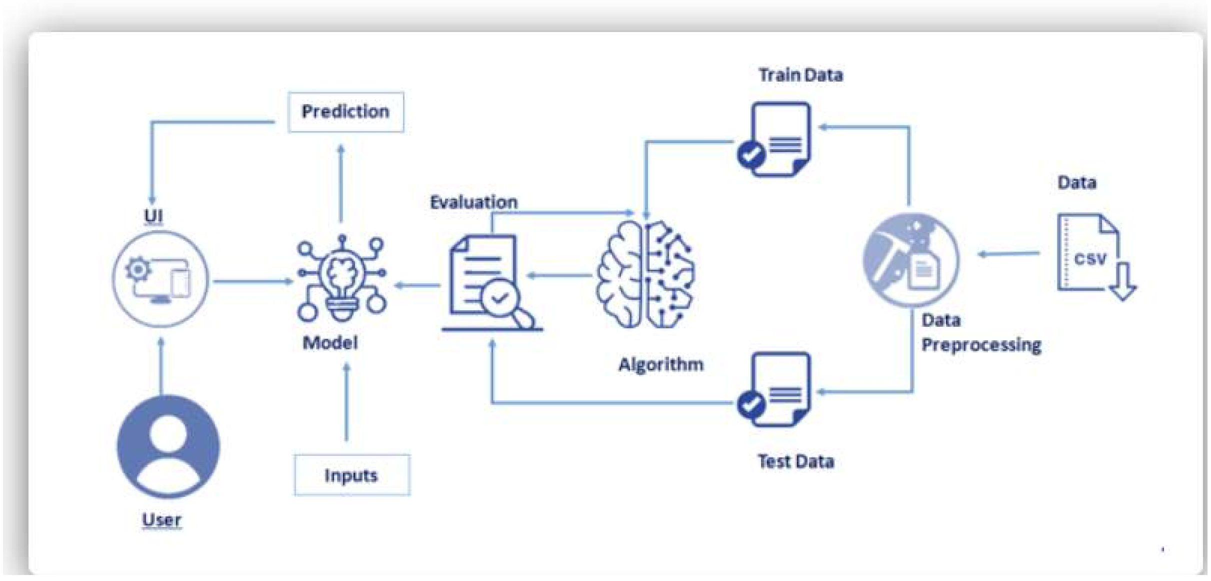
University Admit Eligibility Predictor

Problem Statement

Students are often worried about their chances of admission to University. The aim of this project is to help students in shortlisting universities with their profiles. 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

Technical Architecture



Pre Requisites:

To complete this project you should need the following:

Jupyter Notebook for programming, which can be installed by Anaconda IDE.

Python packages

Install Anaconda

Anaconda/ PyCharm IDE is Ideal to complete this project

To install Anaconda, please refer to Anaconda Installation Steps

Install Python Packages

We need to install the following packages:
1.Numpy: This package is used to perform numerical computations. This package is pre-installed in anaconda.
2.Pandas: Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures, and data analysis tools. This package is pre-installed in anaconda.
3.Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. This package is pre-installed in anaconda.
4.Scikit-learn: This is a machine learning library for the Python programming language. This package is pre-installed in anaconda.
5.Flask: Flask is a lightweight WSGI web application framework.

If not installed use the following
Open anaconda prompt as administrator.

- Type “pip install numpy” and click enter.
- Type “pip install pandas” and click enter.
- Type “pip install matplotlib” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type “pip install Flask” and click enter.

The above steps allow you to install the packages in the anaconda environment

Prior Knowledge

One should have knowledge on the following Concepts :

Supervised and unsupervised learning:

Project Flow

You will go through all the steps mentioned below to complete the project.

- User interacts with the UI (User Interface) to enter Data
- The entered data is analyzed by the model which is integrated
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
- Collect the dataset or Create the dataset
- Data Preprocessing.
- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Taking care of Missing Data.
- Label encoding.
- One Hot Encoding.
- Feature Scaling.
- Splitting Data into Train and Test.
- Model Building
- Training and testing the model
- Evaluation of Model
- Application Building
- Create an HTML file
- Build a Python Code

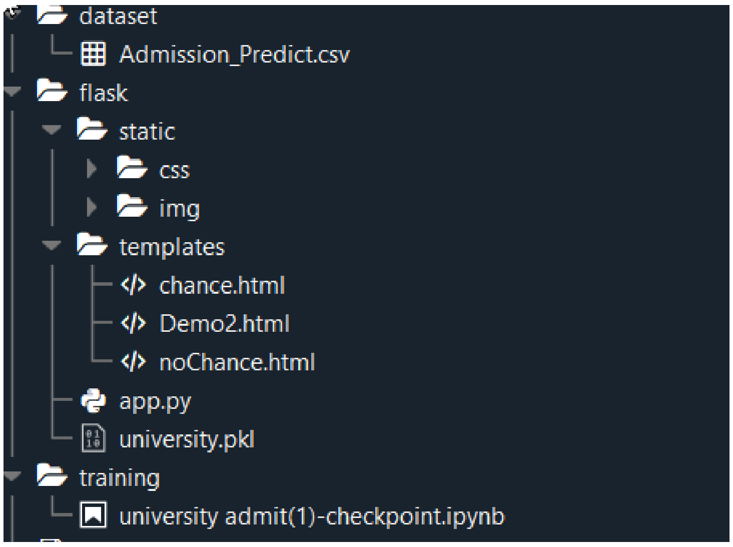
Project Objectives

By the end of this project you'll be able to understand :

- Regression and Classification Problems
- To grab insights from data through visualization.
- Applying different algorithms according
- Evaluation metrics
- how to build a web application using the Flask framework.

Project Structure

- A python file called app.py for server-side scripting.
- We need the model which is saved and the saved model in this content is university.pkl(will be created when the model is built)
- Templates folder which contains the Demo2.HTML file, chance.HTML file, noChance.HTML file.
- Static folder which contains a CSS folder which contains styles.css.



```
dataset
└ Admission_Predict.csv
flask
  static
    css
    img
  templates
  ├ chance.html
  ├ Demo2.html
  └ noChance.html
  app.py
  university.pkl
training
└ university admit(1)-checkpoint.ipynb
```

Data Collection

ML depends heavily on data, without data, it is impossible for an “AI” to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

Download The Dataset

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.

The dataset used for this project was obtained from Kaggle.

Data Preprocessing

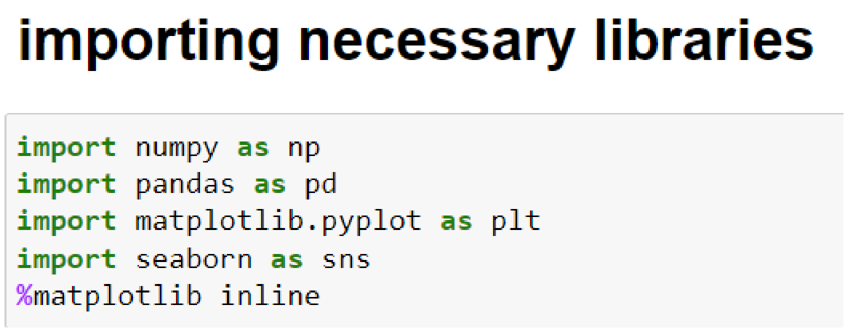
Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn’t feasible for the analysis.
Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing Follow the following steps to process your Data

- Import the Libraries
- Importing the dataset
- Taking care of Missing Data
- Label encoding
- One Hot Encoding
- Feature Scaling
- Splitting Data into Train and Test

Importing The Libraries

It is important to import all the necessary libraries such as pandas, numpy, matplotlib.

- **Numpy**- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.
- **Pandas**- It is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.
- **Seaborn**- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Matplotlib**- Visualisation with python. It is a comprehensive library for creating static, animated, and interactive visualizations in Python



importing necessary libraries

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

Reading The Dataset

You might have your data in .csv files, .excel files Let's load a .csv data file into pandas using read_csv() function.We will need to locate the directory of the CSV file at first If your dataset is in some other location ,Then see below command

Data=pd.read_csv(r"File_location/filename.csv")

Our Dataset Admission_Predict contains following Columns

1.Serial No.

2.GRE Score

3.TOEFL Score

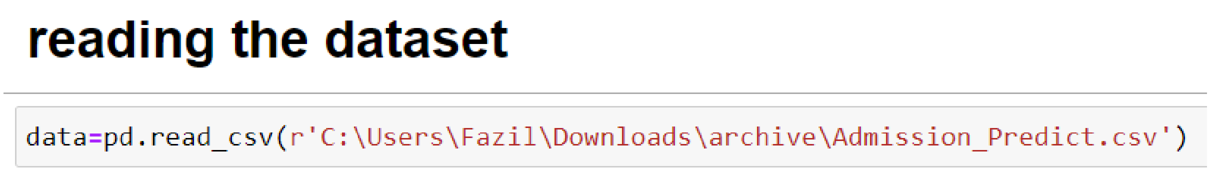
4.University Rating

5.SOP

6.LOR

7.CGPA

8.Chance of Admit

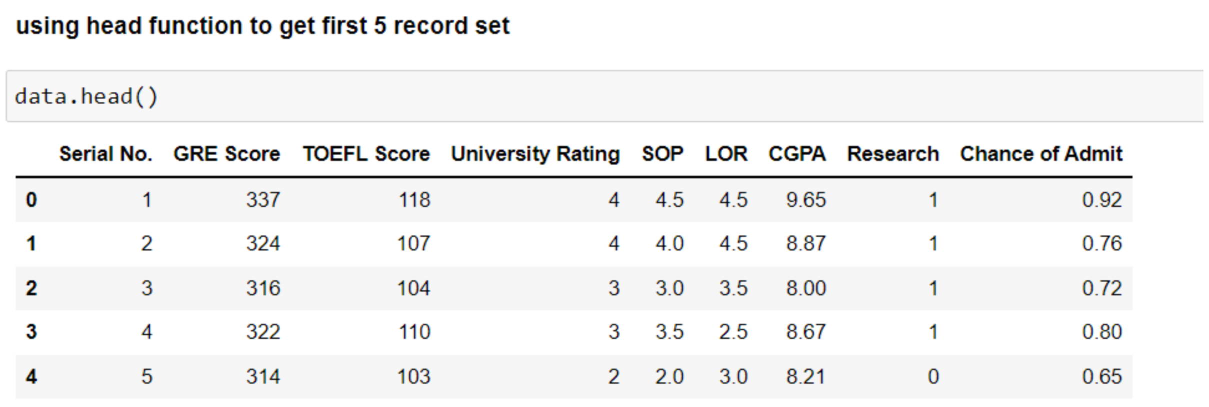


reading the dataset

```
data=pd.read_csv(r'C:\Users\Fazil\Downloads\archive\Admission_Predict.csv')
```

Analyze The Data

- head() method is used to return top n (5 by default) rows of a DataFrame or series.

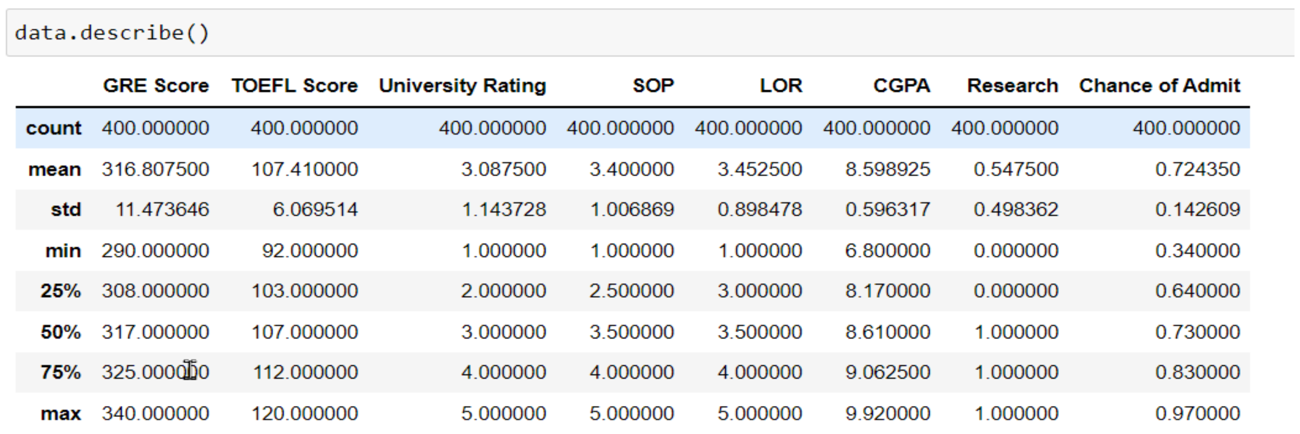


using head function to get first 5 record set

```
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

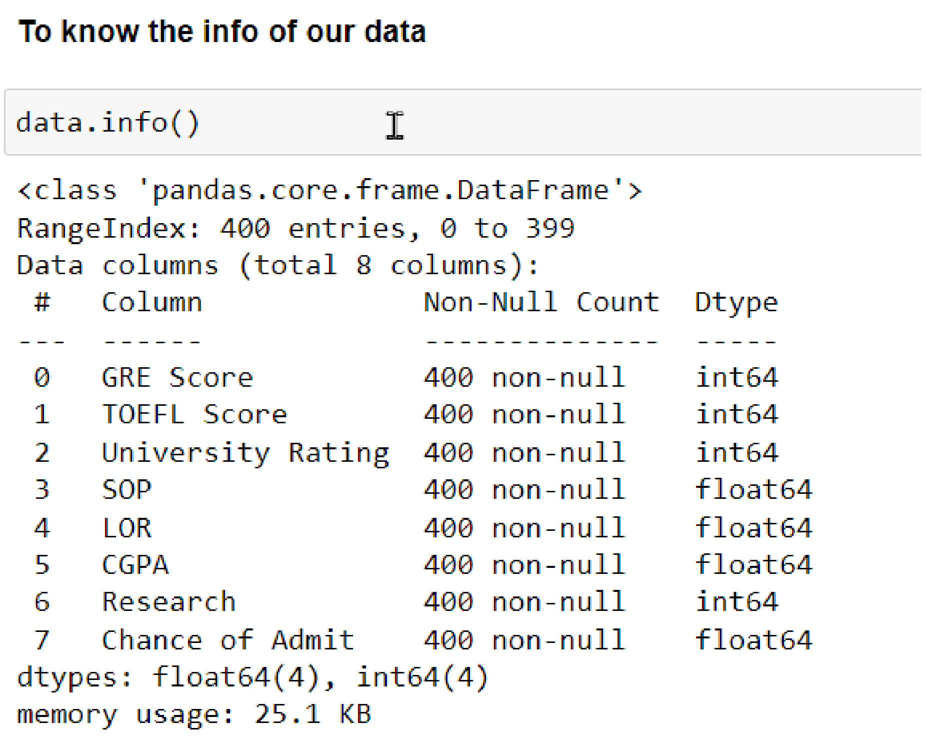
- describe() method computes a summary of statistics like count, mean, standard deviation, min, max, and quartile values



```
data.describe()
```

	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
mean	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

- info() gives information about the data



To know the info of our data

```
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
```

Handling Missing Values

After loading it is important to check the complete information of data as it can indicate many of the hidden information such as null values in a column or a row Check for the null values. if it is present then the following steps can be performed

- Imputing data using the Imputation method in sklearn.
- Filling NaN values with mean, median, and mode using fillna() method.

You can check the null values with the function i**snull().any()**

- If the dataset contains null values then the above functions return as true. But if you look at the dataset you can observe that the dataset does not have any null values.
- You can also check the number of null values present in the columns by the using **isnull().sum()** function

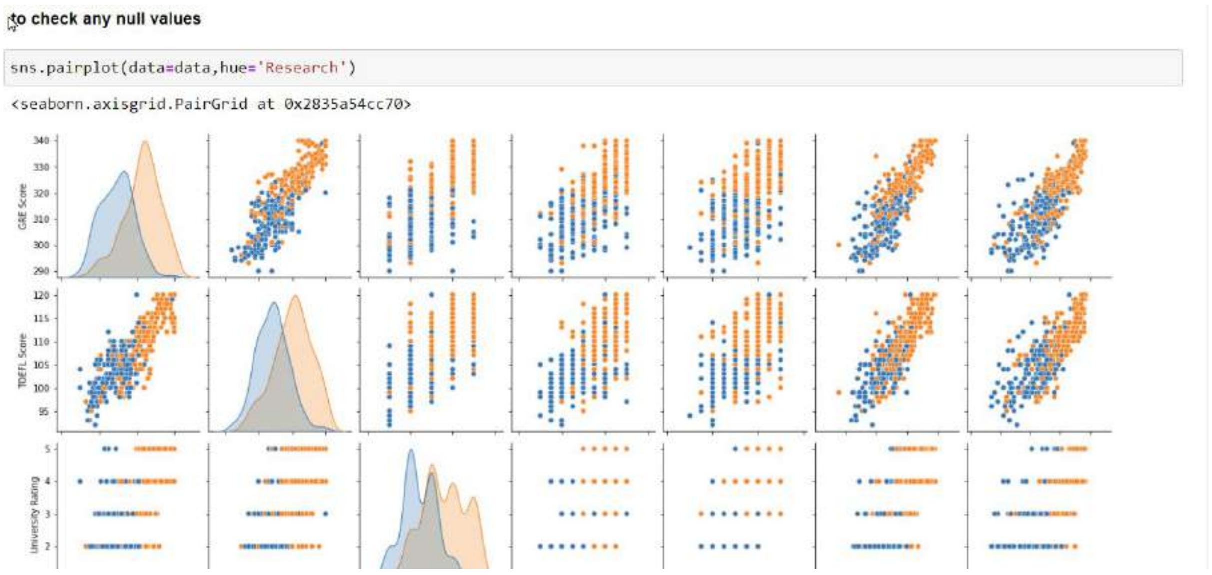
As we don't have categorical data then we can skip the steps of label encoding and one-hot encoding

```
data.isnull().any()
```

```
GRE Score            False
TOEFL Score          False
University Rating    False
SOP                  False
LOR                  False
CGPA                 False
Research             False
Chance of Admit      False
dtype: bool
```

Data Visualization

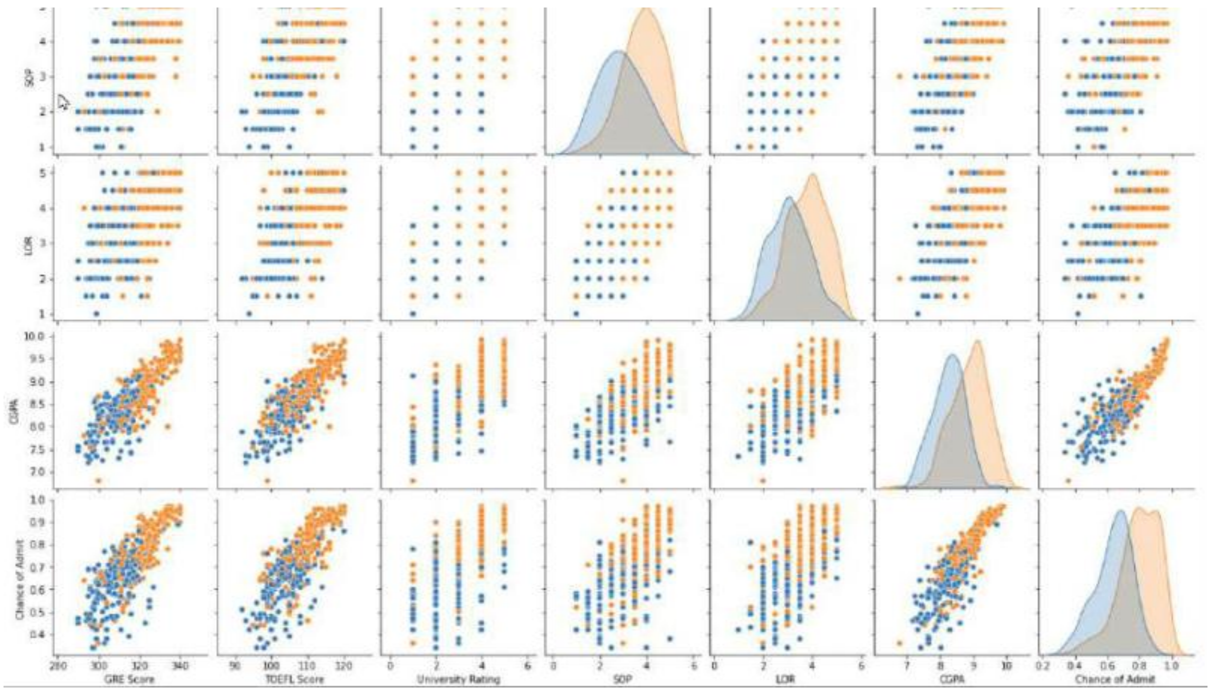
Data visualization is where a given dataset is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data. Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. Machine learning models will perform poorly on data that wasn't visualized and understood properly. To visualize the dataset we need libraries called Matplotlib and Seaborn. The Matplotlib library is a Python 2D plotting library that allows you to generate plots, scatter plots, histograms, bar charts etc.



to check any null values

```
sns.pairplot(data=data,hue='Research')
```

```
<seaborn.axisgrid.PairGrid at 0x2835a54cc70>
```



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

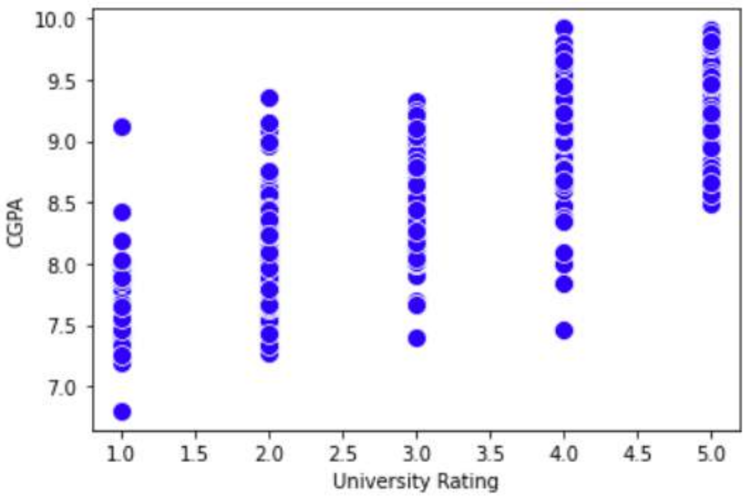
From the above pair plot we infer that

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.

```
sns.scatterplot(x= 'University Rating',y= 'CGPA',data=data,color='blue',s=100)
```

```
<AxesSubplot:xlabel='University Rating', ylabel='CGPA'>
```

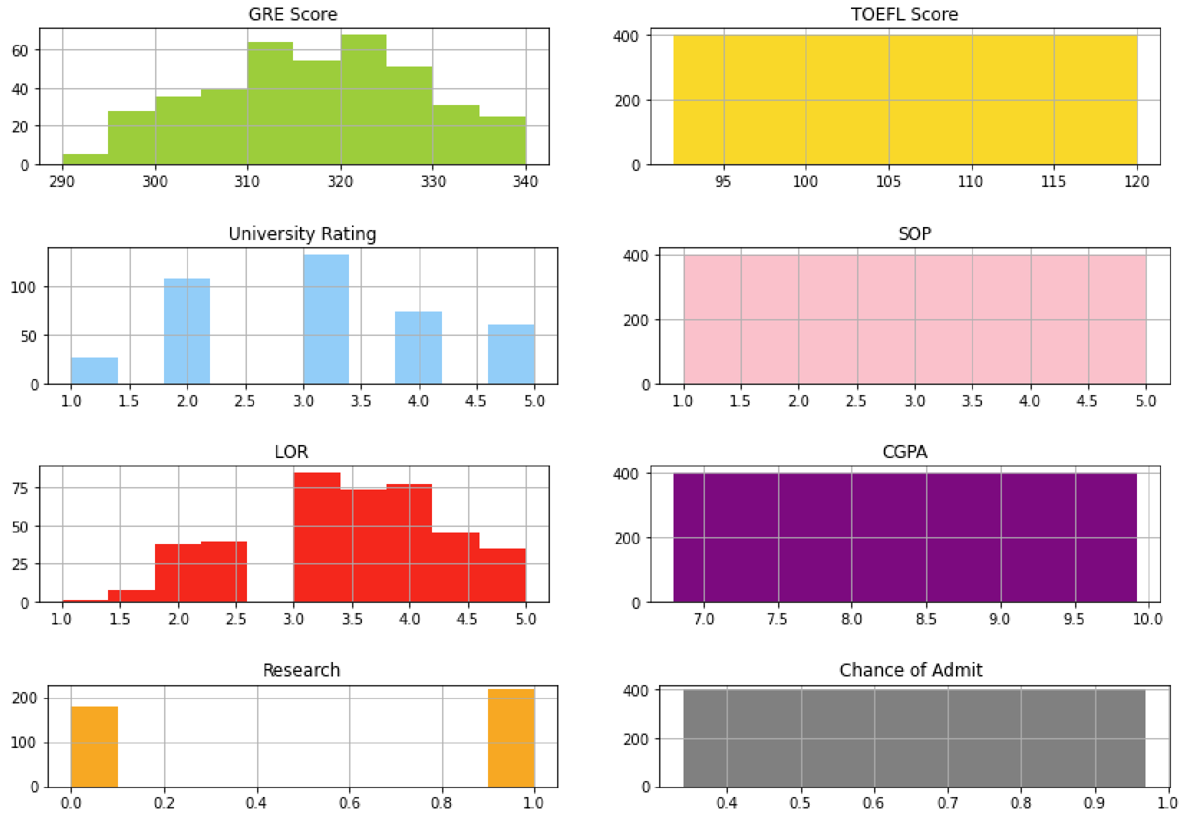


above scatter shows that the higher CGPA the higher chance of getting admission

```
category = ['GRE Score','TOEFL Score','University Rating','SOP','LOR ','CGPA','Research','Chance of Admit ']
color = ['yellowgreen','gold','lightskyblue','pink','red','purple','orange','grey']
start = True

for i in np.arange(4):
    fig = plt.figure(figsize=(14,8))
    plt.subplot2grid((4,2), (i,0))
    data[category[2*i]].hist(color=color[2*i],bins=10)
    plt.title(category[2*i])
    plt.subplot2grid((4,2), (i,1))
    data[category[2*i+1]].hist(color=color[2*i+1],bins=1)
    plt.title(category[2*i+1])

plt.subplots_adjust(hspace = 0.7, wspace = 0.2)
plt.show()
```



```
print('Mean CGPA Score is :',int(data['CGPA'].mean()))
print('Mean GRE Score is :',int(data['GRE Score'].mean()))
print('Mean TOEFL Score is :',int(data['TOEFL Score'].mean()))
```

```
Mean CGPA Score is : 8
Mean GRE Score is : 316
Mean TOEFL Score is : 107
```

The chance of admission is high is the aspirant score is more then the mean score

Splitting Dependent And Independent Columns

We need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column.

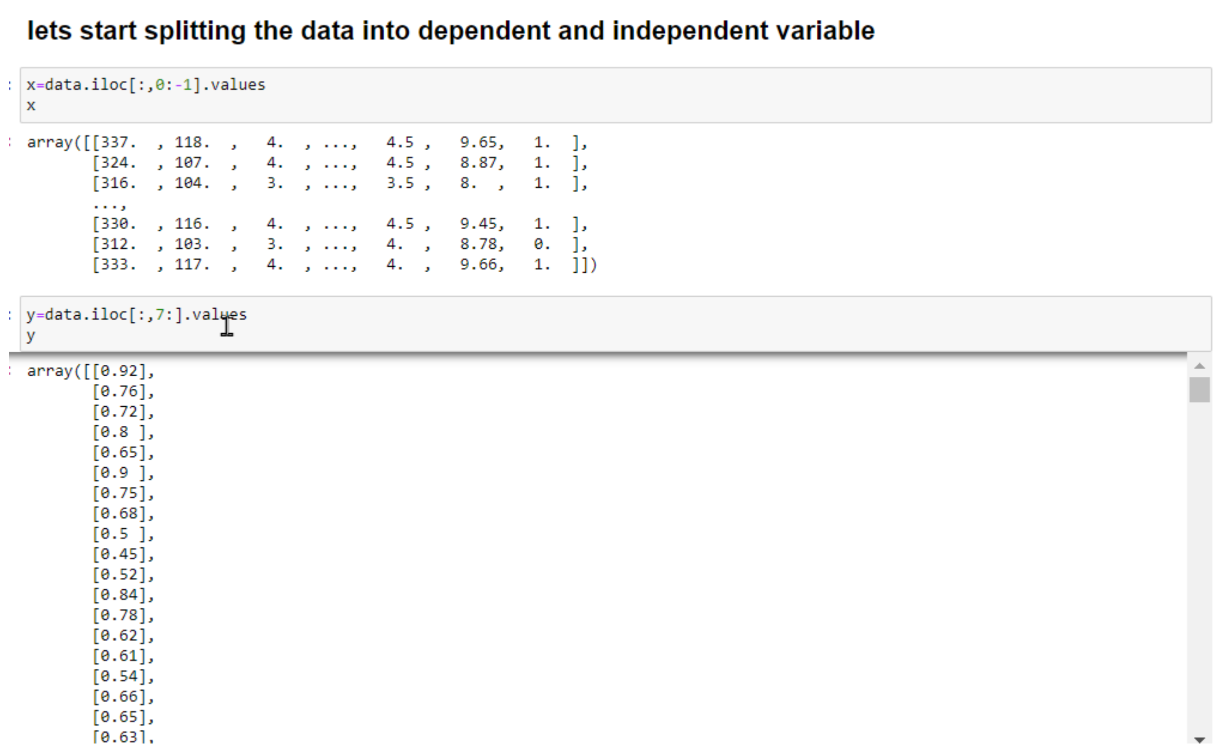
To read the columns, we will use iloc of pandas (used to fix the indexes for selection) which takes two parameters — [row selection, column selection].

Let’s split our dataset into independent and dependent variables.

From the above code “:” indicates that you are considering all the rows in the dataset and “0:7” indicates that you are considering columns 0 to 7 such as year, month, and day as input values and assigning them to variable x. In the same way in the second line “:” indicates you are considering all the rows and “7:” indicates that you are considering only the last column as output value and assigning them to variable y.

Let's Check the shape of x and Y

You can see in x we have 1991 rows with 7 columns and y has 1 column with the same number of rows



lets start splitting the data into dependent and independent variable

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

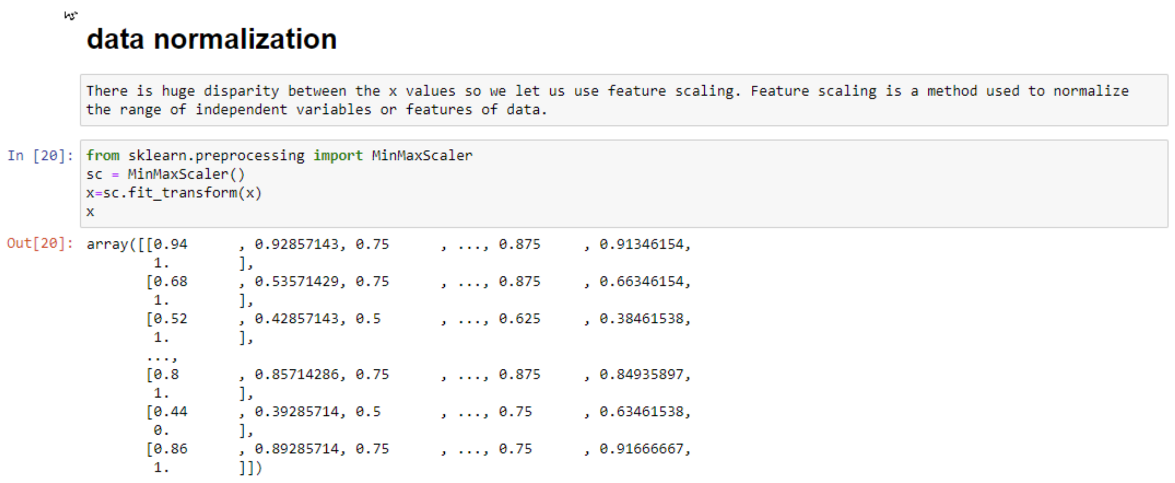
```
array([[337.  , 118.  ,   4.  , ...,   4.5 ,   9.65,   1.  ],
       [324.  , 107.  ,   4.  , ...,   4.5 ,   8.87,   1.  ],
       [316.  , 104.  ,   3.  , ...,   3.5 ,   8.  ,   1.  ],
       ...,
       [330.  , 116.  ,   4.  , ...,   4.5 ,   9.45,   1.  ],
       [312.  , 103.  ,   3.  , ...,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

```
y=data.iloc[:,7:].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],
```

data normalization

There is huge disparity between the x values so we let us use feature scaling. Feature scaling is a method used to normalize the range of independent variables or features of data.



data normalization

There is huge disparity between the x values so we let us use feature scaling. Feature scaling is a method used to normalize the range of independent variables or features of data.

```
In [20]: from sklearn.preprocessing import MinMaxScaler
         sc = MinMaxScaler()
         x=sc.fit_transform(x)
         x
```

```
Out[20]: 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 The Data Into Train And Test

When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you will need a dataset that is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is, 'train_test_split.' Using this we can easily split the dataset into the training and the testing datasets in various proportions.

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

Train Dataset: Used to fit the machine learning model.

Test Dataset: Used to evaluate the fit machine learning model.

In general, you can allocate 80% of the dataset to the training set and the remaining 20% to the test set. We will create 4 sets

x_train

x_test

y_train

y_test .

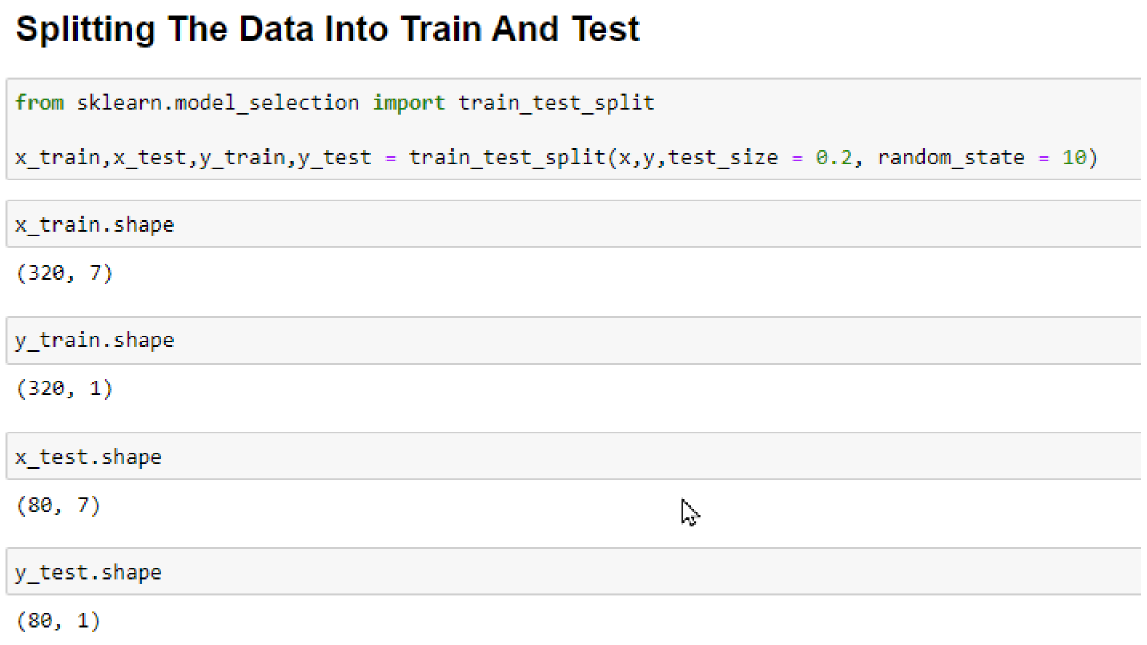
There are a few other parameters that we need to understand before we use the class:

test_size: this parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset and remaining a train dataset

random_state: here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the Random_state class, which will become the number generator. If you don’t pass anything, the Random_state instance used by np.random will be used instead.

If you observe y column it has the probability of student getting admitted into a university. I would like to convert this probability as either yes or no

so let's write a logic to do so



Splitting The Data Into Train And Test

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2, random_state = 10)
```

```
x_train.shape
```

```
(320, 7)
```

```
y_train.shape
```

```
(320, 1)
```

```
x_test.shape
```

```
(80, 7)
```

```
y_test.shape
```

```
(80, 1)
```

Model Building

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have it may be Classification algorithms are Regression algorithms.

Example: 1. Linear Regression.

2. Logistic Regression.

3. Random Forest Regression / Classification.

4. Decision Tree Regression / Classification.

You will need to train the datasets to run smoothly and see an incremental improvement in the prediction rate.

Now we apply the Logistic Regression algorithm on our dataset.

Training And Testing The Model

Once after splitting the data into train and test, the data should be fed to an algorithm to build a model.

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have it may be Classification algorithms are Regression algorithms. As it is a kind of classification problem you can apply any of the following Algorithms

1.Logistic Regression

2.Decision Tree Classifier

3.Random Forest Classifier

4.KNN

Logistic Regression:

Logistic Regression is used when the dependent variable (target) is categorical. For example,

To predict whether an email is a spam (1) or (0)

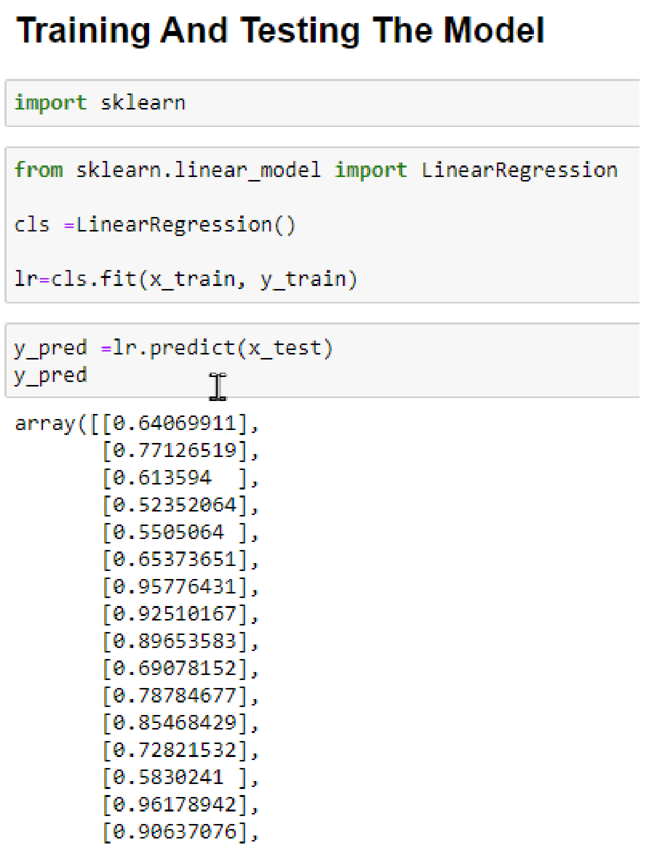
Whether the tumor is malignant (1) or not (0)

Out of all the algorithms Logistic Regresson got the highest accuracy so let's build a model with Logistic regression

We’re going to use x_train and y_train obtained above in the train_test_split section to train our decision tree regression model. We’re using the fit method and passing the parameters as shown below.

We are using the algorithm from Scikit learn library to build the model as shown below, Once the model is trained, it’s ready to make predictions. We can use the predict method on the model and pass x_test as a parameter to get the output as y_pred.

Notice that the prediction output is an array of real numbers corresponding to the input array.


Training And Testing The Model

```
import sklearn
```

```
from sklearn.linear_model import LinearRegression

cls =LinearRegression()

lr=cls.fit(x_train, y_train)
```

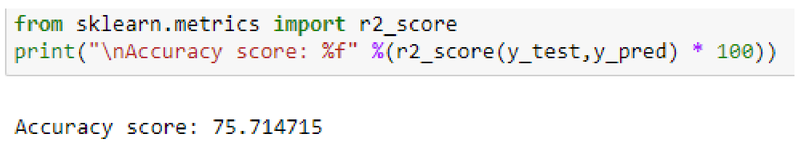
```
y_pred =lr.predict(x_test)
y_pred
```

```
array([[0.64069911],
       [0.77126519],
       [0.613594  ],
       [0.52352064],
       [0.5505064 ],
       [0.65373651],
       [0.95776431],
       [0.92510167],
       [0.89653583],
       [0.69078152],
       [0.78784677],
       [0.85468429],
       [0.72821532],
       [0.5830241 ],
       [0.96178942],
       [0.90637076],
```

Model Evaluation

There are many numbers of model evaluation techniques for the classification type of machine learning models. the following are widely used

- Accuracy_score
- Confusion matrix
- Roc- Auc Curve

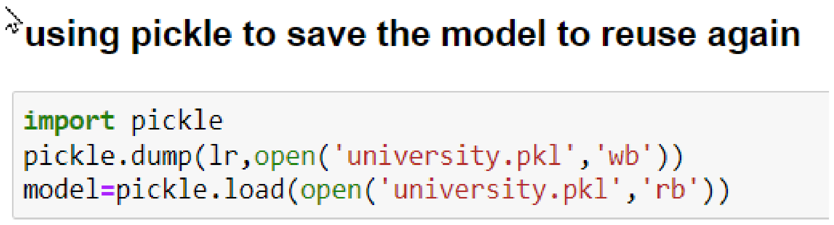

```
from sklearn.metrics import r2_score
print("\nAccuracy score: %f" %(r2_score(y_test,y_pred) * 100))
```

```
Accuracy score: 75.714715
```

Save The Model

After building the model we have to save the model.

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions or transport data over the network. wb indicates the write method and rd indicates the read method.



using pickle to save the model to reuse again

```python
import pickle
pickle.dump(lr,open('university.pkl','wb'))
model=pickle.load(open('university.pkl','rb'))
```

Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.
This section has the following tasks

- Building HTML Pages
- Building server-side script

Build HTML Code

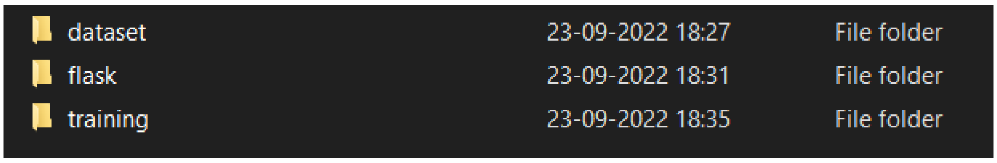
In this HTML page, we will create the front end part of the web page. On this page, we will accept input from the user and Predict the values.

In our project we have 3 HTML files, they are

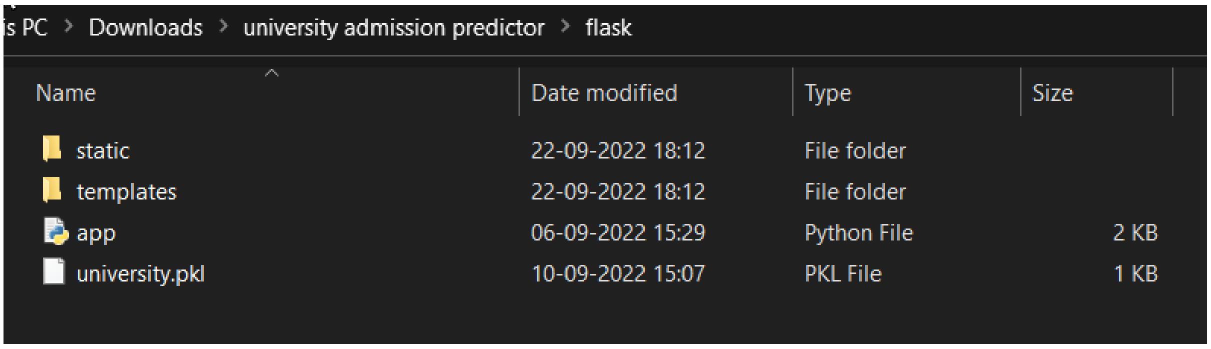
1.Demo2.html

2.chance.html

3.noChance.html

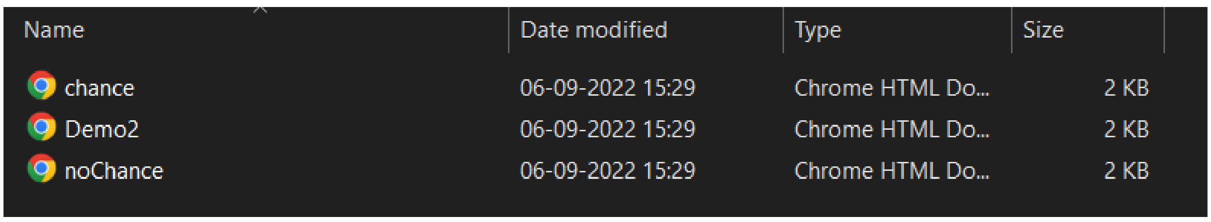


dataset	23-09-2022 18:27	File folder
flask	23-09-2022 18:31	File folder
training	23-09-2022 18:35	File folder



is PC › Downloads › university admission predictor › flask

Name	Date modified	Type	Size
static	22-09-2022 18:12	File folder	
templates	22-09-2022 18:12	File folder	
app	06-09-2022 15:29	Python File	2 KB
university.pkl	10-09-2022 15:07	PKL File	1 KB



Name	Date modified	Type	Size
chance	06-09-2022 15:29	Chrome HTML Do...	2 KB
Demo2	06-09-2022 15:29	Chrome HTML Do...	2 KB
noChance	06-09-2022 15:29	Chrome HTML Do...	2 KB

Build Python Code

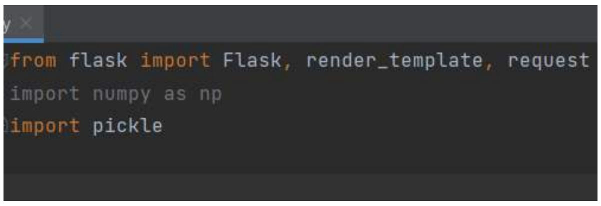
Let us build the app.py flask file which is a web framework written in python for server-side scripting. Let's see step by step procedure for building the backend application.

In order to develop a web App with respect to our model, we basically use Flask framework which is written in python.

- Line 1-3 We are importing necessary libraries like Flask to host our model request
- Line 4 Initialise the Flask application
- Line 5 Loading the model using pickle
- Line 7 Routes the API URL
- Line 9 Rendering the template. This helps to redirect to the home page. In this home page, we give our input and ask the model to predict
- Line 19 we are taking the inputs from the form
- Line 21-23 Feature Scaling the inputs
- Line 24 Predicting the values given by the user
- Line 27-30 If the output is false render noChance template
- If the output is True render chance template
- Line 31 The value of __name__ is set to __main__ when the module run as the main   program otherwise it is set to name of the modul

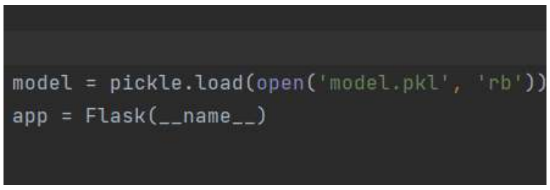
Activity 2: Build Python code:

Import the libraries



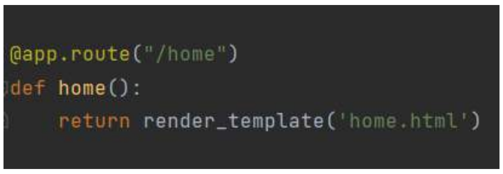
```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.



```
model = pickle.load(open('model.pkl', 'rb'))
app = Flask(__name__)
```

Render HTML page:

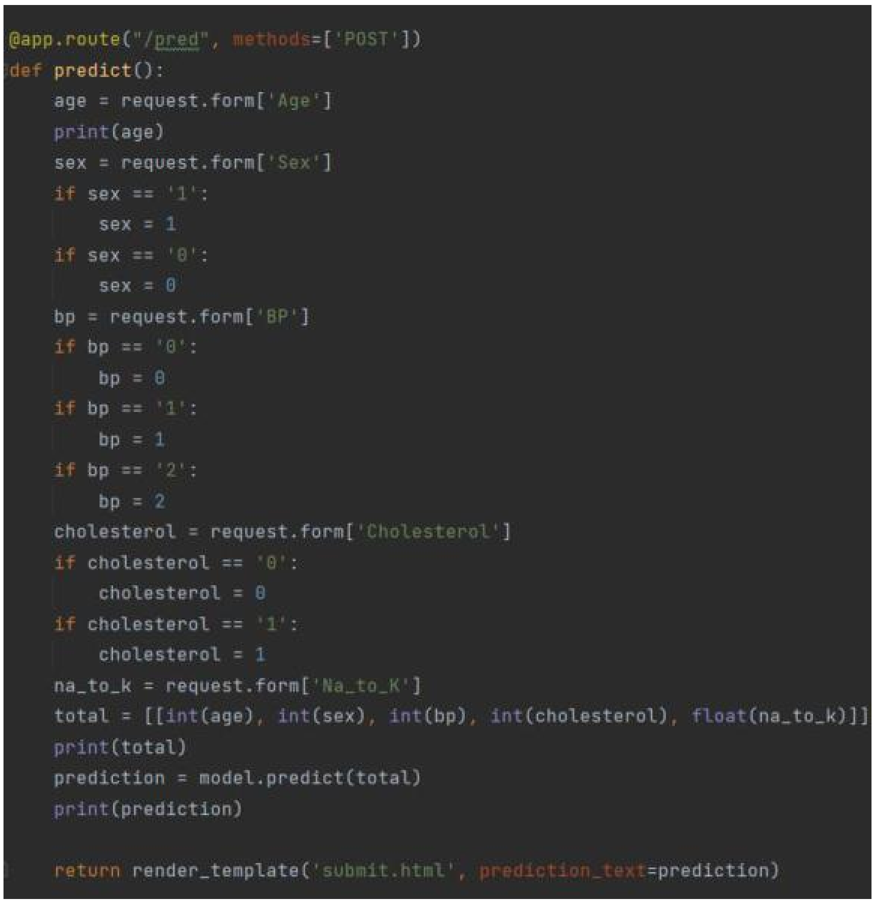


```
@app.route("/home")
def home():
    return render_template('home.html')
```

Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:



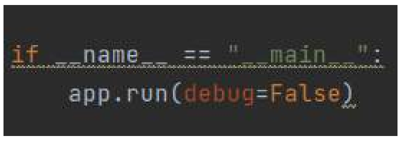
```
@app.route("/pred", methods=['POST'])
def predict():
    age = request.form['Age']
    print(age)
    sex = request.form['Sex']
    if sex == '1':
        sex = 1
    if sex == '0':
        sex = 0
    bp = request.form['BP']
    if bp == '0':
        bp = 0
    if bp == '1':
        bp = 1
    if bp == '2':
        bp = 2
    cholesterol = request.form['Cholesterol']
    if cholesterol == '0':
        cholesterol = 0
    if cholesterol == '1':
        cholesterol = 1
    na_to_k = request.form['Na_to_K']
    total = [[int(age), int(sex), int(bp), int(cholesterol), float(na_to_k)]]
    print(total)
    prediction = model.predict(total)
    print(prediction)

    return render_template('submit.html', prediction_text=prediction)
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict()

function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the submit.html page earlier.

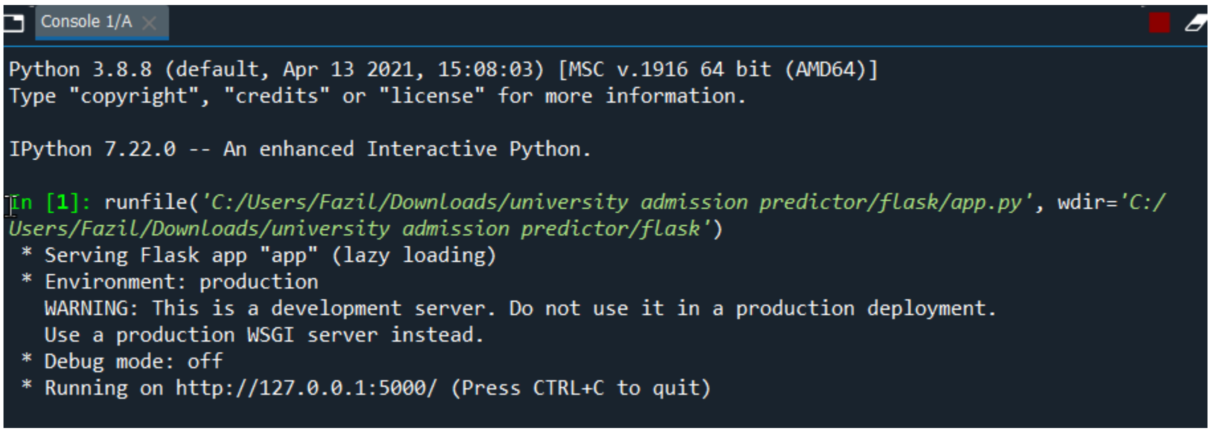
Main Function:



```
if __name__ == "__main__":
    app.run(debug=False)
```

Activity 3: Run the application

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

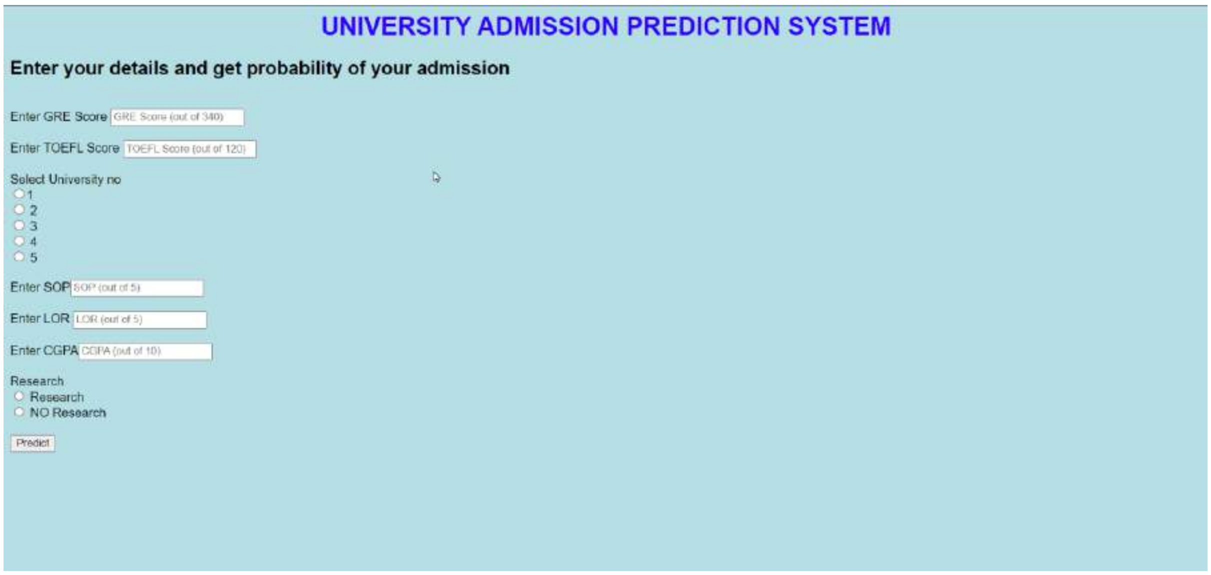


Console 1/A

```
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.22.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Fazil/Downloads/university admission predictor/flask/app.py', wdir='C:/
Users/Fazil/Downloads/university admission predictor/flask')
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



UNIVERSITY ADMISSION PREDICTION SYSTEM

Enter your details and get probability of your admission

Enter GRE Score: GRE Score (out of 340)

Enter TOEFL Score: TOEFL Score (out of 120)

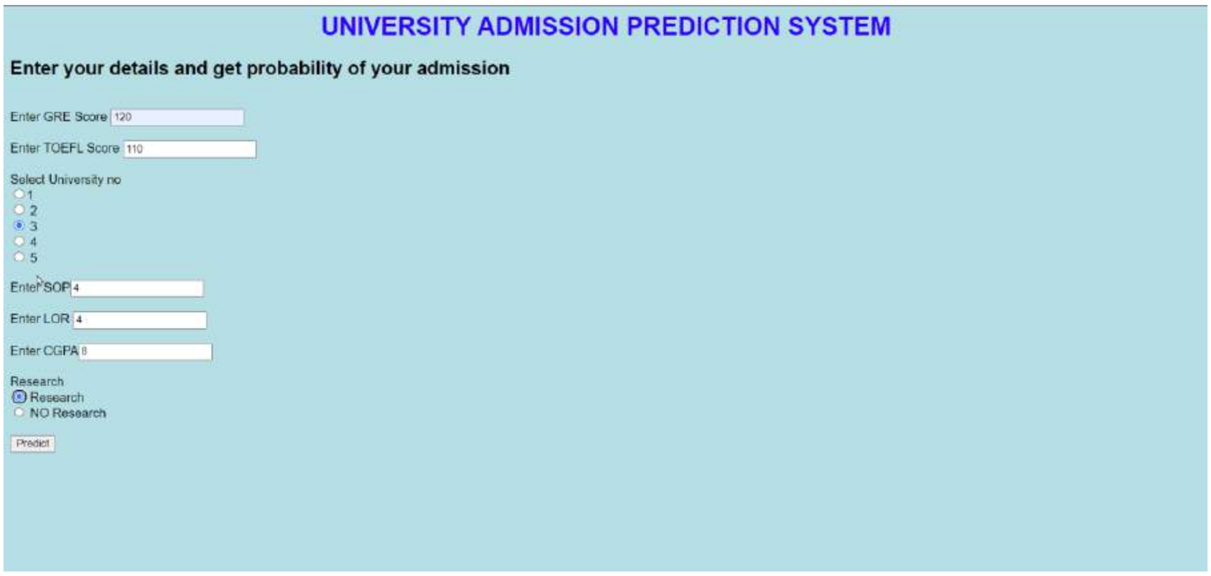
Select University no
- 1
- 2
- 3
- 4
- 5

Enter SOP: SOP (out of 5)

Enter LOR: LOR (out of 5)

Enter CGPA: CGPA (out of 10)

Research
- Research
- NO Research



UNIVERSITY ADMISSION PREDICTION SYSTEM

Enter your details and get probability of your admission

Enter GRE Score: 120

Enter TOEFL Score: 110

Select University no
- 1
- 2
- [✓ 3]
- 4
- 5

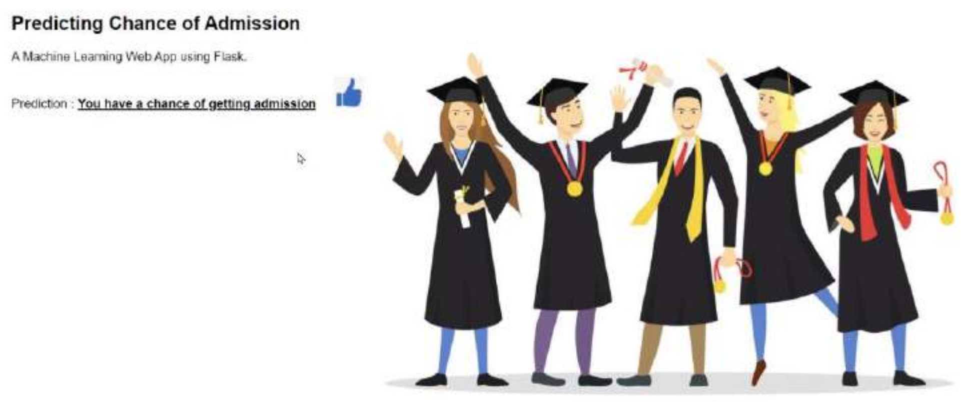
Enter SOP: 4

Enter LOR: 4

Enter CGPA: 8

Research
- [✓ Research]
- NO Research

THUS PROVIDING THE VARIABLES WE CAN PREDICT THE CHANCE OF ADMISSION



Predicting Chance of Admission

A Machine Learning Web App using Flask.

Prediction : You have a chance of getting admission