

*NAAN MUDHALVAN PROJECT BASED EXPERIENTIAL LEARNING*

***A PROJECT REPORT***

***ON***

***INTELLIGENT ADMISSIONS: THE FUTURE OF UNIVERSITY  
DECISION MAKING USING MACHINE LEARNING.***



***Department of Computer Applications***

***Arignar Anna Government Arts College (W)***

***(Affiliated to Thiruvalluvar University)***

***WALAJAPET – 632 513.***

***Submitted By:***

***Team ID: NM2023TMID22186***

***Team Leader :PRIYA MAHALAKSHMI P***

***Team member: PAVITHRA P***

***Team member: POOJA S***

***Team member: PRIYA P***

***Team member: RAJASHREE L***

## **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 position 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 examination 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 .*

*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 enlisted in America are from Asian nations like India, Pakistan, Srilanka, 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 position 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 accuentuation 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. Conformation is a dataset with 400 lines that contains 7 distinct autonomous factors which are:*

- *Graduate Record Exam I (GRE) score. The score will be out of 340 focuses.*
- *Trial of English as a Foreigner Language (TOEFL) score, which will be out of 120 focuses.*
- *University Rating (Uni.Rating) that demonstrates the Bachelor University positioning among different colleges. The score will be out of 5.*
- *Statement of direction (SOP) which is a record written to show the applicant's life, driven and the inspiration for the picked degree/college. The score will be out of 5 focuses.*
- *Letter of Recommendation Strength (LOR) which confirms the applicant proficient experience, fabricates validity, supports certainty and guarantees your ability. The score is out of 5 focuses.*
- *Undergraduate GPA (CGPA) out of 10.*

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

## **OVERVIEW:**

- *University admission is the process by which students are selected to attend a college or university.*
- *The process typically involves several steps, including submitting an application, taking entrance exams, and participating in interviews or other evaluations.*
- *Students are often worried about their chances of admission in university. The university admission process can be demanding, but by being well-informed, prepared, and organized, students can increase their chances of being admitted to the university of their choice.*
- *The aim of this project is to help students in short listing universities with their profiles. Machine learning algorithms are then used to train a model on this data, which can be used to predict the chances of future applicants being admitted. With this project, students can make more informed decisions about which universities to apply to, and universities can make more efficient use of their resources by focusing on the most promising applicants*
- *The predicted output gives them a fair idea about their admission chances in a particular university. This analysis should also help students who are currently preparing or will be preparing to get a better idea.*

## **PURPOSE:**

- ❖ *It helps student to make right decisions for choosing their college. In which students can register with their personal as well as marks details to prediction the admission in colleges and the administrator can allot the seats for the students.*
- ❖ *Identifying fraud in admissions applications.*
- ❖ *Predicting application acceptances by students and thereby identify was to improve acceptance rates and to improve capacity planning.*
- ❖ *Document recognition, validation and approval.*
- ❖ *Scanning for and identifying policy violations.*
- ❖ *Argument your admissions process by making recommendations regarding specific applications and flagging applications that need follow up for various reasons.*
- ❖ *Chat bots to take inbound inquires, answer question related to ATAR scores and helping to build course schedules.*

## **PROBLEM DEFINITION & DESIGNN THINKING:**

### ***Empathy Map:***

*An empathy map is a template that organizes a user's behaviors and feelings to create a sense of empathy between the user and your team. The empathy map represents a principle user and helps teams better understand their motivations, concerns, and user experiences.*



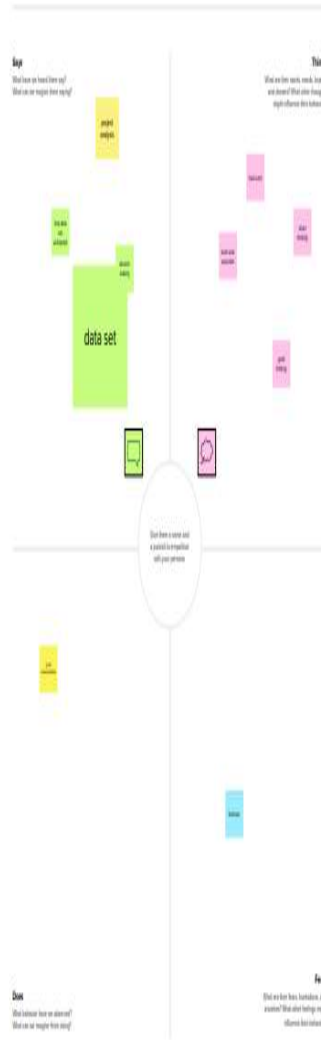
## Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

[View template details](#)

### Build empathy

The information you add here should be representative of the observations and research you've done about your users.



Need some inspiration?

See a broader view of the template to better your work.

[Get more](#)



## IDEATION & BRAINSTORMING MAP:

*Brainstroming is a group problem solving method that involves the spontaneous contribution of creative ideas and solutions.*

### We" Questions

rou created  
1 by selecting  
on what seems  
on in the areas

### Brainstorm solo

Have each participant begin in the "solo brainstorm space" by silently brainstorming ideas and placing them into the template. This "silent-storming" avoids group-think and creates an inclusive environment for introverts and extroverts alike. Set a time limit. Encourage people to go for quantity.

🕒 10 minutes

### Brainstorm as a group

Have everyone move their ide template and have the team si group them by thematic topics that arise. Encourage "Yes, and the way.

🕒 15 minutes

Rajashree L



Priya Mahalakshmi P



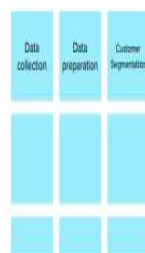
Pavithra P



Pooja S



Priya P



Person 6



Person 7



Person 8



## **Define problem:**

*In this milestone, we will see the define problem/Problem Understanding.*

### **1)Specify the Business Problem:**

- *University admission is the process by which students are selected to attend a college or university.*
- *The process involves several steps, including submitting an application, taking entrance exam, and participating in interviews or other evaluations.*
- *The aim of this project is to help students in short listing universities with their profiles.*
- *The predicted output gives them a fair idea about their admission chances in a particular university.*

### **2)Business Requirements:**

- *The business requirements for a machine learning model to predict chances of student admission in the university.*
- *A project aims to predict the chances of a student getting admitted to a particular university based on certain factors.*
- *The business value of this project is that it will help students make more informed decisions about which universities to apply.*

### **3)Literature Survey:**

- *The university chances of admission project is a well-researched topic in the field of education and machine learning*



- *Many studies have been conducted to predict university admission using different machine learning techniques.*

- *The study found that random forest algorithm performed the best with an accuracy of 85.5%*

- *Study found that the KNN algorithm performed well with an accuracy of 81.2%.*

#### ***4)Social or Business Impact:***

##### ***Social Impact:***

- *The ability to accurately predict the chances of university admission and gaining access to higher education.*

##### ***Business Impact:***

- *Using machine learning models to predict university admission and also lead to increase in revenue for universities.*

- *The company provides the prediction service.*

#### ***Data Collection & Preparation:***

##### ***Collect the dataset:***

- *There are many popular open sources for collecting the data.*
- *In this project we have used .csv data. This data is downloaded from kaggle.com.*

*Next we open google colab.*

##### ***Importing the libraries:***

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

*Read the Dataset:*

*In pandas we have a function called `read_csv()` to read the dataset. As a parameter we have to give the directory of the csv file.*

```
data = pd.read_csv('/content/Admission_Predict.csv')
```

*Handling missing values:*

*Let's find the first 5 rows in dataset.*

```
data.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 find the datatype of our dataset first.

A screenshot of a Jupyter Notebook interface. On the left, a file explorer shows a folder named 'sample\_data' containing a file 'Admission\_Predict.csv'. The main area shows a code cell with the command `data.info()`. The output displays the DataFrame's structure: 400 entries, 8 columns. The columns are GRE Score, TOEFL Score, University Rating, SOP, LOR, CGPA, Research, and Chance of Admit. Data types are int64 for GRE Score, TOEFL Score, University Rating, and Research; float64 for SOP, LOR, CGPA, and Chance of Admit. Memory usage is 25.1 KB.

```
[ ] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   GRE Score              400 non-null   int64  
1   TOEFL Score            400 non-null   int64  
2   University Rating      400 non-null   int64  
3   SOP                    400 non-null   float64 
4   LOR                    400 non-null   float64 
5   CGPA                   400 non-null   float64 
6   Research               400 non-null   int64  
7   Chance of Admit        400 non-null   float64 
dtypes: float64(4), int64(4)
memory usage: 25.1 KB
```

Let us rename the column, in python have a inbuilt function `rename()`. We can easily rename the column names.

A screenshot of a Jupyter Notebook code cell. The code is `data.rename(columns = {'Chance of Admit ':'Chance of Admit'})`. The output shows the command was executed successfully.

```
[3] data.rename(columns = {'Chance of Admit ':'Chance of Admit'})
```

## Exploratory Data Analysis:

### **Descriptive statistical:**

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called `describe`. With this `describe` function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

A screenshot of a Jupyter Notebook code cell. The code is `data.describe()`. The output is a summary table of the dataset's statistics.

```
data.describe()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	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	8.170000	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	9.062500	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

### **Visual analysis:**

- Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data.

- It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.
- *Univariate analysis* In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as `distplot` and `countplot`.
- The Seaborn package provides a wonderful function `distplot`. With the help of `distplot`, we can find the distribution of the feature. To make multiple graphs in a single plot, we use `subplot`.

```
sns.distplot(data['GRE Score'])
```

`<ipython-input-5-64e93544a305>:1: UserWarning:`

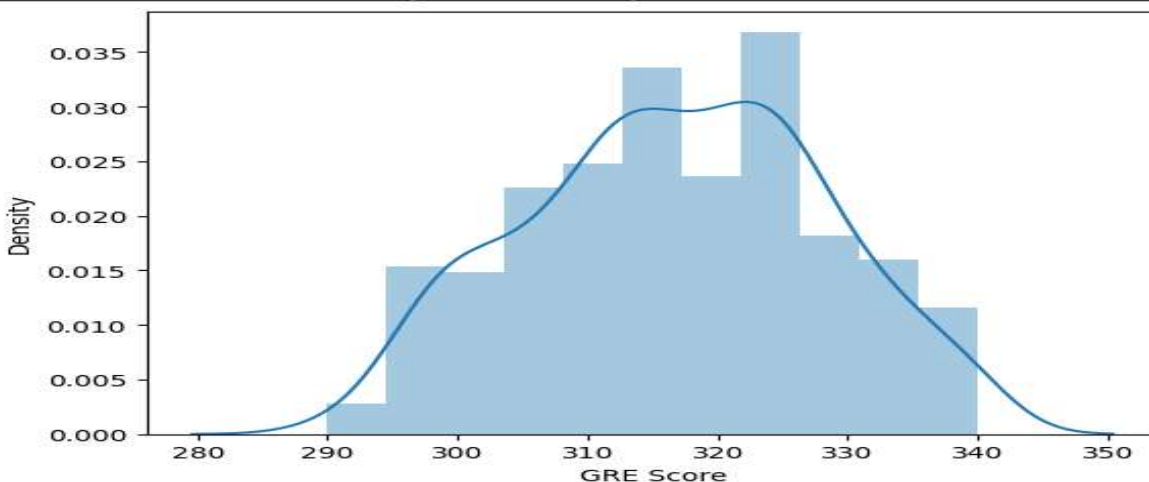
``distplot` is a deprecated function and will be removed in seaborn v0.14.0.`

Please adapt your code to use either ``displot`` (a figure-level function with similar flexibility) or ``histplot`` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(data['GRE Score'])
```

`<Axes: xlabel='GRE Score', ylabel='Density'>`



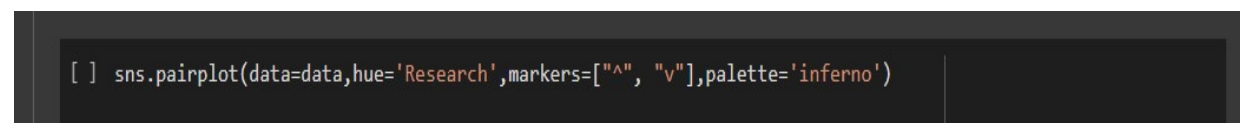
## ***Bivariate analysis:***

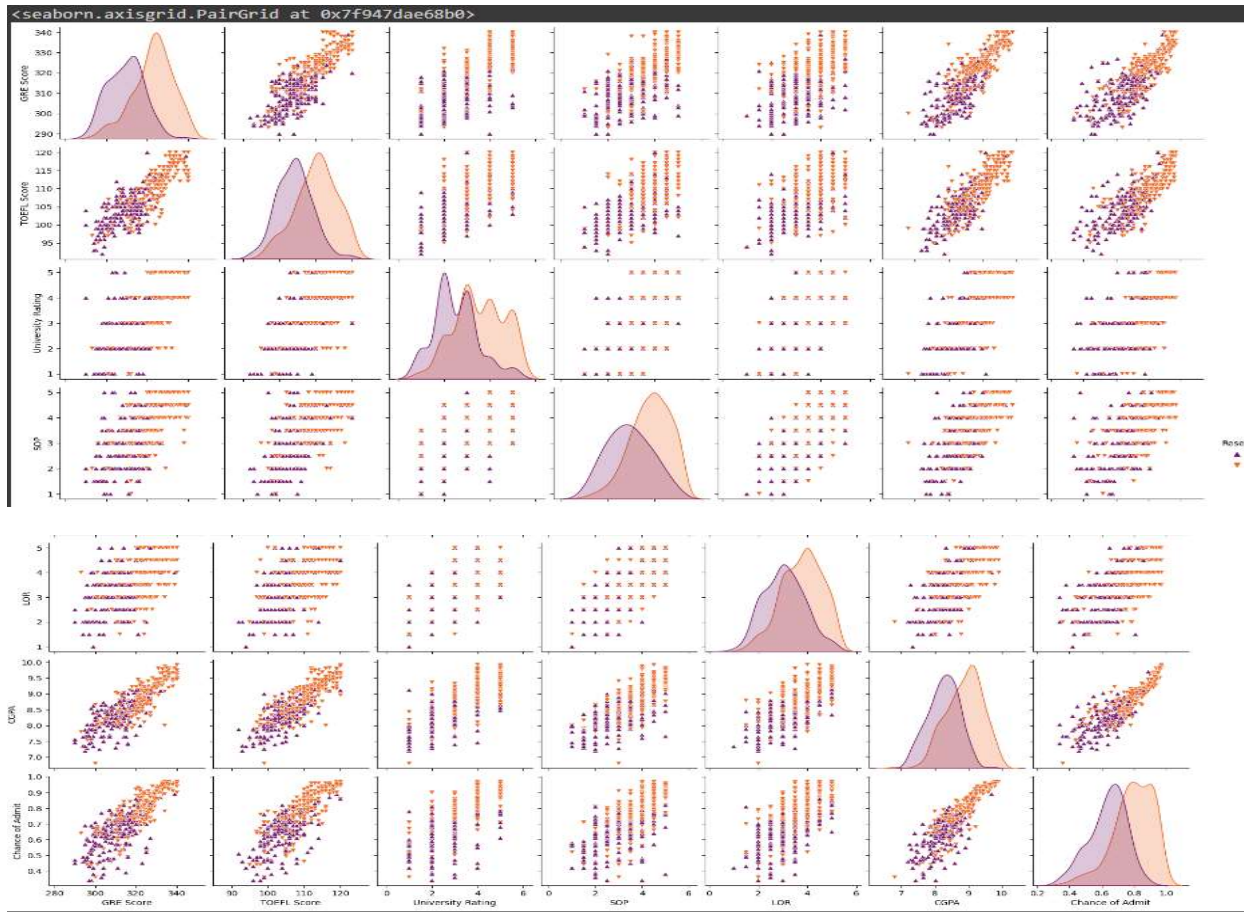
*We see that the output variable "Chance of Admit" depends on CGPA, GRE, TOEFEL. The columns SOP, LOR and Research have less impact on university admission.*



## ***Pair Plot:***

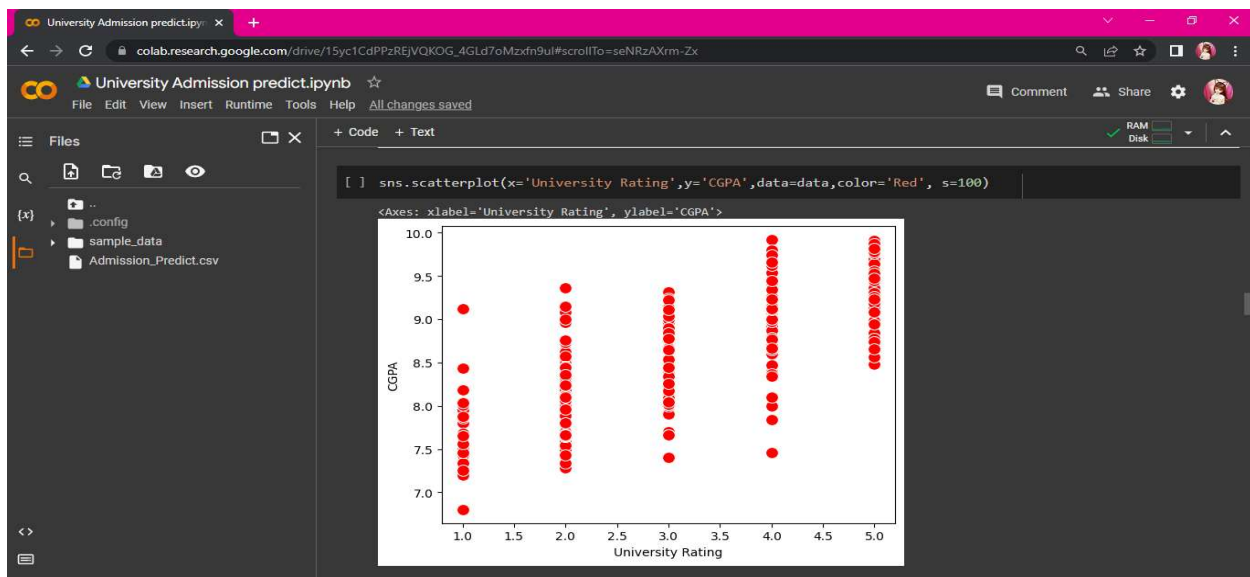
*Plot pairwise relationships in a dataset.*





Pair plot usually gives pair wise relationships of the columns in the dataset

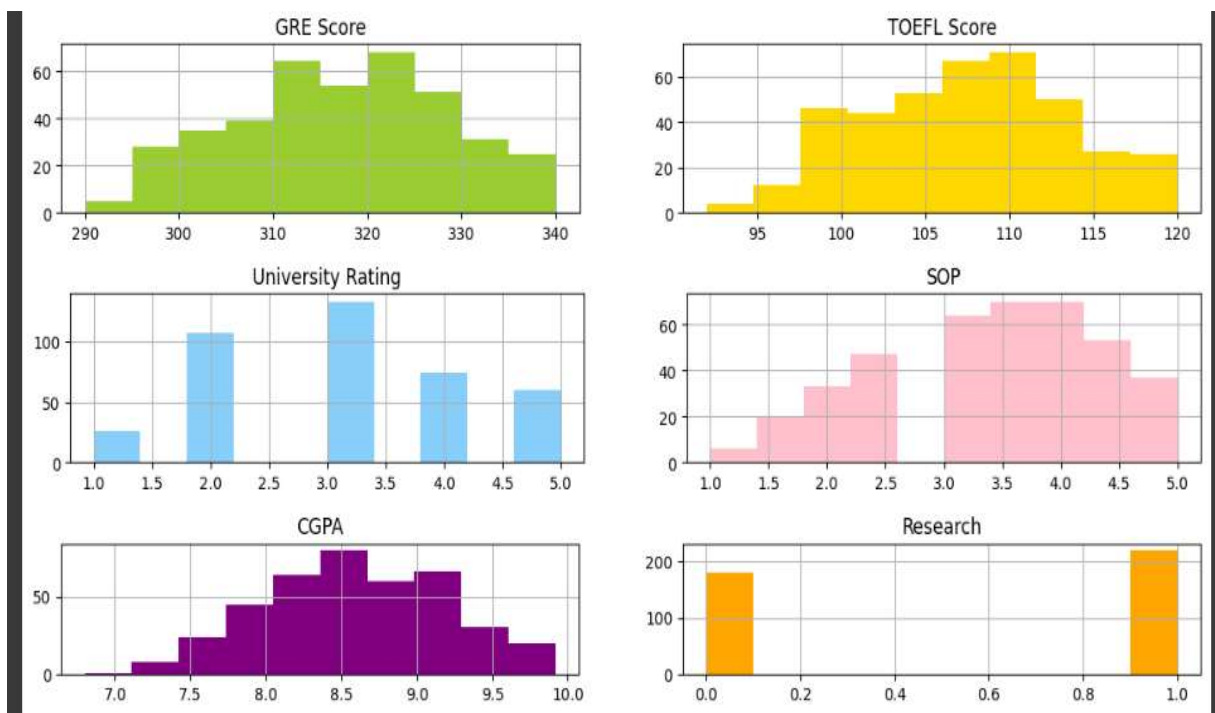
1. GRE score TOEFL score and CGPA all are linearly related to each other.
2. Students in research score high in TOEFL and GRE compared to non research candidates.





*Visualizing the Each column in a dataset using subplot( ).*

```
[ ] category = ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'CGPA', 'Research'],  
color = ['yellowgreen', 'gold', 'lightskyblue', 'pink', 'purple', 'orange']  
start = True  
for i in np.arange(3):  
    fig = plt.figure(figsize=(14,6))  
    plt.subplot2grid((3,2),(i,0))  
    data[category[2*i]].hist(color=color[2*i],bins=10)  
    plt.title(category[2*i])  
    plt.subplot2grid((3,2),(i,1))  
    data[category[2*i+1]].hist(color=color[2*i+1],bins=10)  
    plt.title(category[2*i+1])  
  
plt.subplots_adjust(hspace = 0.7, wspace = 0.2)  
plt.show()
```



*Scaling the Data Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction Models such as KNN, Logistic regression need scaled data, as they follow distance based method and Gradient Descent concept.*

```

[ ] from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
x=sc.fit_transform(x)
x

array([[0.94      , 0.92857143, 0.75      , ..., 0.875      , 0.91346154,
        1.        ],
       [0.68      , 0.53571429, 0.75      , ..., 0.875      , 0.66346154,
        1.        ],
       [0.52      , 0.42857143, 0.5       , ..., 0.625      , 0.38461538,
        1.        ],
       ...,
       [0.8       , 0.85714286, 0.75      , ..., 0.875      , 0.84935897,
        1.        ],
       [0.44      , 0.39285714, 0.5       , ..., 0.75       , 0.63461538,
        0.        ],
       [0.86      , 0.89285714, 0.75      , ..., 0.75       , 0.91666667,
        1.        ]])

```

*Splitting data into x and y Now let's split the Dataset into x and y.*

```

[ ] x=data.iloc[:,0:-1].values
x

array([[337. , 118. , 4. , ..., 4.5 , 9.65, 1. ],
       [324. , 107. , 4. , ..., 4.5 , 8.87, 1. ],
       [316. , 104. , 3. , ..., 3.5 , 8. , 1. ],
       ...,
       [330. , 116. , 4. , ..., 4.5 , 9.45, 1. ],
       [312. , 103. , 3. , ..., 4. , 8.78, 0. ],
       [333. , 117. , 4. , ..., 4. , 9.66, 1. ]])

```

```

[ ] y=data['Chance of Admit'].values
y

array([0.92, 0.76, 0.72, 0.8 , 0.65, 0.9 , 0.75, 0.68, 0.5 , 0.45, 0.52,
       0.84, 0.78, 0.62, 0.61, 0.54, 0.66, 0.65, 0.63, 0.62, 0.64, 0.7 ,
       0.94, 0.95, 0.97, 0.94, 0.76, 0.44, 0.46, 0.54, 0.65, 0.74, 0.91,
       0.9 , 0.94, 0.88, 0.64, 0.58, 0.52, 0.48, 0.46, 0.49, 0.53, 0.87,
       0.91, 0.88, 0.86, 0.89, 0.82, 0.78, 0.76, 0.56, 0.78, 0.72, 0.7 ,
       0.64, 0.64, 0.46, 0.36, 0.42, 0.48, 0.47, 0.54, 0.56, 0.52, 0.55,
       0.61, 0.57, 0.68, 0.78, 0.94, 0.96, 0.93, 0.84, 0.74, 0.72, 0.74,
       0.64, 0.44, 0.46, 0.5 , 0.96, 0.92, 0.92, 0.94, 0.76, 0.72, 0.66,
       0.64, 0.74, 0.64, 0.38, 0.34, 0.44, 0.36, 0.42, 0.48, 0.86, 0.9 ,
       0.79, 0.71, 0.64, 0.62, 0.57, 0.74, 0.69, 0.87, 0.91, 0.93, 0.68,
       0.61, 0.69, 0.62, 0.72, 0.59, 0.66, 0.56, 0.45, 0.47, 0.71, 0.94,
       0.94, 0.57, 0.61, 0.57, 0.64, 0.85, 0.78, 0.84, 0.92, 0.96, 0.77,
       0.71, 0.79, 0.89, 0.82, 0.76, 0.71, 0.8 , 0.78, 0.84, 0.9 , 0.92,
       0.97, 0.8 , 0.81, 0.75, 0.83, 0.96, 0.79, 0.93, 0.94, 0.86, 0.79,
       0.8 , 0.77, 0.7 , 0.65, 0.61, 0.52, 0.57, 0.53, 0.67, 0.68, 0.81,
       0.78, 0.65, 0.64, 0.64, 0.65, 0.68, 0.89, 0.86, 0.89, 0.87, 0.85,
       0.9 , 0.82, 0.72, 0.73, 0.71, 0.71, 0.68, 0.75, 0.72, 0.89, 0.84,
       0.93, 0.93, 0.88, 0.9 , 0.87, 0.86, 0.94, 0.77, 0.78, 0.73, 0.73,
       0.7 , 0.72, 0.73, 0.72, 0.97, 0.97, 0.69, 0.57, 0.63, 0.66, 0.64,
       0.68, 0.79, 0.82, 0.95, 0.96, 0.94, 0.93, 0.91, 0.85, 0.84, 0.74,
       0.76, 0.75, 0.76, 0.71, 0.67, 0.61, 0.63, 0.64, 0.71, 0.82, 0.73,
       0.74, 0.69, 0.64, 0.91, 0.88, 0.85, 0.86, 0.7 , 0.59, 0.6 , 0.65,
       0.7 , 0.76, 0.63, 0.81, 0.72, 0.71, 0.8 , 0.77, 0.74, 0.7 , 0.71,
       0.93, 0.85, 0.79, 0.76, 0.78, 0.77, 0.9 , 0.87, 0.71, 0.7 , 0.7 ,
       0.75, 0.71, 0.72, 0.73, 0.83, 0.77, 0.72, 0.54, 0.49, 0.52, 0.58,
       0.78, 0.89, 0.7 , 0.66, 0.67, 0.68, 0.8 , 0.81, 0.8 , 0.94, 0.93,

```





*A LogisticRegression algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.*

```
[x] [ ] def logreg(x_train,x_test,y_train,y_test):  
      lr = LogisticRegression(random_state=0)  
      lr.fit(x_train,y_train)  
      y_lr_tr = lr.predict(x_train)  
      print(accuracy_score(y_lr_tr,y_train))  
      yPred_lr = lr.predict(x_test)  
      print(accuracy_score(yPred_lr,y_test))  
      print("****Logistic Regression****")  
      print("confusion_Matrix")  
      print(confusion_matrix(y_test,yPred_lr))  
      print("Classification Report")  
      print(classification_report(y_test,yPred_lr))
```

```
[ ] logreg(x_train,x_test,y_train,y_test)

0.934375
0.875
***Logistic Regression***
confusion_Matrix
[[ 0 10]
 [ 0 70]]
Classification Report
      precision    recall  f1-score   support

 False      0.00      0.00      0.00        10
  True      0.88      1.00      0.93        70

 accuracy      0.44
 macro avg      0.44      0.50      0.47        80
weighted avg      0.77      0.88      0.82        80

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
warn_prf(average, modifier, msg_start, len(result))
```

```
[ ] lr = LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
print("Predicting on test values")
lr_pred = lr.predict(x_test)
print("output is: ",lr_pred)
print("Predicting on random input")
lr_pred_own = lr.predict(sc.transform([[337,118,4,4.5,4.5,9.65,1]]))
print("output is: ",lr_pred_own)

Predicting on test values
output is: [ True True True True True True True True True True True True True True True True
 True True True True True True True True True True True True True True True True
 True True True True True True True True True True True True True True True True
 True True True True True True True True True True True True True True True True
 True True True True True True True True True True True True True True True True
 True True True True True True True True True True True True True True True True]

Predicting on random input
output is: [ True]
```

## 2) *DecisionTree:*

```
[ ] def decisionTree(x_train,x_test,y_train,y_test):
    dtc = DecisionTreeClassifier(criterions="entropy",random_state=0)
    dtc.fit(x_train,y_train)
    y_dt_tr = dtc.predict(x_train)
    print(accuracy_score(y_dt_tr,y_train))
    yPred_dt = dtc.predict(x_test)
    print(accuracy_score(yPred_dt,y_test))
    print("****Decision Tree****")
    print("confusion_Matrix")
    print(confusion_matrix(y_test,yPred_dt))
    print("Classification Report")
    print(classification_report)
```

```
[ ] dtc =DecisionTreeClassifier(criterion="entropy",random_state=0)
dtc.fit(x_train,y_train)
print("Predicting on test values")
dtc_pred =dtc.predict(x_test)
print("output is:",dtc_pred)
print("Predicting on random input")
dtc_pred_own = dtc.predict(sc.transform([[337,118,4,4.5,4.5,9.65,1]]))
print("output is: ",dtc_pred_own)

Predicting on test values
output is: [ True  True  True  True  True  True  True  True False  True  True  True  True
 False  True  True  True  True False  True  True False  True  True  True
 True  True  True  True  True  True  True  True False  True  True
 True  True  True  True  True  True  True  True True  True  True  True
 True False  True False False  True  True  True  True True  True  True
 True  True  True  True  True  True  True  True]
Predicting on random input
output is: [ True]
```

### 3) RandomForest:

```
[ ] def RandomForest(x_train,x_test,y_train,y_test):
    rf = RandomForestClassifier(criterion= "entropy",n_estimators=10,random_state=0)
    rf.fit(x_train,y_train)
    y_rf_tr = rf.predict(x_train)
    print(accuracy_score(y_rf_tr,y_train))
    yPred_rf = rf.predict(x_test)
    print(accuracy_score(yPred_rf,y_test))
    print("***Random Forest***")
    print("confusion_Matrix")
    print(confusion_matrix(y_test,yPred_rf))
    print("Classification Report")
    print(classification_report(y_test,yPred_rf))
```

```
[ ] RandomForest(x_train,x_test,y_train,y_test)

0.996875
0.9
***Random Forest***
confusion_Matrix
[[ 2  8]
 [ 0 70]]
Classification Report
precision    recall  f1-score   support

   False      1.00      0.20      0.33        10
    True       0.90      1.00      0.95        70

 accuracy      0.95      0.60      0.90        80
 macro avg       0.95      0.60      0.64        80
weighted avg       0.91      0.90      0.87        80
```

### ANN model:

*Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epochs.*

```
ANN Model

[ ] import keras
    from keras.models import Sequential
    from keras.layers import Dense

[ ] classifier = Sequential()

[ ] classifier.add(Dense(units=7, activation= 'relu',input_dim=7))

[ ] classifier.add(Dense(units=7, activation= 'relu'))

[ ] classifier.add(Dense(units=1, activation= 'linear'))

[ ] classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

## Testing the model:

*In ANN we first have to save the model to the test the inputs.*

```
[ ] classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

[ ] model = classifier.fit(x_train, y_train, batch_size=10,validation_split=0.33, epochs=20)

Epoch 1/20
22/22 [=====] - 1s 21ms/step - loss: 2.7661 - accuracy: 0.1215 - val_loss: 1.4374 - val_accuracy: 0.2170
Epoch 2/20
22/22 [=====] - 0s 5ms/step - loss: 1.4508 - accuracy: 0.3598 - val_loss: 0.9712 - val_accuracy: 0.5943
Epoch 3/20
22/22 [=====] - 0s 5ms/step - loss: 0.9475 - accuracy: 0.5374 - val_loss: 0.7155 - val_accuracy: 0.6415
Epoch 4/20
22/22 [=====] - 0s 5ms/step - loss: 0.7291 - accuracy: 0.5607 - val_loss: 0.5631 - val_accuracy: 0.6415
Epoch 5/20
22/22 [=====] - 0s 6ms/step - loss: 0.5854 - accuracy: 0.5701 - val_loss: 0.4672 - val_accuracy: 0.6604
Epoch 6/20
22/22 [=====] - 0s 7ms/step - loss: 0.4882 - accuracy: 0.5981 - val_loss: 0.5321 - val_accuracy: 0.6887
Epoch 7/20
22/22 [=====] - 0s 5ms/step - loss: 0.4231 - accuracy: 0.6636 - val_loss: 0.5997 - val_accuracy: 0.7264
Epoch 8/20
22/22 [=====] - 0s 5ms/step - loss: 0.4249 - accuracy: 0.7523 - val_loss: 0.5620 - val_accuracy: 0.8019
Epoch 9/20
22/22 [=====] - 0s 5ms/step - loss: 0.3742 - accuracy: 0.8318 - val_loss: 0.5348 - val_accuracy: 0.8962
Epoch 10/20
22/22 [=====] - 0s 7ms/step - loss: 0.3297 - accuracy: 0.9019 - val_loss: 0.5426 - val_accuracy: 0.9245
Epoch 11/20
22/22 [=====] - 0s 5ms/step - loss: 0.2917 - accuracy: 0.9159 - val_loss: 0.6137 - val_accuracy: 0.9057
Epoch 12/20
22/22 [=====] - 0s 5ms/step - loss: 0.2596 - accuracy: 0.9533 - val_loss: 0.7005 - val_accuracy: 0.9245
Epoch 13/20
22/22 [=====] - 0s 5ms/step - loss: 0.2373 - accuracy: 0.9439 - val_loss: 0.6832 - val_accuracy: 0.9245
```

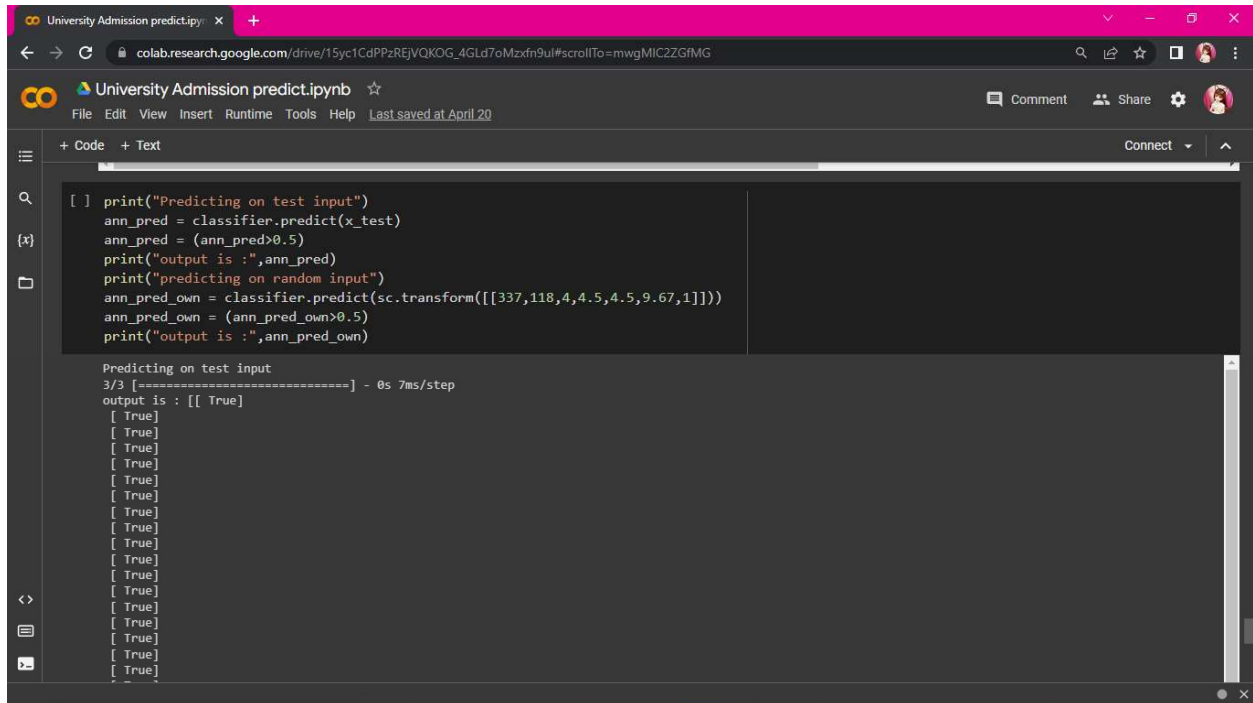
```
ann_pred = classifier.predict(x_test)
[ ] ann_pred = (ann_pred>0.5)
print(accuracy_score(ann_pred,y_test))
print("***ANN Model***")
print("Confusion_Matrix")
print(confusion_matrix(y_test,ann_pred))
print("Classification Report")
print(classification_report(y_test,ann_pred))

3/3 [=====] - 0s 3ms/step
0.875
***ANN Model***
Confusion_Matrix
[[ 0 10]
 [ 0 70]]
Classification Report
              precision    recall  f1-score   support

   False       0.00        0.00        0.00        10
    True       0.88        1.00        0.93        70

 accuracy       0.44        0.50        0.47        80
 macro avg       0.44        0.50        0.47        80
 weighted avg       0.77        0.88        0.82        80

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
_warn_prf(average, modifier, msg_start, len(result))
```



### **Model Deployment Activity:**

***1) Save the best model:***

*Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving it's weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.*

```
[ ] pickle.dump(lr,open('university.pkl','wb'))
```

**RESULT:**

- *Open anaconda prompt from the start menu.*
- *Navigate to the folder where our python script is.*
- *Now type “python app.py” command.*
- *Navigate to the localhost where we can view our web page.*
- *Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.*

127.0.0.1:5000

# UNIVERSITY ADMISSION PREDICTION SYSTEM

Enter your details and get probability of your admission

Enter GRE Score

Enter TOEFL Score

Select University no

- ☐ 1
- ☐ 2
- ☐ 3
- ☒ 4
- ☐ 5


Enter SOP

Enter LOR

Enter CGPA

Research

- ☐ Research
- ☒ No Research




*Now, click the predict button from the banner and will get redirected to the prediction page.*

Predicting Chance of Admission

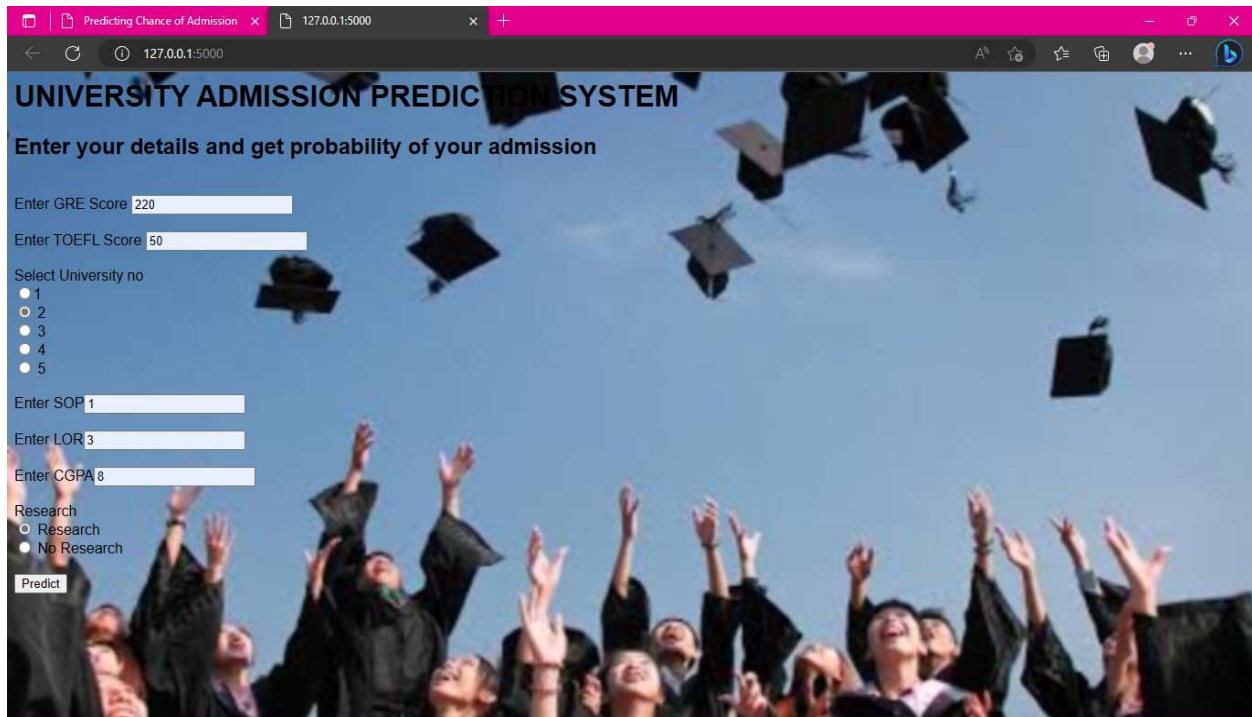
A MACHINE LEARNING WEB APP using Flask.

Prediction: You have a chance of getting admission 🎉





*Again run the web browser, give the different values,*



The screenshot shows a web browser window with the title "Predicting Chance of Admission". The address bar shows "127.0.0.1:5000". The page content is titled "UNIVERSITY ADMISSION PREDICTION SYSTEM" and "Enter your details and get probability of your admission". The form includes the following fields and options:

- Enter GRE Score: 220
- Enter TOEFL Score: 50
- Select University no:
  - ☐ 1
  - ☐ 2
  - ☐ 3
  - ☐ 4
  - ☐ 5
- Enter SOP: 1
- Enter LOR: 3
- Enter CGPA: 8
- Research:
  - ☐ Research
  - ☒ No Research
- Predict button

*Now, we will get the predicted result after clicking onto the predict button.*



### Predicting No Chance of Admission

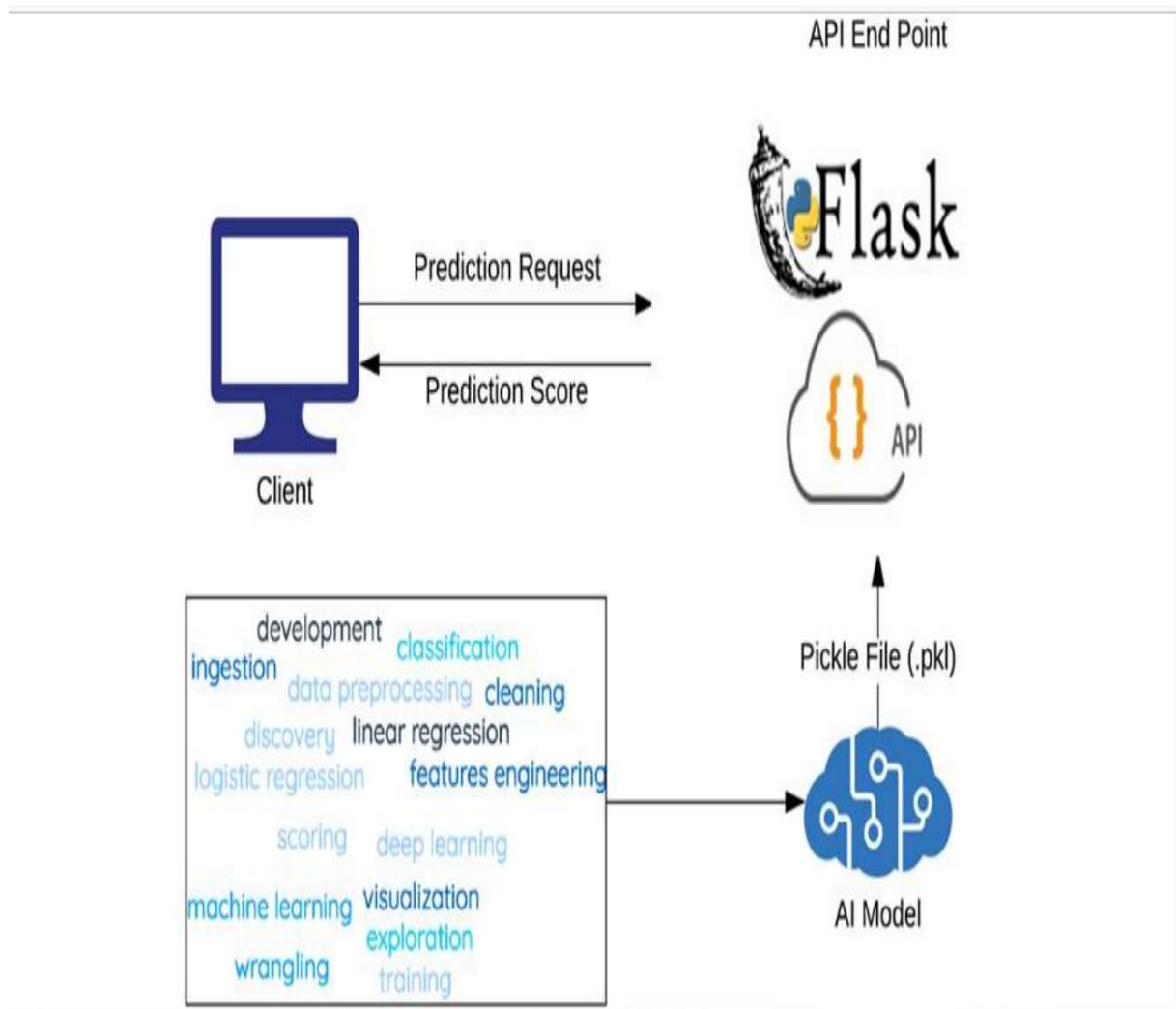
A MACHINE LEARNING WEB APP using Flask.

Prediction: **You Dont have a chance of getting admission** 📄



## **PROPOSED SYSTEM:**

*The proposed system consist of four regression models. Out of those we use Linear Regression using Dimensionality Reduction which is also a high accurate model. A user interface is provided through which an actor can interact with the system. The algorithm with improved accuracy will act as a backend for the user interface. Whenever any actor (Student/Consultancy) provides the data to the user interface it will show the result of Chance of Admission which is ranging 0 to 1.*



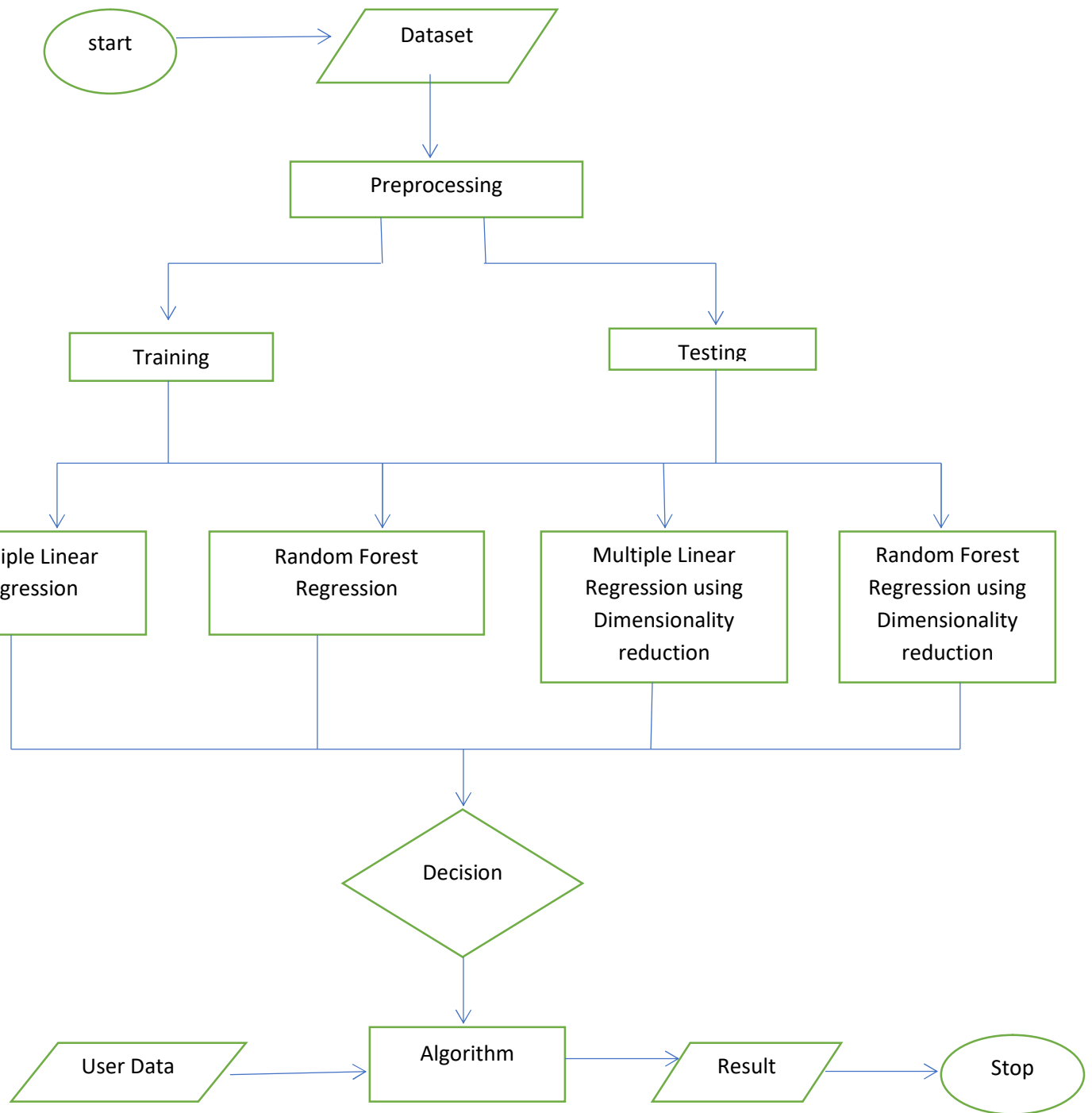


***User Manual There are several steps for using this:***

- 1. Initially the user has to open our website and provide all the requested values.*
- 2. The user has to give his GRE score within the range of 270 to 340.*
- 3. The user has to give his TOEFL score within the range of 100 to 120*
- 4. The user has to give his LOR number within the range of 1 to 5*
- 5. The user has to give his CGPA score within the range of 6.5 to 10*
- 6. In the same way he/she has to select values within given ranges or given options.*
- 7. All the given inputs are displays in screen and prediction is also displayed output is ranging from 0 to 1*

***BENEFITS OF PROPOSED SYSTEM:***

- High R2 score*
- Low Root Mean Square Error (RMSE)*



### **ADVANTAGES:**

- *It helps student for making decision for choosing a right college.*
- *Here the chance of occurrence of error is less when compared with the existing system.*
- *It is fast, efficient and reliable.*
- *Avoids data redundancy and inconsistency.*
- *Very user-friendly.*
- *Easy accessibility of data.*
- *Number of personnel required is considerably less.*
- *Provides more security and integrity to data.*

### **DISADVANTAGES:**

- *Required active internet connection.*
- *System will provide inaccurate results if data entered incorrectly.*

### **APPLICATION:**

*Minimum hardware requirements: RAM: 16 GB.*

### **SOFTWARE CONFIGURATION:**

*Operating System: windows 10*

*Application server: Google chrome*

*Front End: HTML, PYTHON*

*Softwares: Anaconda, pycharm*

## **CONCLUSION:**

*Student admission problem is very important in educational institutions. In this project addresses machine learning models to predict the chance of a student to be admitted.*

*This will assist students to know in advance if they have a chance to get accepted.*

*Machine learning models were performed to predict the opportunity of a student to get admitted to a master's program.*

*The machine learning models included are multiple linear regression, random forest, Multiple Linear Regression with Backward Elimination and random forest regression with backward elimination.*

*Experiments show that the Linear Regression model surpasses other models. Our aim would be to predict the "Chance of Admit" based on the different parameters that are provided in the dataset.*

*We will achieve this aim by using the Linear Regression model. Based on the data that we have, we will split out data into training and testing sets.*

*The Training set will have features and labels on which our model would be trained. The label here is the "Chance of Admit". If you think from a no-technical standpoint then label is basically the output that we want and features are the parameters that drive us towards the output.*

*Once our model is trained, we will use the trained model and run it on the test set and predict the output.*

*Then we will compare the predicted results with the actual results that we have to see how our model performed.*

*This whole process of training the model using features and known labels and later testing it to predict the output is called Supervised Learning.*

### **FUTURE SCOPE:**

- ❖ *Modifying dataset to convert this problem into Classification problem.*
- ❖ *Currently, the dataset is limited to only predicting the chance of admission.*
- ❖ *We plan to add another class to the dataset which provides which college the student will be admitted to based on his academic performance.*
- ❖ *Creating the model with additional parameters such as Work Experience, Technical Papers, Written, and Content of Letter of Recommendation etc.*
- ❖ *Creating a model based on the graph of admitted vs enrolled students of previous year to predict the increase or decrease in cutoff scores among applicants.*
- ❖ *Comparing different universities based on applied vs admitted data.*

### **Source code:**

*Integrate with Web Framework:*

#### ***Building Html Pages:***

*For this project create two HTML files namely*

- ***Demo.html***
- ***chance.html***
- ***nochance.html***

*and save them in the templates folder.*

## Demo.html:

```
main.py  demo.html x  chance.html  nochance.html

1 <html>
2 <style>
3   body
4   {
5     background-image: url('../static/images/images.jpg');
6     background-position: center;
7     font-family: sans-serif;
8     background-size: cover;
9   }
10 </style>
11
12 <body>
13   <div class="login">
14
15
16   <h1>UNIVERSITY ADMISSION PREDICTION SYSTEM<span class="label label-default"></span></h1>
17   <h2>Enter your details and get probability of your admission <span class="label label-default"></span></h2><br>
18
19
20
21   <form action="{url_for('y_predict')}}" method="post">
22
23     Enter GRE Score <input type="text" name="gre" placeholder="GRE Score (out of 340)" required="required"/><br><br>
24     Enter TOEFL Score <input type="text" name="toefl" placeholder="TOEFL Score (out of 120)" required="required"/> <br> <br>
25     Select University no<br> <input type="radio" name="rating" value="1"> 1 <br>
26     ..... <input type="radio" name="rating" value="2"> 2 <br>
27     ..... <input type="radio" name="rating" value="3"> 3 <br>
28     ..... <input type="radio" name="rating" value="4"> 4 <br>
29     ..... <input type="radio" name="rating" value="5"> 5 <br><br>
30
31     Enter SOP<input type="text" name="sop" placeholder="sop (out of 5)" required="required"/><br><br>
32     Enter LOR<input type="text" name="lor" placeholder="LOR (out of 5)" required="required"/><br><br>
33     Enter CGPA<input type="text" name="cgpa" placeholder="CGPA (out of 10)" required="required"/><br><br>
34
35     Research<br> <input type="radio" name="research" value="1"> Research <br>
36     ..... <input type="radio" name="research" value="0"> No Research <br><br>
37
38
39
40     <button type="submit" class="btn btn-default">Predict</button>
41
42   </form>
43   {{prediction_text}}
44 </div>
45 </body>
46 </html>
47
```

## *chance.html:*

```
main.py  demo.html  chance.html x  nochance.html
1  <html lang="en" dir="ltr">
2      <head>
3          <meta charset="utf-8">
4          <title>Predicting Chance of Admission</title>
5          <link rel="shortcut icon" href="{{url_for('static', filename='diabetes-favicon.ico')}}">
6          <link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/styles.css')}}">
7          <script src="https://kit.fontawesome.com/5f3f547070.js" crossorigin="anonymous"></script>
8          <link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">
9      </head>
10     <style>
11     body
12     {
13     background-image:url('../static/images/chance.jpg');
14     background-position: center;
15     font-family:sans-serif;
16     background-size:cover;
17     }
18     </style>
19
20     <body>
21         <br><br><br><br><br><br>
22         <!--Website Tittle-->
23         <div style="...">
24             <div class="container">
25                 <h2 class='container-heading'><span class="heading_font">Predicting Chance of Admission</span></h2>
26                 <div class='description'>
27                     <p>A MACHINE LEARNING WEB APP using Flask.</p>
28                 </div>
29             </div>
30
31             <!--Result -->
32             <div class="results">
33
34                 <p>Prediction: <b><u>{{prediction_text}}</u></b><img src='../static/images/thumbs-1'></p></div>
35             </div>
36         </div>
37     </body>
html > style
```

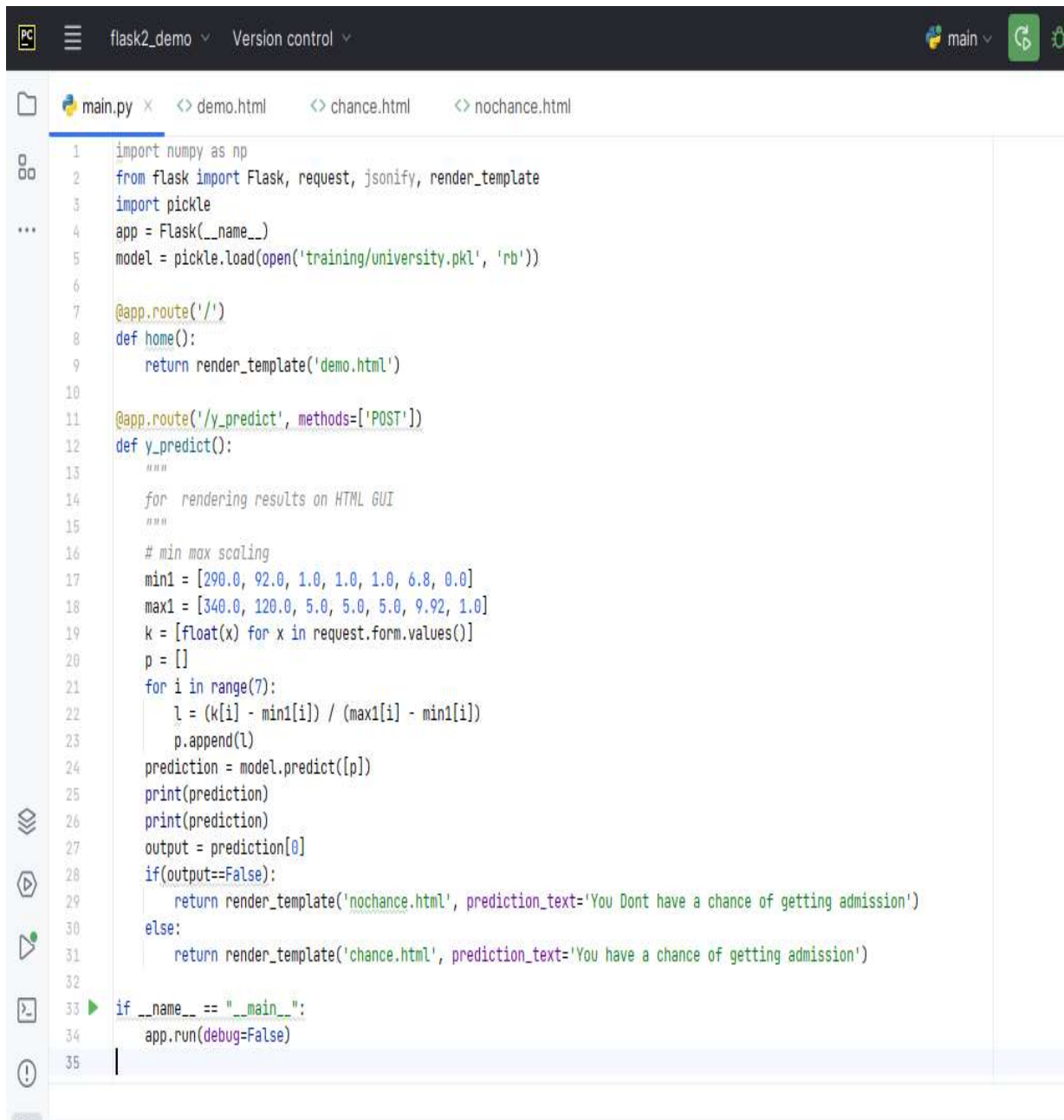
## ***nochance.html:***

```
main.py  demo.html  chance.html  nochance.html x
1 <html lang="en" dir="ltr">
2   <head>
3     <meta charset="utf-8">
4     <title>Predicting No Chance of Admission</title>
5     <link rel="shortcut icon" href="{{url_for('static', filename='diabetes-favicon.ico')}}">
6     <link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/styles.css')}}">
7     <script src="https://kit.fontawesome.com/5f3f547070.js" crossorigin="anonymous"></script>
8     <link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">
9   </head>
10  <style>
11    body
12    {
13      background-image:url('../static/images/no chance.jpg');
14      background-position: center;
15      font-family:sans-serif;
16      background-size:cover;
17    }
18  </style>
19
20  <body>
21    <br><br><br><br><br><br><br>
22    <!--Website Title-->
23    <div style="padding-left:200px">
24      <div class="container">
25        <h2 class="container-heading"><span class="heading_font">Predicting No Chance of Admission</span></h2>
26        <div class="description">
27          <p>A MACHINE LEARNING WEB APP using Flask.</p>
28        </div>
29      </div>
30
31      <!--Result -->
32      <div class="results">
33
34        <p>Prediction: <b><u>{{prediction_text}}</u></b> <img src='../static/images/thumbs-1.png">
35      </div>
36    </div>
37  </body>
38 </html>
39
```



## Build Python code:

*main.py:*



```
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3 import pickle
4 app = Flask(__name__)
5 model = pickle.load(open('training/university.pkl', 'rb'))
6
7 @app.route('/')
8 def home():
9     return render_template('demo.html')
10
11 @app.route('/y_predict', methods=['POST'])
12 def y_predict():
13     """
14     for rendering results on HTML GUI
15     """
16     # min max scaling
17     min1 = [290.0, 92.0, 1.0, 1.0, 1.0, 6.8, 0.0]
18     max1 = [340.0, 120.0, 5.0, 5.0, 5.0, 9.92, 1.0]
19     k = [float(x) for x in request.form.values()]
20     p = []
21     for i in range(7):
22         l = (k[i] - min1[i]) / (max1[i] - min1[i])
23         p.append(l)
24     prediction = model.predict([p])
25     print(prediction)
26     print(prediction)
27     output = prediction[0]
28     if(output==False):
29         return render_template('nochance.html', prediction_text='You Dont have a chance of getting admission')
30     else:
31         return render_template('chance.html', prediction_text='You have a chance of getting admission')
32
33 if __name__ == "__main__":
34     app.run(debug=False)
35
```

***THANK YOU.***