

    plt.scatter(X_test[:, 0],
               X_test[:, 1],

```



```
c=",  
  
edgecolor='black',  
alpha=1.0,  
linewidth=1,  
marker='o',  
s=100,  
label='test set')
```

```
#read the data into a pandas dataframe
```

```
df = pd.read_csv('Admission_Predict_Ver1.1.csv')  
df.dropna(inplace=True)
```

```
#read the data into a pandas dataframe
```

```
df = pd.read_csv('Admission_Predict.csv')  
df.dropna(inplace=True)
```

```
df.columns = ["No", "GRE", "TOEFL", "UR", "SOP", "LOR", "CGPA", "RES", "CoA"]
```

```
#print(df.describe())
```

```
X = df[['GRE', 'TOEFL', 'CGPA']]
```

```
y = df['CoA']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=1)
```

```
# chose 0.82 because it is the 3rd quartile for chance of admit
```

```
ty_train=[1 if CoA > 0.82 else 0 for CoA in y_train] # learned from internet
```

```
ty_train=np.array(ty_train)
```

```
ty_test=[1 if CoA > 0.82 else 0 for CoA in y_test] #learned from internet
```

```
ty_test=np.array(ty_test)
```

```
forest = RandomForestClassifier(criterion='entropy', n_estimators=25, random_state=1, n_jobs=2)
forest.fit(X_train, ty_train)
rf_pred = forest.predict(X_test)
```

```
X_combined_std = np.vstack((X_train[['CGPA', 'TOEFL']], X_test[['CGPA', 'TOEFL']]))
y_combined = np.hstack((ty_train, ty_test))
```

```
plot_decision_regions(X=X_combined_std, y=y_combined,
                      classifier=RandomForestClassifier(criterion='entropy', n_estimators=25, random_state=1, n_jobs=2))
plt.savefig("random_forest.png")
plt.show()
```

```
print("Random Forest Accuracy: %.3f" % accuracy_score(ty_test, rf_pred))
print("Random Forest F1-Score: %.3f" % f1_score(ty_test, rf_pred))
```

```
print("Random Forest Precision: %.3f" % precision_score(ty_test, rf_pred))
print("Random Forest Recall: %.3f" % recall_score(ty_test, rf_pred))
```

5.4.4 Decision Tree:

```
# import statements
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from pydotplus import graph_from_dot_data
```

```

from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    classifier.fit(X, y)

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)

    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha=0.8,
                    c=colors[idx],
                    marker=markers[idx],
                    label=cl,
                    edgecolor='black')

```

```

# highlight test examples
if test_idx:
    # plot all examples
    X_test, y_test = X[test_idx, :], y[test_idx]

plt.scatter(X_test[:, 0],
            X_test[:, 1],
            c="",
            edgecolor='black',
            alpha=1.0,
            linewidth=1,
            marker='o',
            s=100,
            label='test set')

#read the data into a pandas dataframe
df = pd.read_csv('Admission_Predict_Ver1.1.csv')
df.dropna(inplace=True)
df.columns = ["No", "GRE", "TOEFL", "UR", "SOP", "LOR", "CGPA", "RES", "CoA"]

#print(df.describe())
X = df[['GRE', 'TOEFL', 'CGPA']]
y = df['CoA']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=1)

# chose 0.82 because it is the 3rd quartile for chance of admit
ty_train=[1 if CoA > 0.82 else 0 for CoA in y_train] # learned from internet
ty_train=np.array(ty_train)

ty_test=[1 if CoA > 0.82 else 0 for CoA in y_test] #learned from internet
ty_test=np.array(ty_test)

```

```
tree_model = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=1)
tree_model.fit(X_train, ty_train)
dt_pred = tree_model.predict(X_test)
```

```
dot_data = export_graphviz(tree_model,
                            filled=True,
                            rounded=True,
                            class_names=['Rejected', 'Admitted'],
                            feature_names=['GRE', 'TOEFL', 'CGPA'],
                            out_file=None)
graph = graph_from_dot_data(dot_data)

graph.write_png("tree.png")
```

```
X_combined_std = np.vstack((X_train[['CGPA', 'TOEFL']], X_test[['CGPA', 'TOEFL']]))
y_combined = np.hstack((ty_train, ty_test))
plot_decision_regions(X=X_combined_std, y=y_combined,
                      classifier=DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=1))
plt.savefig("decision_tree.png")
plt.show()
```

```
print("Decision Tree Accuracy: %.3f" % accuracy_score(ty_test, dt_pred))
print("Decision Tree F1-Score: %.3f" % f1_score(ty_test, dt_pred))
print("Decision Tree Precision: %.3f" % precision_score(ty_test, dt_pred))
print("Decision Tree Recall: %.3f" % recall_score(ty_test, dt_pred))
```

CHAPTER 6

SCREENSHOTS AND RESULTS

6.1 EXPLORATORY DATA ANALYSIS RESULTS (EDA):

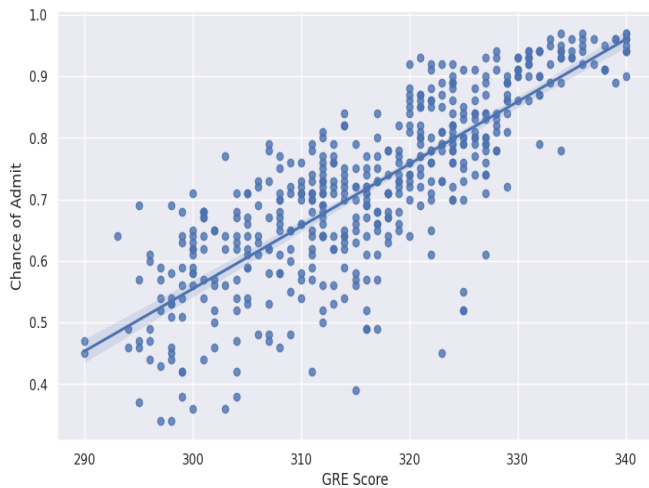


Fig 6.1 Regression plot: GRE score vs COA

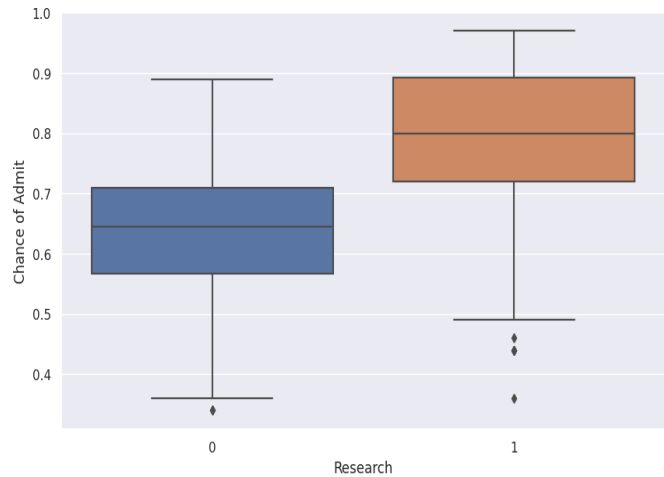


Fig 6.2 Box plot: Research vs COA

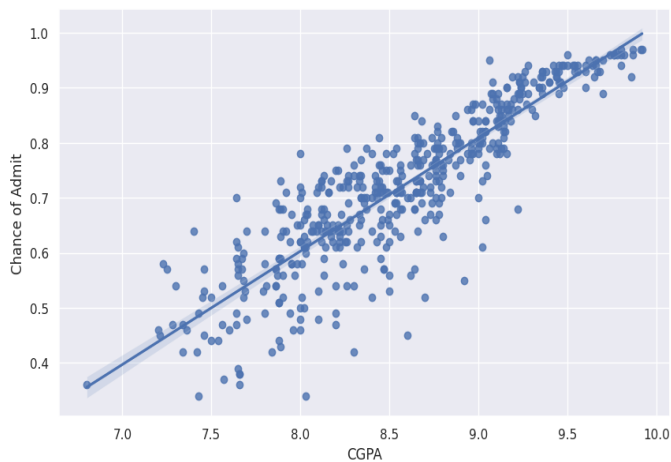


Fig 6.3 Regression plot: CGPA vs COA

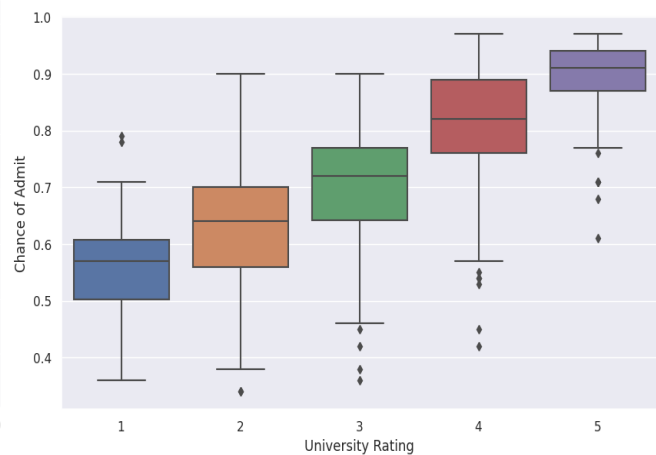


Fig 6.4 Box plot: UR vs COA

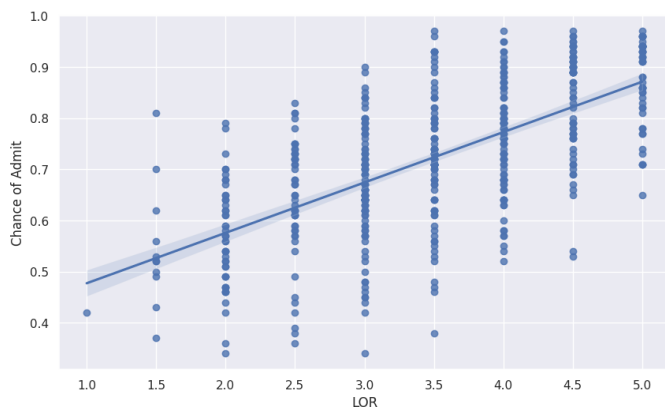


Fig 6.5 Regression plot: LOR vs COA

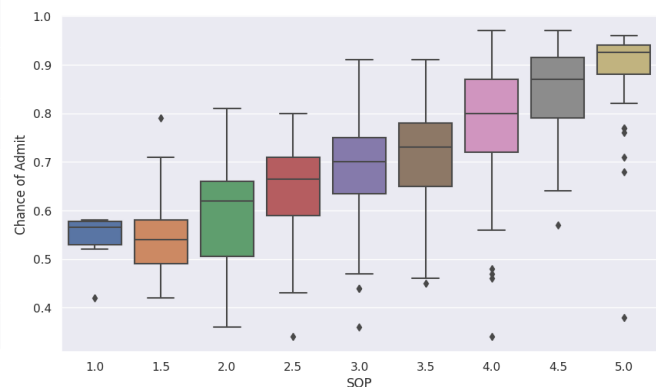


Fig 6.6 Box plot: SOP vs COA

6.2 HEATMAP:

A correlation heatmap is a graphical tool that displays the correlation between multiple variables as a color-coded matrix. Well, by looking at the heatmap, we can infer that there exists great multicollinearity in the data.

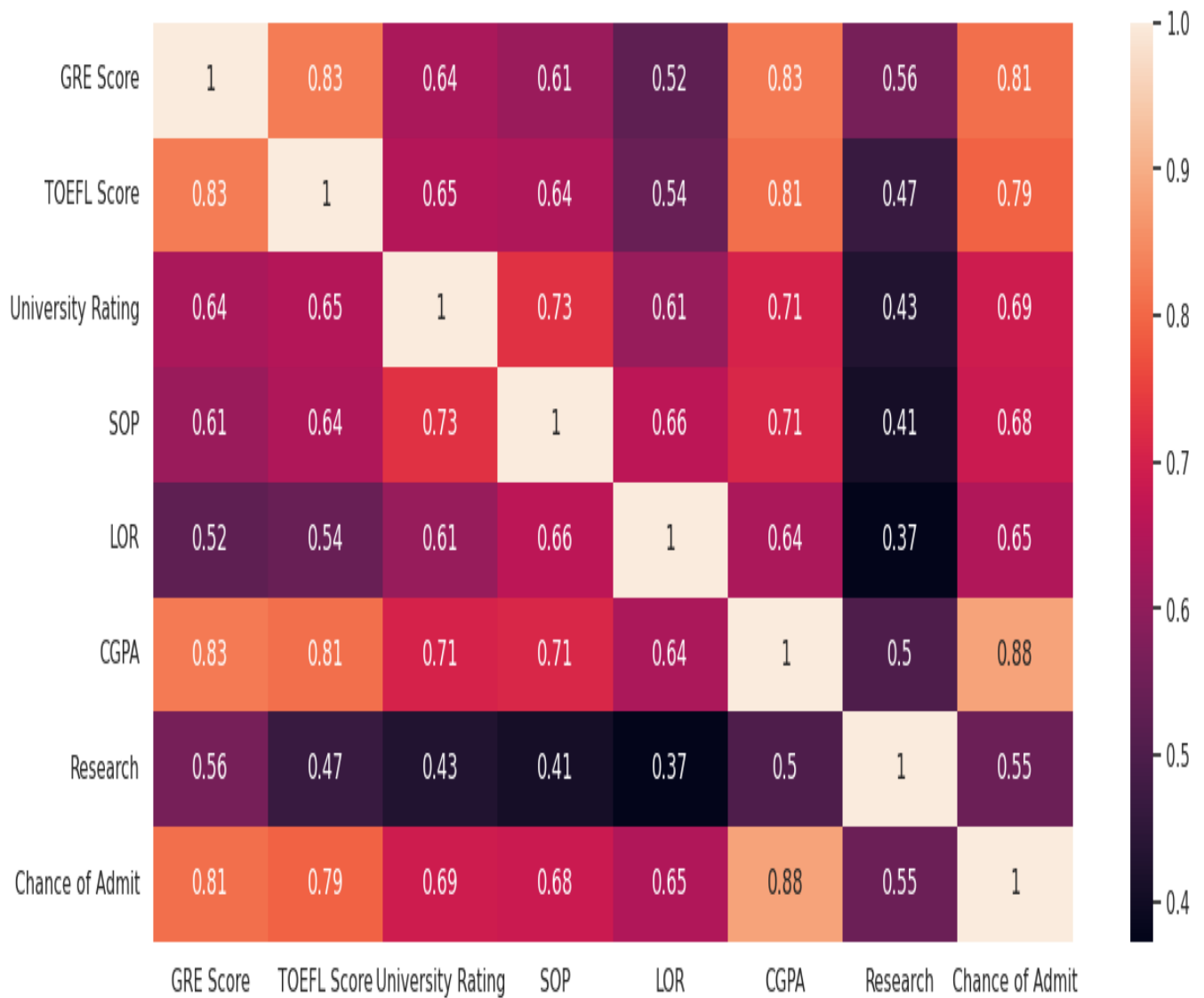


Fig 6.7 Heatmap of all the parameters

6.3 CONFUSION MATRIX:

Confusion Matrix helps us to display the performance of a model or how a model has made its prediction in Machine Learning. Confusion Matrix helps us to visualize the point where our model gets confused in discriminating two classes. It can be understood well through a 2×2 matrix where the row represents the actual truth labels, and the column represents the predicted labels.

This matrix consists of 4 main elements that show different metrics to count a number of correct and incorrect predictions. Each element has two words either as follows:

- True or False
- Positive or Negative

If the predicted and truth labels match, then the prediction is said to be correct, but when the predicted and truth labels are mismatched, then the prediction is said to be incorrect. Further, positive and negative represents the predicted labels in the matrix.

6.3.1 SVM

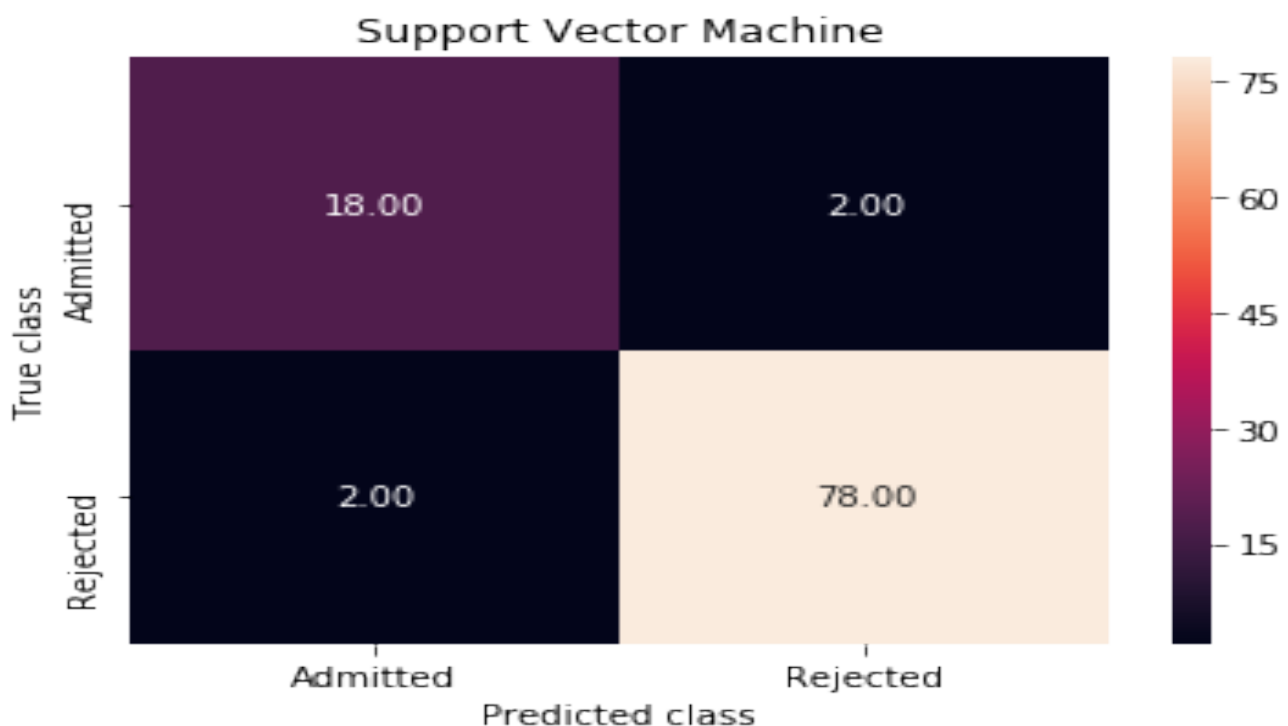


Fig 6.8 Confusion matrix of SVM

6.3.2 LOGISTIC REGRESSION

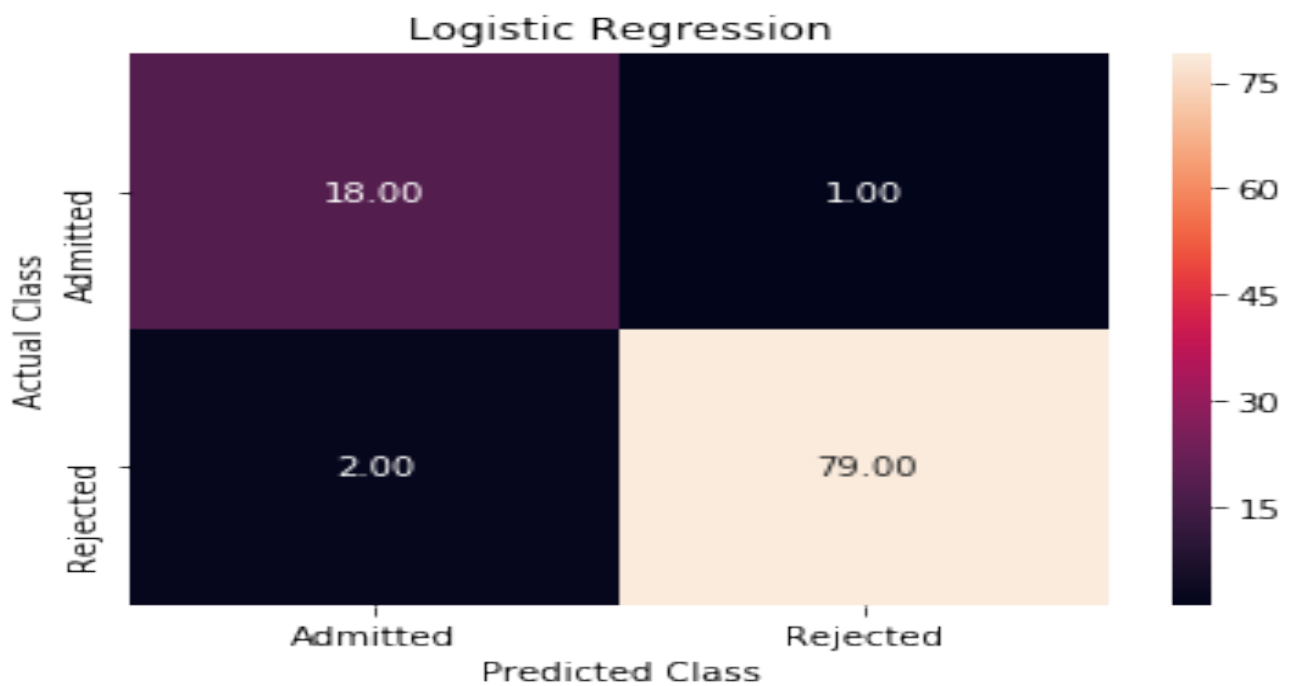


Fig 6.9 Confusion matrix of Logistic Regression

6.3.3 RANDOM FOREST

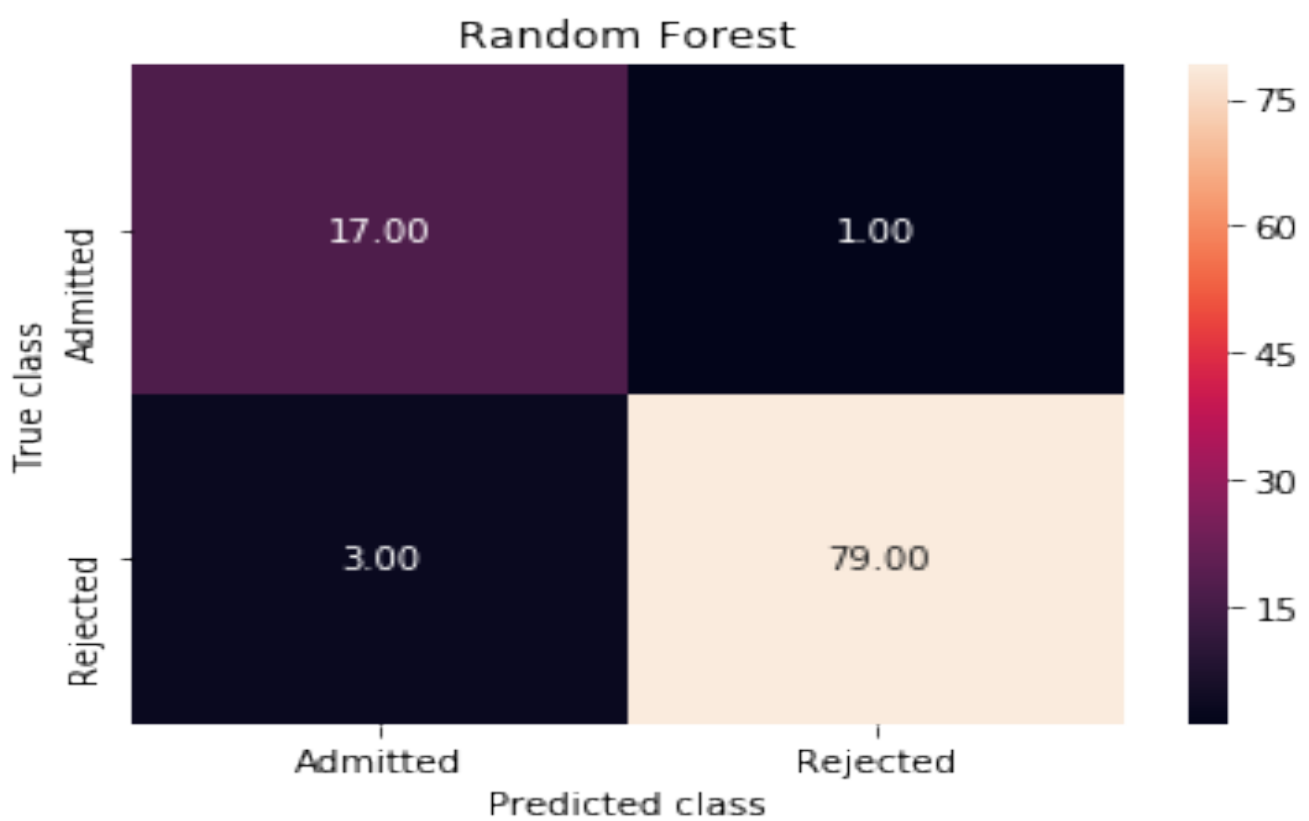


Fig 6.10 Confusion matrix of Random Forest

6.4 METRICS COMPARISON OF MODELS

6.4.1 ACCURACY

Accuracy is a highly intuitive metric, so you should not experience any challenges in understanding it. The Accuracy score is calculated by dividing the number of correct predictions by the total prediction number.

$$\text{Accuracy} = \frac{\text{TrueNegatives} + \text{TruePositive}}{\text{TruePositive} + \text{FalsePositive} + \text{TrueNegative} + \text{FalseNegative}}$$

6.4.2 F-1 SCORE

The F-1 score(also known as F-measure) is a metric used to evaluate performance of a Machine Learning model. It combines precision and recall into a single score.

F-measure formula:

$$\text{F-score} = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

6.4.3 PRECISION

Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (either correctly or incorrectly).

$$\text{Precision} = \text{True Positive} / \text{True Positive} + \text{False Positive}$$

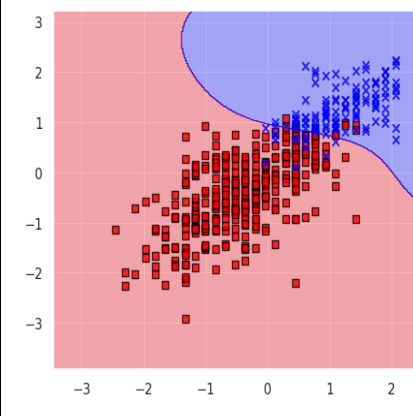
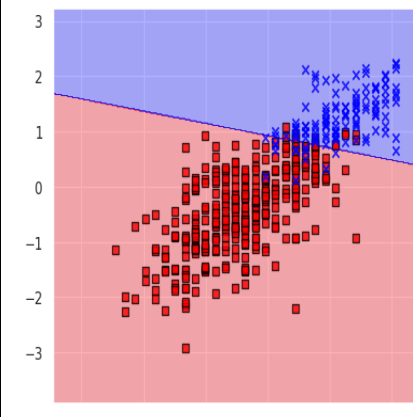
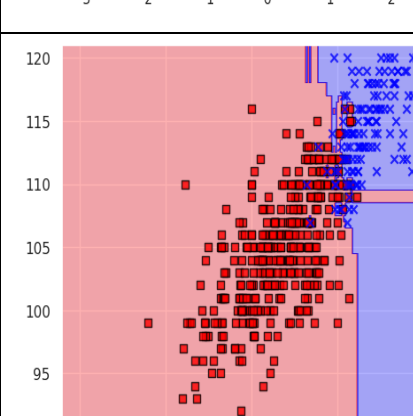
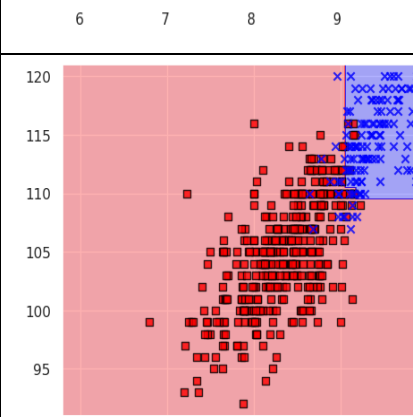
$$\text{Precision} = \text{TP} / \text{TP} + \text{FP}$$

6.4.4 RECALL

The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected.

1. $\text{Recall} = \text{True Positive} / \text{True Positive} + \text{False Negative}$

Table 6.1 Metrics comparison of models

MODEL	ACCURACY (3 parameters)	F1- SCORE	PRECISION	RECALL	ACCURACY (all parameters)	PLOT
Support Vector Machine	0.920	0.898	0.917	0.880	0.950	
Logistic Regression	0.958	0.941	0.960	0.923	0.871	
Random Forest	0.930	0.863	0.846	0.880	0.825	
Decision Tree	0.940	0.880	0.880	0.880	0.620	

6.5 COMPARISON OF MODELS:

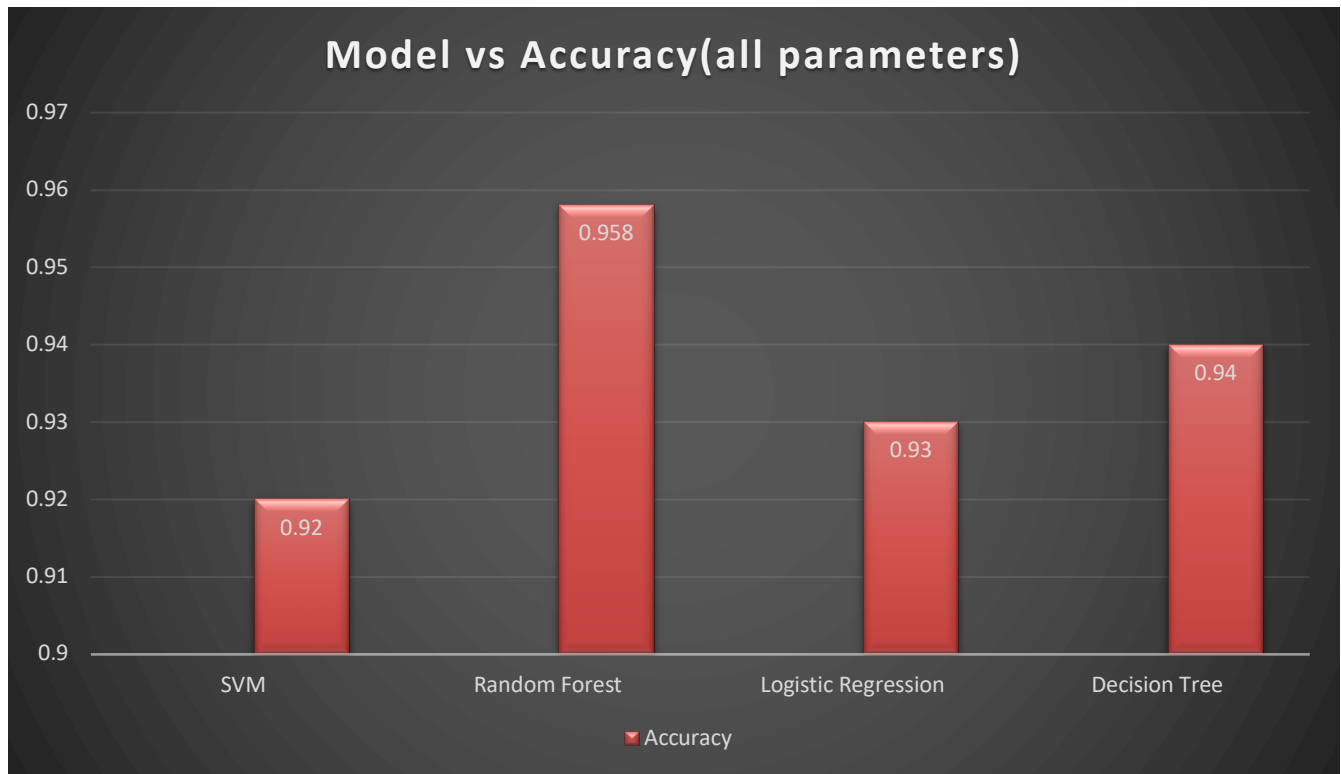


Fig 6.11 Comparison of model vs accuracy with all parameters taken into consideration

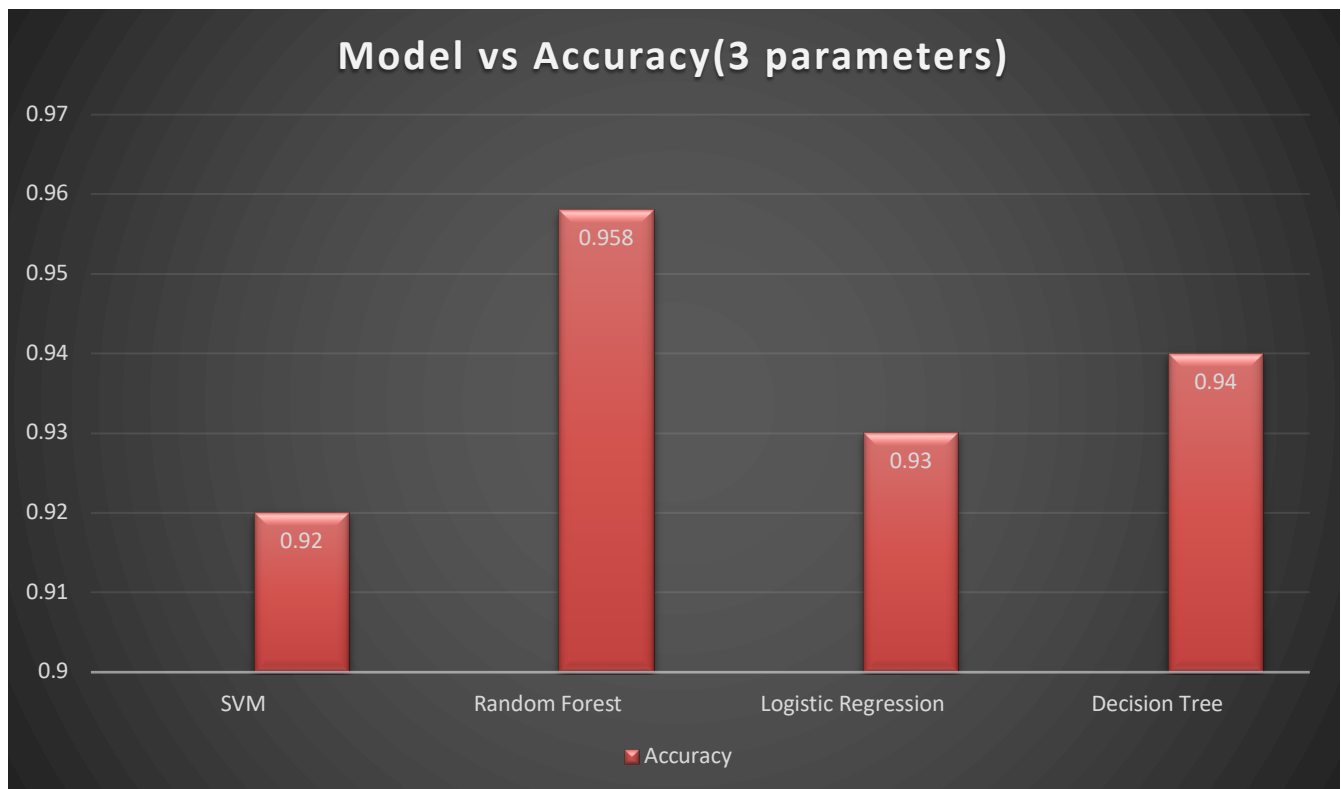


Fig 6.12 Comparison of model vs accuracy with 3 parameter taken into consideration

6.5.1 COMPARISON OF OTHER MODELS:

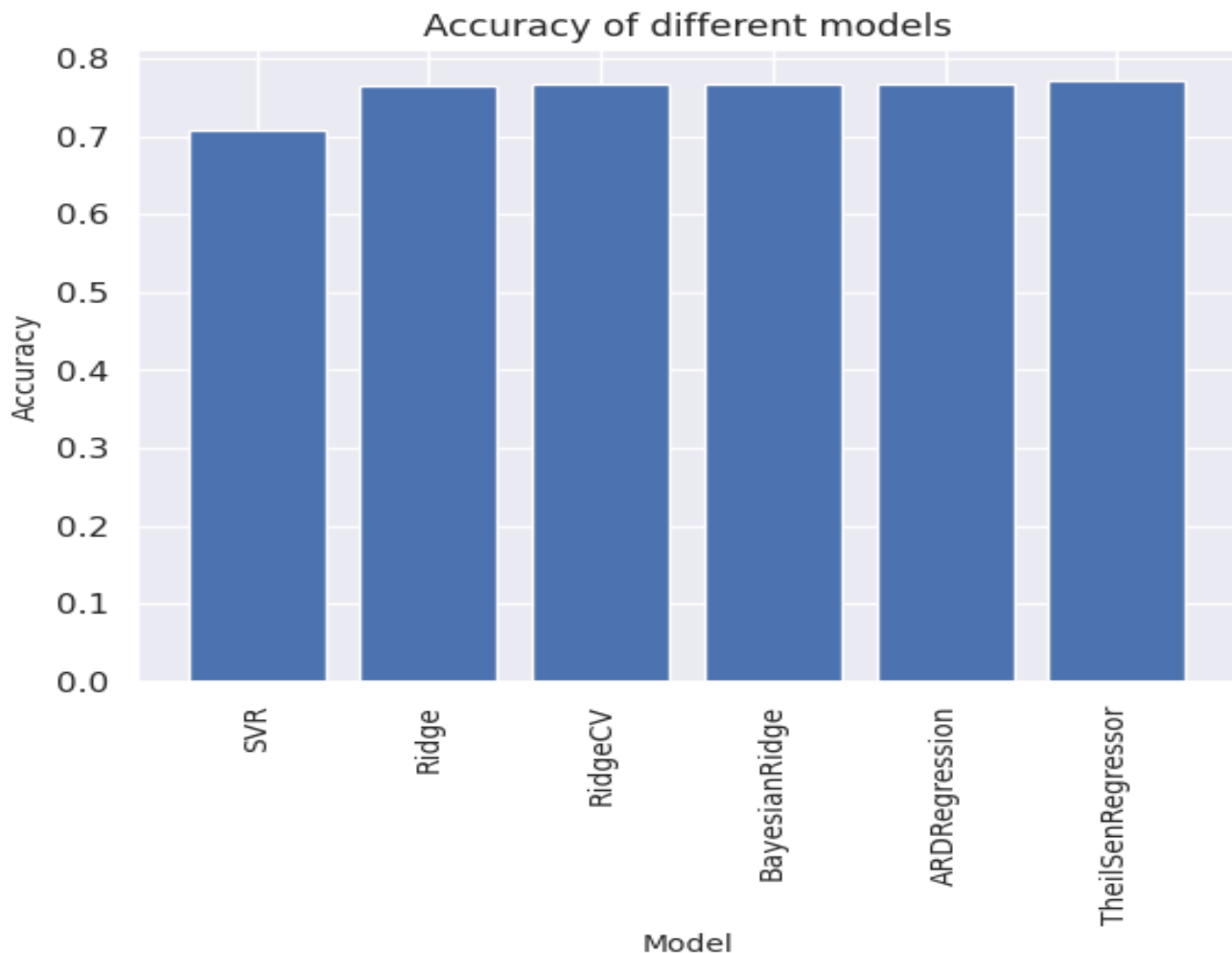


Fig 6.13 Comparison of other models

INPUT:

We input the parameters of an applicant that is GRE score, TOEFL score, LOR, number of research papers published, UR, CGPA and SOP

Here, we implemented on Random Forest model

```
newPerson = [[330, 110, 3, 4.0, 4.5, 9.8, 1]]
```

```
accuracy_RandomForest = RandomForest_model.score(xtest, ytest)
```

```
print(accuracy_RandomForest)
```

OUTPUT:

The chance of output is predicted based on the parametric data entered. In the above case chance of admit is 82.58%

0.8258437126942657

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 CONCLUSION:

Accuracy should not be the sole metric you use to decide which model performs the best against others. Accuracy is very dependent on the data you are processing, so if you have a dataset that leans heavily towards one class during training and testing, the model will have a high accuracy as a result of the skewed data. To get the best results for accuracy, you would want a balanced data set, however, that is not always possible.

Accuracy is also an easy metric to alter. For example, if the threshold were to be adjusted, one way or the other, predictions that were previously classified as a certain class may change to another. This change across the entire dataset will change the accuracy. One would be able to fib their comparisons between models if they only used accuracy and adjusted the threshold of a singular model in a favorable way. In total, we used accuracy, F1 score, precision score, and the recall score to analyze each models' performance.

There was some debate between choosing between the logistic regression model, SVM, and the random forest, but in the end, logistic regression was deemed the best. Although this model did not have the highest metrics in all categories, it was consistent with its performance with all the metrics measured. For example, in the random forest, the random forest a high F1 score and an absurd precision score, but the forest's recall score was 0.885. Even though logistic regression does not have that level of precision, it had a higher F1 score because its recall was on the same level as its precision, hovering around 0.94. There was a large difference between random forest's scores: $|\text{precision (1.00)} - \text{recall (0.885)}| = 0.115$. However, logistic regression was level with its scores: $|\text{precision (0.960)} - \text{recall (0.923)}| = 0.037$.

With the logistic regression model, we used three features to help with the classification of admission. We decided on a C value of 100 because that allowed us to maximize the metrics, and C at 100 was a balance between allowing points to be erroneous but not having too many. The random state of the model does not matter. The type of algorithm used to optimize the model did not matter much because out of the four algorithms the model could use (newton-cg, sag, saga, lbfgs), due to the multiple features being considered, they all concluded with the same metric scores, and we chose 'lbfgs' in the end. We also chose a multi-class option of 'ovr' because we want to classify a binary value (admitted or not) for each feature that was considered.

SVM and logistic regression did have the same performance across the four metrics that we measured, but we determined that logistic regression is the better model to use over SVM because it is a less computationally expensive model. This decision could be re-evaluated if we gather more data and find that SVM yields better results. Additionally, with the way we are attempting to classify the data, the logistic regression model would be a better fit because we are basically classifying between two classes. A SVM model would be more useful if we did not binarize the Chance of Admit feature.

7.2 FUTURE ENHANCEMENTS

An improvement can also be made and the system can always be made more efficient and accurate using NLP or natural language processing methods which can help rate the quality of the written essays and paragraphs during the exams, so that those features can also help narrow down the chances of admissions over time.

Build a ChatBot to get to interact with the applicant and get the information from applicants.

- a) To predict their chance of admission.
- b) To recommend the university for his profile.
- c) To continuously train our model for better prediction.

REFERENCES

- [1] Zain Bitar,Amjed Al-Mousa(2020), "Prediction of Graduate Admission using Multiple Supervised Machine Learning Models", SoutheastCon.
- [2] Sashank Sridhar,Mootha, Santosh Kolagati(2020), "A University Admission Prediction System using Stacked Ensemble Learning",Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA).
- [3] Md. Omaer Faruq Goni,Abdul Matin,Tonmoy Hasan,Md. Abu Ismail Siddique,Oishi Jyoti,Fahim MD Sifnatul Hasnain(2020), "Graduate Admission Chance Prediction Using Deep Neural Network",IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE).
- [4] Fahim MD, Sifnatul Hasnan (2021), "A Comparative Study on University Admission Predictions Using Machine Learning Techniques", International Journal of Scientific Research in Computer Science, Engineering and Information Technology.
- [5] Kruthika CS, Apeksha B, Chinmaya GR, Madhumathi JB, Veena MR (2021), "University Admission Prediction using Machine Learning. Am J Glob J Res Rev Vol: 8 No:7.," in IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4724-4734.
- [6] Sujay S (2018),"Supervised Machine Learning Modelling & Analysis For Graduate Admission Prediction" ,Published in International Journal of Trend in Research and Development (IJTRD).
- [7] M. S. Acharya, A. Armaan and A. S. Antony(2019), "A Comparison of Regression Models for Prediction of Graduate Admissions",International Conference on Computational Intelligence in Data Science (ICCIDS), Chennai, India.

