MACHINE LEARNING PROJECT

PHASE 2

Balasani Dheeraj Redddy	AM.EN.U4CSE20115
Vanguru Vekata Varun Kumar Reddy	AM.EN.U4CSE20172
Chinthakuntla Purushottam Reddy	AM.EN.U4CSE20120
S N V V S Gowtham Tadavarthy	AM.EN.U4CSE20160
Joshua Wilson Philip	AM.EN.U4CSE20135
Maddipati Umesh Chandra	AM.EN.U4CSE20142

Problem statement

It is very expensive to apply for top universities around the world. Predicting the chances for scholar for a particular collage can save then a lot of money and time. We ran a different algorithm on previous year dataset to get best results. So that scholars can guess their capabilities before they apply.

2.

Datasets

- Brief on the datasets used in the project.
- At least 3 datasets should be chosen for the project.

This data set is about the students who applied to the universities and their chance of getting admitted.

The dataset is mainly:

GRE Scores (290 to 340)

The Graduate Record Examinations (GRE): is a the most popular test for graduate schools' admission, it consists of three sections : Analytical Writing, Verbal and Quantitative

TOEFL Scores: (92 to 120)

Test of English as a Foreign Language (TOEFL) is a very popular test for English language amongst universities worldwide, it is marked based on three sections: Reading, Listening, Speaking, and Writing, each one of them is out of 30, yielding a maximum score of 120 and a minimum of 0.

University Rating (1 to 5):

The rating of the university the student completed his undergraduate degree from

Statement of Purpose (1 to 5)

Statement of Purpose (SOP) is a letter written by the student himself to state his purpose and motivation for completing a graduate degree in addition to his goals while and after he completes his study. Many universities find this letter significant because it better describe the student from a personal perspective.

Letter of Recommendation Strength (1 to 5)

Letter of Recommendation (LOR) is a letter written by a person that knows the student and recommends that the university accept his admission, this person can be a professor in his undergraduate degree or a professional whom the student have worked with.

Undergraduate CGPA (6.8 to 9.92)

To Join in any foreign university the minimum cut-off range is 6.89 and the maximum cut-off range is 9.92

Research Experience (0 or 1)

To Join in any foreign university the minimum Research Experience must have 0 or 1

Chance of Admit (0.34 to 0.97)

If all the conditions are satisfied for a students then the chance of getting admissionis 0.34% to 0.97%

3. Prepare Data

Preprocessing Data:

- filling missing values with most frequent values using simpile imputer
- Normalizing and standardizing the data to ensure that all features are on a similar scale
- reduce the dimensionality of the data
- renamed some column names

```
In [127]: data = pd.read_csv("Admission_Predict.csv")
          data.shape

Out[127]: (400, 9)
```

- Statistical summary of all attributes

```
In [130]: data.describe()
```

Out[130]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	399.000000	399.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.619048	107.383459	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301	11.391182	6.053848	1.143728	1.006869	0.898478	0.597325	0.498362	0.142609
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.167500	0.000000	0.640000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.072500	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

- Breakdown of the data by the class variable.

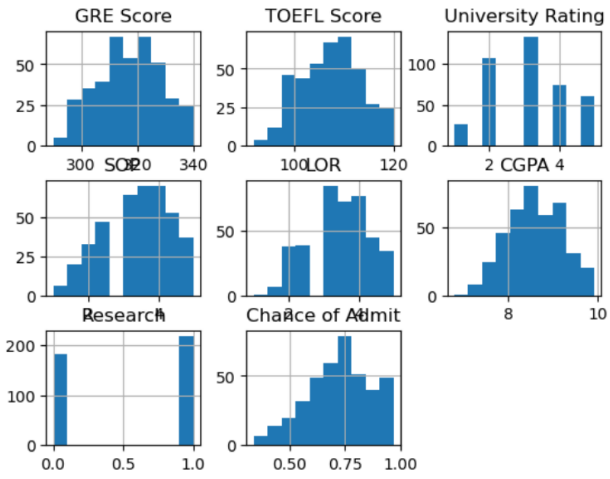
```
In [129]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         400 non-null    int64
 1   GRE Score          399 non-null    float64
 2   TOEFL Score        399 non-null    float64
 3   University Rating  400 non-null    int64
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64
 8   Chance of Admit    400 non-null    float64
dtypes: float64(6), int64(3)
memory usage: 28.2 KB
```

- Data Visualization:

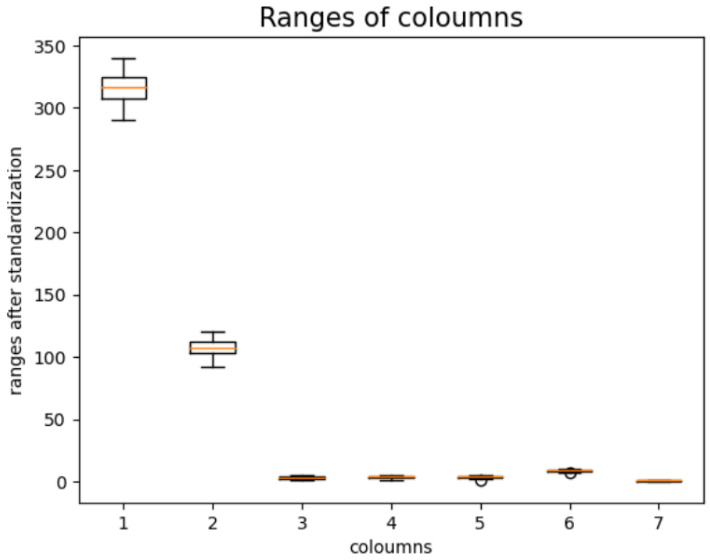
```
In [137]: import matplotlib.pyplot as plt
          data.hist()
```

```
Out[137]: array([[<AxesSubplot:title={'center':'GRE Score'}>,
                  <AxesSubplot:title={'center':'TOEFL Score'}>,
                  <AxesSubplot:title={'center':'University Rating'}>],
                 [<AxesSubplot:title={'center':'SOP'}>,
                  <AxesSubplot:title={'center':'LOR'}>,
                  <AxesSubplot:title={'center':'CGPA'}>],
                 [<AxesSubplot:title={'center':'Research'}>,
                  <AxesSubplot:title={'center':'Chance of Admit'}>, <AxesSubplot:>]],
                dtype=object)
```

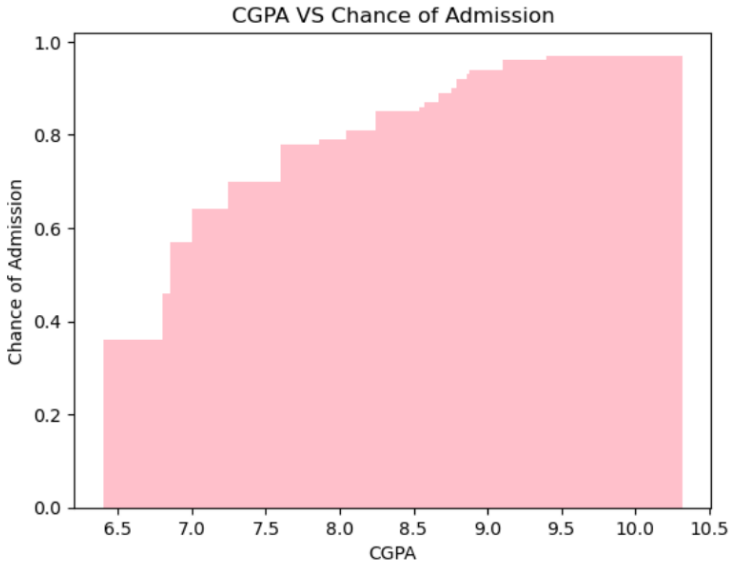


Visualize the data using various plots like scatterplot, histograms, box plot etc and record your interpretations with varying values

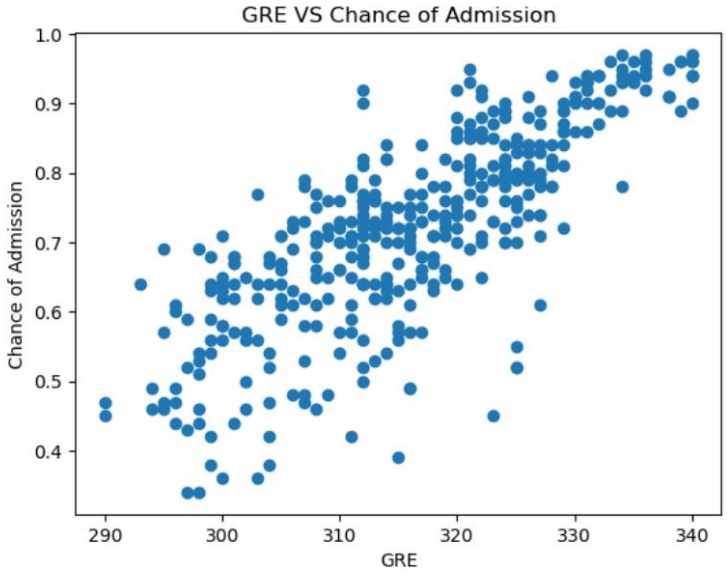
In [143]:
```
plt.boxplot(X[:,:])
plt.title('Ranges of coloumns',fontsize=15)
plt.xlabel("coloumns")
plt.ylabel("ranges after standardization")
plt.show()
```



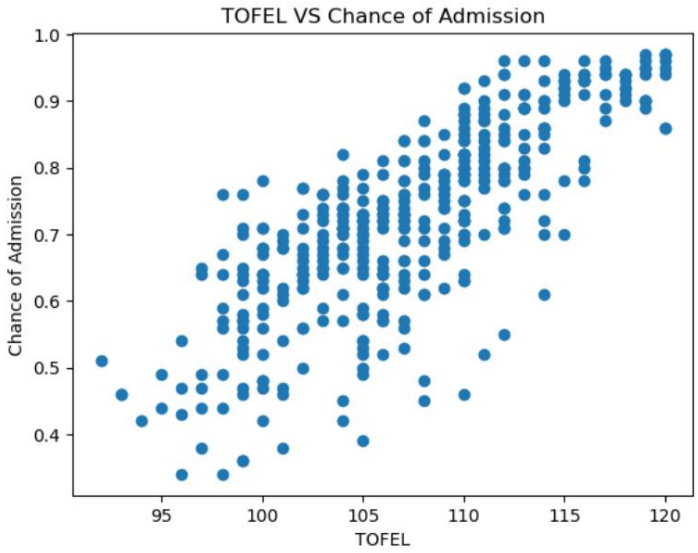
In [139]:
```
plt.bar(X[:,5], Y[:,0],color = "pink")
plt.title("CGPA VS Chance of Admission")
plt.xlabel("CGPA")
plt.ylabel("Chance of Admission")
plt.show()
```



In [141]:
```
plt.scatter(X[:,0], Y)
plt.title("GRE VS Chance of Admission")
plt.xlabel("GRE")
plt.ylabel("Chance of Admission")
plt.show()
```

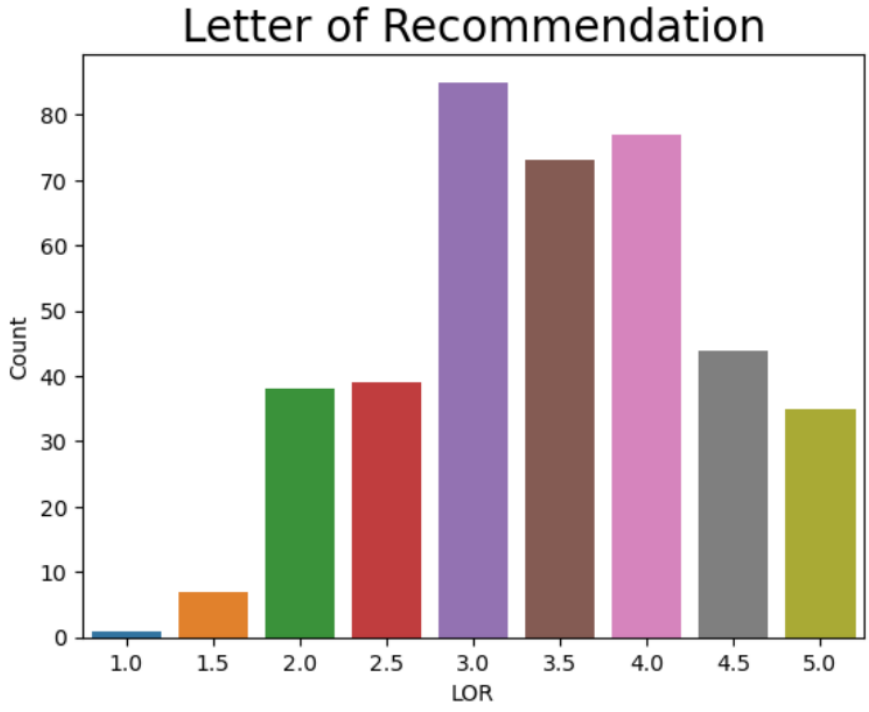


In [142]:
```
plt.scatter(X[:,1], Y)
plt.title("TOFEL VS Chance of Admission")
plt.xlabel("TOFEL")
plt.ylabel("Chance of Admission")
plt.show()
```

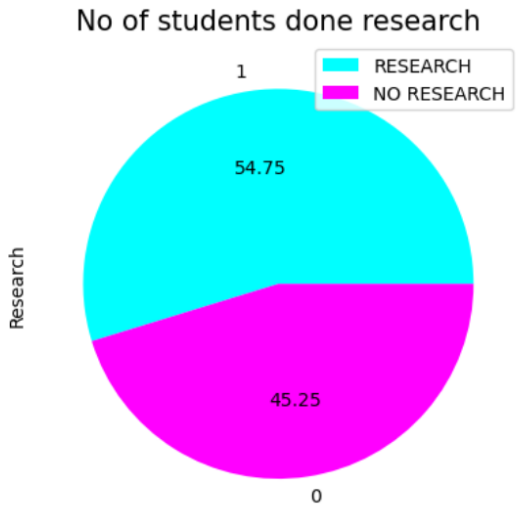


In [145]:
```
LOR = pd.DataFrame(data.groupby(['LOR']).count()['GRE Score'])
LOR.rename({'GRE Score':'Count'}, axis=1, inplace=True)
sns.barplot(LOR.index, LOR['Count']).set_title('Letter of Recommendation', size='20')
plt.show()
```

```
C:\Users\dheer\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
gs: x, y. From version 0.12, the only valid positional argument will be `data`, and pass
keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
In [144]: data['Research'].value_counts().plot(kind='pie',textprops={'color':'black'},autopct='%.2f',cmap='cool')
          plt.title('No of students done research',fontsize=15)
          plt.legend(['RESEARCH','NO RESEARCH'])
          plt.show()
```



4. Python packages

Brief on the python packages used for implementation of machine learning algorithms pertaining to your project.

NumPy: is a fundamental package for scientific computing with Python, providing support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays.

Pandas: is a high-performance, easy-to-use data structures and data analysis toolkit. It provides powerful data manipulation and analysis capabilities, such as indexing, reshaping, merging, and joining of data sets.

Scikit-Learn: is a popular machine learning library for Python, providing implementations of a wide range of algorithms for classification, regression, clustering, and more. It also includes many useful tools for model evaluation and selection, as well as for preprocessing and feature extraction.

Matplotlib: is a popular and widely used plotting library for Python. It provides a high-level interface for creating attractive and informative statistical graphics in Python. With Matplotlib, you can create line plots, scatter plots, histograms, bar charts, and many other types of plots using a simple, intuitive API. Additionally, Matplotlib integrates well with other popular Python libraries, such as Pandas and NumPy, making it easy to use with large, real-world datasets.

5. Learning Algorithms

In this project we have used the five ML Algorithms are:
➔Logistic Regression
➔Random Forest
➔Gaussian Navie Bayes
➔Support Vector Classifiers(SVCs)
➔K-nearest Neighnours(KNN)

Logistic Regression: it is a simple and widely used algorithm that is often used for binary classification tasks. It can be used to predict whether a student is likely to be accepted or rejected based on their scores and college ranking.

Random Forest: Random forests are an ensemble learning method that combines multiple decision trees to make more accurate predictions. They can be used in the same way as decision trees to predict a student's likelihood of acceptance based on their scores and college ranking.

Gaussian Navie Bayes: is a classification algorithm that is based on the idea of using Bayes' theorem to make predictions. It assumes that the features in the data are mutually independent and follows a Gaussian distribution. This allows the algorithm to make predictions based on the probabilities of different classes given the input data.

Support Vector Classifiers(SVCs): These are type of an algorithm that can be used for classification tasks. They work by finding the hyperplane in the feature space that maximally separates the different classes in the data. This allows the algorithm to make predictions based on the distance of new data points to the hyperplane. support vector classifier could potentially be a useful algorithm to consider. It could be trained on the data to predict a student's likelihood of acceptance based on their scores and college ranking

K-nearest Neighbors(KNN): is a classification algorithm that is based on the idea of using the "k" closest data points in the feature space to make predictions for new data. It works by calculating the distances between the new data point and the "k" nearest points in the training set, and then using these distances to determine the class of the new data point. KNN classifier could potentially be a useful algorithm to consider. It could be trained on the data to predict a student's likelihood of acceptance based on their scores and college ranking

Split your data into training, validation, and testing

Use k-fold cross validation to evaluate your ML algorithm

CROSS VALIDATION SCORES

```
In [174]: from sklearn.model_selection import cross_val_score

In [175]: cross_val_score(LogisticRegression(),x_train,y_train)#average=0.93125
Out[175]: array([0.921875, 0.890625, 0.984375, 0.921875, 0.9375  ])

In [176]: cross_val_score(SVC(),x_train,y_train)#average=0.928125
Out[176]: array([0.921875, 0.890625, 0.984375, 0.921875, 0.921875])

In [177]: cross_val_score(RandomForestClassifier(),x_train,y_train)#average=0.946875
Out[177]: array([0.921875, 0.90625 , 1.      , 0.9375  , 0.96875 ])

In [178]: cross_val_score(GaussianNB(),x_train,y_train)#average=
Out[178]: array([0.921875, 0.890625, 0.9375  , 0.875   , 0.9375  ])

In [179]: cross_val_score(KNeighborsClassifier(),x_train,y_train)#average=
Out[179]: array([0.9375  , 0.921875, 0.96875 , 0.921875, 0.90625 ])
```

Create models and estimate their accuracy on unseen data using the specified ML algorithms.

•Example: If Logistic regression, SVM, and KNN are used for classification, create models for different algorithms. Select the best model.

Logistic Regression

LogisticRegression

```
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression()
classifier.fit(x_train, y_train)
```

Random Forest

RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
RFC=RandomForestClassifier()
RFC.fit(x_train,y_train)
y_pred_RFC = RFC.predict(x_test)
print("score: ", RFC.score(x_test,y_test))
```

Gaussian Navie Bayes

Naive Bayes Classifiers

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train, y_train)
y_pred_gnb = gnb.predict(x_test)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy:",metrics.accuracy_score(y_test, y_pred_gnb))
```

Support Vector Classifiers(SVCs)

SVC

```
from sklearn.svm import SVC
svm = SVC(random_state = 1)
svm.fit(x_train,y_train)
y_pred_svm = svm.predict(x_test)
print("score: ", svm.score(x_test,y_test))
```

```
score:  0.9375
```

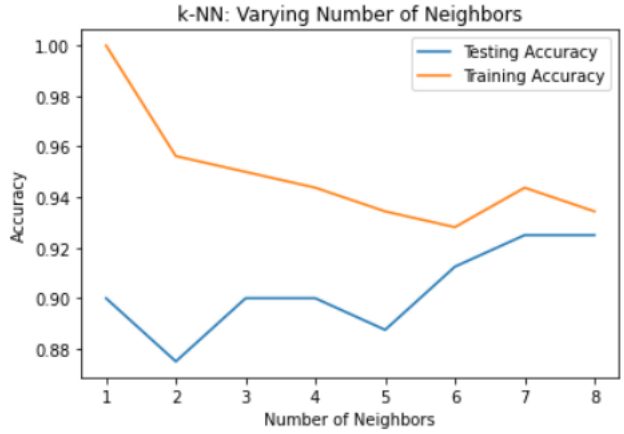
K-nearest Neighnours(KNN)

KNeighborsClassifier

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train, y_train)
y_pred_knn = knn.predict(x_test)
print("KNN model accuracy:",metrics.accuracy_score(y_test, y_pred_knn))
```

```
KNN model accuracy: 0.925
```

Plot accuracy for different values of k.



•Plot a comparison graph showing the accuracy comparison of various algorithms on each of your datasets.

accuracy comparison of various algorithms

```
models=['RandomForest','knn','svc','logistic_regression','naive_bayes']
accuracy=[metrics.accuracy_score(y_test, y_pred_RFC),metrics.accuracy_score(y_test, y_pred_knn),metrics.accuracy_score(y_test, y_pred_svm), metrics.ac

plt.title('accuracy comparison of various algorithms')
plt.plot(models, accuracy)
plt.legend()
plt.xlabel('MODELS')
plt.ylabel('Accuracy')
plt.show()
```