(UCS411) Artificial Intelligence

PROJECT REPORT

COLLEGE ADMISSION PREDICTION

Submitted to: Dr. Jhilik Bhattacharya

By:

Nipun Tank	102153011
Yuvam Sharma	102153012
Manav Singh	102283005
Vinayak Kapur	102103864
Hansika Sachdeva	102103309

TABLE OF CONTENTS

ABSTRACT

In our project we aim to find the most suitable colleges for applying based on a student's profile. We can judge across a wide variety of domains that include MS (international), M.Tech (India) and MBA (India and International). For the accurate predictions we plan on training a machine learning model in order to provide results. The dataset contains information on the student profile and the university and course details with a field detailing if the admission was positive or not. We also want to create a portal with additional features such as GPA converter, connecting with graduates from universities and a system that advises on what aspects of a student's profile must be improved. We could provide two types of outputs for a profile that includes a probability score as to whether a university is suitable for that student, or a list of universities that have high probability of accepting the student and is ranked based on the university scores.

INTRODUCTION

For anyone pursuing their postgraduate studies, it would be difficult for them to find out what college they may join, based on their GPA, Quants, Verbal, TOEFL and AWA Scores. People may apply to many colleges that look for candidates with a higher score set, instead of applying to colleges at which they have a chance of getting into. This would be detrimental to their future. It is very important that a candidate should apply to colleges that he/she has a good chance of getting into, instead of applying to colleges that they may never get into.

There aren't many efficient ways to find out the colleges that one can get into, relatively quickly. The Education Based Prediction System helps a person decide what colleges they can apply to with their scores. The dataset that is used for processing consists of the following parameters: College names, Quants and Verbal Scores (GRE) TOEFL and AWA Scores. The GRE Test (Graduate Record Examinations) is a standardized test used by many universities and graduate schools around the world as part of the graduate admissions process.

Other factors are also taken into consideration while applying to colleges, such as Letter of Prediction (LoR), Statement of Purpose (SoP), Co-curricular activities and Research papers as well (research papers from journals that are not well known or have a high percentage of plagiarism are not taken into consideration for this case).

When a person has completed their undergraduate degree and wants to pursue a Postgraduate degree in a field of their choice, more often than not, it is very confusing for the person to figure out what colleges they should apply to with the scores that they have obtained in GRE and TOEFL, along with their GPA at the time of their graduation. Many candidates may apply to colleges that do not fall

under their score requirements and hence waste a lot of time. Applying to many colleges with scores also increases the cost. There are not many efficient methods that are available to help address this issue and hence an Education Predictor System has been developed.

In the system proposed, a person can enter their scores in the respective fields provided. The system then processes the data entered and produces an output of the list of colleges that a person could get into, with their scores. This is relatively quick and helps conserve time and money. In order to achieve this we have proposed a novel method utilising Machine Learning and Data mining algorithms.

LITERATURE REVIEW

The prediction systems collect the information regarding items. They gather preferences and profiles and analyse the same to advise the user to make right decisions regarding products, people, policies, and services.[1]. As day after day, the availability of electronic and web content is growing fast, researchers are relying more on content to extract vital information for better predictions. So, prediction systems became popular in assisting numerous decision-making contexts. The user is an entity to which the prediction is provided, and an item is a product/service being predicted. Prediction analysis predicts the future preferences by analysing the previous interaction between users and items because past behaviours are often good indicators of future choices. It is the toughest job to design and implement a large-scale online service that can find what is to be predicted to the customers based on the past purchase history. For example, Amazon gives product predictions, yahoo makes web page predictions. The process of constructing an efficient and effective predictions system is a challenging task. The underlying reason is the large size of the product (object) space and context space.

The main goal of prediction systems is to assist its users in finding their preferred objects from the large set of available objects. The voting of a particular customer on a particular object is learned through a random payoff and this payoff is received by the predictor system based on the response details of the customer to the prediction system. Zhibo Wang, et al. [2] proposed a unique similarity-based metric to find the similarity details of users in terms of their lifestyles and they have constructed a Friend book system to predict friends based on their lifestyles. Prediction systems have developed in parallel with the web technology J.

Bobadilla et al. [3]. At the initial time of their existence, they were based on demographic, content-based and collaborative filtering. Now they are in a position to incorporate social information also.

A knowledge-based prediction system considers user centric requirements rather than his/her past history in order to make predictions. Hector Nunez, et al. [4] discussed the comparison of different similarity measures for improving the classification process. Authors said that automatic knowledge acquisition and management methods are needed to build consistent, robust, reliable, fault tolerant, and effective decision support systems. Fazeli Soude, et al. [7] said that predictor systems are being used (have been using) in many real-world applications such as e-commerce-based applications – Amazon and eBay. Predictor systems must be accurate and useful to as many numbers of users as possible. The fundamental goal of the educational predictor systems is to satisfy many quality features such as accuracy, usefulness, effectiveness, novelty, completeness, and diversity. Recommender systems must satisfy user-centric requirements. User-centric based prediction systems are more useful than datacentric prediction systems. The methods existing for the prediction are contentbased filtering, collaborative filtering, and rule mining approaches. Contentbased filtering approach predicts an item to a user by clustering the items and the user pairs into groups. This clustering is used to gain similarity between user and item. Personal information of the user is not considered here. Queen Esther Booker creates a prototype of a system for course predictions [10]. The system accepts user requirements as keywords and predicts courses for students. Collaborative filtering (CF) approach predicts an item to a user by grouping similar users based on user profiles and predicts the user interests towards the items. Hana introduces a system based on CF approach to predict courses for a student by analysing and matching the student's academic records

[8]. Then the system analyses and predicts a course that meets the student's profile. Elham S. Khorasani et al. proposed a Collaborative Filtering model based on Markov Chain to predict courses based on historical data [7].

Rule mining approach focuses on predicting a series of items to a user by discovering the association rules. Itmazi and Megias developed a prediction system based on rule mining to predict learning objects [9]. Akrivi Vlachou, et al. [11] said that finding the most influential database tuples from a given database of tuples is very useful in real-world applications such as market data analysis and decision making. Authors proposed two algorithms for finding most influential database objects. The first one uses properties of the sky-band (SB) set for limiting the maximum number of resultant candidate objects and the second one follows branch and bound (BB) algorithm paradigm and it uses upper bound on influence score.

Amit Singh, et al. [14] proposed an approximate solution to answer reverse nearest neighbour queries in high dimensional spaces. Authors said that the approach is mainly based on a feature called strong co-relation between k-nearest neighbour (k-NN) and reverse the nearest neighbour (RNN) in connection with Boolean range query (BRQ). Elke Achtert, et al. [6] said that all the existing generalized reverse k-nearest neighbour (RkNN) search methods are only applicable to Euclidian distances but not for general metric objects. As a result, authors proposed first approach for efficient reverse k-nearest neighbour search in arbitrary metric spaces (RkNNSAMS) and k value will be given at query run time. Duc Thang et al. [5] said that fast, usability, simplicity and with reasonably good performance features are always better than the best performing algorithm only in some cases and rare usage of the algorithm because of high complexity. DINO IENCO, et al. [15] said that the process of clustering data objects containing only

categorical attributes is a tedious task because defining a distance value between pairs of categorical attributes is difficult.

Authors proposed a framework to find a distance measure between categorical attributes. Madhavi et al. [3] formulated measures on the data containing categorical attributes. They categorized existing measures as context free and context-sensitive measures for categorical data. Usue Mori et al. [12] said that the most famous Euclidian distance and the common measures used for non-temporal data are not always the best methods for finding similarity between time series data because they do not deal with noise and misalignments in the time series data. Authors said that Euclidian distance suffers from noise and outlier problems. Yung-Shen Lin et al. [13] said that similarity measures are being used extensively in text classification and clustering. In the literature, various methods used for similarity comparison are - Euclidian distance, Manhattan distance, taxicab distance, cosine similarity measure, city-block distance, Bray-Curties measure, Jaccard coefficient, extended Jaccard coefficient, Hamming distance, Dice coefficient, IT-Sim and so on.

Bouzekri E, et al. [20] in their work the highlight is on engineering issues related to the development of a predicter system in a critical context. They propose a generic architecture to decompose Predicter Systems. This generic architecture integrates existing proposals from the Predicter Systems community with current knowledge in interactive systems architectures. In order to engineer Predicter Systems compliant with the entire list of requirements identified, they propose to use a set of complementary integrated modelbased approaches from the literature. Zhang, et al. [22] go against the classic way of implicitly assuming the underlying classification is binary, that is, a candidate item is either predicted or not. Here they propose an alternate framework that integrates three-way decision and random forests to build predicter systems by considering both misclassification

cost and teacher cost. With these costs, a threeway decision model is built, and rational settings for positive and negative threshold values α* and β* are computed. Next, they construct a random forest to compute the probability P that a user will like an item. Experimental results show that the (α*, β*)-pair determined by threeway decision is optimal.

Y. Subba Reddy, et al. [16] proposed a prediction system for college/course selection. The experimental results showed that applying WCLUSTER in this domain is superior to traditional and previous approaches. To speed up the query execution process a multidimensional indexing structure called R-Tree was used. Deokate Monali, et al. [17] present a framework to choose a desired college using a prediction system on the basis of college NAAC grade, NBA grade, campus placement and review from alumni student and also add Semantic analysis algorithm which will capture positive and negative sentiments, combining Naive Bayes and Adaboost algorithm which was used to rank the branches as well as colleges.

Tulasi K.Paradarami, et al. [18] show that a set of content and collaborative features allows for the development of a neural network model with the goal of minimizing logloss and rating misclassification error using stochastic gradient descent optimization algorithm; and that the hybrid approach is a very promising solution when compared to standalone memory-based collaborative filtering method. K Qazanfari, et al. [19] present a novel text-content-based predicter system as a valuable tool to predict user interests. It developed a specific procedure to create user models and item feature-vectors by soliciting from a user a few keywords and expanding those keywords into a list of weighted nearsynonyms, resulting in higher precision and accuracy in comparison to wellknown feature-vector-generating methods like Glove and Word2Vec.

PROBLEM STATEMENT

Educational organizations are one of the important parts of our society and playing a vital role for growth and development of any individual. There are different college prediction apps and websites being maintained contemporarily, but using them is tedious to some extent, due to the lack of articulate information regarding colleges, and the time consumed in searching the best deserving college.

The problem statement, hence being tackled, is to design a college prediction/prediction system and to provide a probabilistic insight into college administration for overall rating, cut-offs of the colleges, admission intake and preferences of students. Also, it helps students avoid spending time and money on counsellor and stressful research related to finding a suitable college.

Students often face the challenge of finding suitable universities and colleges for future studies. They may know what stream they would want to get into, however it is difficult for them to find colleges based on their Academic and extracurricular performance. What we aim to do is provide a portal which can give a probabilistic output as to how likely it is to get into a university given their details.

DATASET USED

https://www.kaggle.com/mohansacharya/graduate-admissions

Data Explorer

28.4 KB

Admission_Predict.csv

Admission_Predict_Ver1.1.csv

Summary

2 files

18 columns

Admission_Predict.csv (12.6 KB)

Detail | Compact | Column

9 of 9 columns

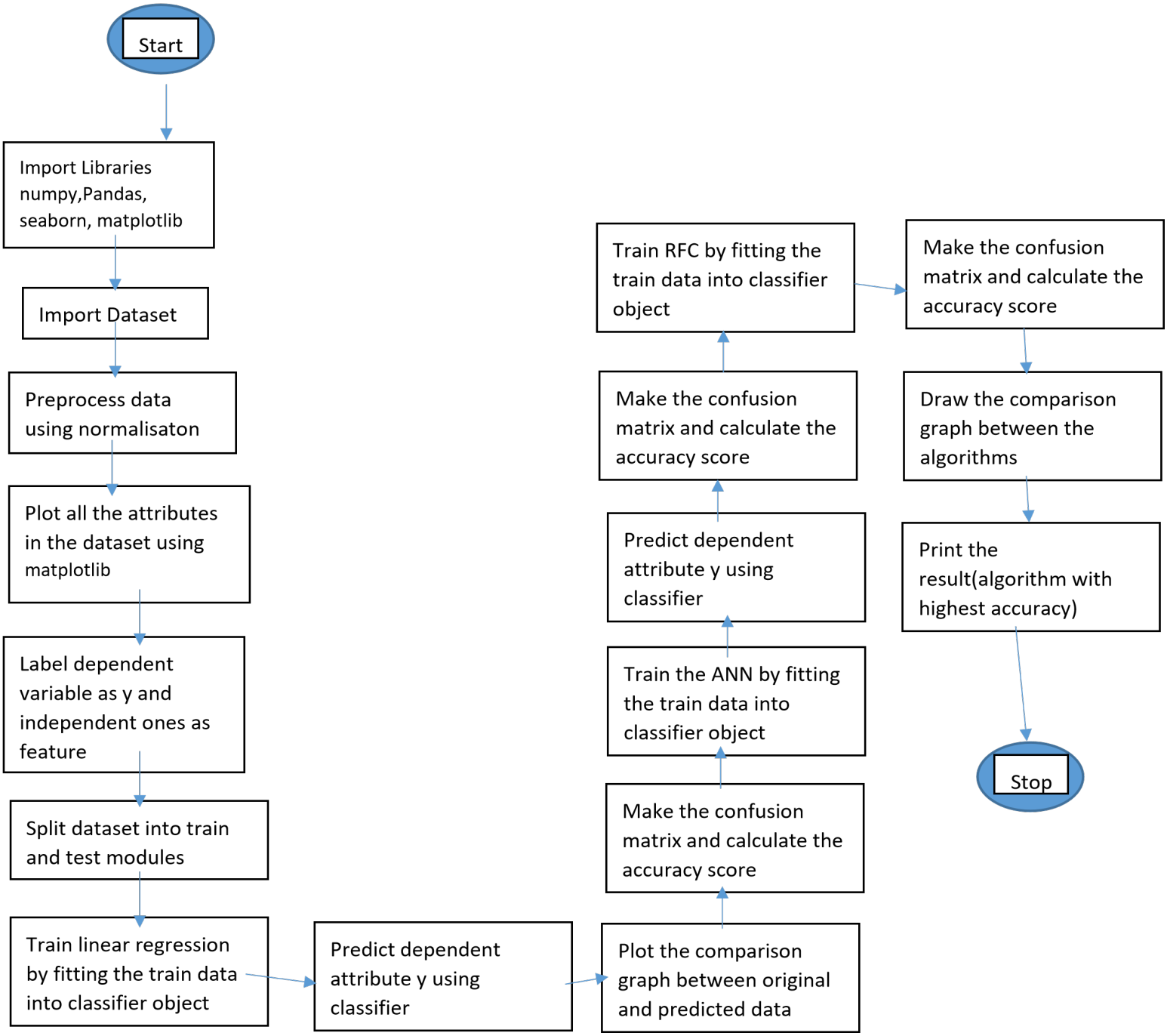
# Serial No.	# GRE Score	# TOEFL Score	# University Rating	# SOP	#
1 – 400	290 – 340	92 – 120	1 – 5	1 – 5	1
1	337	118	4	4.5	4.
2	324	107	4	4	4.
3	316	104	3	3	3.
4	322	110	3	3.5	2.
5	314	103	2	2	3
6	330	115	5	4.5	3
7	321	109	3	3	4
8	308	101	2	3	4
9	302	102	1	2	1.
10	323	108	3	3.5	3
11	325	106	3	3.5	4
12	327	111	4	4	4.
13	328	112	4	4	4.
14	307	109	3	4	3
15	311	104	3	3.5	2
16	314	105	3	3.5	2.

Data Tasks (3) Code (709) Discussion (44) Activity Metadata Download (29 kB) New Notebook

Metadata Feedback

Usage Information	License	CC0: Public Domain
	Visibility	Public
Maintainers	Dataset owner	Mohan S Acharya
Updates	Expected update frequency	Not specified
	Last updated	2018-12-28
	Date created	2018-03-04
	Current version	Version 2

ARCHITECTURE



ALGORITHMS USED

1. Linear Regression

Simple linear regression is an approach for predicting a response using a single feature. It is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used. It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

2. Artificial Neural Network

Neural networks are artificial systems that were inspired by biological neural networks. These systems learn to perform tasks by being exposed to various datasets and examples without any task-specific rules. The idea is that the system generates identifying characteristics from the data they have been passed without being programmed with a pre-programmed understanding of these datasets. Artificial Neural Networks contain artificial neurons which are called units. These units are arranged in a series of layers that together constitute the whole Artificial Neural Networks in a system. A layer can have only a dozen units or millions of units as this depends on the complexity of the system. Commonly, Artificial Neural Network has an input layer, output layer as well as hidden layers. The input layer receives data from the outside world which the neural network needs to analyze or learn about. Then this data passes through one or multiple hidden layers that transform the input into data that is valuable for the output layer. Finally, the output layer provides an output in the form of a response of the Artificial Neural Networks to input data provided.

3. Decision Trees

A Decision Tree is constructed by asking a serious of questions with respect to a record of the dataset we have got. Each time an answer is received, a follow-up question is asked until a conclusion about the class label of the record. The series of questions and their possible answers can be organised in the form of a decision tree, which is a hierarchical structure consisting of nodes and directed edges. A tree has three types of nodes:

- root node that has no incoming edges and zero or more outgoing edges.
- Internal nodes, each of which has exactly one incoming edge and two or more outgoing edges.
- Leaf or terminal nodes, each of which has exactly one incoming edge and no outgoing edges.

In a decision tree, each leaf node is assigned a class label. The non-terminal nodes, which include the root and other internal nodes, contain attribute test conditions to separate records that have different characteristics.

4. Random Forests

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

IMPLEMENTATION & RESULT

Our implementation can be accessed by using the following link:

https://colab.research.google.com/drive/1TNvmjMWKy_0hTuUqnqY7oVGMyb8Punhs?usp=sharing

https://github.com/Nipuntank/TIET-DSA-lab-assignment/blob/main/AI_Project.ipynb

Artificial Intelligence Project Implementation

Members:

Nipun Tank

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

+ Code + Text

```
# read the csv file
admission_df = pd.read_csv('Admission_Predict_Ver1.1.csv')
```

```
admission_df.head()
```

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

```
# Let's drop the serial no.
admission_df.drop('Serial No.', axis = 1, inplace = True)
admission_df
```

...	...	...	...	...	...	...	...	...
495	332	108	5	4.5	4.0	9.02	1	0.87
496	337	117	5	5.0	5.0	9.87	1	0.96
497	330	120	5	4.5	5.0	9.56	1	0.93
498	312	103	4	4.0	5.0	8.43	0	0.73
499	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 8 columns

```
[ ] # checking the null values
    admission_df.isnull().sum()
```

```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

```
[ ] # Check the dataframe information
    admission_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   GRE Score          500 non-null    int64  
 1   TOEFL Score        500 non-null    int64  
 2   University Rating  500 non-null    int64  
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64  
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 31.4 KB
```

```
[ ] # Statistical summary of the dataframe
    admission_df.describe()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000

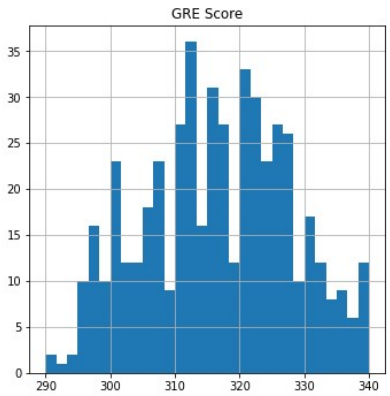
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

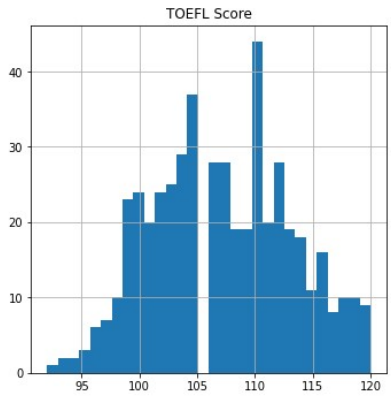
```
# Grouping by University ranking
df_university = admission_df.groupby(by = 'University Rating').mean()
df_university
```

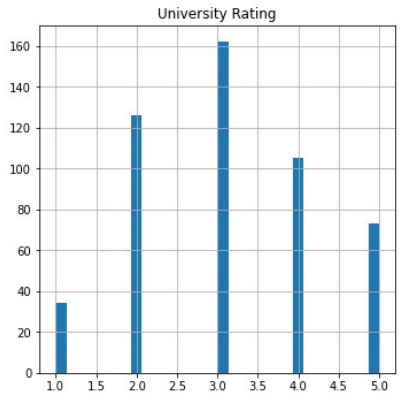
University Rating	GRE Score	TOEFL Score	SOP	LOR	CGPA	Research	Chance of Admit
1	304.911765	100.205882	1.941176	2.426471	7.798529	0.294118	0.562059
2	309.134921	103.444444	2.682540	2.956349	8.177778	0.293651	0.626111
3	315.030864	106.314815	3.308642	3.401235	8.500123	0.537037	0.702901
4	323.304762	110.961905	4.000000	3.947619	8.936667	0.780952	0.801619
5	327.890411	113.438356	4.479452	4.404110	9.278082	0.876712	0.888082

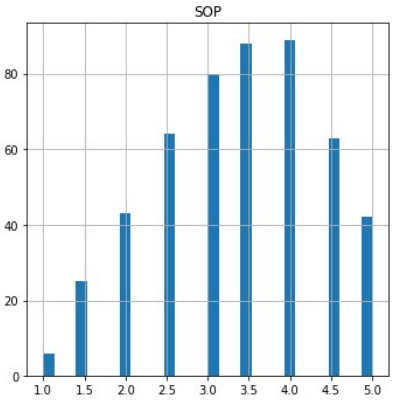
```
admission_df.hist(bins = 30, figsize =(20,20))
```

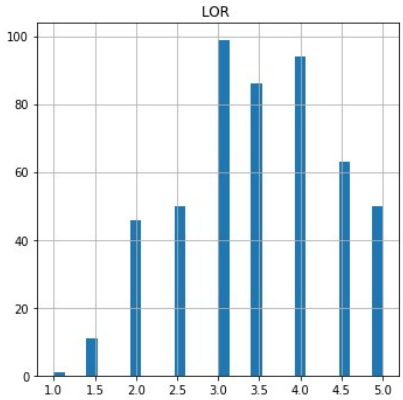
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f68c7957b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f68c78aad10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f68c7873350>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f68c7827950>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f68c77ddf50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f68c779e590>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f68c7755b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f68c771b110>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f68c7841990>]],
      dtype=object)
```

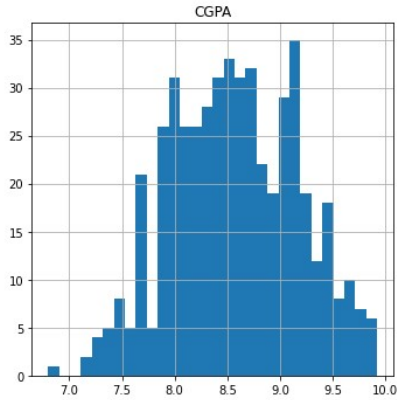


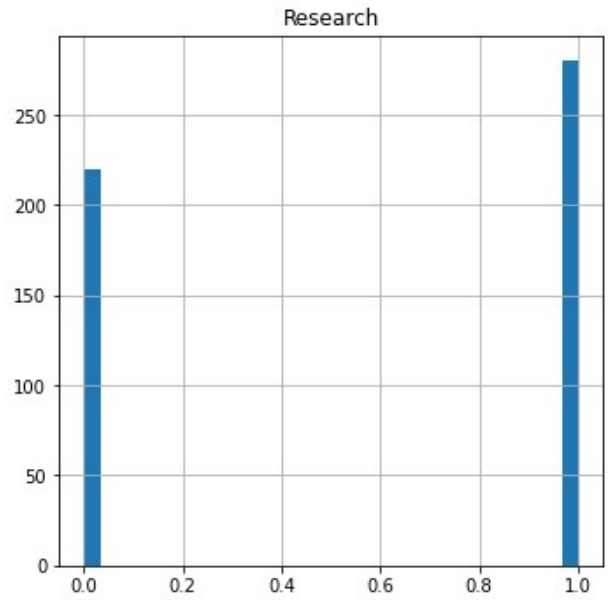


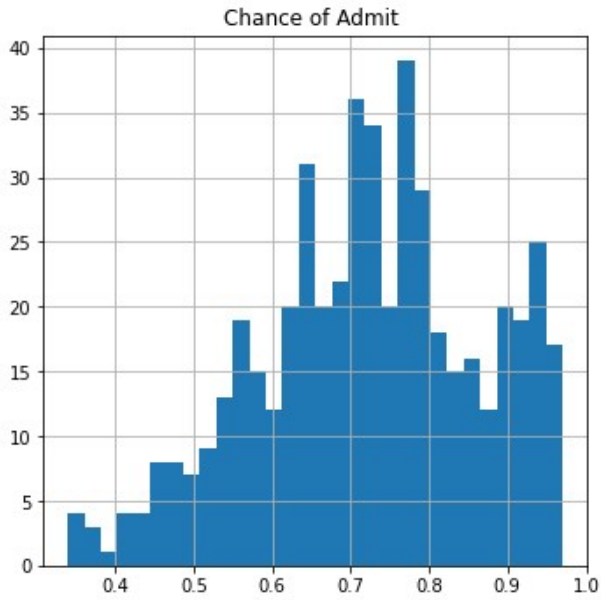




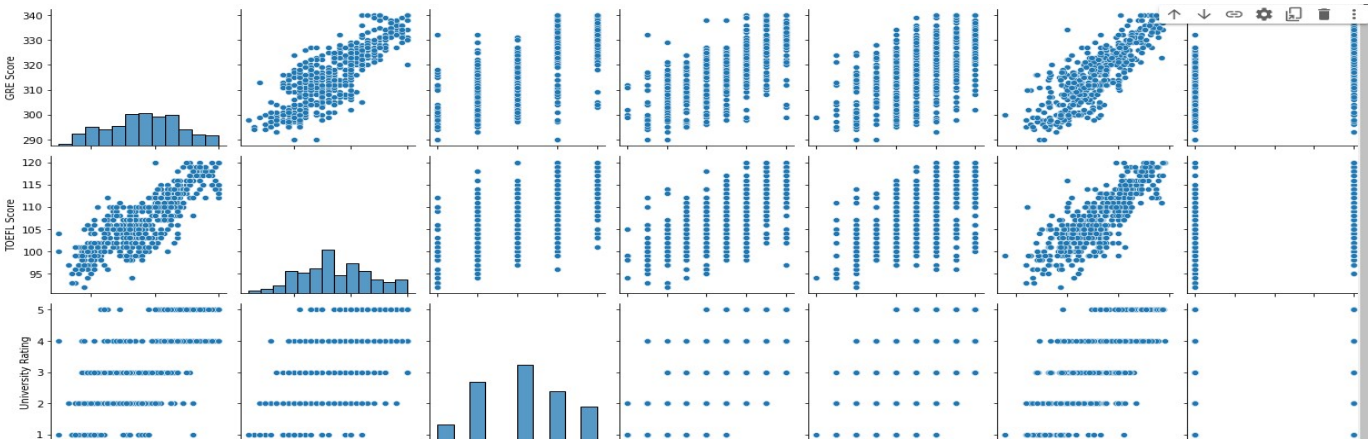


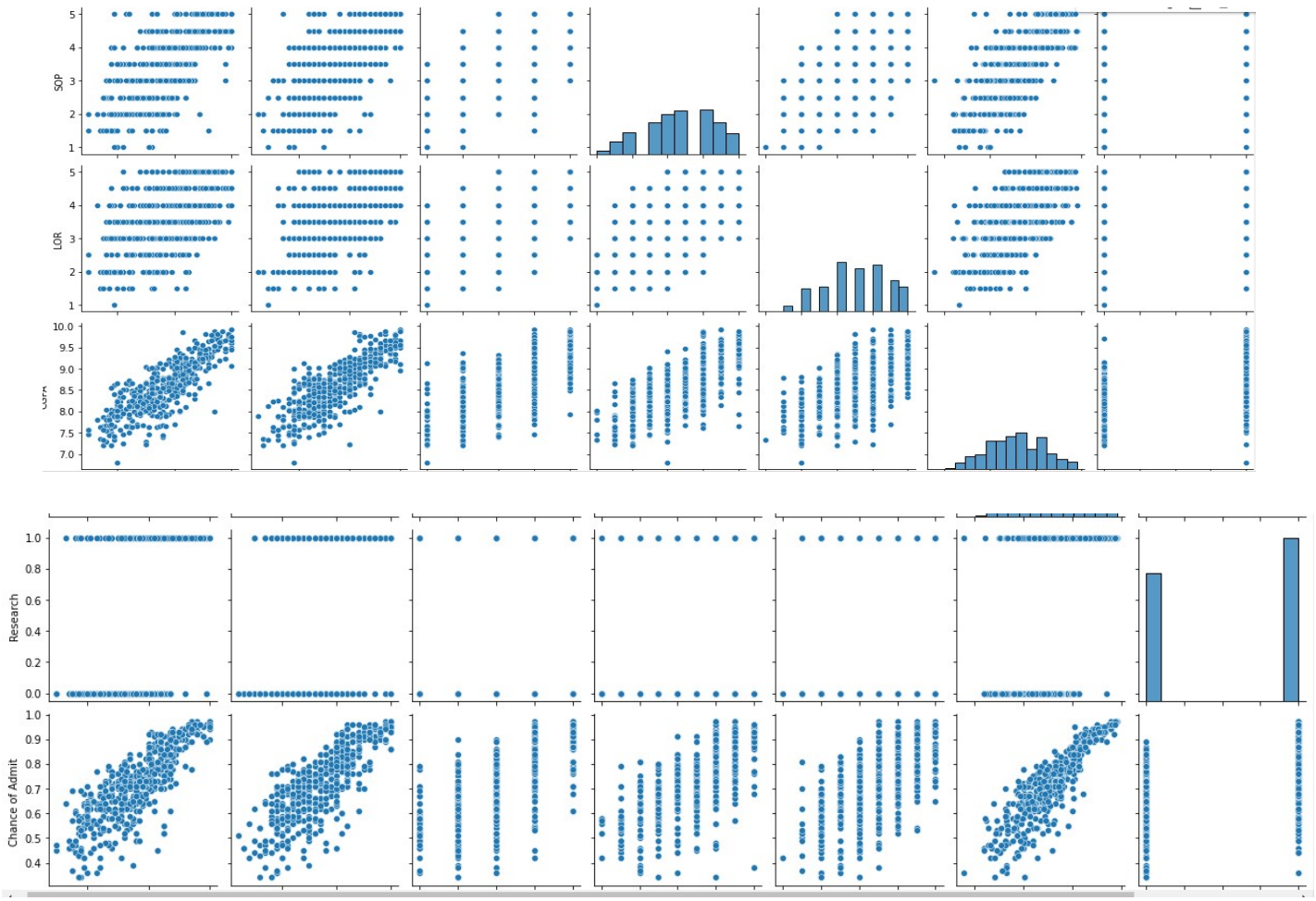




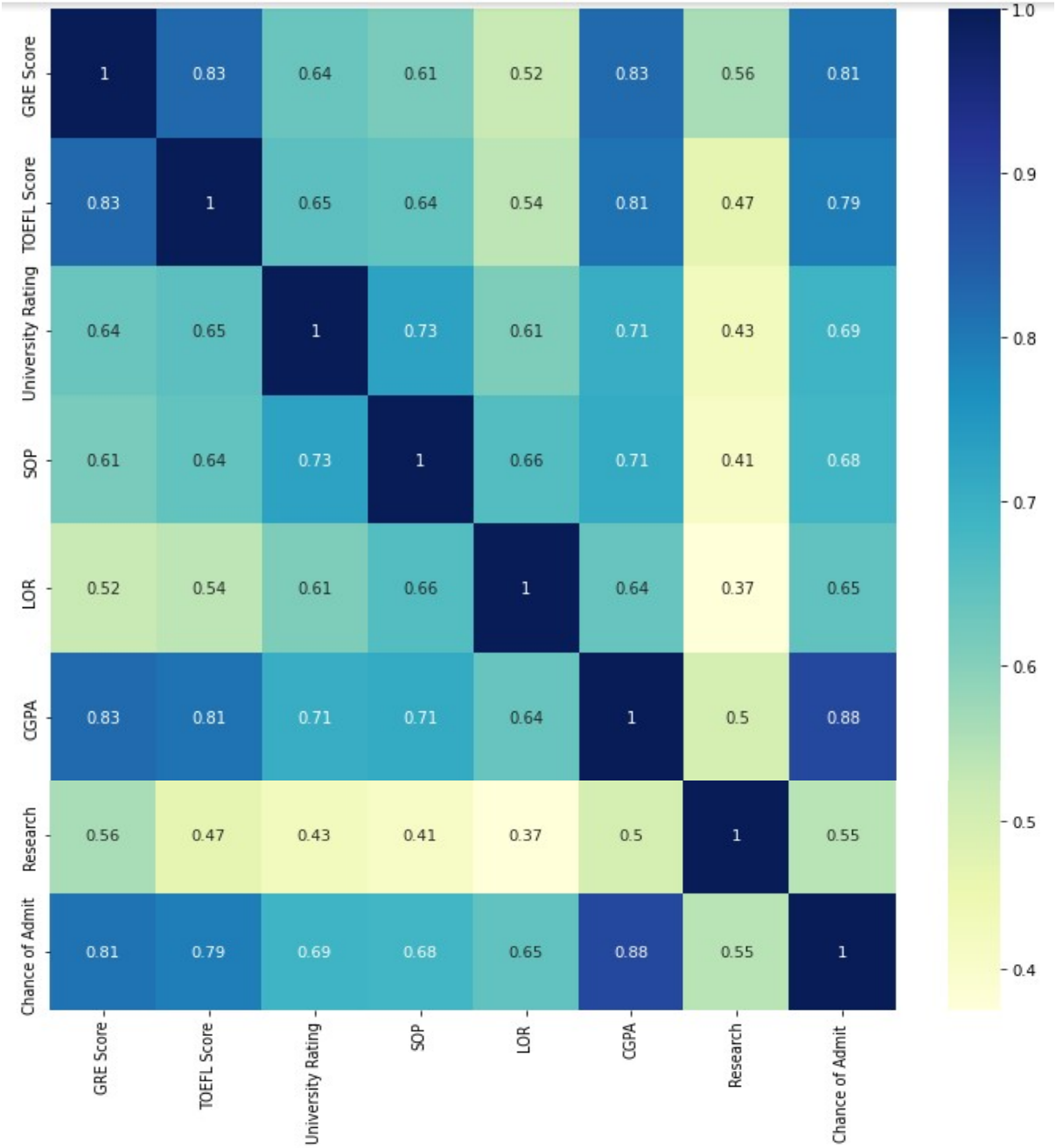


sns.pairplot(admission_df)






```
corr_matrix = admission_df.corr()
plt.figure(figsize = (12,12))
sns.heatmap(corr_matrix , annot = True, cmap="YlGnBu")
plt.show()
```


GRE Score
TOEFL Score
University Rating
SOP
LOR
CGPA
Research
Chance of Admit
1 0.83 0.64 0.61 0.52 0.83 0.56 0.81
0.83 1 0.65 0.64 0.54 0.81 0.47 0.79
0.64 0.65 1 0.73 0.61 0.71 0.43 0.69
0.61 0.64 0.73 1 0.66 0.71 0.41 0.68
0.52 0.54 0.61 0.66 1 0.64 0.37 0.65
0.83 0.81 0.71 0.71 0.64 1 0.5 0.88
0.56 0.47 0.43 0.41 0.37 0.5 1 0.55
0.81 0.79 0.69 0.68 0.65 0.88 0.55 1
1.0
0.9
0.8
0.7
0.6
0.5
0.4

```
[ ] admission_df.columns
```

```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

```
[ ] X = admission_df.drop(columns = ['Chance of Admit '])
```

```
[ ] y = admission_df['Chance of Admit ']
```

```
[ ] X.shape
```

```
(500, 7)
```

```
[ ] y.shape
```

```
(500,)
```

```
[ ] y
```

```
0      0.92
1      0.76
2      0.72
3      0.80
4      0.65
       ...
495    0.87
496    0.96
497    0.93
498    0.73
499    0.84
Name: Chance of Admit , Length: 500, dtype: float64
```

```
X = np.array(X)
y = np.array(y)
```

```
y = y.reshape(-1,1)
y.shape
```

```
(500, 1)
```

```
# scaling the data before training the model
from sklearn.preprocessing import StandardScaler, MinMaxScaler
scalar_x = StandardScaler()
X = scalar_x.fit_transform(X)
```

```
[ ]  scalar_y = StandardScaler()
     y = scalar_y.fit_transform(y)
```

```
[ ]
     # spliting the data in to test and train sets
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X , y , test_size=0.5)
```

LINEAR REGRESSION

```
[ ]  from sklearn.linear_model import LinearRegression
     from sklearn.metrics import mean_squared_error, accuracy_score
```

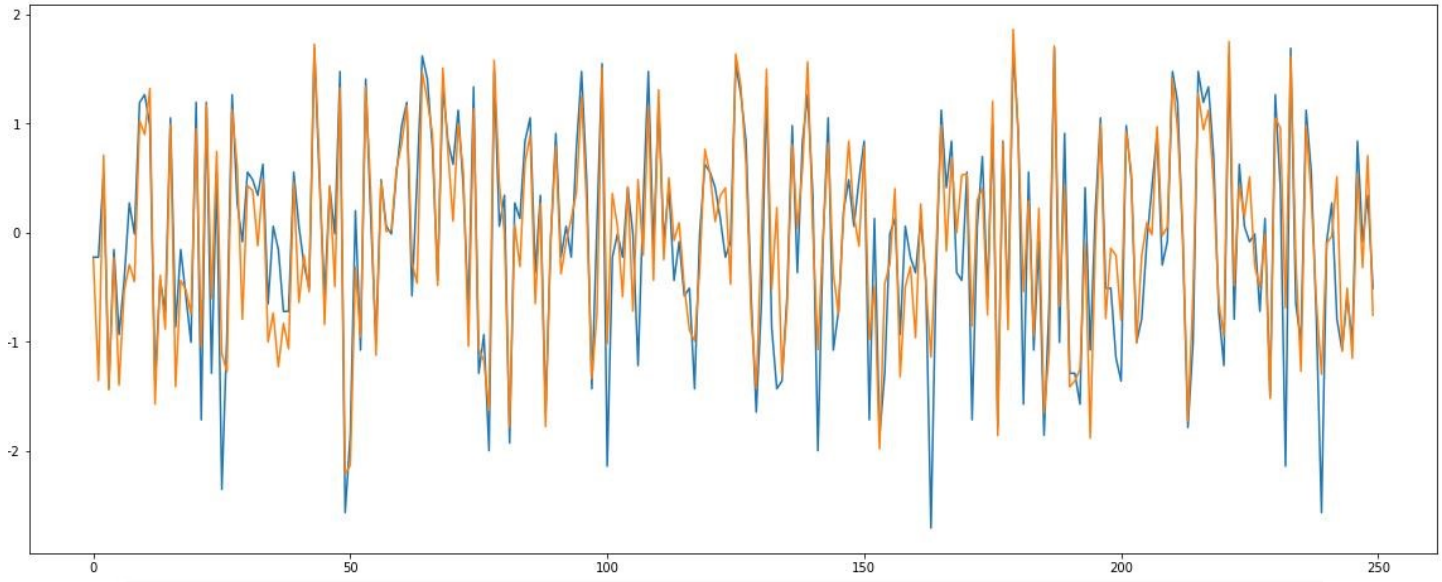
```
[ ]
     LinearRegression_model = LinearRegression()
     LinearRegression_model.fit(X_train, y_train)
```

```
     LinearRegression()
```

```
[ ]  y_pred = LinearRegression_model.predict(X_test)
```

```
[ ]  from matplotlib import rcParams
     rcParams['figure.figsize']=20,8
     plt.plot(y_test)
     plt.plot(y_pred)
```

```
     [<matplotlib.lines.Line2D at 0x7f68426cd410>]
```




```
[ ]  accuracy_LinearRegression = LinearRegression_model.score(X_test, y_test)
     accuracy_LinearRegression
```

```
0.8060189050302291
```

NEURAL NETWORKS

```
[ ]  import tensorflow as tf
     from tensorflow import keras
     from tensorflow.keras.layers import Dense, Activation, Dropout
     from tensorflow.keras.optimizers import Adam
```

```
[ ]  ANN_model = keras.Sequential()
     ANN_model.add(Dense(50, input_dim = 7))
     ANN_model.add(Activation('relu'))
     ANN_model.add(Dense(150))
     ANN_model.add(Activation('relu'))
     ANN_model.add(Dropout(0.5))
     ANN_model.add(Dense(150))
     ANN_model.add(Activation('relu'))
     ANN_model.add(Dropout(0.5))
     ANN_model.add(Dense(50))
     ANN_model.add(Activation('linear'))
```

```
[ ] ANN_model.add(Dense(1))
    ANN_model.compile(loss = 'mse', optimizer = 'adam')
    ANN_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                400

 activation (Activation)     (None, 50)                0

 dense_1 (Dense)             (None, 150)               7650

 activation_1 (Activation)   (None, 150)               0

 dropout (Dropout)           (None, 150)               0

 dense_2 (Dense)             (None, 150)               22650

 activation_2 (Activation)   (None, 150)               0

 dropout_1 (Dropout)         (None, 150)               0

 dense_3 (Dense)             (None, 50)                7550

 activation_3 (Activation)   (None, 50)                0

 dense_4 (Dense)             (None, 1)                 51

=================================================================
Total params: 38,301
Trainable params: 38,301
Non-trainable params: 0
_________________________________________________________________
```

```
ANN_model.compile(optimizer='Adam', loss='mean_squared_error')
```

```
epochs_hist = ANN_model.fit(X_train, y_train, epochs = 100, batch_size = 20)
```

```
[ ] result = ANN_model.evaluate(X_test, y_test)
    accuracy_ANN = 1 - result
    print("Accuracy : {}".format(accuracy_ANN))
```

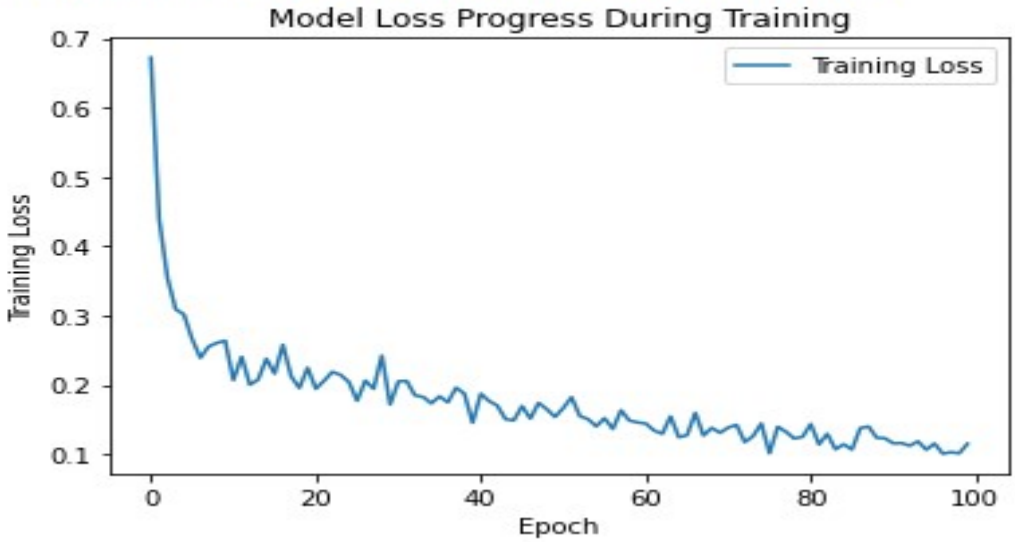
```
8/8 [==============================] - 0s 3ms/step - loss: 0.2803
Accuracy : 0.7196816504001617
```

```
[ ] epochs_hist.history.keys()
```

```
dict_keys(['loss'])
```

```
[ ] plt.plot(epochs_hist.history['loss'])
    plt.title('Model Loss Progress During Training')
    plt.xlabel('Epoch')
    plt.ylabel('Training Loss')
    plt.legend(['Training Loss'])
```

```
<matplotlib.legend.Legend at 0x7f6849819610>
```



DECISION TREE AND RANDOM FOREST

```
[ ] from sklearn.tree import DecisionTreeRegressor
    DecisionTree_model = DecisionTreeRegressor()
    DecisionTree_model.fit(X_train,y_train)
```

```
DecisionTreeRegressor()
```

```
[ ] accuracy_DecisionTree = DecisionTree_model.score(X_test,y_test)
    accuracy_DecisionTree
```

```
0.47098342179659125
```

Random Forest Classification

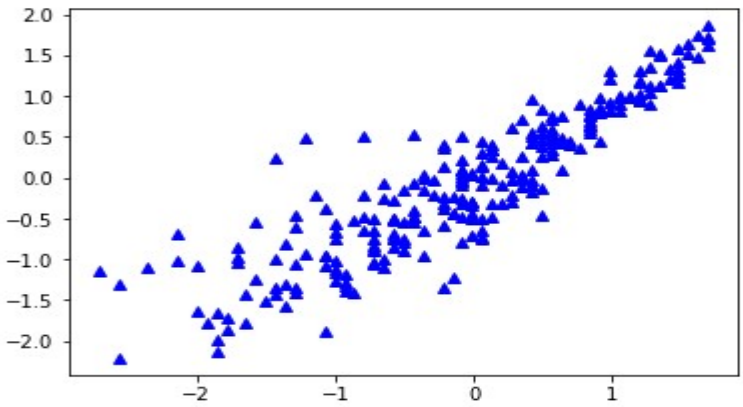
```
[ ] from sklearn.ensemble import RandomForestRegressor
    RandomForest_model = RandomForestRegressor(n_estimators=100,max_depth=10)
    RandomForest_model.fit(X_train,y_train)
```

```
[ ]  accuracy_RandomForest = RandomForest_model.score(X_test,y_test)
     accuracy_RandomForest
```

```
0.7714815134186287
```

```
y_predict = LinearRegression_model.predict(X_test)
plt.plot(y_test, y_predict, '^',color = 'b')
```

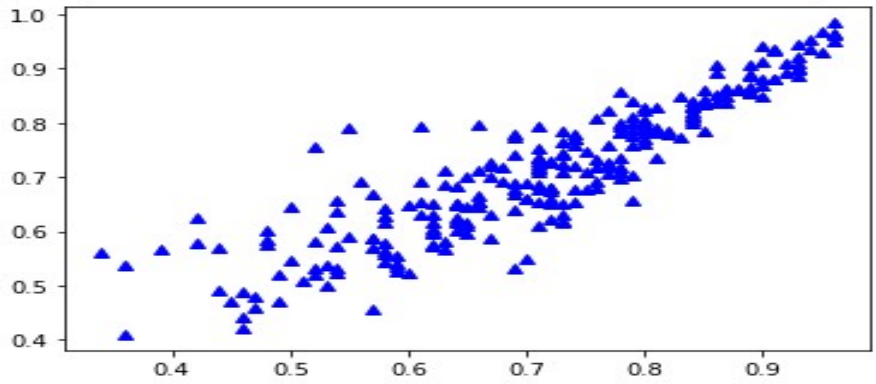
```
[<matplotlib.lines.Line2D at 0x7f6842868f50>]
```




```
[ ]  y_predict_orig = scalar_y.inverse_transform(y_predict)
     y_test_orig = scalar_y.inverse_transform(y_test)
```

```
[ ]  plt.plot(y_test_orig,y_predict_orig,'^',color = 'b')
```

```
[<matplotlib.lines.Line2D at 0x7f684285ba50>]
```




```
[ ]  k = X_test.shape[1]
     n = len(X_test)
     n
```

```
250
```

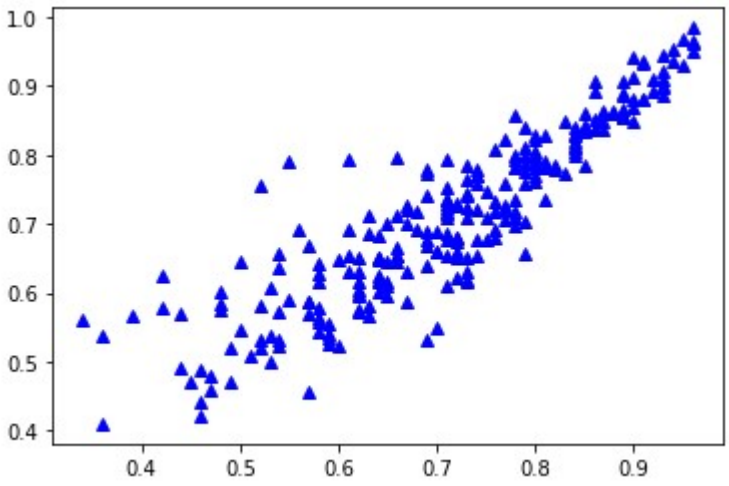
```
[ ]  from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
     from math import sqrt
     RMSE = float(format(np.sqrt(mean_squared_error(y_test_orig, y_predict_orig)),'.3f'))
     MSE = mean_squared_error(y_test_orig, y_predict_orig)
     MAE = mean_absolute_error(y_test_orig, y_predict_orig)
     r2 = r2_score(y_test_orig, y_predict_orig)
     adj_r2 = 1-(1-r2)*(n-1)/(n-k-1)
```

```
[ ]  print('RMSE =',RMSE, '\nMSE =',MSE, '\nMAE =',MAE, '\nR2 =', r2, '\nAdjusted R2 =',adj_r2)

     RMSE = 0.061
     MSE = 0.003697039133566072
     MAE = 0.043119079892106865
     R2 = 0.8060189050302291
     Adjusted R2 = 0.8004078816220126
```

Results

In predicting the chance of acceptance of a student we saw the highest accuracy in the Linear Regression algorithm 80.6% followed by Random Forest with 77.14% followed by ANN with 71.96% and the least accuracy was observed in decision tress which was 47.09%.




```
RMSE = 0.061
MSE = 0.003697039133566072
MAE = 0.043119079892106865
R2 = 0.8060189050302291
Adjusted R2 = 0.8004078816220126
```

RMSE- Root-mean-square deviation
MSE-mean_squared_error
MAE-mean_absolute_error
R2- r2_score

CONCLUSION

Every year, lakhs of students apply to colleges to continue/start their education. A lot of them are misguided and are not cautious with their approach, and this leads to them applying to the wrong colleges, which wastes a lot of time, effort and money. Through our project we would like to help such prospective students, to find better colleges. This would be detrimental to their future. It is very important that a candidate should apply to colleges that he/she has a good chance of getting into, instead of applying to colleges that they may never get into. Our prepared models work to a satisfactory level of accuracy, and may be of great assistance to such people. This is a project with good future scope, especially for students of our age group who want to pursue their higher education in their dream college.

Future Works

As part of future works we would like to improve on our accuracy of algorithms and also use new and better algorithms and make them work in ensemble, we would also try to include more attributes on calculating the chance of acceptance of a student. Also display the list of colleges a particular student should apply to based on his/her attributes. So we would display the colleges in a ranking manner to apply to base on their acceptance score.