

## 

**Flight Delay Prediction for**

**Aviation industry using Machine Learning**

**TEAM MEMBERS:**

* **V.MANJULA**
* **K.SARANYA**
* **S.KEERTHANA**
* **V.THENMOZHI**

**INTRODUCTION**:

As people increasingly choose to travel by air, the amount of flights that fail to take off on time also increases. This growth exacerbates the crowded situation at airports and causes financial difficulties within the airline industry. Air transportation delay indicates the lack of efficiency of the aviation system. It is a high cost to both airline companies and their passengers. According to the estimation by the Total Delay Impact Study, the total cost of air transportation delay to air travelers and the airline industry in 2007 was $32.9 billion in the US, resulting in a $4 billion reduction in GDP [[**1**](https://dl.acm.org/doi/fullHtml/10.1145/3497701.3497725#bib1)]. Therefore, predicting flight delays can improve airline operations and passenger satisfaction, which will result in a positive impact on the economy.

**1.1 Over view:**

**Project description**:

Over the last twenty years, air travel has been increasingly preferred among travelers, mainly because of its speed and in some cases comfort. This has led to phenomenal growth in air traffic and on the ground. An increase in air traffic growth has also resulted in massive levels of aircraft delays on the ground and in the air. These delays are responsible for large economic and environmental losses. According to, taxi-out operations are responsible for 4,000 tons of hydrocarbons, 8,000 tons of nitrogen oxides and 45,000 tons of carbon monoxide emissions in the United States in 2007. Moreover, the economic impact of flight delays for domestic flights in the US is estimated to be more than $19 Billion per year to the airlines and over $41 Billion per year to the national economy In response to growing concerns of fuel emissions and their negative impact on health, there is active research in the aviation industry for finding techniques to predict flight delays accurately in order to optimize flight operations and minimize delays.

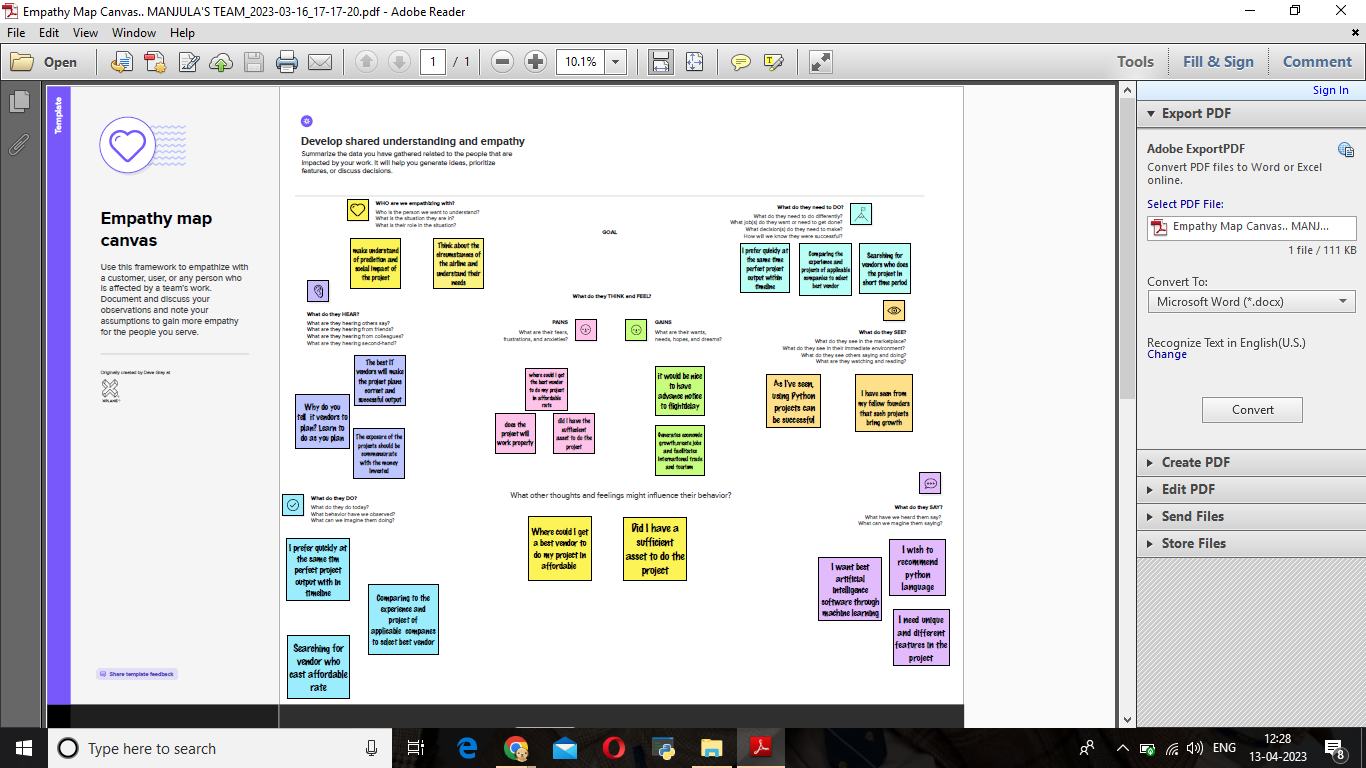
Using a machine learning model, we can predict flight arrival delays. The input to our algorithm is rows of feature vector like departure date, departure delay, distance between the two airports, scheduled arrival time etc. We then use decision tree classifier to predict if the flight arrival will be delayed or not. A flight is delayed when difference between scheduled and actual arrival times is greater than 15 minutes. Furthermore, we compare decision tree classifier with logistic regression and a simple neural network for various figures of merit. Finally, it will be integrated to web based application

**1.2 Purpose:**

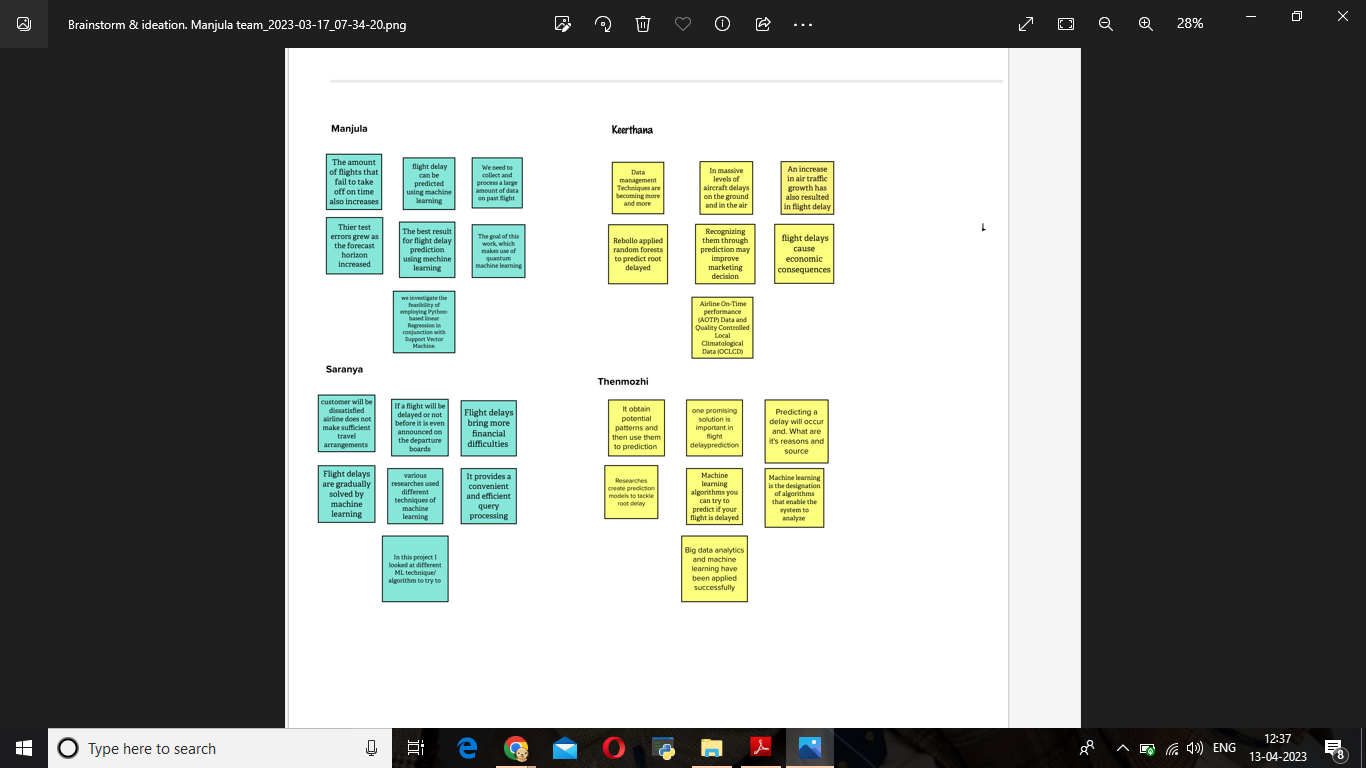
Therefore, predicting flight delays can **improve airline operations and passenger satisfaction**, which will result in a positive impact on the economy. In this study, the main goal is to compare the performance of machine learning classification algorithms when predicting flight delays.

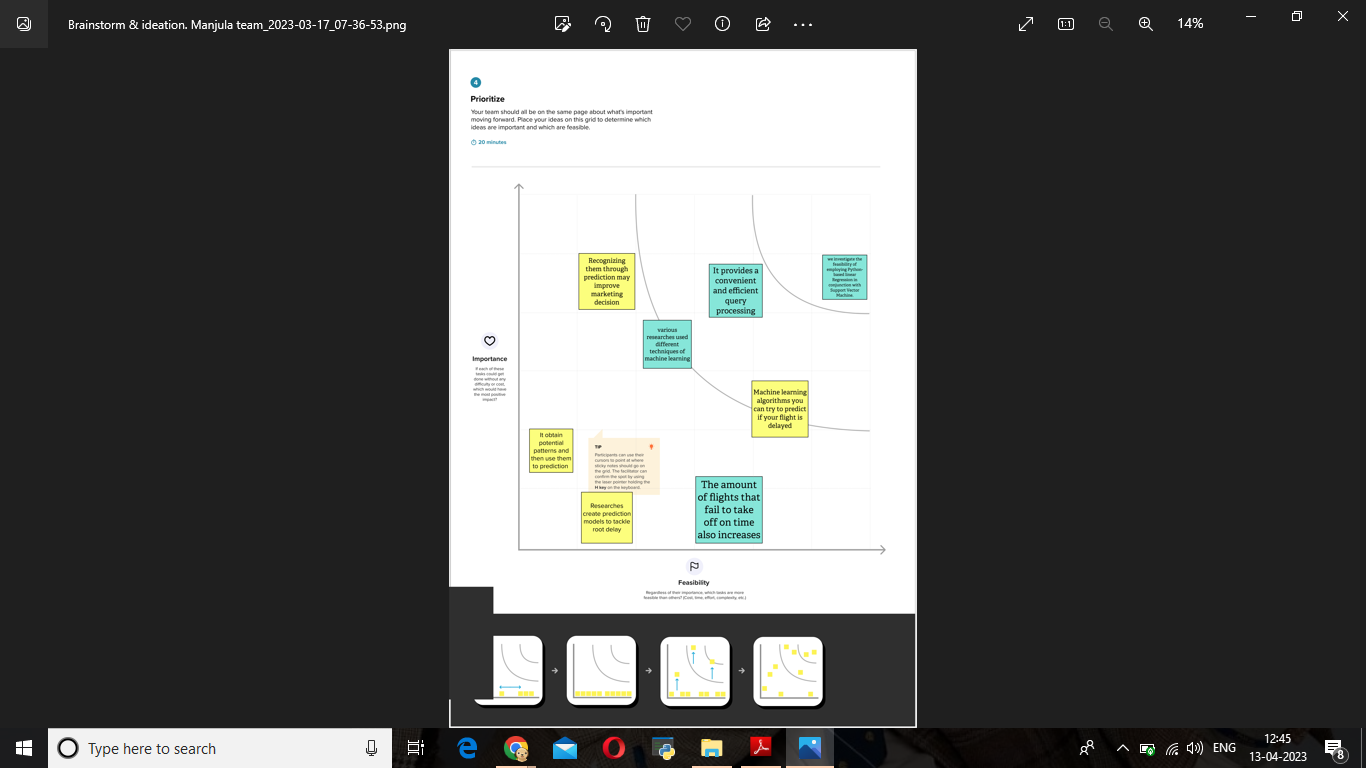
**Problem Definition & Design Thinking**

**2.1 Empathy map:**

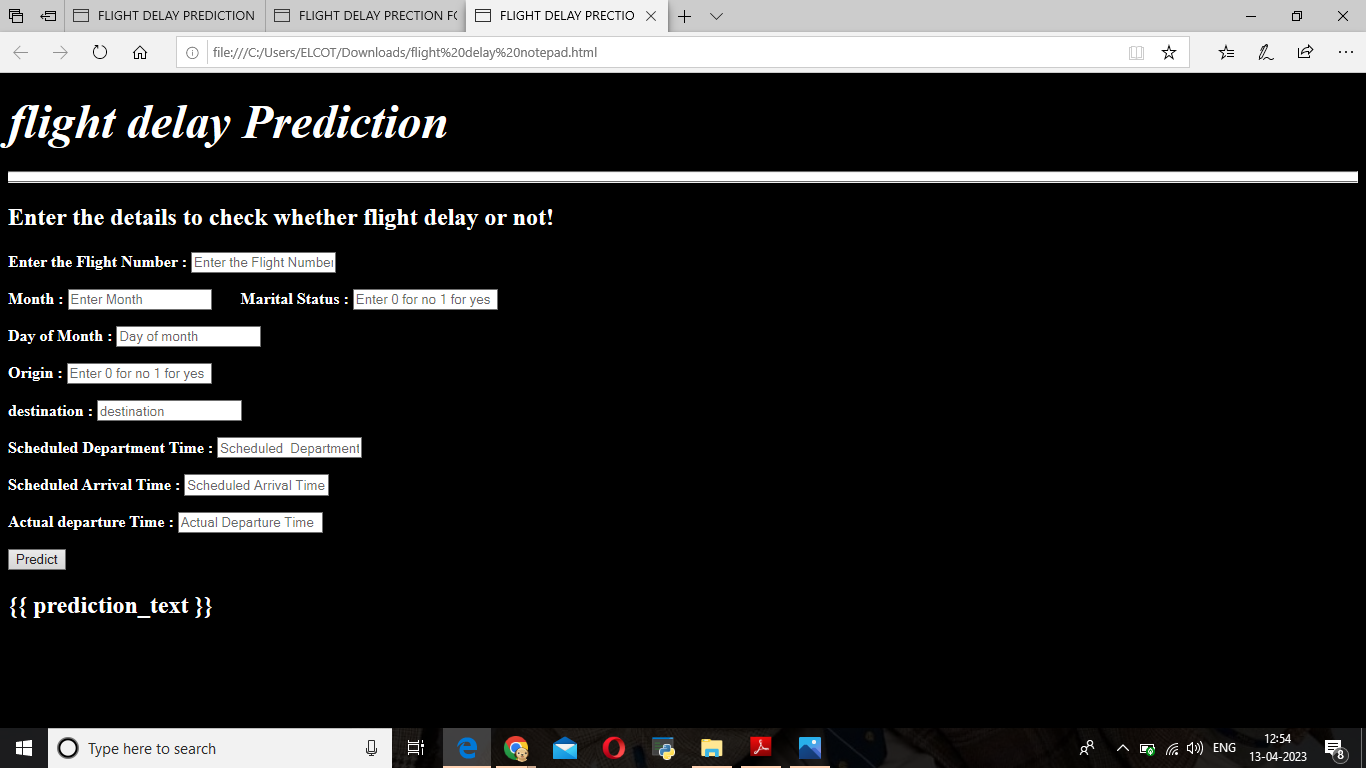


**2.2 Ideation & Brainstorming Map**





**Result:**



***Advantages & Disadvantages:***

***Advantages:***

* *Therefore , predicting flight delays can improve airline operations and passenger satisfaction , which will result in a positive impact on the economy.*

***Disadvantages:***

* *Carriers attribute flight delay to serveral causes such as bad weather conditions , airport congestion ,airspace congestion ,and use of smaller aircraft by airlines.*
* *These delays and cancellations tarnish the airline’s reputation , often resulting in loss of demand by passengers.*

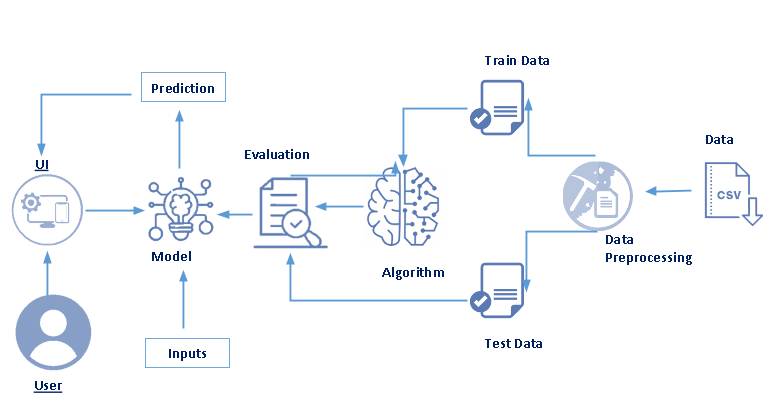
***Appliction:***

* ***Open a anaconda prompt from the start menu***
* ***Nagative to the folder where your python script is.***
* ***Now type “python app.py” command***

***Conclusion:***

The paper performed a prediction of the occurrence of flight delays by adapting it into a machine learning problem. A supervised machine learning approach in the form of binary classification was used for the prediction. Seven algorithms were used for delay prediction, and four measures were used for algorithms performance evaluation. Due to the imbalanced nature of the data set, evaluation measures were weighted to eliminate the dominant effect of non-delayed flights over delayed flights. After applying classifiers to the delay prediction, the values of their four measures were compared to evaluate the performance of each model.

***Technical* Architecture:**



**Project Flow:**

* User interacts with the UI to enter the input.
* Entered input is analysed by the model which is integrated.
* Once model analyses the input the prediction is showcased on the ul

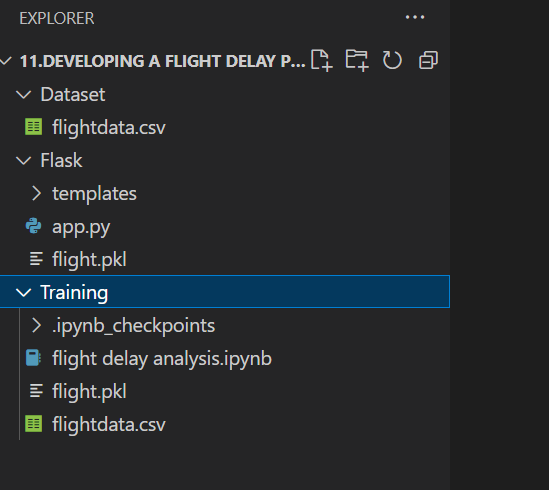
**To accomplish this, we have to complete all the activities listed below,**

* Define Problem / Problem understanding
* Specify the business problem
* Business requirements
* Literature Survey
* Social or Business impact.
* Data Collection & Preparation
* Collect the dataset
* Data Preparaton
* Exploratory Data Analysis
* Descriptive statistical
* Visual Analysis
* Model Building
* Training the model in multiple algorithms
* Testing the model

* Project Demonstration & Documentation
* Record explanation Video for project end to end solution
* Project Documentation-Step by step project development procedure

# **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.py for scripting
* Flight.pkl our saved model. Further we will use this model for flask integration.
* Training folder contains a model training file

**APPENDIX**

**A.source code**

**Milestone** 1

### **Define Problem / Problem Understanding**

In this milestone, we go through the problem understanding.

### **ACTIVITY 1: Specify the business requirements**

OVER the last twenty years, air travel has been increasingly preferred among travelers, mainly because of its speed and in some cases comfort. This has led to phenomenal growth in air traffic and on the ground. An increase in air traffic growth has also resulted in massive levels of aircraft delays on the ground and in the air. These delays are responsible for large economic and environmental losses. According to, taxi-out operations are responsible for 4,000 tons of hydrocarbons, 8,000 tons of nitrogen oxides and 45,000 tons of carbon monoxide emissions in the United States in 2007. Moreover, the economic impact of flight delays for domestic flights in the US is estimated to be more than $19 Billion per year to the airlines and over $41 Billion per year to the national economy In response to growing concerns of fuel emissions and their negative impact on health, there is active research in the aviation industry for finding techniques to predict flight delays accurately in order to optimize flight operations and minimize delays.

Using a machine learning model, we can predict flight arrival delays. The input to our algorithm is rows of feature vector like departure date, departure delay, distance between the two airports, scheduled arrival time etc. We then use decision tree classifier to predict if the flight arrival will be delayed or not. A flight is delayed when difference between scheduled and actual arrival times is greater than 15 minutes. Furthermore, we compare decision tree classifier with logistic regression and a simple neural network for various figures of merit. Finally, it will be integrated to web based application.

### **ACTIVITY 2: Business Requirements**

To predict flight delays using machine learning, you will need to collect and process a large amount of data on past flight delays. This data should include information such as the flight's departure and arrival times, the airline, the aircraft type, and the weather conditions at the departure and arrival airports. Once you have collected and cleaned the data, you can use a variety of machine learning techniques such as regression, decision trees, or neural networks to train a model that can predict flight delays based on this data. It is important to note that flight delay prediction is a highly complex task and requires a lot of data, but it is possible with the right resources.

### **ACTIVITY 3: Literature Survey**

To predict flight delays using machine learning, you will need to collect and process a large amount of data on past flight delays. This data should include information such as the flight's departure and arrival times, the airline, the aircraft type, and the weather conditions at the departure and arrival airports. Once you have collected and cleaned the data, you can use a variety of machine learning techniques such as regression, decision trees, or neural networks to train a model that can predict flight delays based on this data. It is important to note that flight delay prediction is a highly complex task and requires a lot of data.

The literature suggests that ML models, specifically decision tree, ANN and random forest models, have been used to predict flight delays with varying degrees of accuracy. Commonly used features include historical flight data, weather conditions, and airport operations. It also shows that a combination of data mining techniques can be used to identify the factors that contribute to flight delays.

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

The social and business impact of flight delay prediction using machine learning (ML) can be significant

From a social perspective, flight delay prediction can help improve the travel experience for passengers. By providing accurate and timely predictions of flight delays, passengers can make more informed decisions about their travel plans and potentially avoid delays or missed connections. This can lead to a reduction in travel-related stress and inconvenience.

From a business perspective, flight delay prediction can help airlines and airports improve their operations and reduce costs. By identifying and addressing the factors that contribute to flight delays, airlines and airports can take proactive measures to mitigate the impact of delays. This can lead to improved on-time performance, which can help airlines and airports attract and retain customers and increase revenue. Additionally, flight delay prediction can help airlines and airports optimize their staffing and resource allocation, resulting in cost savings.

**Milestone 2**

**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.

### **ACTVITY 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: [flightdata.csv - Google Drive](https://drive.google.com/file/d/1HNYx6fX5hvRDX43egcAAUsrQ9sccv4AR/view)

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

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

### **Importing the Libraries:**

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.

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.tree import DecisionTreeClassifier

from sklearn.ensemble import GradientBoostingClassifier,RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import RandomizedSearchCV

import imblearn

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

from google.colab import files

uploads=files.upload()

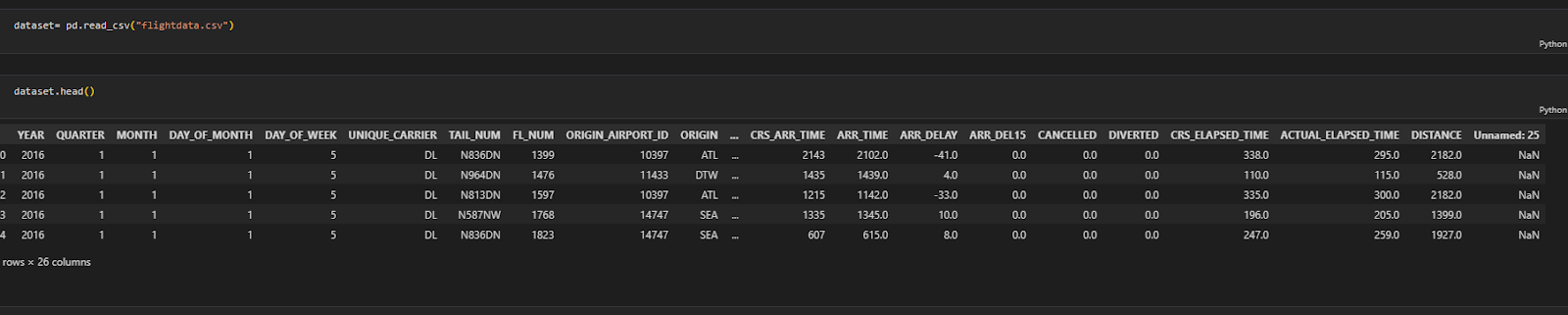
### **ACTIVITY 1.2: Read the Dataset**

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.

dataset=pd.read\_csv("/content/flightdata.csv")

Dataset.head()



### **ACTIVITY 2: 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.

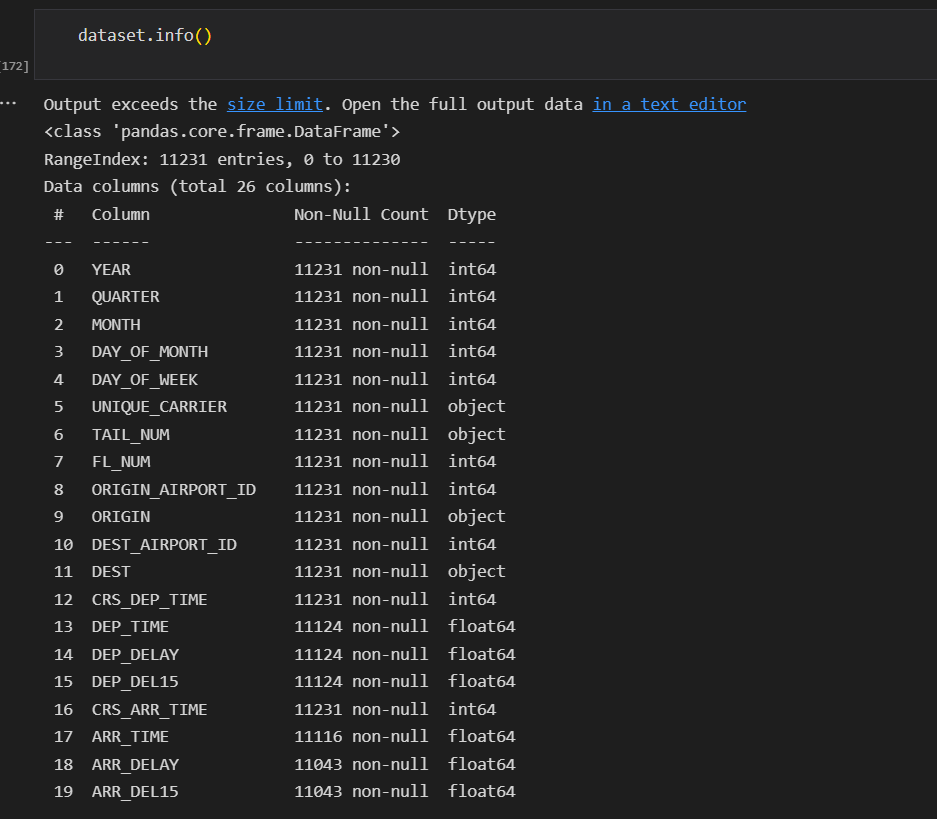
* Handling missing values
* Handling categorical data

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 2.1 : Handling Missing Values**

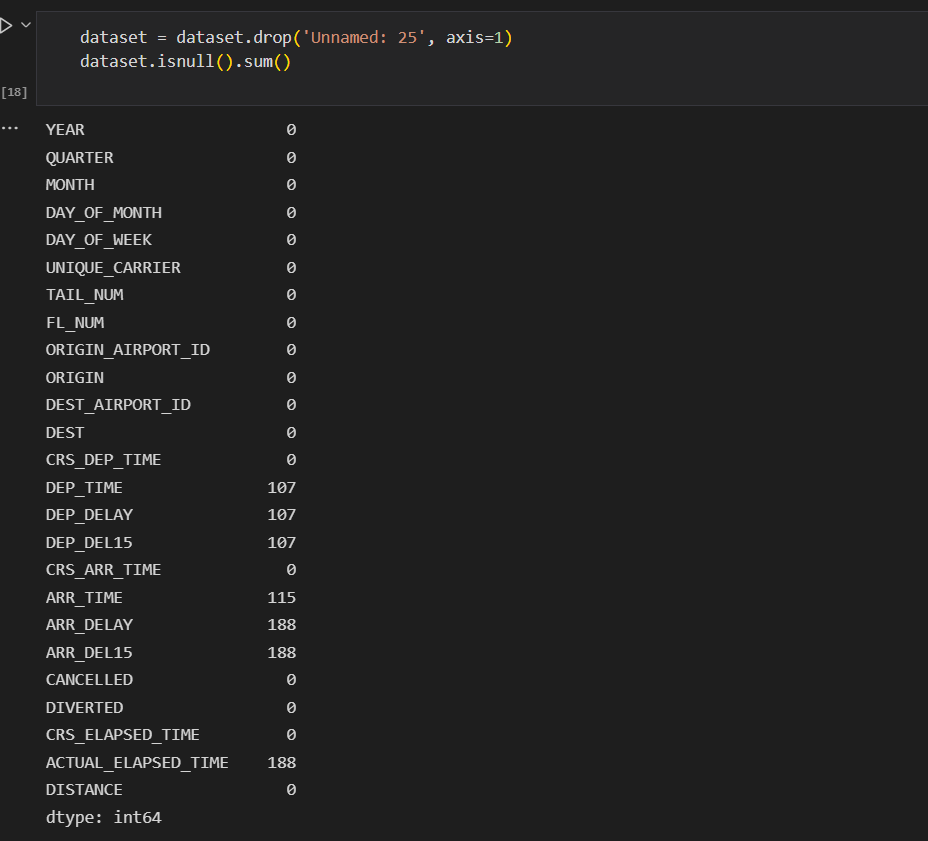
Let’s find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used.

Dataset.info()

.

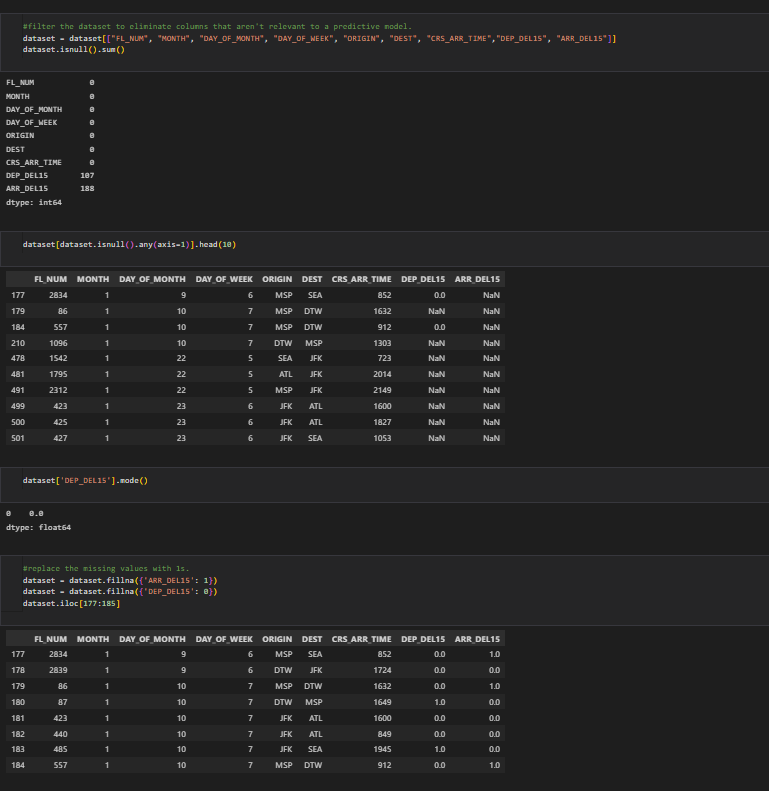
For checking the null values, df.isnull() function is used. To sum those null values we use. sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

dataset.info()

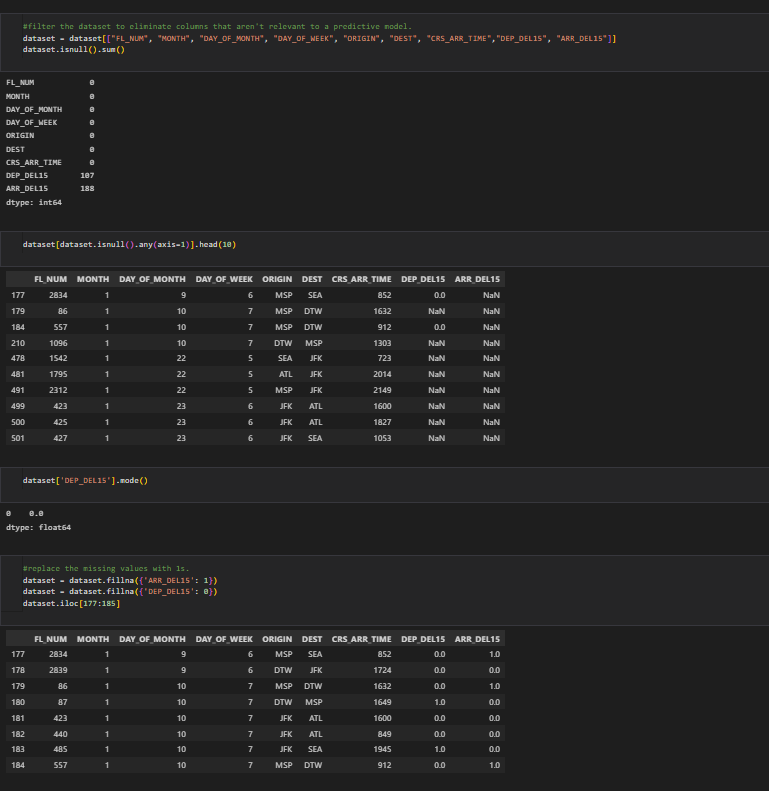


* We will fill in the missing values in the numeric data type using the mean value of that particular column and categorical data type using the most repeated value.

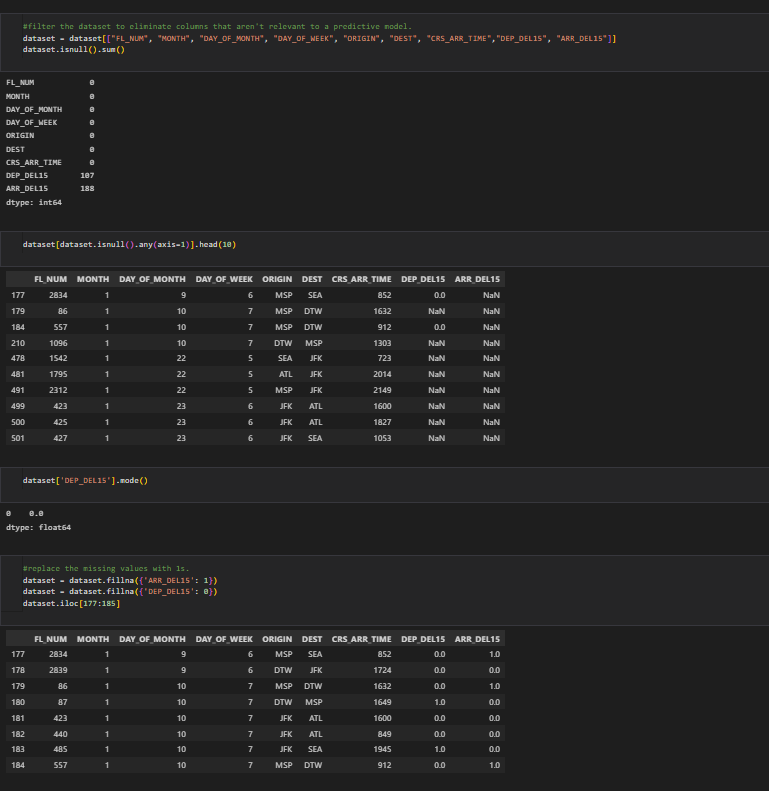
dataset = dataset [["FL\_NUM","MONTH","DAY\_OF\_MONTH","DAY\_OF\_WEEK","ORIGIN","DEST","CRS\_ARR\_TIME","DEP\_DEL15","ARR\_DEL15"]]

dataset.isnull().sum()

dataset[dataset.isnull().any(axis=1)].head(10)



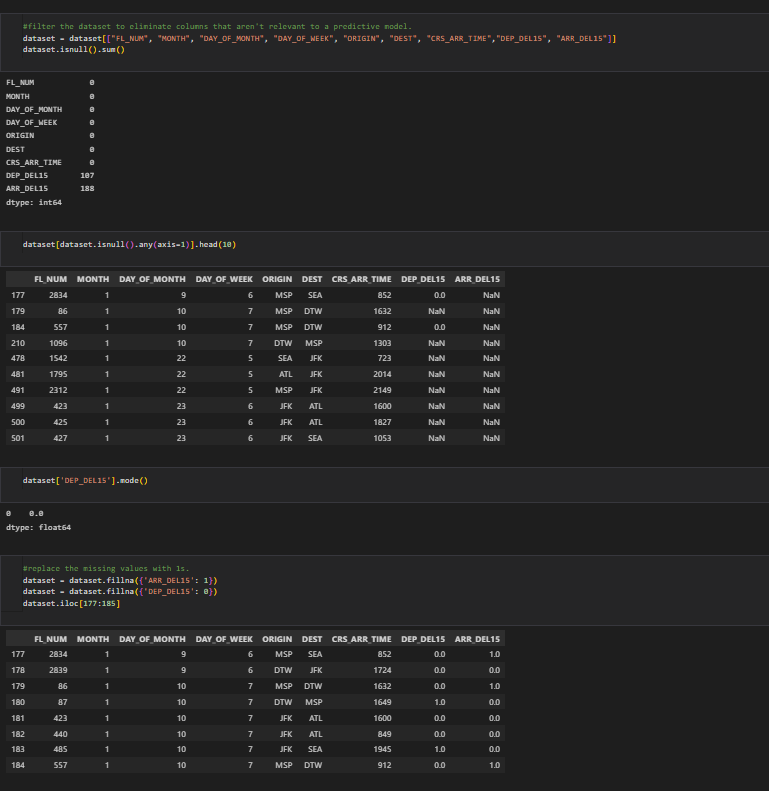
dataset['DEP\_DEL15'].mode()



dataset = dataset.fillna({'ARR\_DEL15':1})

dataset = dataset.fillna({"DEP\_DEL15":0})

dataset.iloc[177:185]



### **ACTIVITY 2.2 : Handling Categorical Values**

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

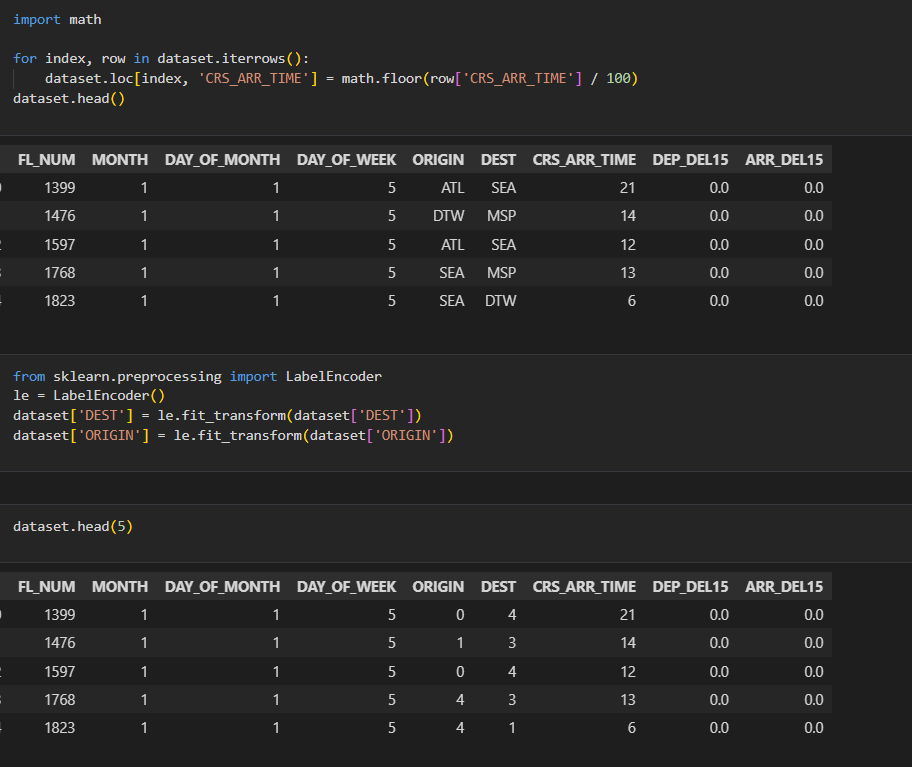
To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension.

Import math

for index,row in dataset.iterrows():

dataset.loc[index, 'CRS\_ARR\_TIME'] = math.floor(row['CRS\_ARR\_TIME'] /100)

dataset.head()



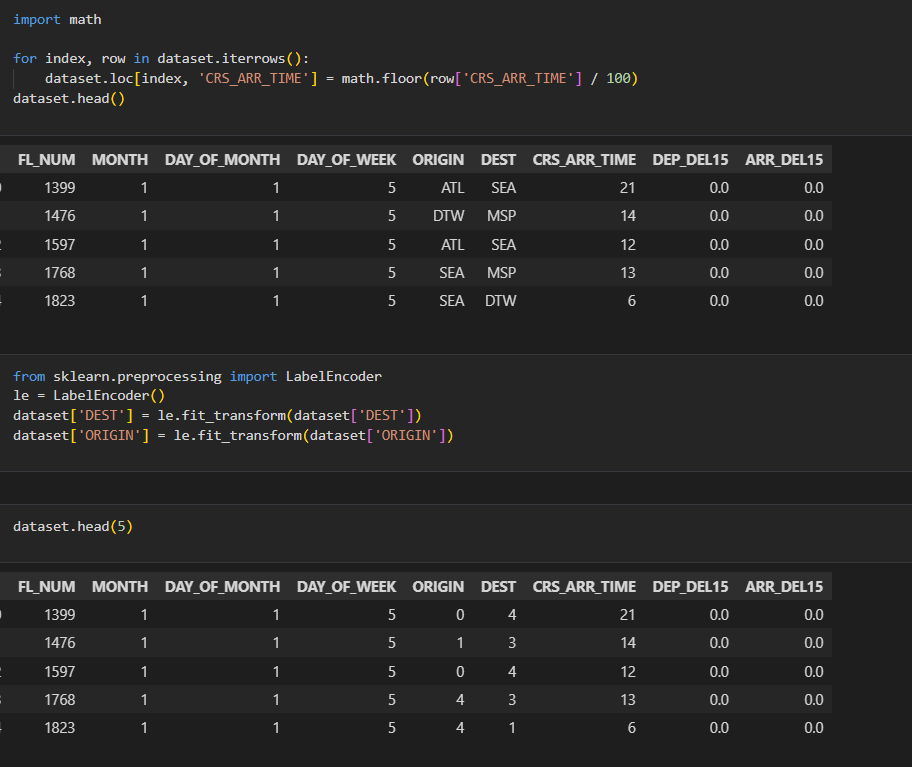
le = Lfrom sklearn.preprocessing import LabelEncoder

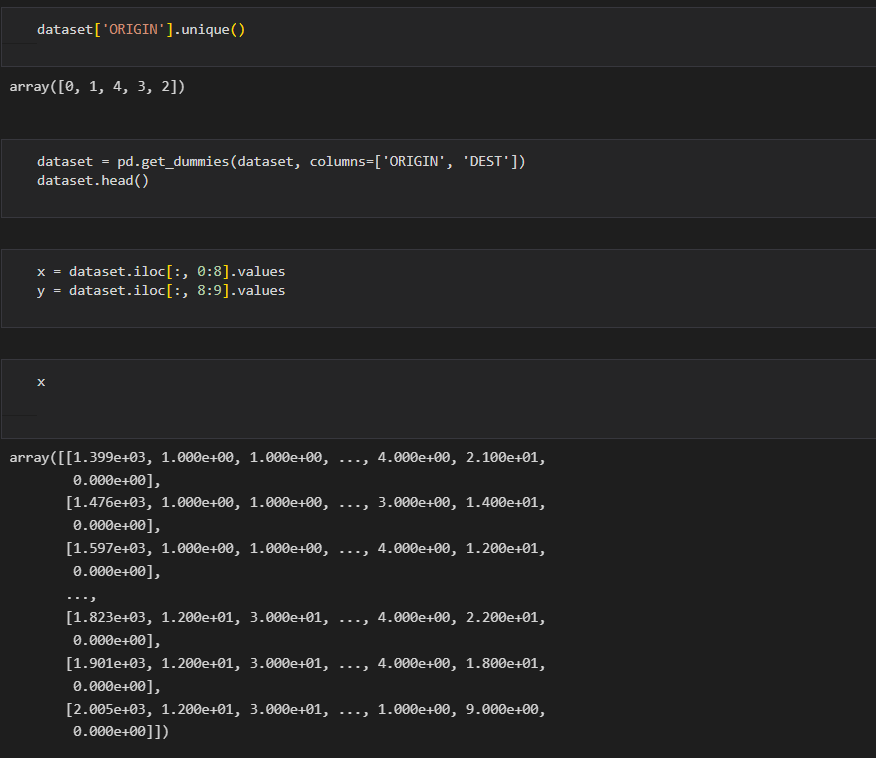
abelEncoder()

dataset['DEST'] = le.fit\_transform(dataset['DEST'])

dataset['ORIGIN'] = le.fit\_transform(dataset['ORIGIN'])

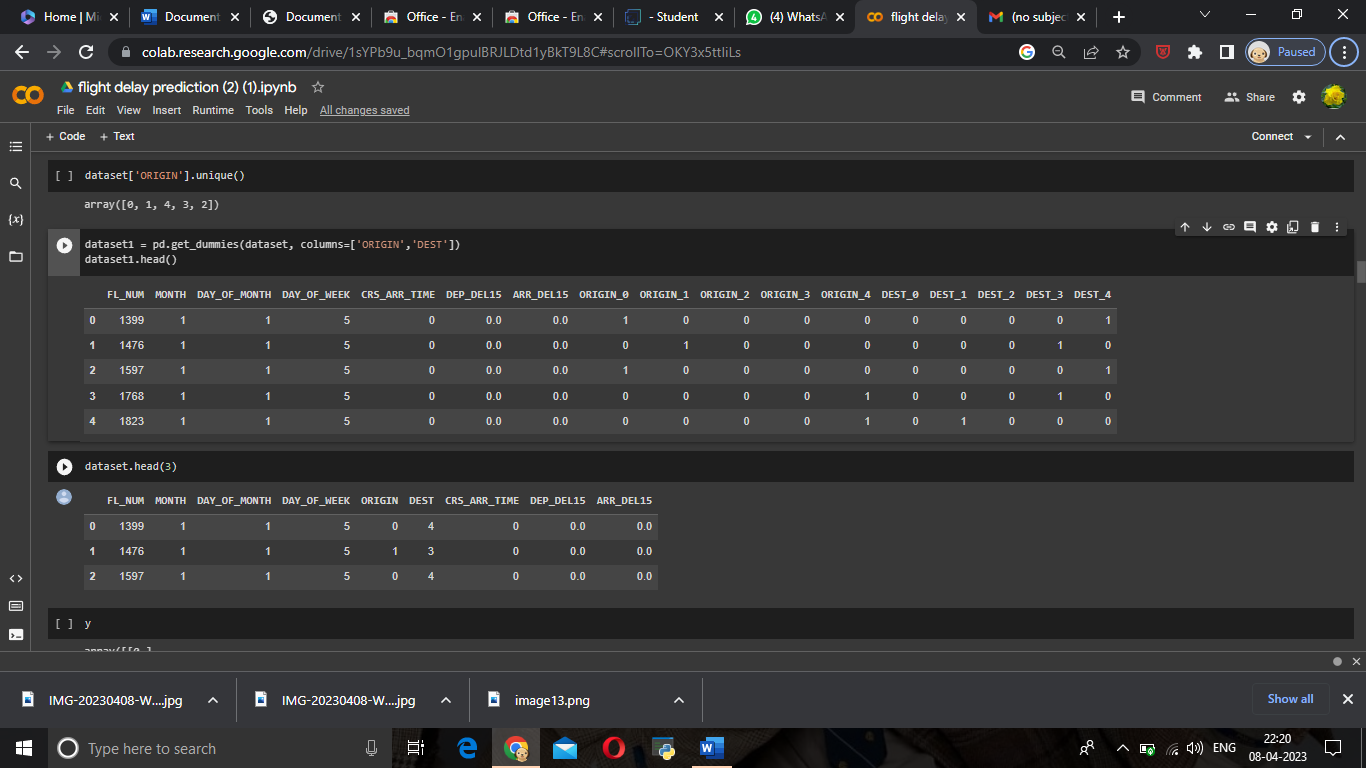
dataset.head(5)



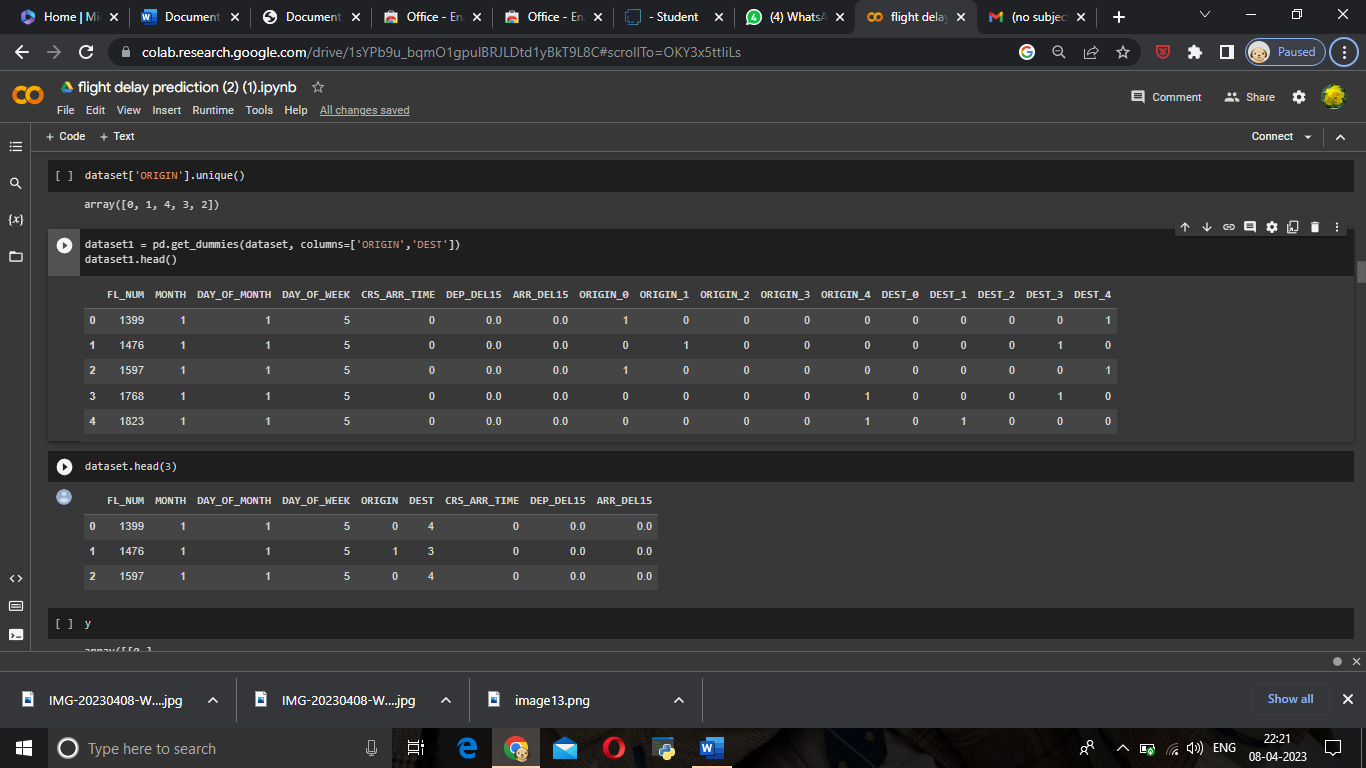
dataset['ORIGIN'].unique()  


dataset1 = pd.get\_dummies(dataset, columns=['ORIGIN','DEST'])

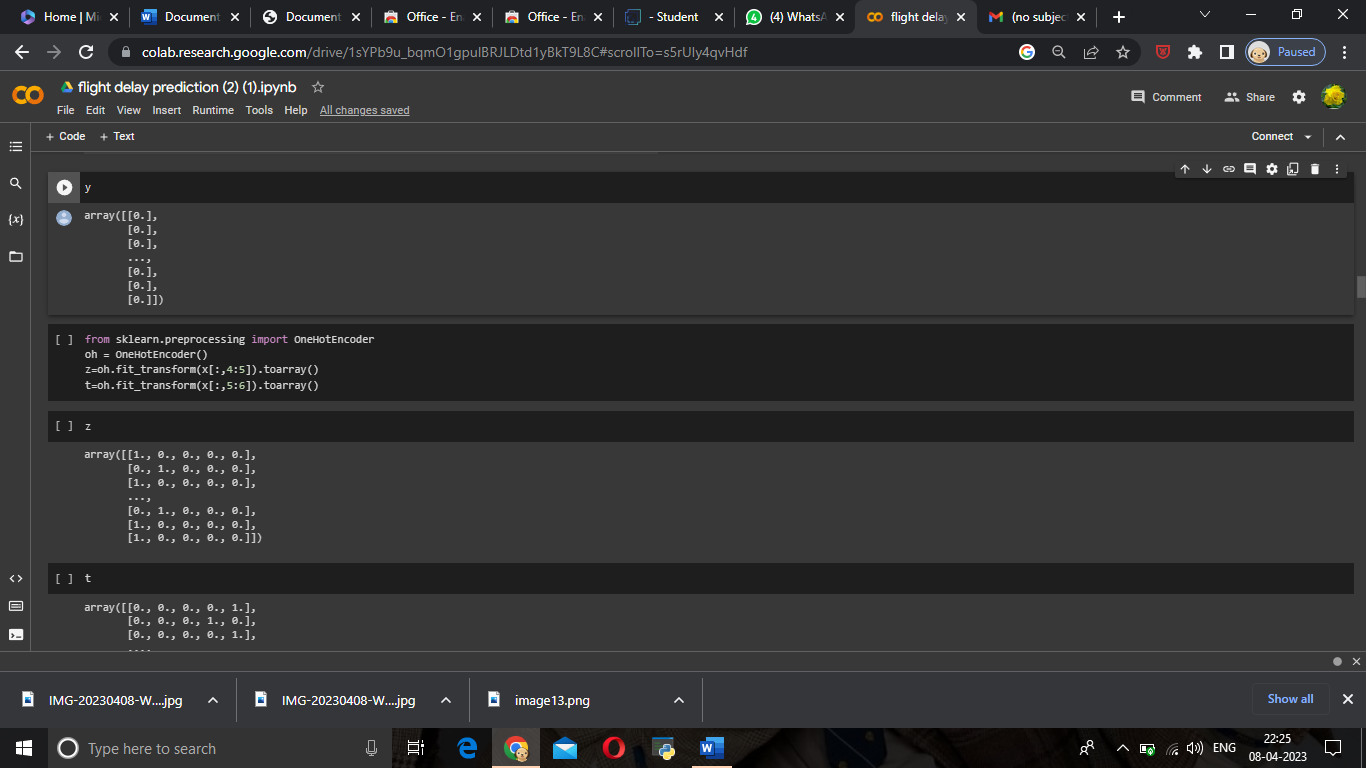
Dataset1.head()



dataset.head(3)



y



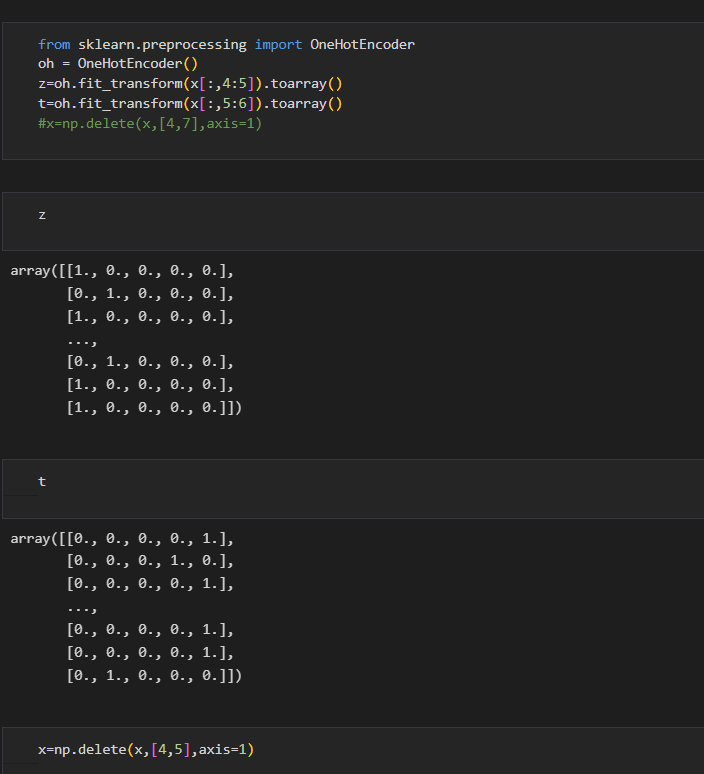
from sklearn.preprocessing import OneHotEncoder

oh = OneHotEncoder()

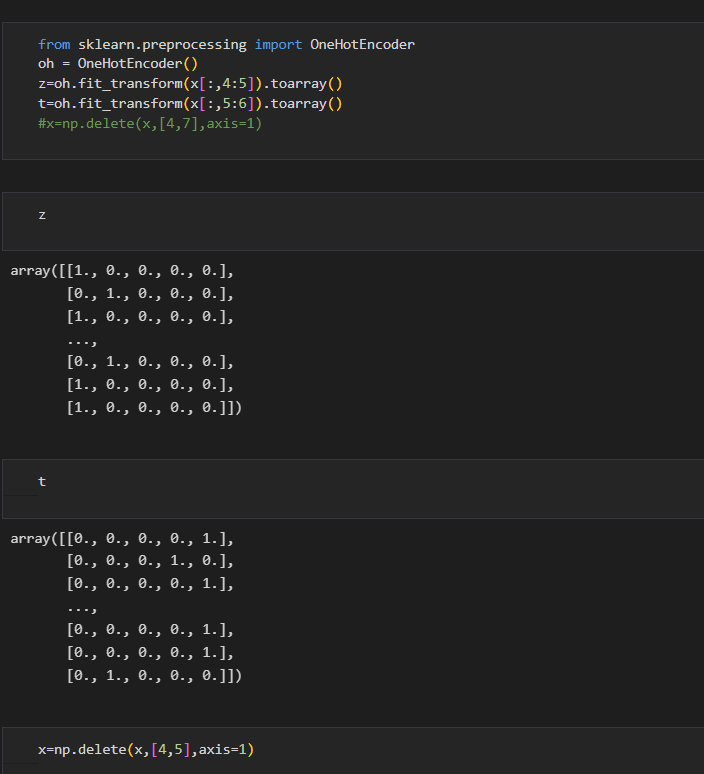
z=oh.fit\_transform(x[:,4:5]).toarray()

t=oh.fit\_transform(x[:,5:6]).toarray()

Z



t



x=np.delete(x,[4,5],axis=1)

**Milestone3:**

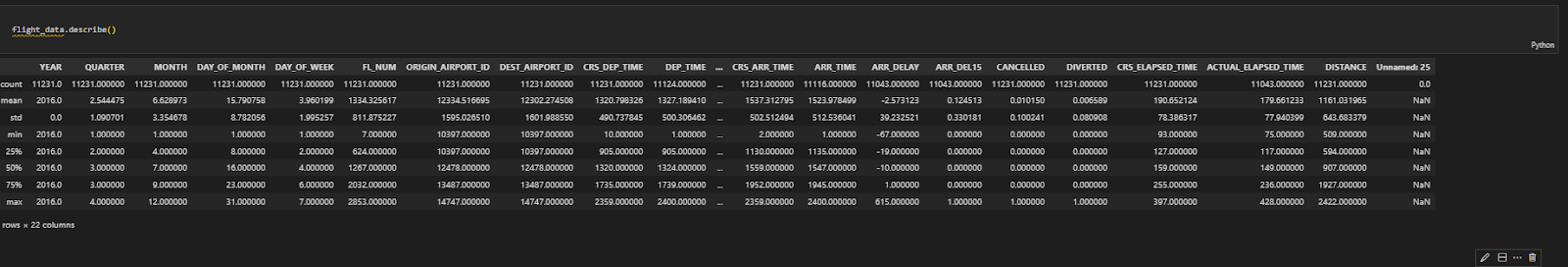
### **Exploratory Data Analysis**

In this milestone, we will go through the exploratory data analysis

### **ACTIVITY 1: 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.

dataset.describe()



### **ACTIVITY 2 : Visual Analysis**

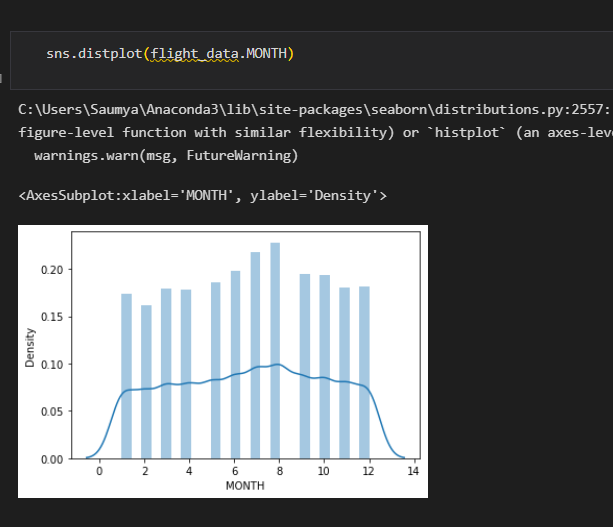
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

### **ACTIVITY 2.1 : 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(dataset.MONTH)

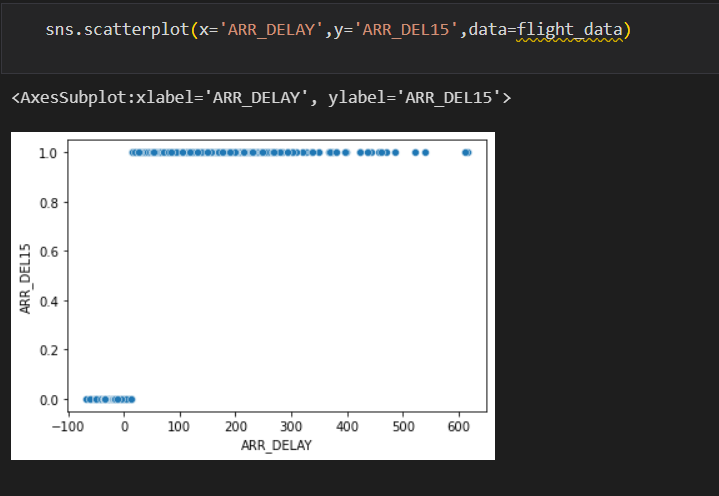


* In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.
* From the plot we came to know, Applicants income is skewed towards left side, where as credit history is categorical with 1.0 and 0.0

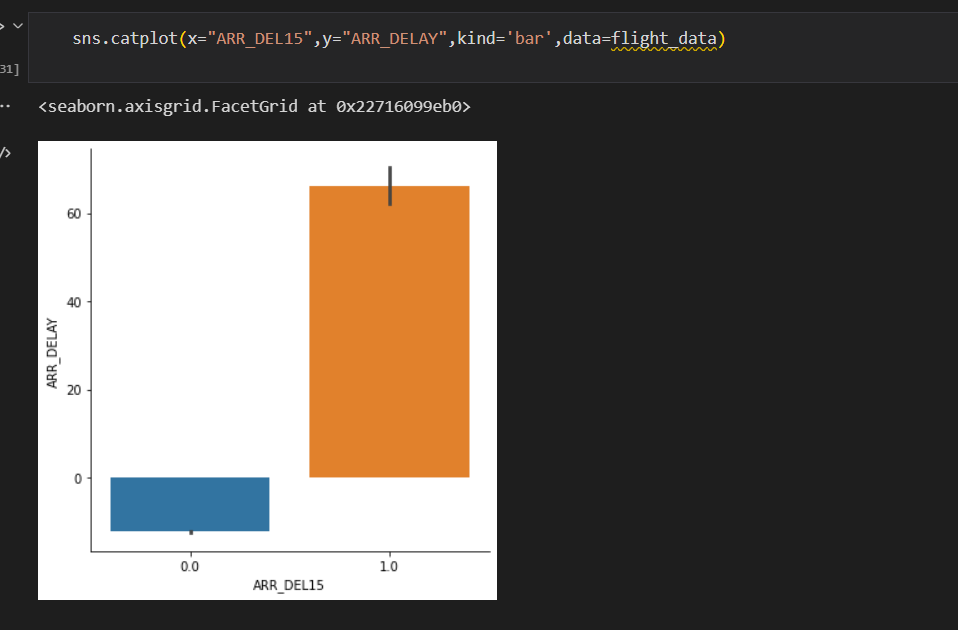
**Countplot:**

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot() , so you can compare counts across nested variables.

From the graph we can infer that , gender and education is a categorical variables with 2 categories , from gender column we can infer that 0-category is having more weightage than category-1,while education with 0,it means no education is a underclass when compared with category -1, which means educated .

**ACTIVITY 2.2: Bivariate Analysis** sns.scatterplot(x='DEP\_DEL15',y='ARR\_DEL15',data=dataset

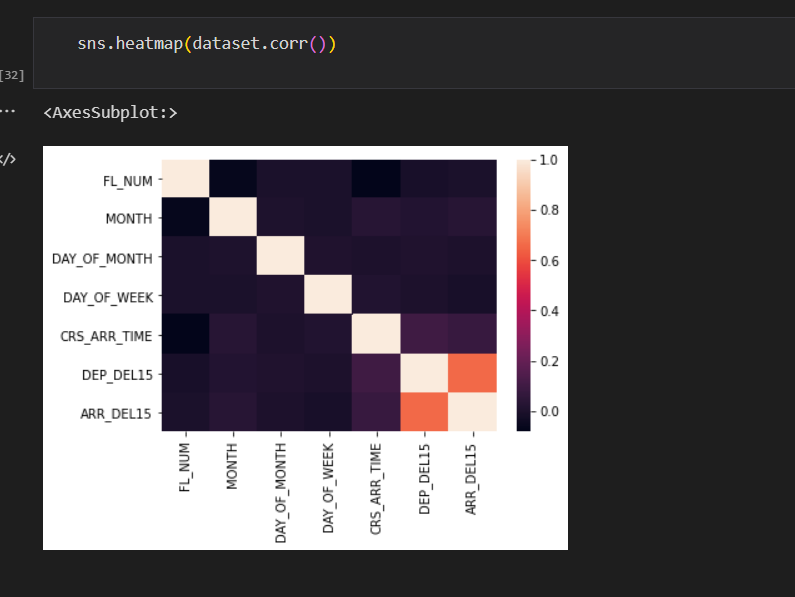
snscatplot.(x="ARR\_DEL15",y="DEP\_DEL15",kind='bar',data=dataset)



**ACTIVITY 2.3 : Multivariate Analysis**

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used a swarm plot from the seaborn package.

sns.heatmap(dataset.corr())



From the above graph we are plotting the relationship all the features.

**Splitting data into dependent and independent variables**

x = dataset.iloc[:,0:8].values

y = dataset.iloc[:,8:9].values

**Splitting data into train and test**

Now let’s split the Dataset into train and test sets  
Changes: first split the dataset into x and y and then split the data set  
Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using the train\_test\_split() function from sklearn. As parameters, we are passing x, y, test\_size, random\_state.

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=0)

from sklearn.model\_selection import train\_test\_split

train\_x, test\_x, train\_y, test\_y = train\_test\_split(dataset.drop('ARR\_DEL15', axis=1),dataset['ARR\_DEL15'],test\_size=0.2,random\_state=0)

x\_test.shape

  
  
x\_train.shape  


y\_test.shape



y\_train.shape



**Scaling the data**  
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.transform(x\_test)

### **Milestone 4: Model Building**

### **ACTIVITY 1: 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 four classification algorithms. The best model is saved based on its performance.

### **ACTIVITY 1.1 : Decision Tree Model**

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier 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. We are going to use x\_train and y\_train obtained above in train\_test\_split section to train our **Decision Tree Classifier** model. We’re using the fit method and passing the parameters as shown below.

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(random\_state = 0)

classifier.fit(x\_train,y\_train)



decisiontree = classifier.predict(x\_test)

decisiontree



from sklearn.metrics import accuracy\_score

desacc = accuracy\_score(y\_test,decisiontree)

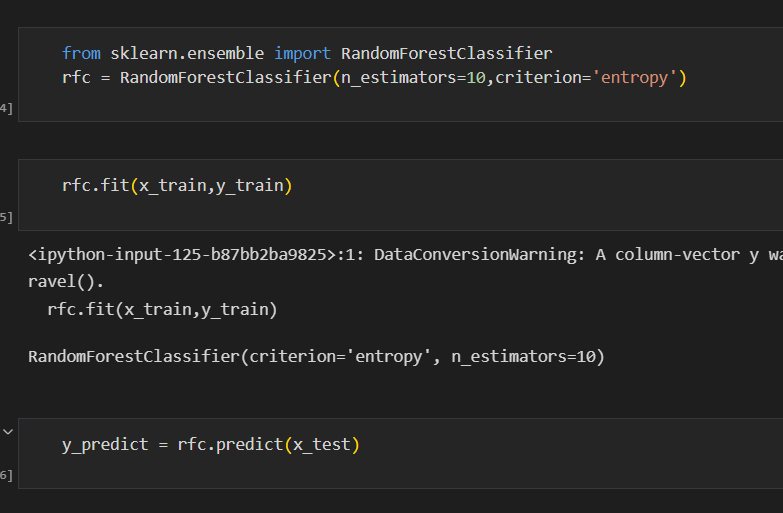
### **ACTIVITY 1.2: Random Forest Model**

A function named random Forest is created and train and test data are passed as the parameters. Inside the function, Random Forest Classifier algorithm is initialized and training data is passed to the model with .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.

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=10,criterion='entropy')

rfc.fit(x\_train,y\_train)



y\_predict = rfc.predict(x\_test)

**ACTIVITY 1.3: 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 Densef 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

.  
import tensorflow

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

classification = Sequential()

classification.add(Dense(30,activation='relu'))

classification.add(Dense(128,activation='relu'))

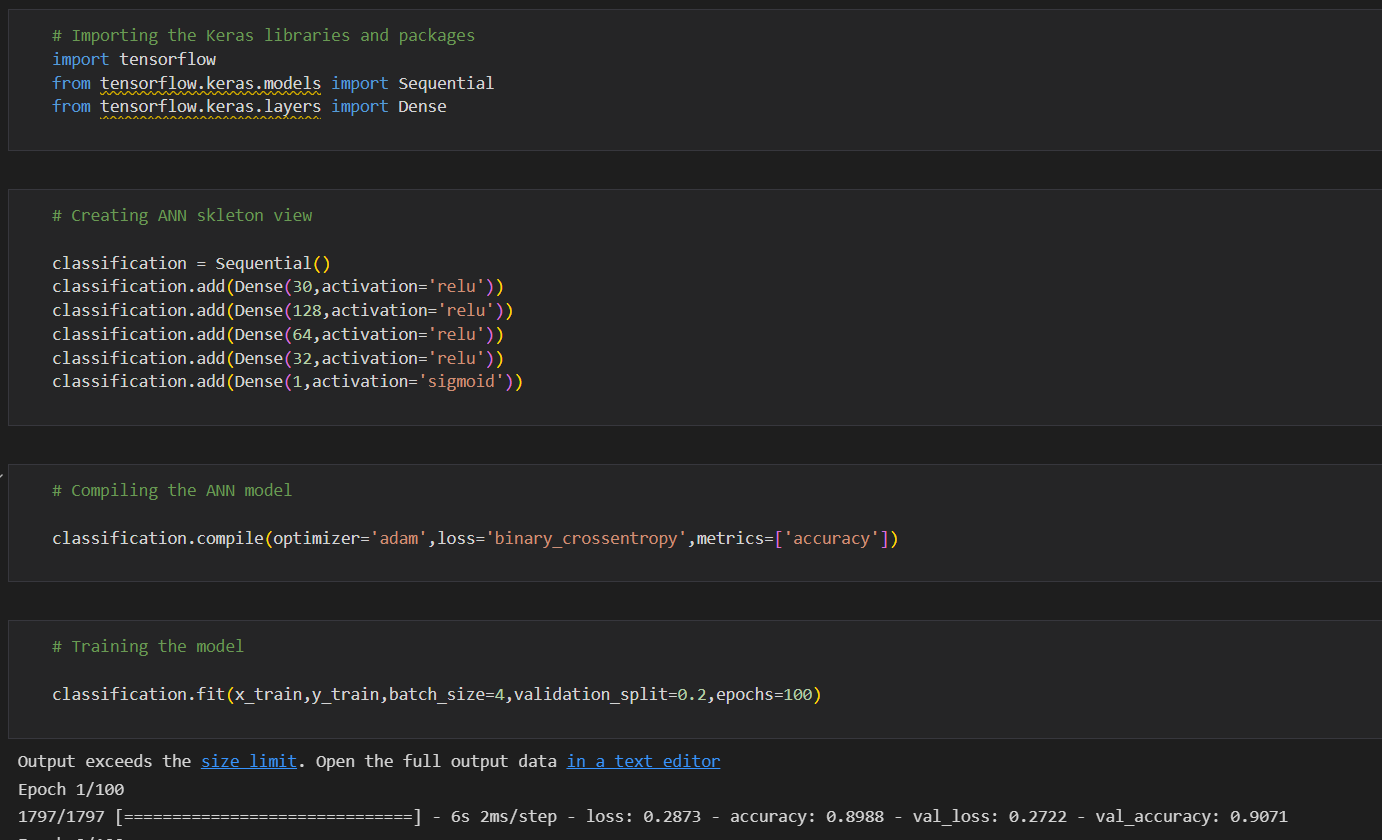
classification.add(Dense(64,activation='relu'))

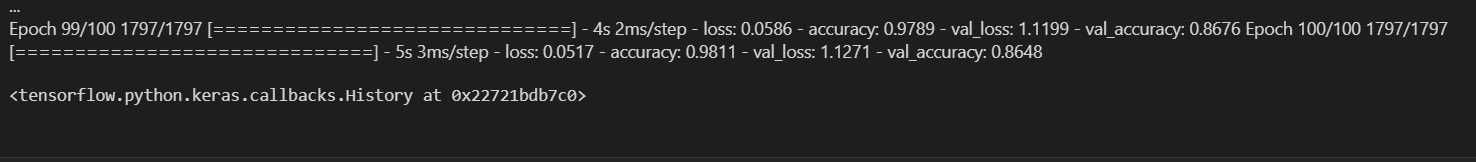
classification.add(Dense(32,activation='relu'))

classification.add(Dense(1,activation='sigmoid'))

classification.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])

classification.fit(x\_train,y\_train,batch\_size=4,validation\_split=0.2,epochs=100)





### **ACTIVITY 2: Test the model**

y\_pred = (y\_pred > 0.5)

y\_pred



y\_pred = rfc.predict([[129,99,1,0,0,1,0,1]])

print(y\_pred)

(y\_pred)

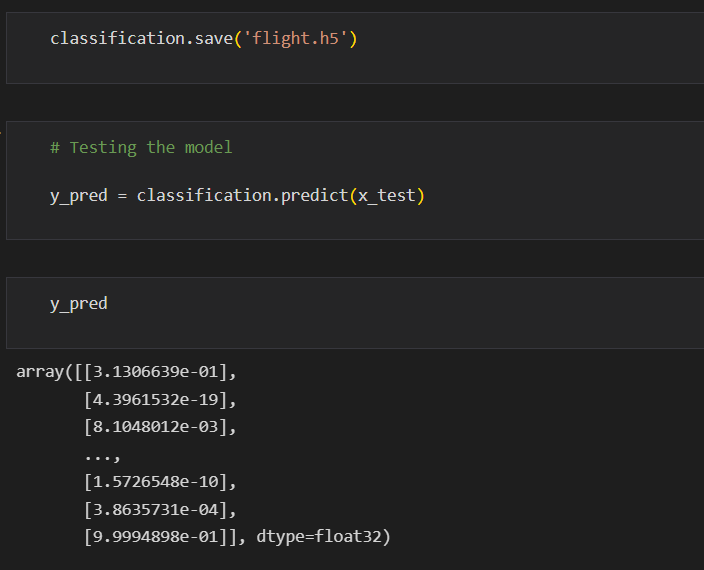


In ANN we first have to save the model to the test the inputs.

classification.save('flight.h5')

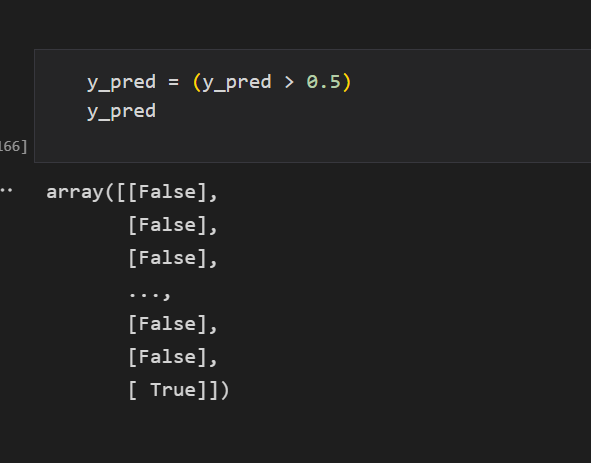
y\_pred = classification.predict(x\_test)

y\_pred



y\_pred = (y\_pred > 0.5)

y\_pred



This code defines a function named "predict\_exit" which takes in a sample\_value as an input. The function then converts the input sample\_value from a list to a numpy array. It reshapes the sample\_value array as it contains only one record. Then, it applies feature scaling to the reshaped sample\_value array using a scaler object 'sc' that should have been previously defined and fitted. Finally, the function returns the prediction of the classifier on the scaled sample\_value

def predict\_exit(sample\_value):

sample\_value = np.array(sample\_value)

sample\_value = sample\_value.reshape(1, -1)

sample\_value = sc.transform(sample\_value)

return classifier.predict(sample\_value)

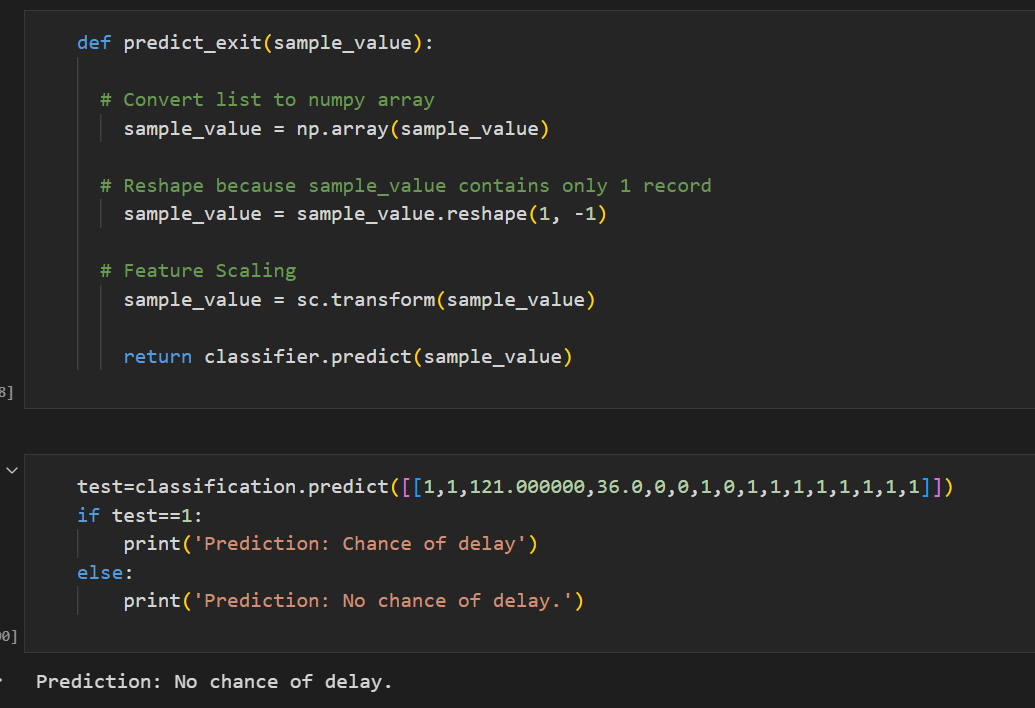
test=classification.predict([[1,1,121.000000,36.0,0,0,1,0]])

if test==1:

print('Prediction:Chance of delay')

else:

print('Prediction:No chance of delay.')



**Milestone 5: Performance Testing & Hyperparameter Tuning**

**Activity 1: Testing model with multiple evaluation metricss**

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

**Activity 1.1: Compare the model**

For comparing the above three models

from sklearn import model\_selection

from sklearn.neural\_network import MLPClassifier

edict(x\_test)

print(name)

print(classification\_report(y\_test, y\_pred, target\_names=target\_names))

results.append(cv\_results)

names.append(name)

this\_df = pd.DataFrame(cv\_results)

this\_df['model'] = name

dfs.append(this\_df)dfs=[]

models = [

('RF',RandomForestClassifier()),

('DecisionTree',DecisionTreeClassifier()),

('ANN',MLPClassifier())

]

results = []

names = []

scoring = ['accuracy','precision\_weighted','recall\_weighted','f1\_weighted','roc\_auc']

target\_names = ['no delay', 'delay']

for name, model in models:

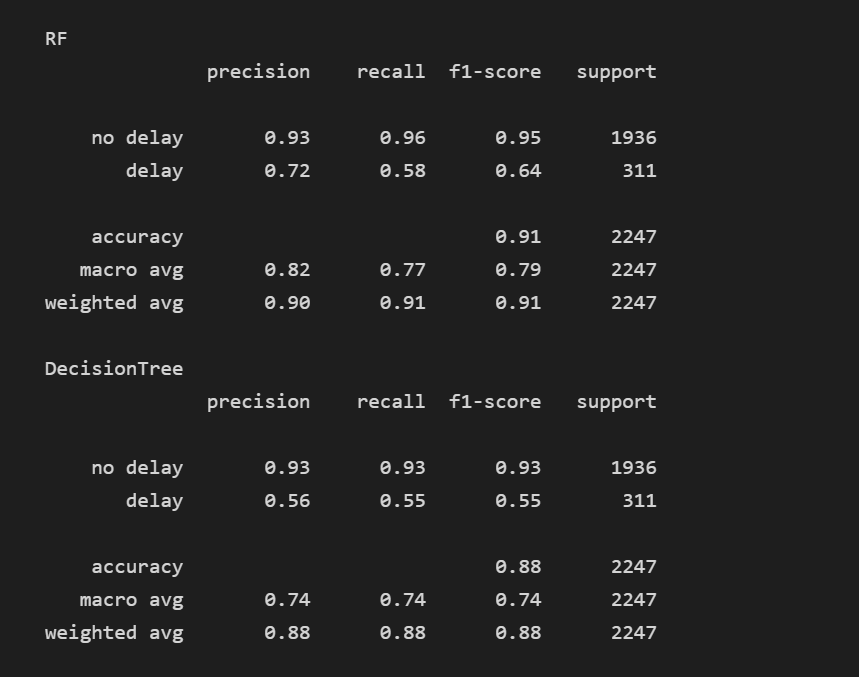
kfold = model\_selection.KFold(n\_splits=5, shuffle=True, random\_state=90210)

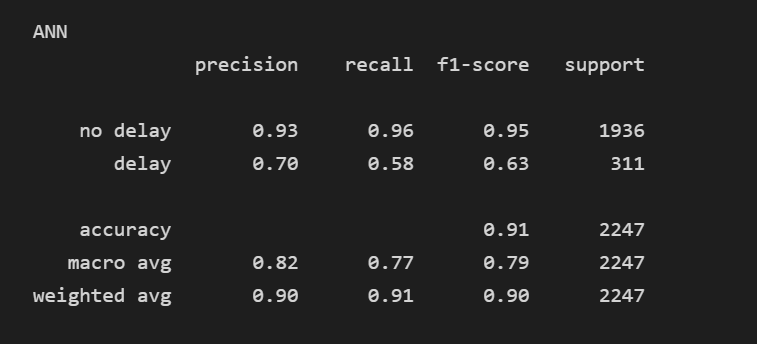
cv\_results = model\_selection.cross\_validate(model, x\_train, y\_train, cv=kfold, scoring=scoring)

clf = model.fit(x\_train,y\_train)

y\_pred = clf.pr

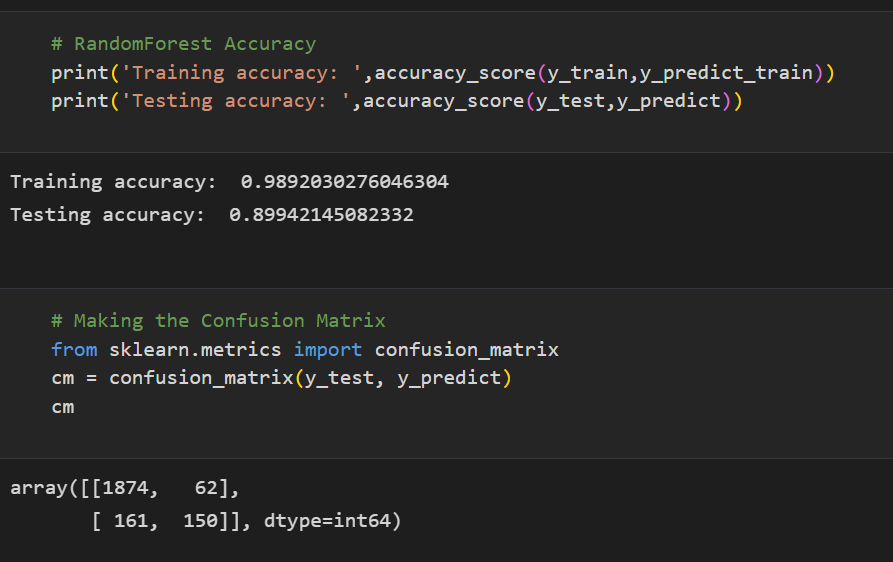
final = pd.concat(dfs, ignore\_index=True)





Print(‘Training accuracy: ‘,accuracy\_score(y\_train,y\_predict\_train))

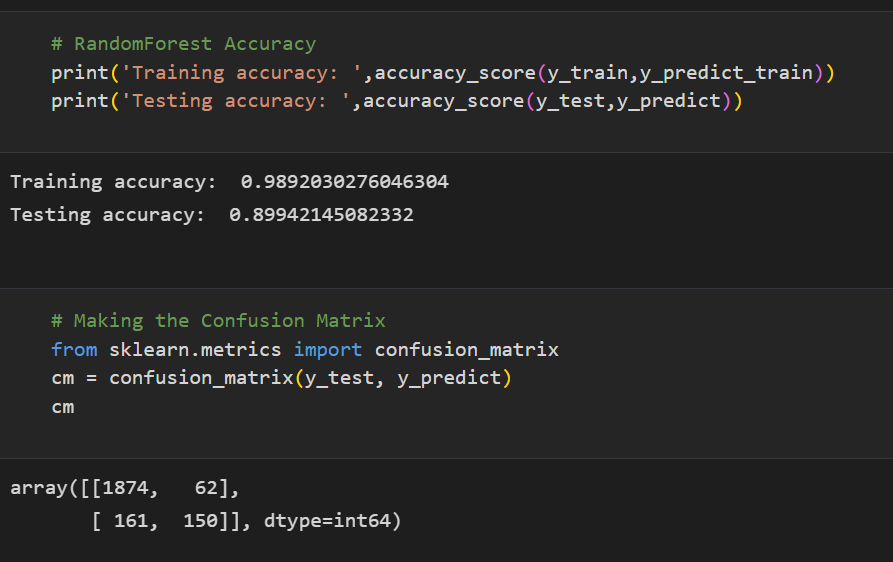
Print(‘Testing accuracy: ‘,accuracy\_score(y\_test,y\_predict))



from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_predict)

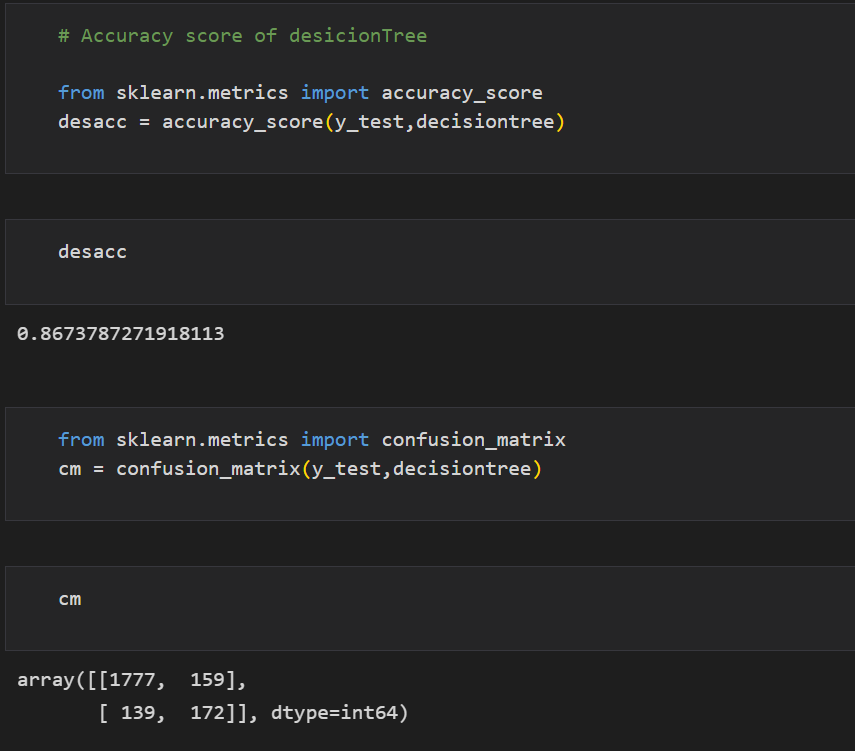
cm



rom sklearn.metrics import accuracy\_score

desacc = accuracy\_score(y\_test,decisiontree)

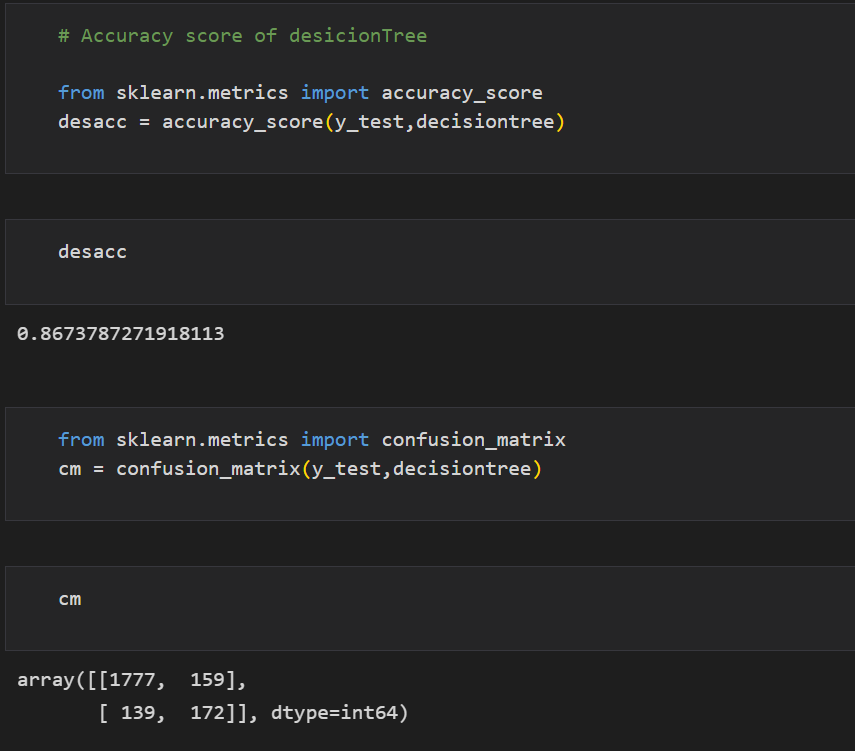
desacc



from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test,decisiontree)

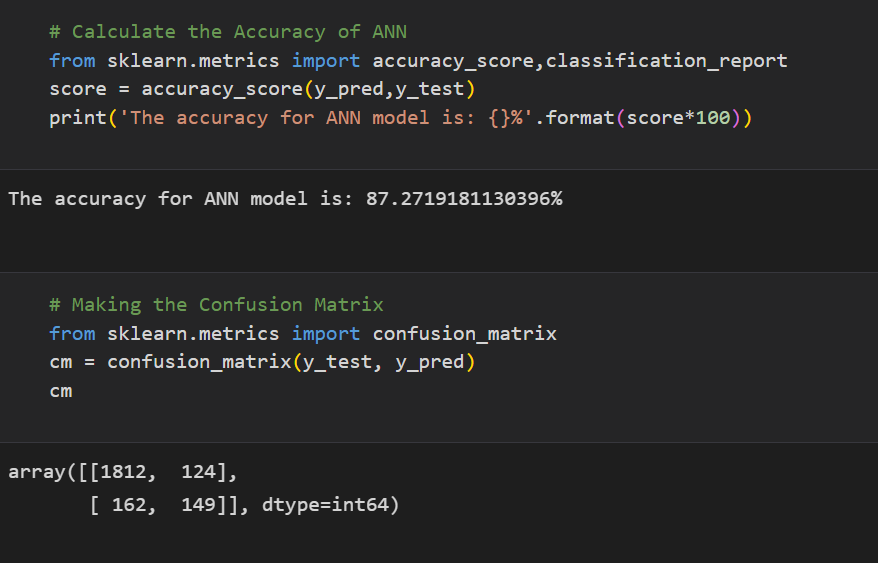
cm



from sklearn.metrics import accuracy\_score,classification\_report

score = accuracy\_score(y\_pred,y\_test)

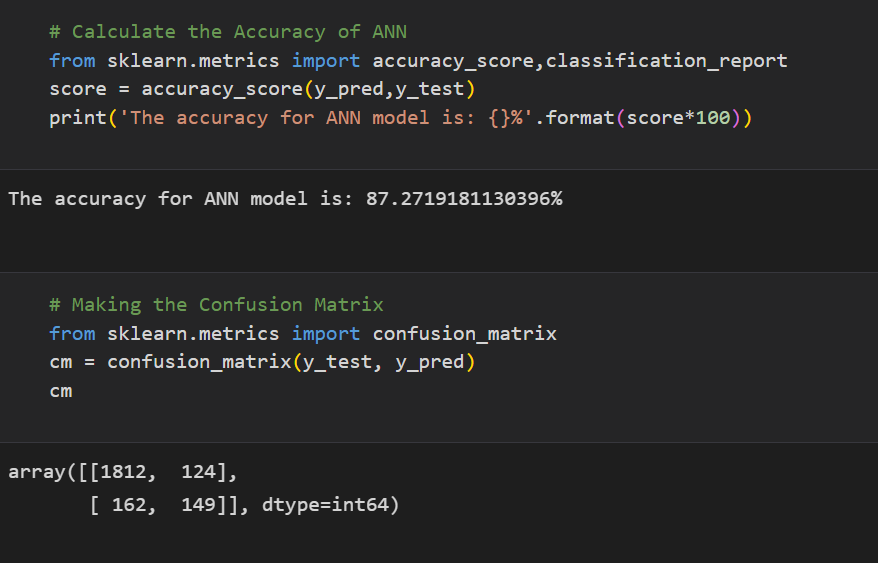
print('The accuracy for ANN model is: {}%'.format(score\*100))



from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test,y\_pred)

Cm



**Activity 2: Comparing model accuracy before & after applying hyperparameter tuning**

Evaluating performance of the model From sklearn, cross\_val\_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().

**Note**: To understand cross validation, refer to this link.

parameters = {

'n\_estimators' :[1,20,30,55,68,74,90,120,115],

'criterion':['gini','entropy'],

'max\_features':["auto", "sqrt", "log2"],

'max\_depth' :[2,5,8,10], 'verbose' :[1,2,3,4,6,8,9,10]

}

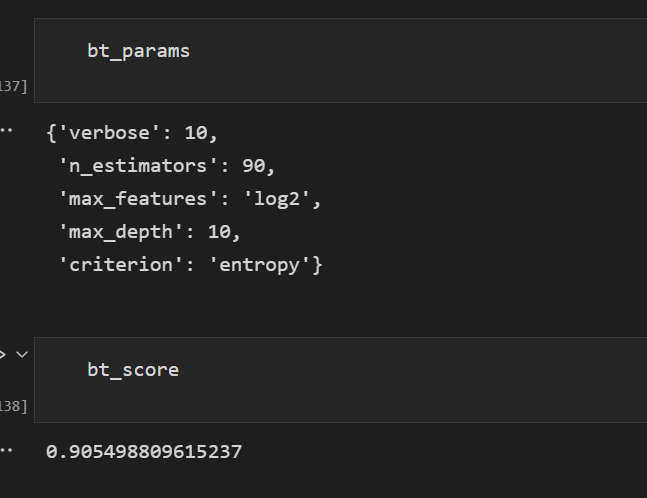
RCV = RandomizedSearchCV(estimator=rfc,param\_distributions=parameters,cv=10,n\_iter=4)

RCV.fit(x\_train,y\_train)

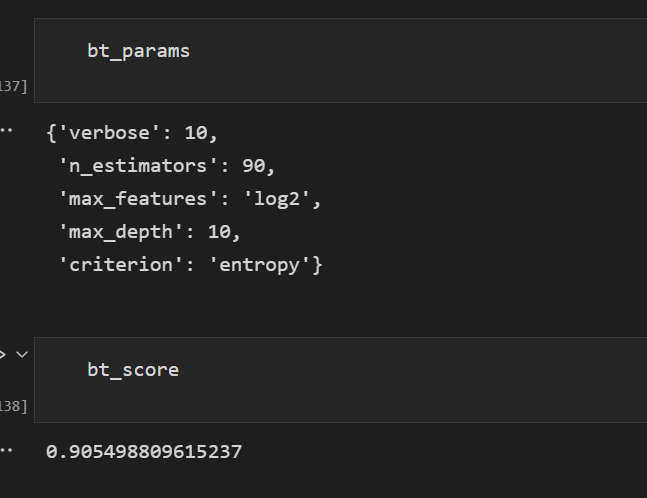
bt\_params = RCV.best\_params\_

bt\_score = RCV.best\_score\_

bt\_params

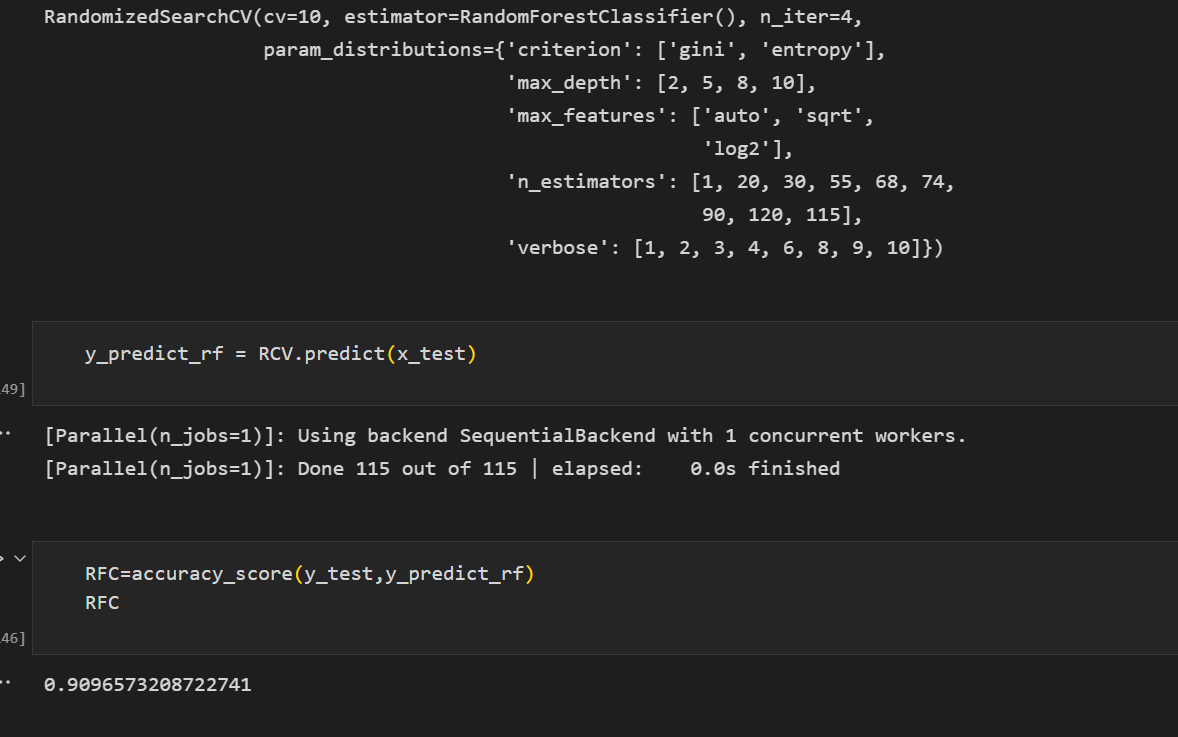


bt\_score

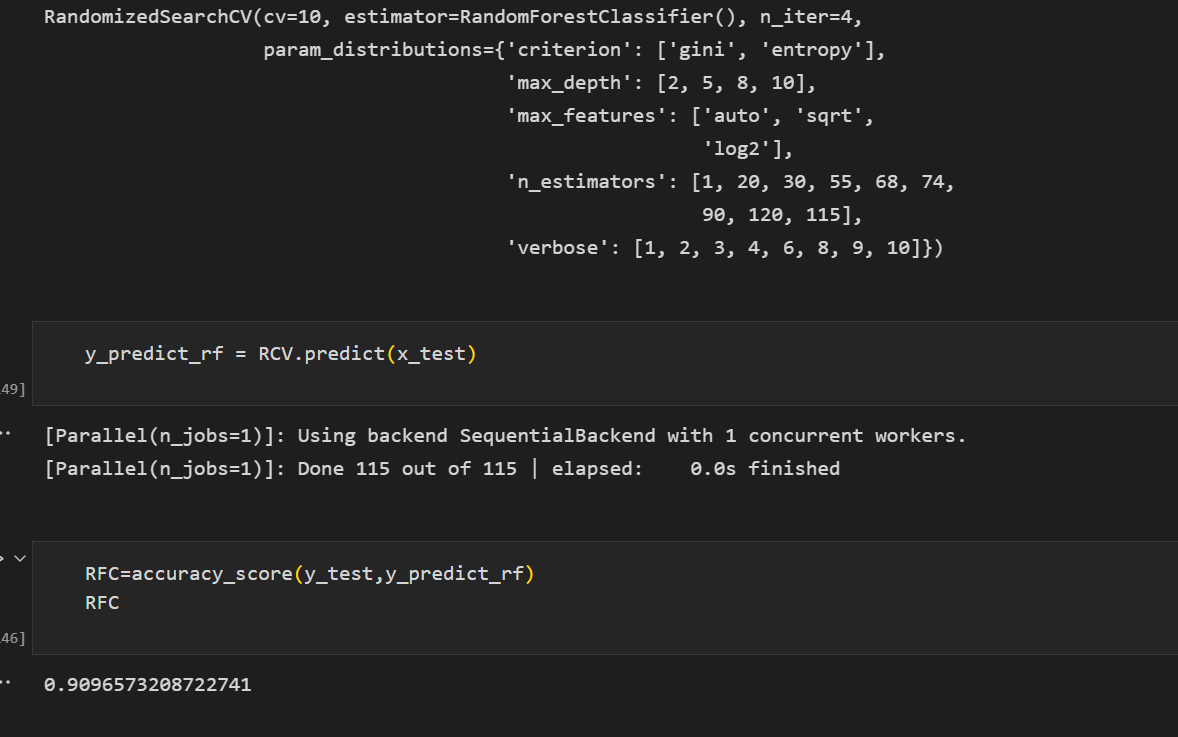


model = RandomForestClassifier(verbose= 10, n\_estimators= 120, max\_features='log2',max\_depth= 10,criterion= 'entropy')

RCV.fit(x\_train,y\_train)

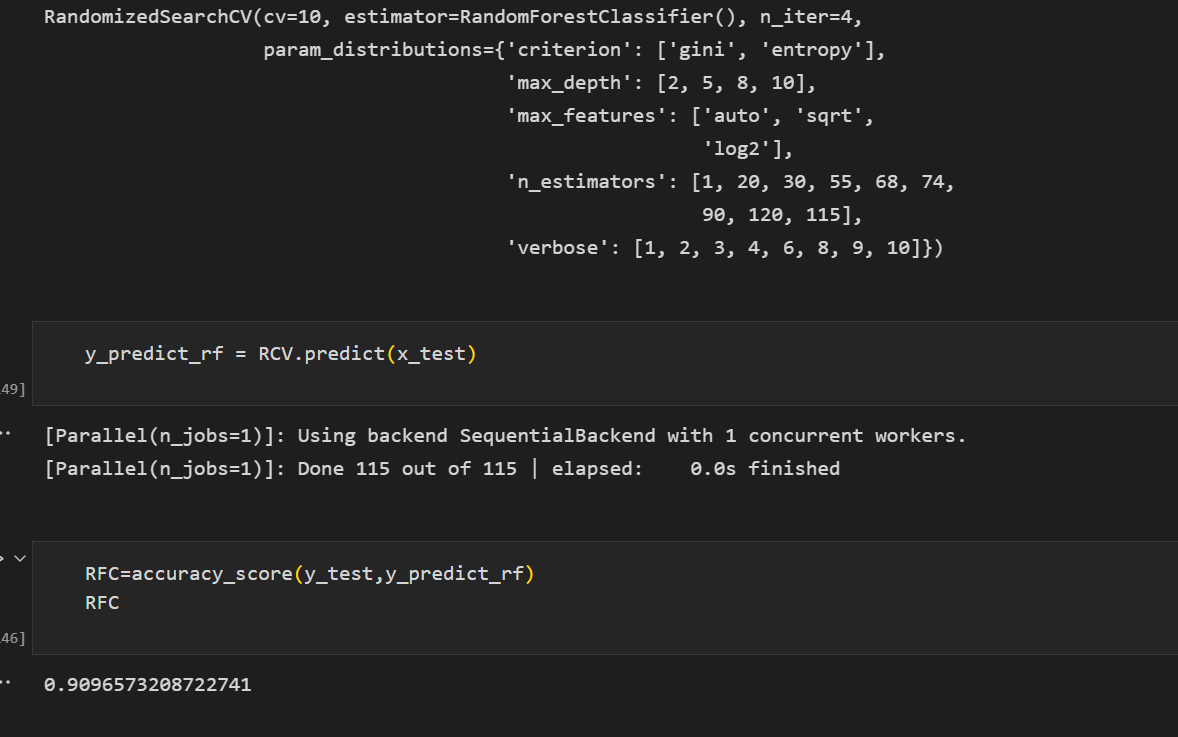


y\_predict\_rfc = RCV.predict(x\_test)



RFC=accuracy\_score(y\_test,y\_predict\_rfc)

RFC



**Milestone 6: 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.

import pickle

pickle.dump(RCV,open('flight.pkl','wb'))

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

**Activity 2.1: Building Html Pages:**

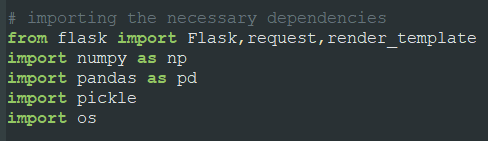
For this project create one HTML files namely

● index.html

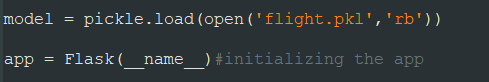
and save them in the templates folder.

**Activity 2.2: Build Python code**:

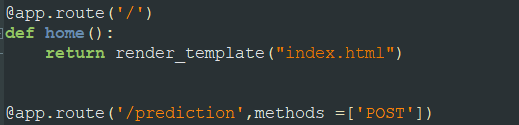
Import the libraries



Load the saved model. 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.



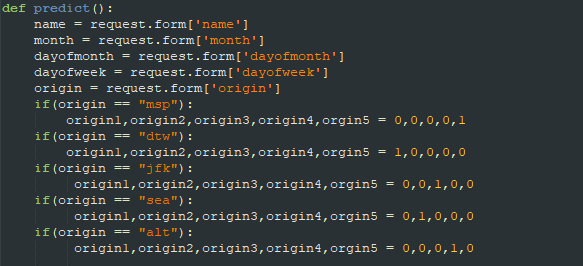
Render HTML page:



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 home.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 the route for prediction is given and necessary steps are performed in order to get the predicted output.

Main Function:



**Activity 2.3: Run the web application**

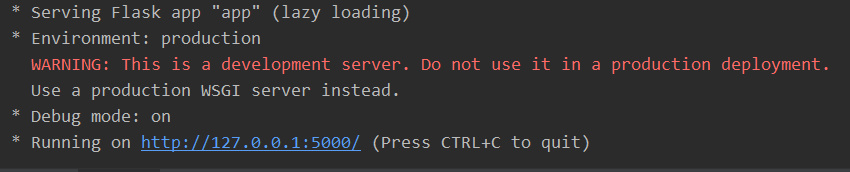
● Open anaconda prompt from the start menu

● 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.

● 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.



Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result



Input 1- Now, the user will give inputs to get the predicted result after clicking onto the submit button.





**Milestone 7: Project Demonstration & Documentation**

Below mentioned deliverables to be submitted along with other deliverables

**Activity 1: - Record explanation Video for project end to end solution**

**Activity 2: - Project Documentation-Step by step project development procedure**.