**AIRLINE REVIEW CLASSIFICATION**

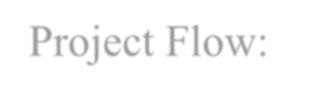
**Project Description:**

In today's interconnected world, the airline industry serves as a critical catalyst for global travel and business. As air travel becomes increasingly accessible, the quality of service provided by airlines plays a pivotal role in shaping passenger experiences. This project focuses on the development of an airline review classification system using Classification models such as Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier etc.,

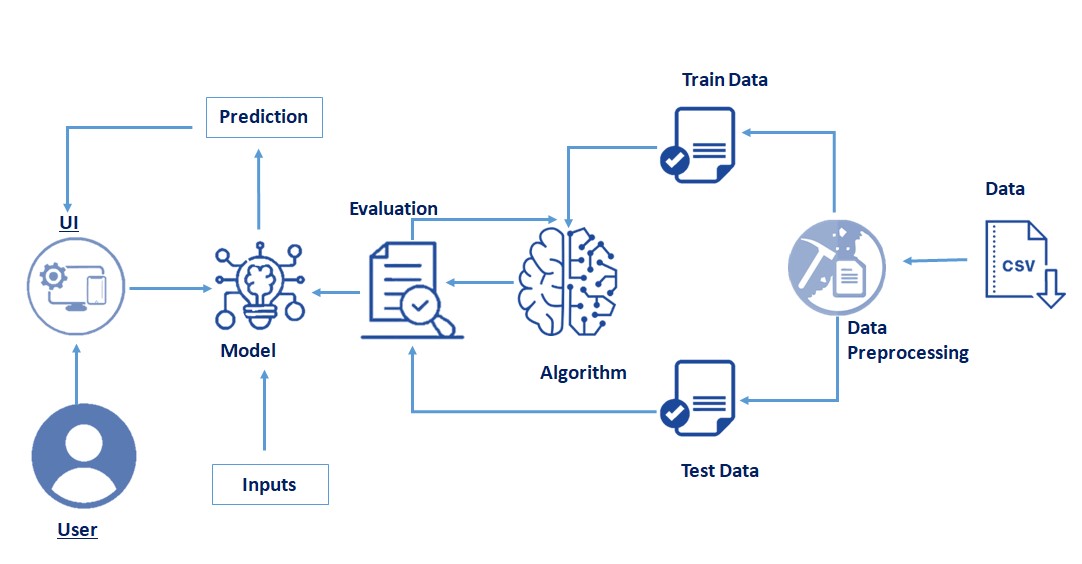
The proliferation of social media platforms, travel websites, and online forums has given rise to a wealth of user-generated content, including airline reviews. Extracting actionable insights from this vast pool of unstructured text data has the potential to provide airlines with valuable information for refining their services and elevating passenger satisfaction.

Throughout this report, we will delve into the methodology employed to preprocess the raw text data, the process of selecting pertinent features, the training and evaluation of the classification model, and the subsequent interpretation of the obtained results.

**Technical Architecture**:



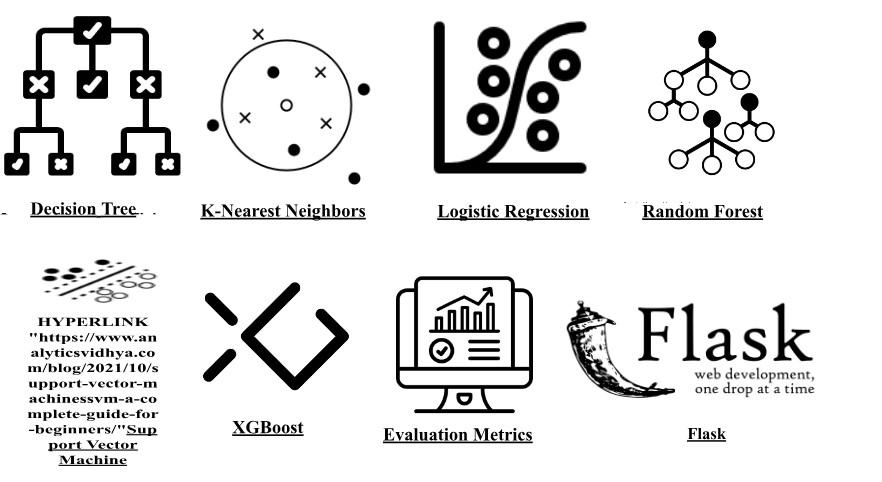
Project Flow:



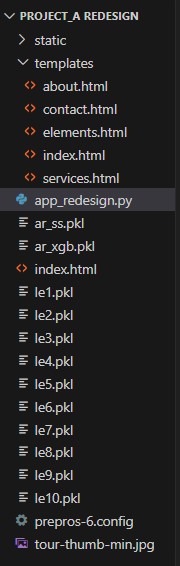
* User interacts with the UI to enter the input.
* Entered input is analyzed by the model which is integrated.
* Once model analyses the input the prediction is showcased on the UI to accomplish this, we have to complete all the activities listed below,
* Data Collection & Preparation
* Collect the dataset
* Data Preparation
* Exploratory Data Analysis
* Descriptive statistical
* Visual Analysis
* Model Building
* Training the model in multiple algorithms
* Testing the model
* Performance Testing
* Testing model with multiple evaluation metrics
* Model Deployment
* Save the best model
* Integrate with Web Framework
* Project Demonstration & Documentation
* Record explanation Video for project end to end solution

**Prior Knowledge:**

You must have prior knowledge of following Supervised Learning topics of Machine Learning to complete this project.



**Project Structure:**

.

Create the project folder which contains files as shown below

* We are building a flask application which needs HTML pages stored in the templates folder and a python script app\_redesign.py for scripting.
* ar\_xgb.pkl is our saved model. Further we will use this model for flask integration.
* ar\_ss.pkl is our pickle file of Standard Scaler Feature.
* [le1.pkl, le2.pkl, le3.pkl, le4.pkl, le5.pkl, le6.pkl, le7.pkl, le8.pkl, le9.pkl, le10.pkl] are the pickle files of Label encoding.

**Milestone 1: Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset. **Activity 1: Collect the dataset**

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project, we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: <https://www.kaggle.com/datasets/khushipitroda/airline-reviews>

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

**Note:** There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques. **Activity 2: Importing the libraries**

Import the necessary libraries as shown in the image.



# Activity 3: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas. 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.



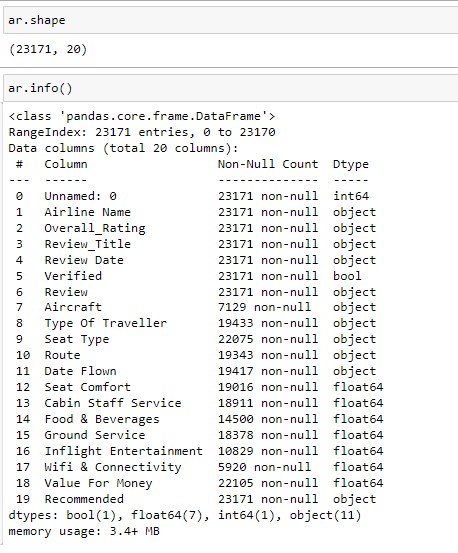
# Activity 4: Data Preparation

As we have understood how the data is, let's pre-process the collected data. The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps

* Handling missing values • Handling categorical data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps. **Activity 5: Handling missing values**

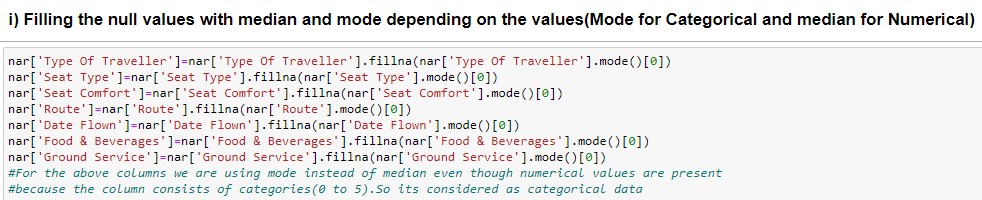
* Let’s find the shape of our dataset first. To find the shape of our data,

the ar.shape method is used. 

* To find the data type, ar.info() function is used.
* For checking the null values, ar.isnull() function is used.
* To sum those null values, we append .sum() to the above null function.
* From the image we found that there are null values present in our dataset. But still there is no need to run that command since we can find it out through ar.info().
* So now we need to replace those null values based with median or mode values depending on feature type.
* But before that , we need to drop unnecessary columns so that we can create an efficient model. We used ar.drop([ ],axis=1) to remove unnecessary rows

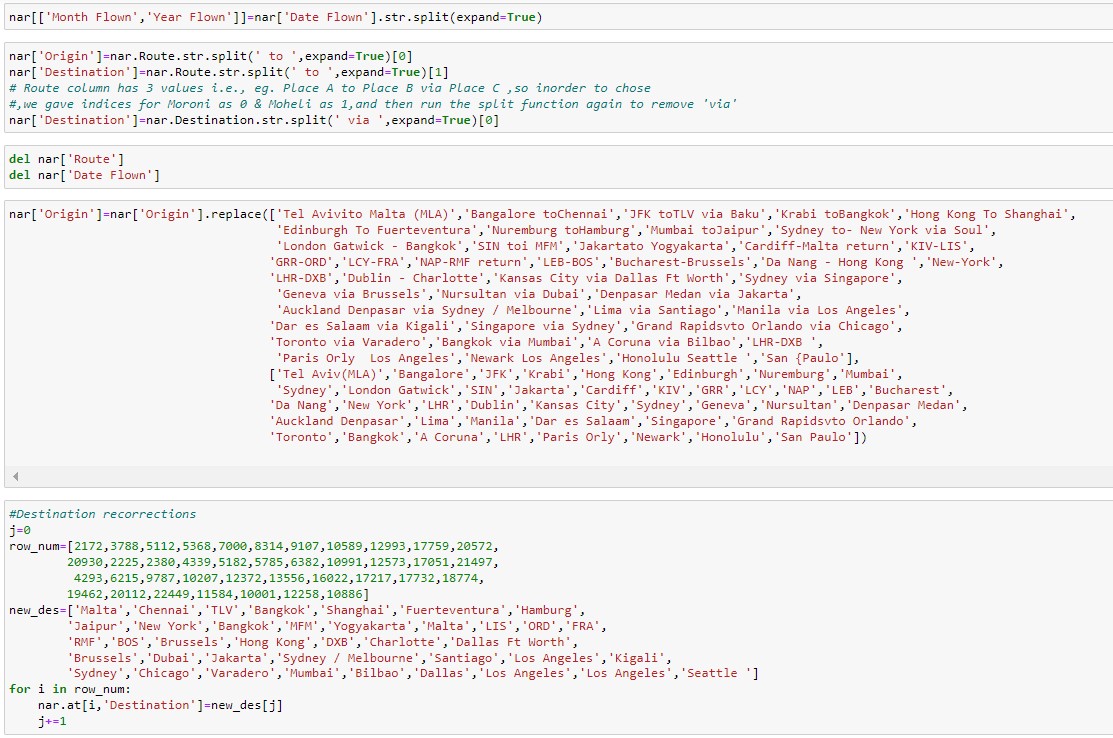


In the above dataset check also for any unnecessary values, upon checking the Overall\_Rating column we found some irrelevant value, so we replace it with relevant value. And later fill all the null values.

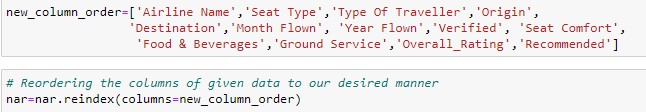


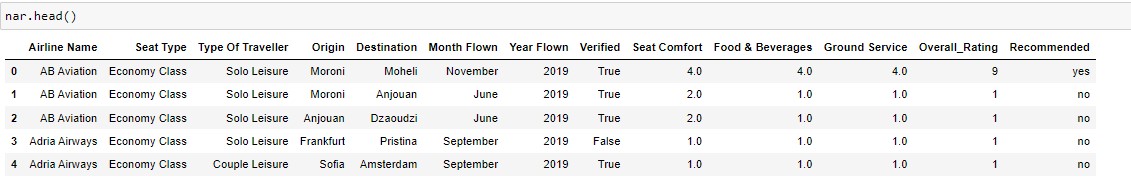
Later , we will modify the existing columns depending on our requirement.

In our dataset I have modified the Date flown and Route columns.

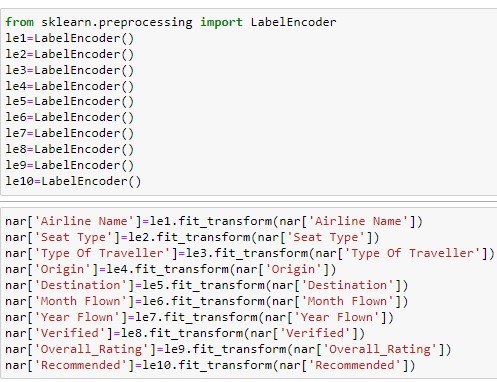


After correcting all the values in the respective features as shown in the above images, we shall re order the columns for our convenience. The resultant dataset looks as below.





**Activity 6: Handling Categorical Values**

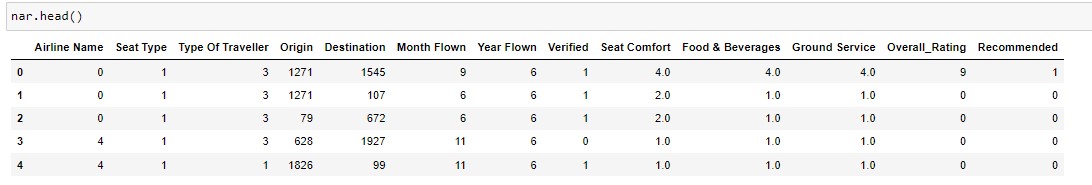


As we can see our dataset has categorical data, we must convert the categorical data to integer encoding or binary encoding. To convert the categorical features into numerical features we use encoding techniques. There are several techniques but, in our project, we are using label encoding. But prior encoding we need to do EDA.

In our project, categorical features are Airline Name, Seat Type,

Type Of Traveller, Origin, Destination, Month Flown, Year Flown, Verified, Overall\_Rating, Recommended. Label encoding is done for those columns.

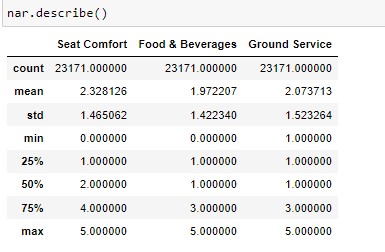
Dataset will be converted as below image



**Milestone 2: Exploratory Data Analysis**

# Activity 1: Descriptive statistics

Descriptive analysis is to study the basic features of data with the statistical

process. 

Here pandas have 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.

# Activity 2: Visual analysis

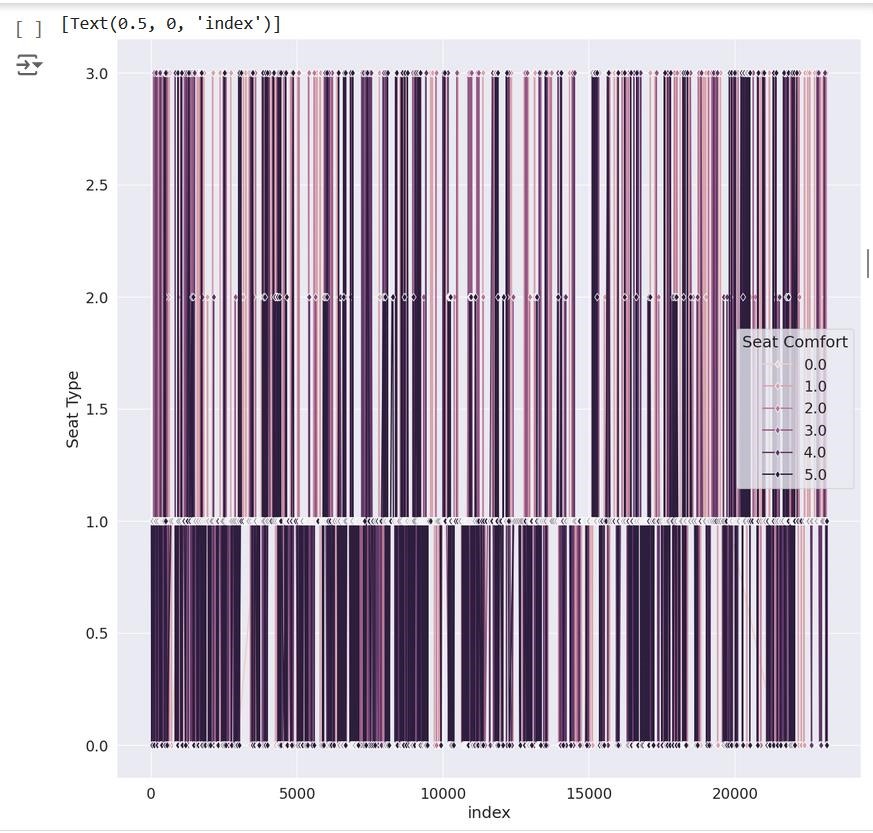
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

# Activity 3: Univariate Analysis

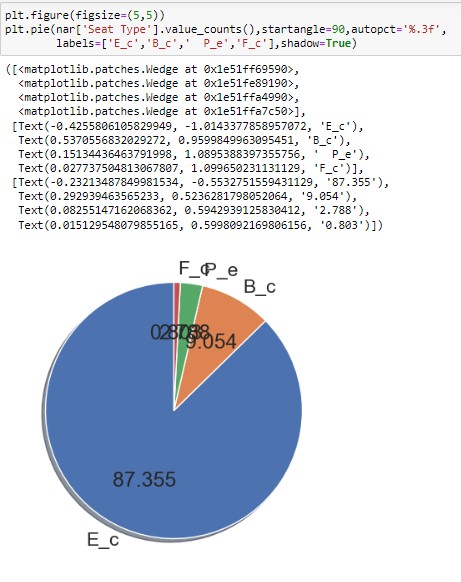
In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and

countplot.

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. Matplotlib function have many plots. We are using bar plot for our dataset.

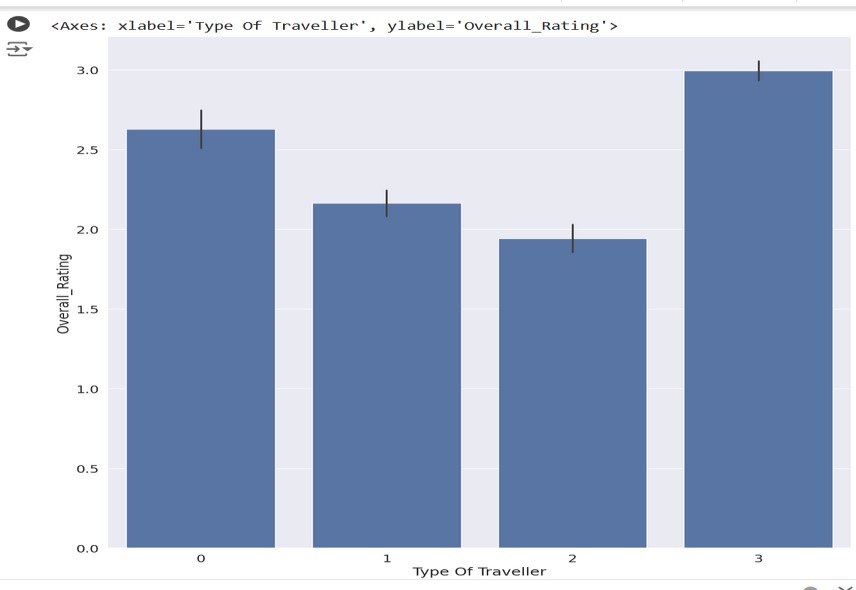


Note: In our dataset we used lineplot, bar plot, pieplot and not using distplot and countplot because our dataset is not having continuous values



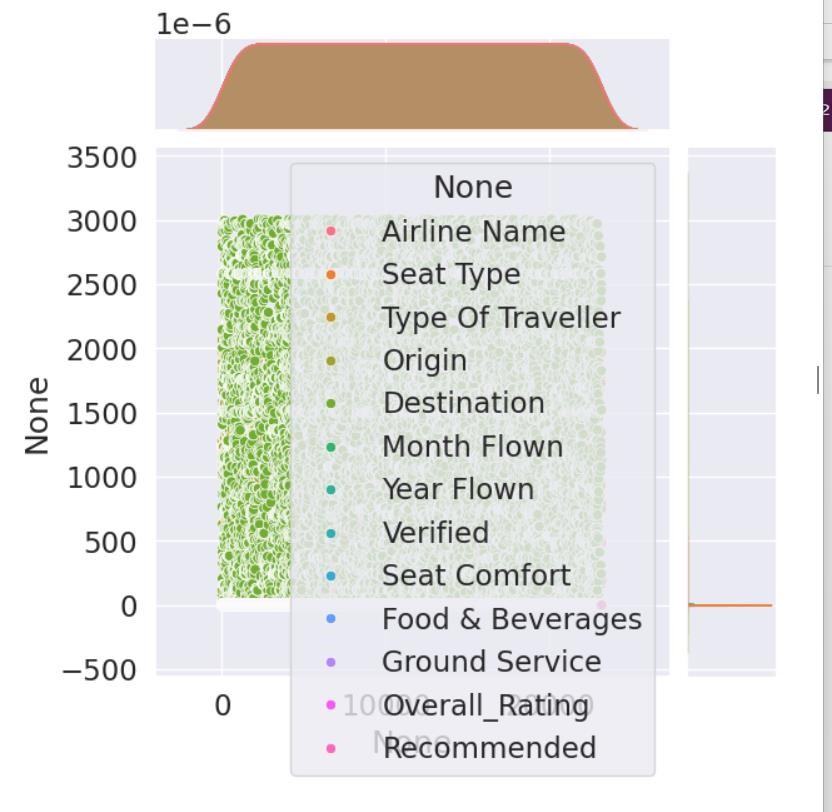
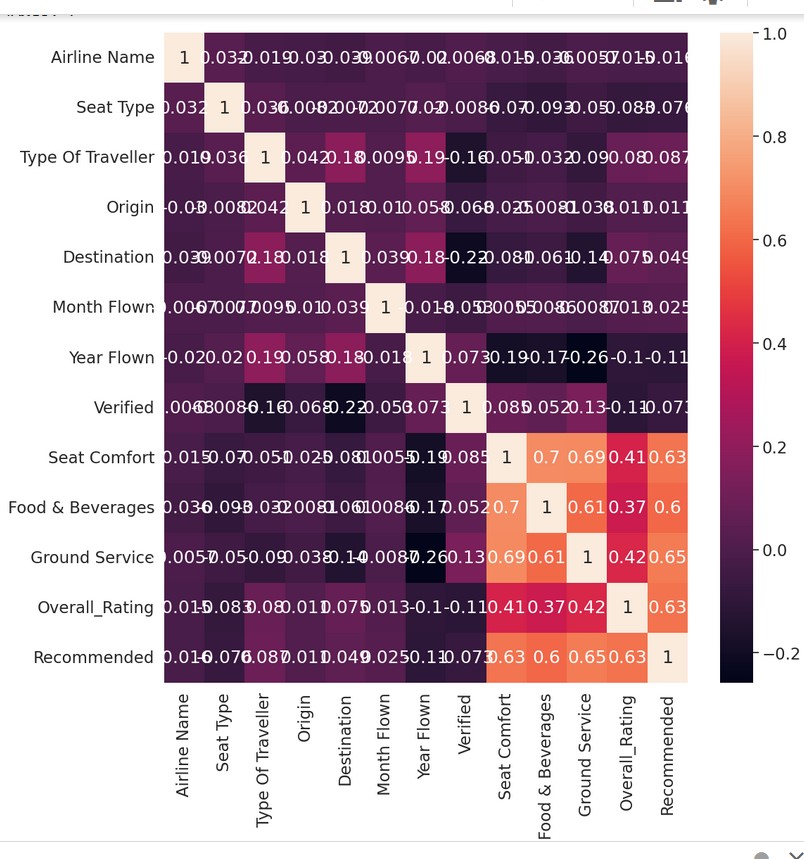
# Activity 4: Bivariate Analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between Type Of Traveller, Overall\_Rating.



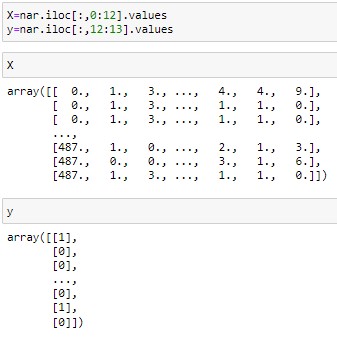
**Activity 5: Multivariate Analysis**

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used jointplot and heatmap(for finding correlation) from seaborn package.



# Milestone 3: Model Building Activity 1: Splitting Data Into Train And Testing

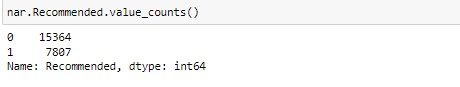
Now let’s split the Dataset into train and test sets. First split the dataset into X and y and then split the dataset. Here X and y variables are

created.

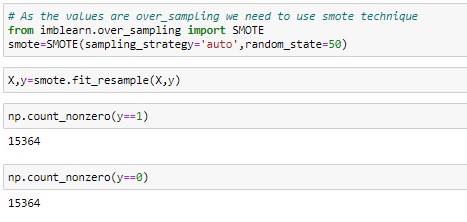
On X variable, nar is passed with dropping the target variable. And on y target variable is passed.

Basically, in target variable some values repeat more often than the other kind of values. To remove that imbalanced data, we will use SMOTE(Synthetic Minority Over Sampling Technique).

After checking ,we got to know that there is an imbalance of values in target variable.

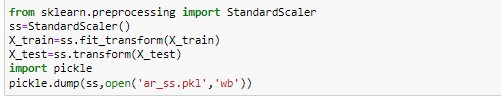


We can observe that after doing SMOTE the count of each value in target variable became equal.



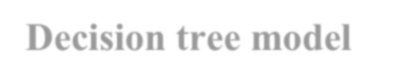
For splitting training and testing data we are using train\_test\_split() function from sklearn. As parameters, we are passing x, y, test\_size, random\_state.

After splitting the train and test data, we shall use StandardScaler function to remove the outliers of our dataset. And also save that model with the help of pickle function

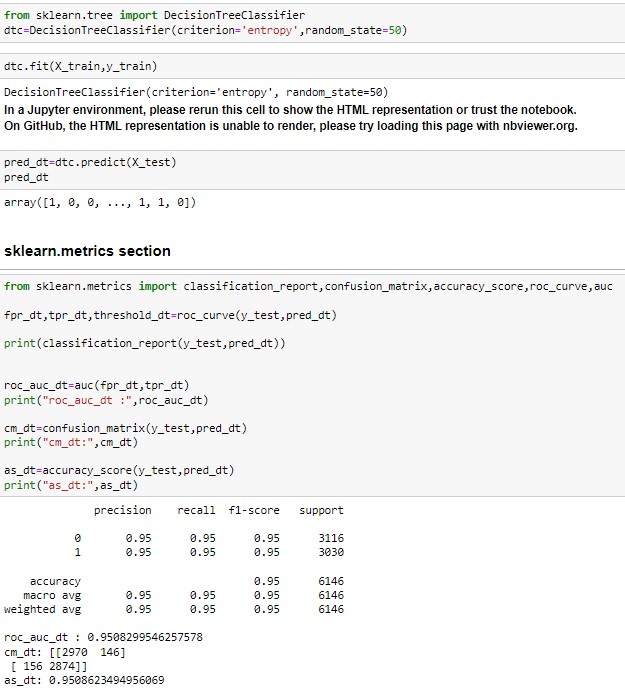


# Activity 2: Training The Model In Multiple Algorithms

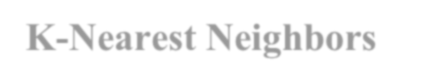
Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.



**Decision tree model**



A function named Decision Tree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialized 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.



**K**

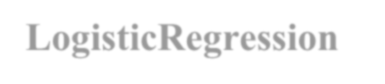
**-**

**Nearest Neighbors**

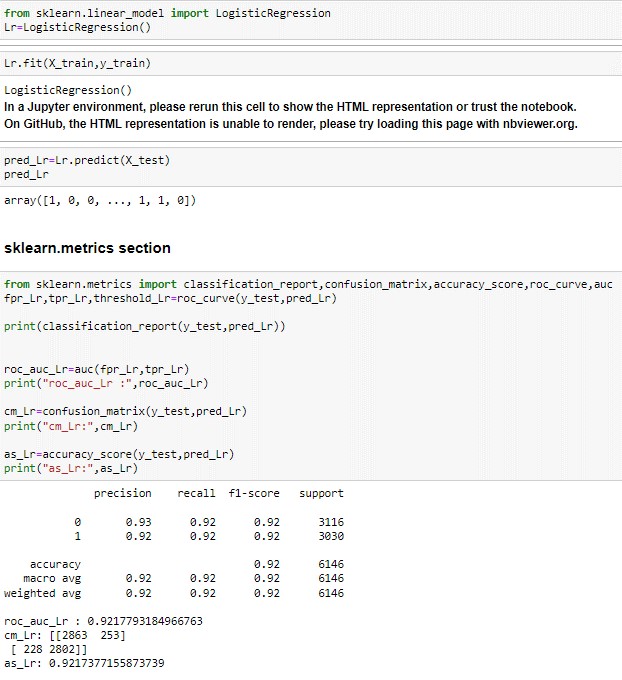


A function named KNeighborsClassifier is created and train and test data are passed as the parameters. Inside the function,

KNeighborsClassifier algorithm is initialized 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.

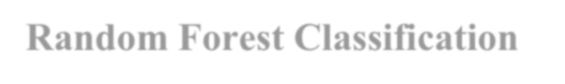


**LogisticRegression**



A function named Logistic Regression is created and train and test data are passed as the parameters. Inside the function, LogisticRegression algorithm is initialized 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. Naive Bayes Classification

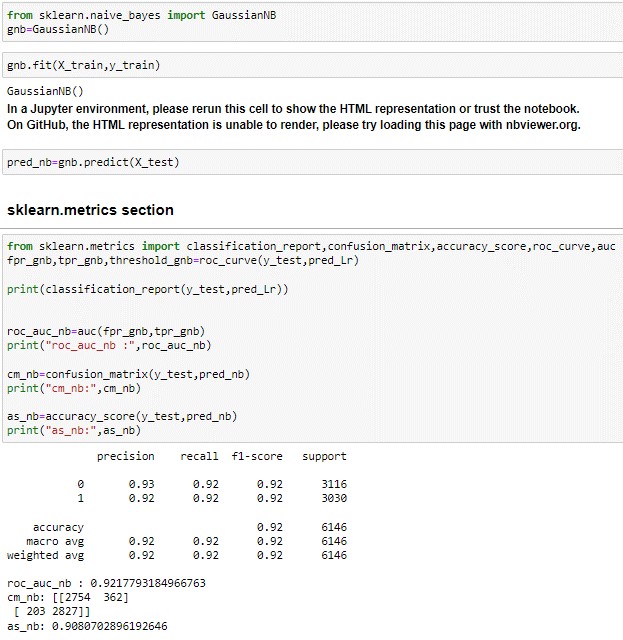
A function named GaussianNB is created and train and test data are passed as the parameters. Inside the function, GaussianNB algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.



**Random Forest Classification**

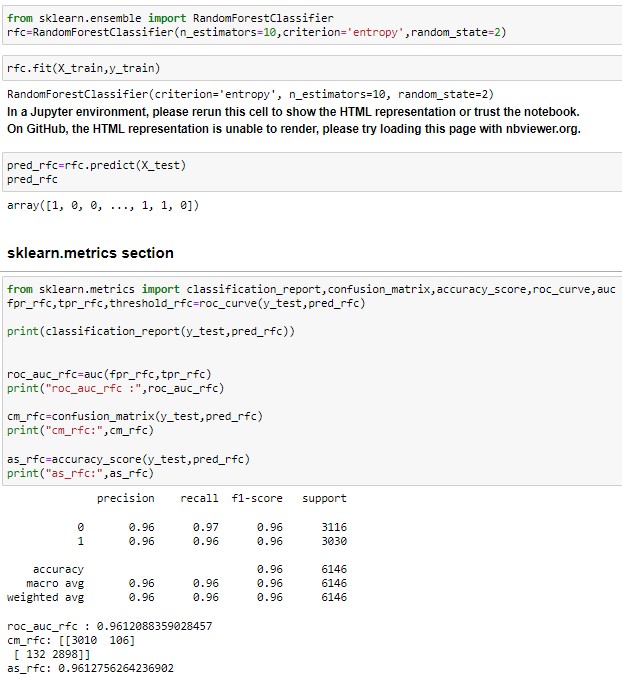


A function named RandomForestClassifier is created and train and test



data are passed as the parameters. Inside the function,

RandomForestClassifier algorithm is initialized 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 model, a confusion matrix and classification report is done.

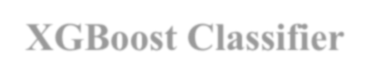


Support Vector Machine

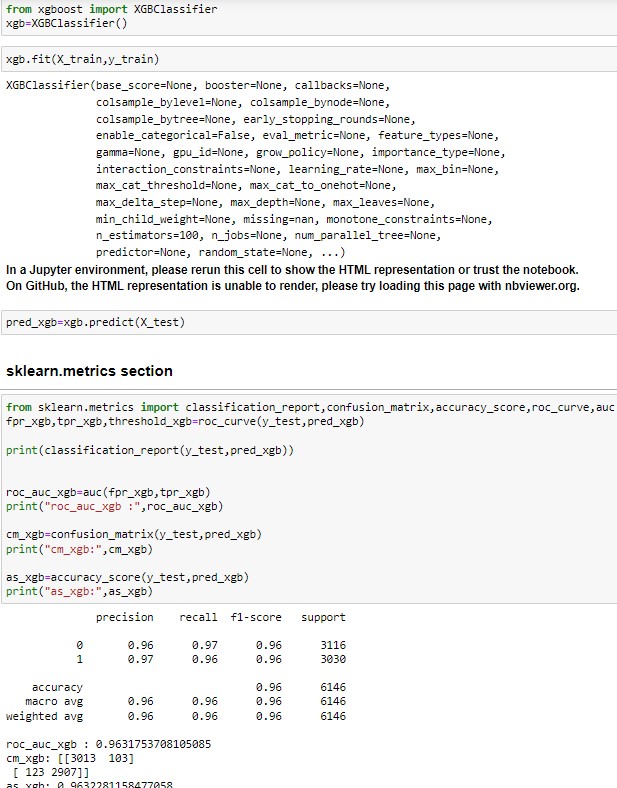
A function named SVC is created and train and test data are passed as the parameters. Inside the function, SVC algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

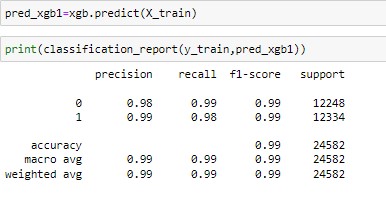


**XGBoost Classifier**



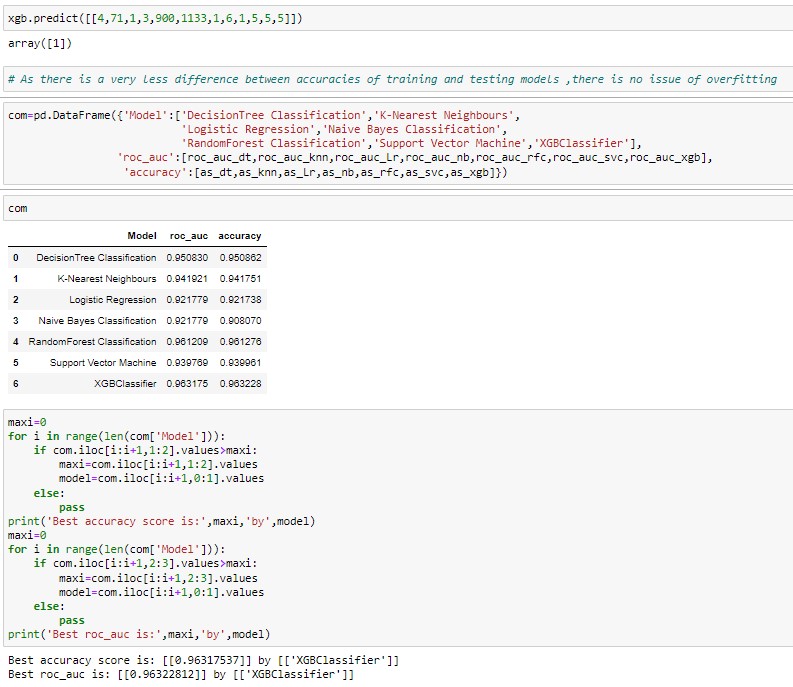
A function named XGBClassifier is created and train and test data are passed as the parameters. Inside the function, XGBClassifier algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done. Let us check the training accuracy also for this algorithm. We get to know that there is no issue of overfitting and at the same time both the testing and training accuracies are best. So will test this model.





# Activity 3 : Testing The Model

Here we have tested with XGBoost algorithm. You can test with all algorithm. With the help of predict() function.



**Milestone 4: 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 its 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.



# Activity 2: Integrate with Web Framework

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
* Run the web application

In this section, we will be building a web application that is integrated into 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
  + Run the web application

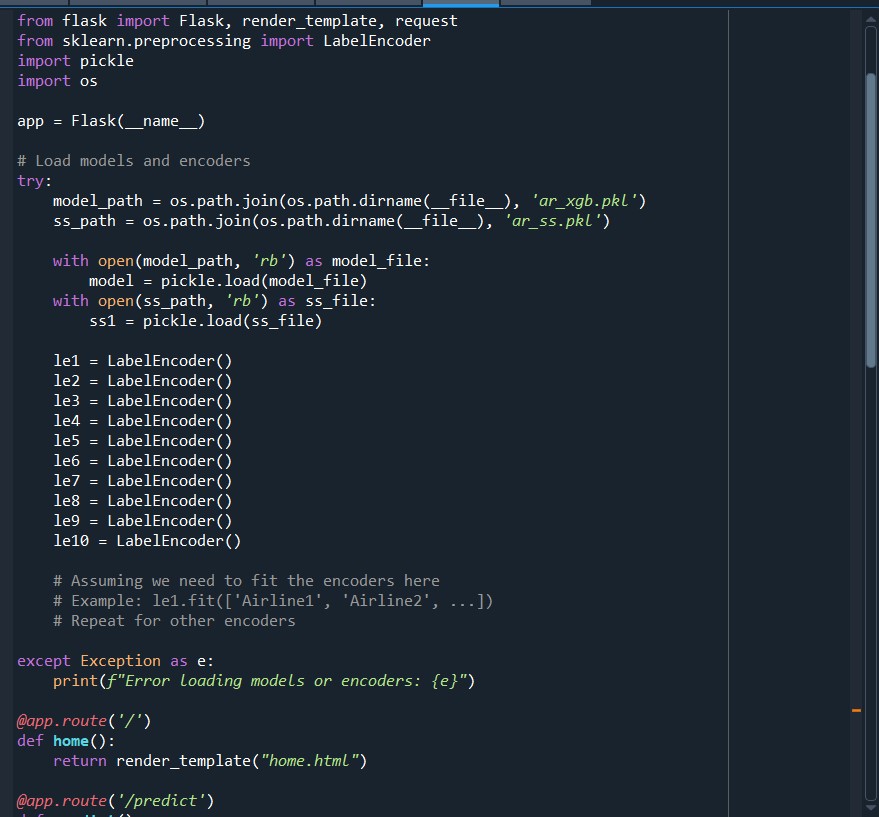
**Activity 3: Building Html Pages:**

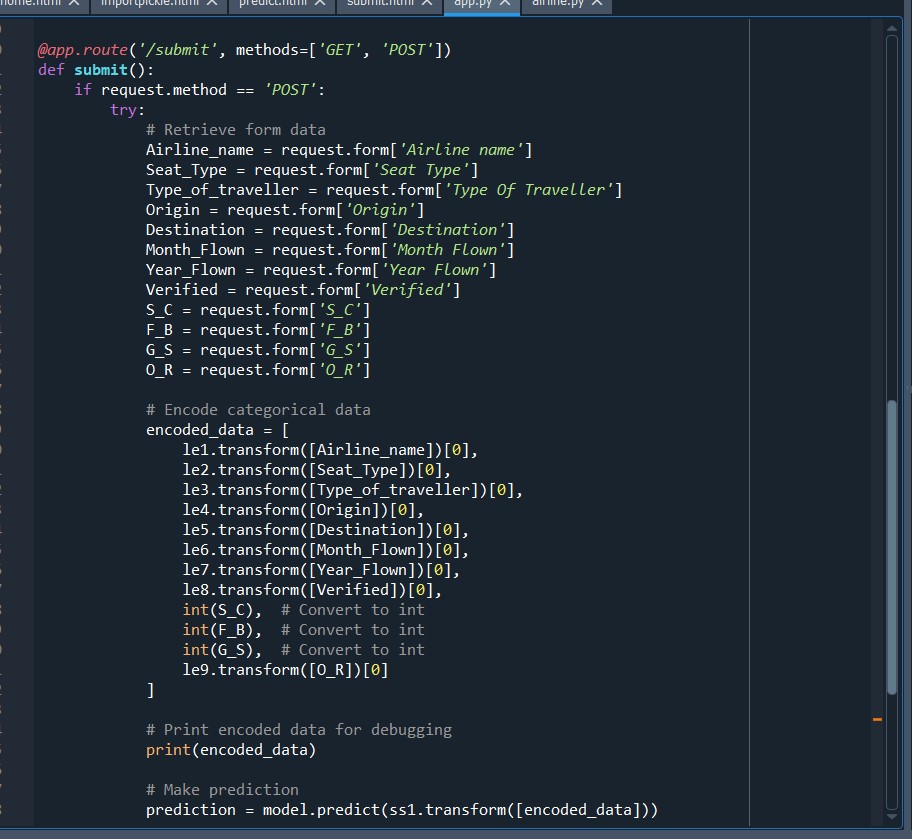
For this project create two HTML files namely and save them in the templates folder. Refer this [link f](https://drive.google.com/drive/folders/1K-C2uvRstV8x6bwsOY_BWBvXdzjbbfPF?usp=share_link)or templates

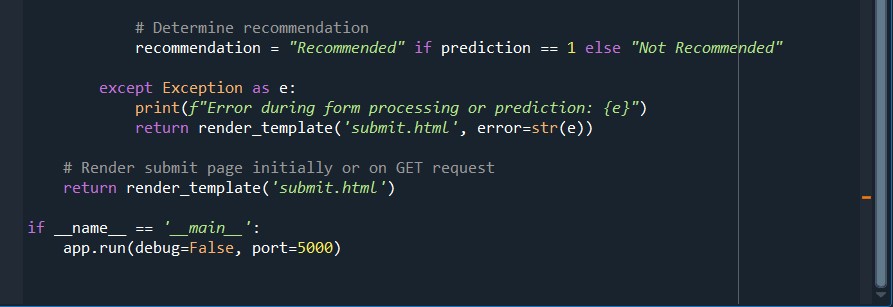
. home.html predict.html submit.html

**Activity 4: Build Python code:**

Importing the 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.







**Render HTML page:**



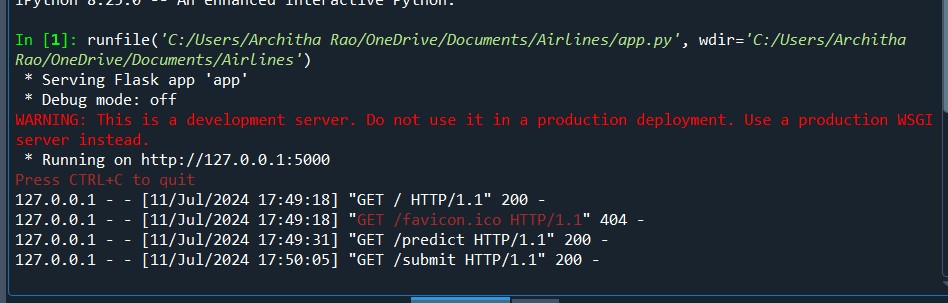
Here we will be using a declared constructor to route to the HTML page which we have created earlier. In the above example, ‘/’ URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method. Retrieves the value from UI:

Here we are routing our app to Guest() 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 be rendered to the text that we have mentioned in the index.html page. Set app.run(debug=True) so that we can edit.

# Activity 5: Run the web 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.

Enter the inputs, click on the Check It button, and see the result/prediction on the web



Now, Go the web browser and write the localhost URL

(http://127.0.0.1:5000) to get the below result

