# PROJECT : FLIGHT DELAY PREDICTION FOR AVIATION INDUSTRY UISNG MACHINE LEARNING.

# TEAM MEMBERS NAME:

1. KATHIRAVAN G(20UCS4260)(LEAD)
2. LOGESHWARAN S(20UCS4261)
3. NAVEENKUMAR A(20UCS4264)
4. VARATHARAJ K(20UCS4276)

# GITHUB LINK: <https://github.com/KathiravanCR7/NaanMudhalvan-Flight-Delay-Prediction-for-aviation-industry-using-Machine-Learning>

# 1. INTRODUCTION

**1.1 Overview**

**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.

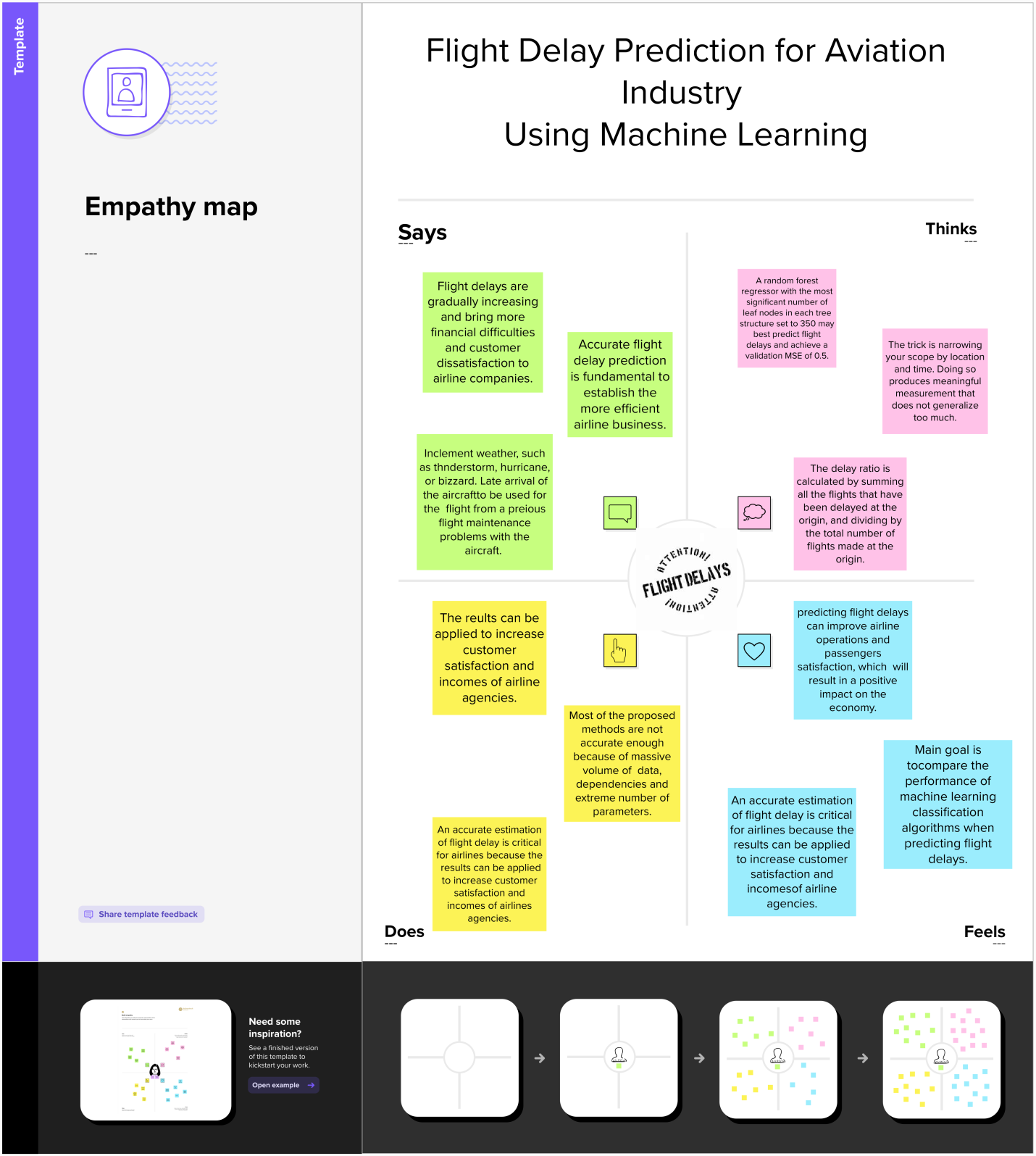
Using a machine learning model, we can predict flight arrival delays. The input to our algorithm is rows of feature vector like departure date, departure delay, distance between the two airports, scheduled arrival time etc. We then use decision tree classifier to predict if the flight arrival will be 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

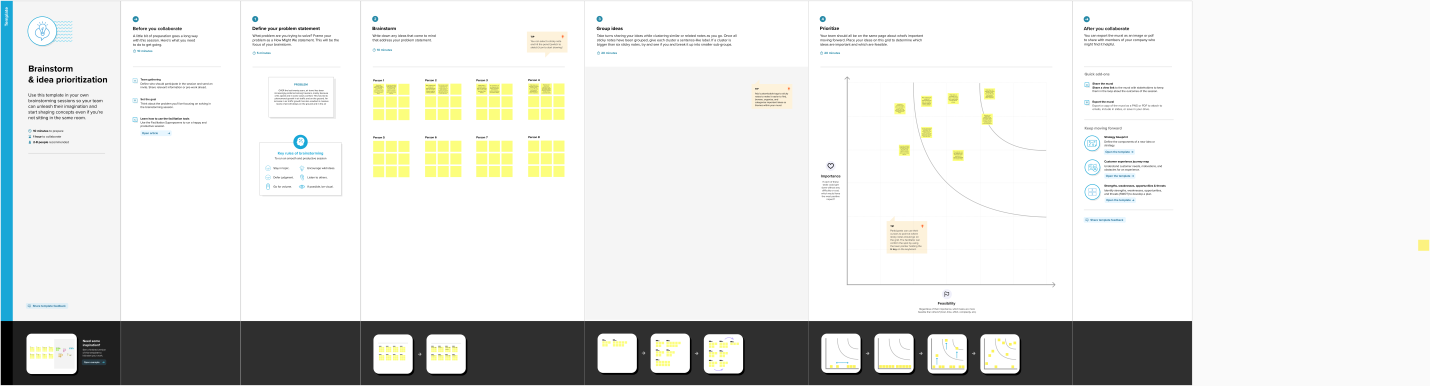
To predict flight delays using machine learning, you will need to collect and process a large amount of data on past flight delays. This data should include information such as the flight's departure and arrival times, the airline, the aircraft type, and the weather conditions at the departure and arrival 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.

# 2. PROBLEM DEFINITION & DESIGN THINKING.

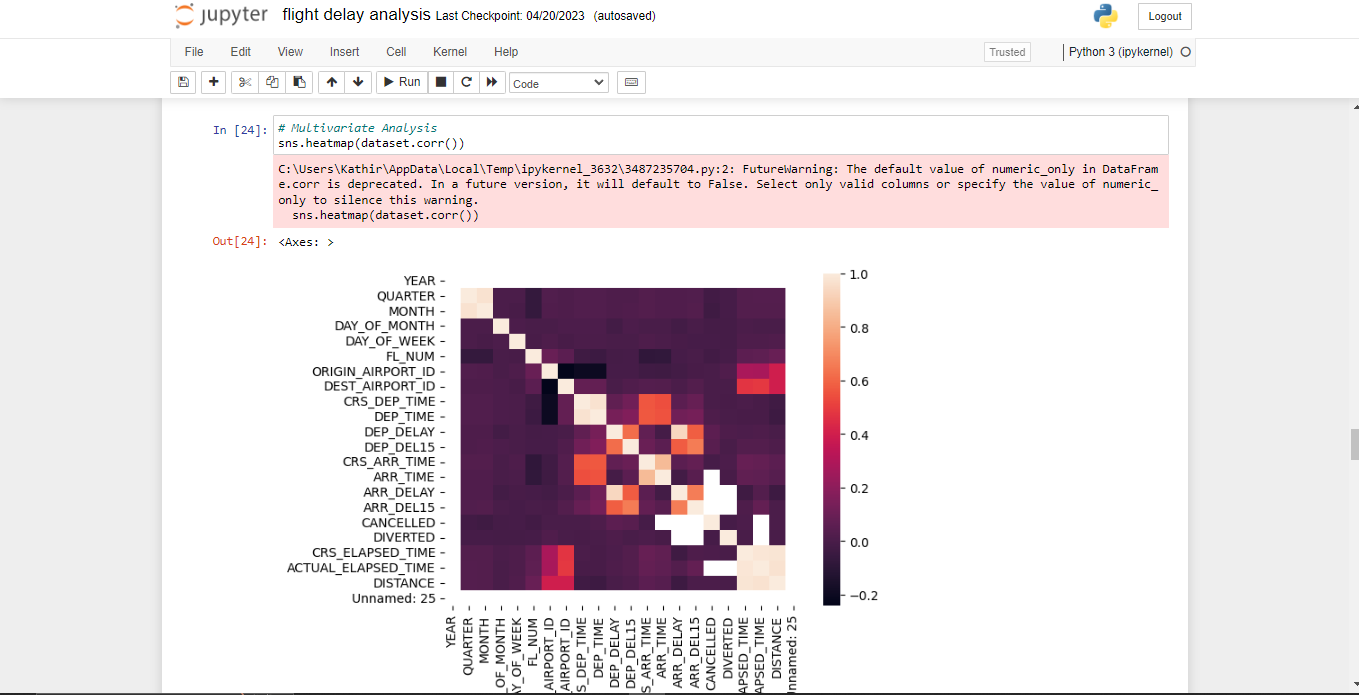
2.1 Empathy Map



2.2 Ideation & Brainstorming Map



# 3. RESULT.



# 4. ADVANTAGES & DISADVANTAGES.

* 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.

# 5. APPLICATIONS.

* 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.

# 6. CONCLUSION.

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.

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.

# 7. FUTURE SCOPE.

* 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.

# 8. APPENDIX.

A. Source Code.

# Importing the Libraries

import pandas as pd

import numpy as np

import pickle

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import sklearn

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

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

from sklearn.preprocessing import StandardScalar

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, f1\_score

# Reading the Dataset

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

dataset.head()

# Handling Missing Values

dataset.info()

#skip handling the missing values step.

dataset = dataset.drop('Unnamed: 25', axis=1)

dataset.isnull().sum()

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()

# replace the missing values with 15

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

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

datset.iloc[177:185]

# Handling Cateogrical Values

import math

for index, row in dataset.iterrows():

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

datset.head()

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

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

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

dataset.head(5)

dataset['ORIGIN'].unique()

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

dataset.head()

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

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

x

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)

# Descriptive Statistical

flight\_data.describe()

# Univariate Analysis

sns.distplot(flight\_data.MONTH)

# Bivariate Analysis

sns.scatterplot(x='ARR\_DELAY', y='ARR\_DEL15', data=flight\_data)

sns.catplotlib(x='ARR\_DEL15', y='ARR\_DELAY', kind='bar', data=flight\_data)

# Multivariate Analysis

sns.heatmap(dataset.corr())

# Splitting data into dependent and independent variables

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

dataset.head()

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

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

# Splitting data into train and test

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

    x, y, test\_size=0.2, random\_state=0)

train\_x, test\_x, train\_y, test\_y = train\_test\_split(dataset.drop(

    'ARR\_DEL15', axis=1), df['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)

# MODEL BUILDING

# Decision Tree Model

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(random\_state=0)

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

DecisionTreeClassifier(random\_state=0)

decisiontree = classifier.predict(x\_test)

decisiontree

from sklearn.metrics import accuracy\_score

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

# Random Forest Model

from sklearn.ensemble import RandomForestClassifier

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

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

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

# ANN MODEL

# Importing the keras libraries and packages

import tensorflow

from tensorflow.keras.models import Sequential

from tensorflow.keras.layer import Dense

# Creating ANN Skleton view

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'))

# Compiling the ANN Model

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

# Training the model

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

# Test the model

#Decision Tree

y\_pred = classifier.predict(

    [[129, 99, 1, 0, 0,  1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1]])

print(y\_pred)

(y\_pred)

# RandomForest

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

print(y\_pred)

(y\_pred)

classification.save('flight.h5')

# Test the model

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

y\_pred

y\_pred = (y\_pred > 0.5)

y\_pred

def predict\_exit(sample\_value):

    # Convert list to numpy array

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

    # Reshape because sample\_value contains only one record

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

    # Feature Scaling

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

    return classifier.predict(sample\_value)

test = classification.predict(

    [[1, 1, 121.000000, 36.0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1]])

if test == 1:

    print('Prediction: Chance of delay')

else:

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

# Performance Testing & Hyperparameter Tuning

# Compare The Model

from sklearn import model\_selection

from sklearn.neural\_network import MLPClassifier

dfs = []

models = [

    ('RF', RandomForestClassifier()),

    ('DecisionTree', DecisionTreeClassifier()),

    ('ANN', MLPClassifier())

]

result = []

names = []

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

target\_nmes = ['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.predict(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)

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

return final

# RanomForest Accuracy

print('Training Accuracy: ', accuracy\_score(y\_train, y\_predict\_train))

print('Testing Accuracy: ', accuracy\_score(y\_test, y\_predict))

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

cm

# Accuracy score of Destination tree

from 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

# Calculate the Accuracy of ANN

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))

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

cm

# Comparing Model Accuracy Before & After Applying Hyperparameter Tuning

# Comare the Model

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']

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

cv\_results = model\_selection.cross\_validate(

    model, x\_train, y\_train, cv=fold, scoring=scoring)

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

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

print(name)

print(classification\_report(y\_tet, 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)

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

return final

# RandomForest Accuracy

print('Training accuracy: ', accuracy\_score(y\_train, y\_predict\_train))

print('Testing accuracy: ', accuracy\_score(y\_test, y\_predict))

# Making the Confusion Matrix

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

cm

# Accuracy Score of DecisionTree

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

desacc

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

cm

# Comparing Model Accuracy Before & After Applying Hyperparameter Tuning

# Giving some parameters that can be used in randomized search cv

parameter = {

    '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]

}

# Performing the randomized CV

RCV = RandomizedSearchCV(

    estimator=rf, param\_distributions=parameters, cv=10, n\_iter=4)

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

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\_rf = RCV.predict(x\_test)

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

RFC

# Model Deployment

# Save the best Model

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

# Importing the necessary dependencies

model = pickle.load(open('flight.pkl', 'rb'))

app = Flask(\_name\_)  # Initializing the app

@app.route('/')

def home():

    return render\_template("index.html")

@app.route('/prediction', methods=['POST'])

# Retrieves the value from UI:

def predict():

    name = request.form['name']

    month = request.form['month']

    dayofmonth = request.form['dayofmonth']

    dayofweek = request.form['dayofweek']

    origin = request.form['origin']

    if (origin == "map"):

        origin1, origin2, origin3, origin4, origin5 = 0, 0, 0, 0, 1

    if (origin == "dtw"):

        origin1, origin2, origin3, origin4, origin5 = 1, 0, 0, 0, 0

    if (origin == "jfk"):

        origin1, origin2, origin3, origin4, origin5 = 0, 0, 1, 0, 0

    if (origin == "sea"):

        origin1, origin2, origin3, origin4, origin5 = 0, 1, 0, 0, 0

    if (origin == "alt"):

        origin1, origin2, origin3, origin4, origin5 = 0, 0, 0, 1, 0

destination = request.form['destination']

if (destination == "map"):

    destination1, destination2, destination3, destination4, destination5 = 0, 0, 0, 0, 1

if (destination == "dtw"):

    destination1, destination2, destination3, destination4, destination5 = 1, 0, 0, 0, 0

if (destination == "jfk"):

    destination1, destination2, destination3, destination4, destination5 = 0, 0, 1, 0, 0

if (destination == "sea"):

    destination1, destination2, destination3, destination4, destination5 = 0, 1, 0, 0, 0

if (destination == "alt"):

    destination1, destination2, destination3, destination4, destination5 = 0, 0, 0, 1, 0

dept = request.form['dept']

arrtime = request.form['arrtime']

actdept = request.form['actdept']

dept15 = int(dept)-int(actdept)

total = [[name, month, dayofmonth, dayofweek, origin1, origin2, origin3, origin4,

          origin5, destination1, destinatioon2, destination3, destination4, destination5]]

# print Total

y\_pred = model.predict(total)

print(y\_pred)

if (y\_pred == [0.]):

    ans = "The Flight will be  on time"

else:

    ans = "The Flight will be delayed"

return render\_template("index.html", showcase=ans)

if \_name\_ == '\_main\_':

    app.run(debug=True)