

GOVERNMENT ARTS AND SCIENCE COLLEGE NANNILAM DEPARTMENT OF COMPUTER SCIENCE

MACHINE LEARNING WITH PYTHON

PROJECT TITLE FLIGHT DELAY PREDICTION FOR AVIATION INDUSTRY USING MACHINE LEARNING

Team ID: NM2023TMID19848

Team Size: 4

Team Leader: VINAYAGAM M

Team member: GOPINATH K

Team member: JANANAYAGAN V

Team member: SURYAKUMAR V

Overview:

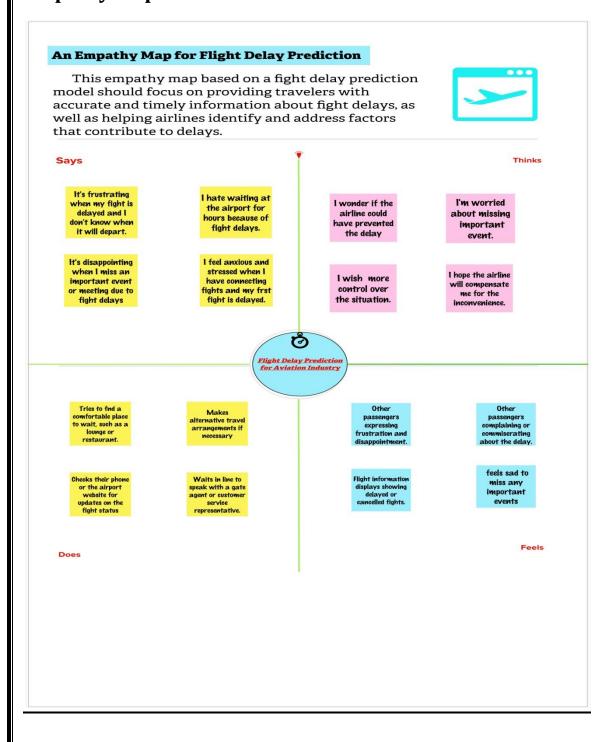
- ♣ The flight delay prediction project using machine learning involves the development of a model that can accurately predict flight delays. This project utilizes historical flight data and machine learning techniques to identify patterns and factors that contribute to flight delays.
- → The model is trained on a large dataset of flight information, which includes various features such as departure and arrival times, weather conditions, airline and airport operations, and other relevant factors that may impact flight delays.
- ♣ Once the model is trained, it can be used to predict the likelihood and duration of flight delays for future flights. This information can be used by airlines and passengers to better plan their travel schedules, reduce the impact of flight delays, and improve overall flight efficiency.
- ♣ The project aims to improve the accuracy of flight delay predictions by utilizing advanced machine learning techniques such as neural networks and decision trees. By doing so, the model can identify complex relationships and interactions between various factors that may impact flight delays.
- →Overall, the flight delay prediction project using machine learning has the potential to greatly improve the efficiency and reliability of air travel by providing more accurate and timely information on flight delays.

Purpose:

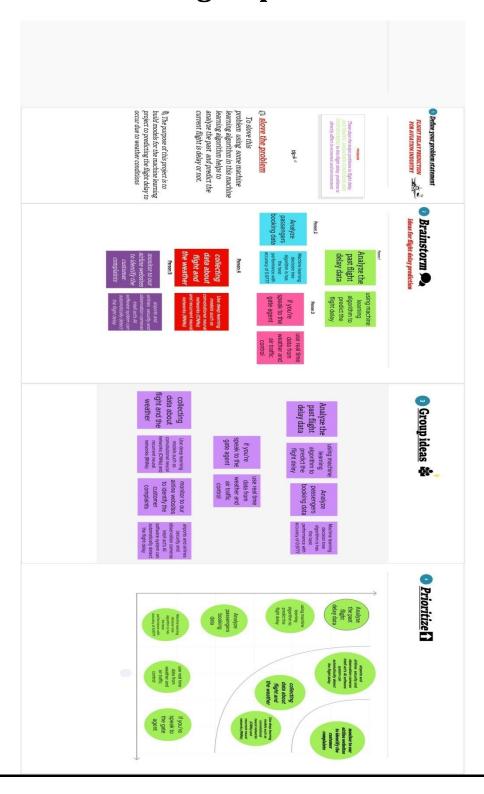
- ✓ Improving airline operations: By accurately predicting flight delays, airlines can better manage their resources, such as gate assignments, crew scheduling, and aircraft utilization. This can lead to improved efficiency and cost savings for airlines.
- ✓ Enhancing customer experience: Accurate flight delay predictions can help airlines to proactively communicate with customers about potential delays and provide alternative travel options. This can help to reduce stress and frustration for travelers, improving their overall experience
- ✓ Increasing safety: Machine learning models can be used to predict potential safety hazards, such as adverse weather conditions, which could impact flight operations. By identifying these hazards in advance, airlines can take appropriate safety measures to protect passengers and crew.
- ✓ Supporting airport planning: Flight delay prediction can help airports to better manage their operations and resources, such as ground handling and gate assignments. This can lead to improved efficiency and reduced congestion at airports.

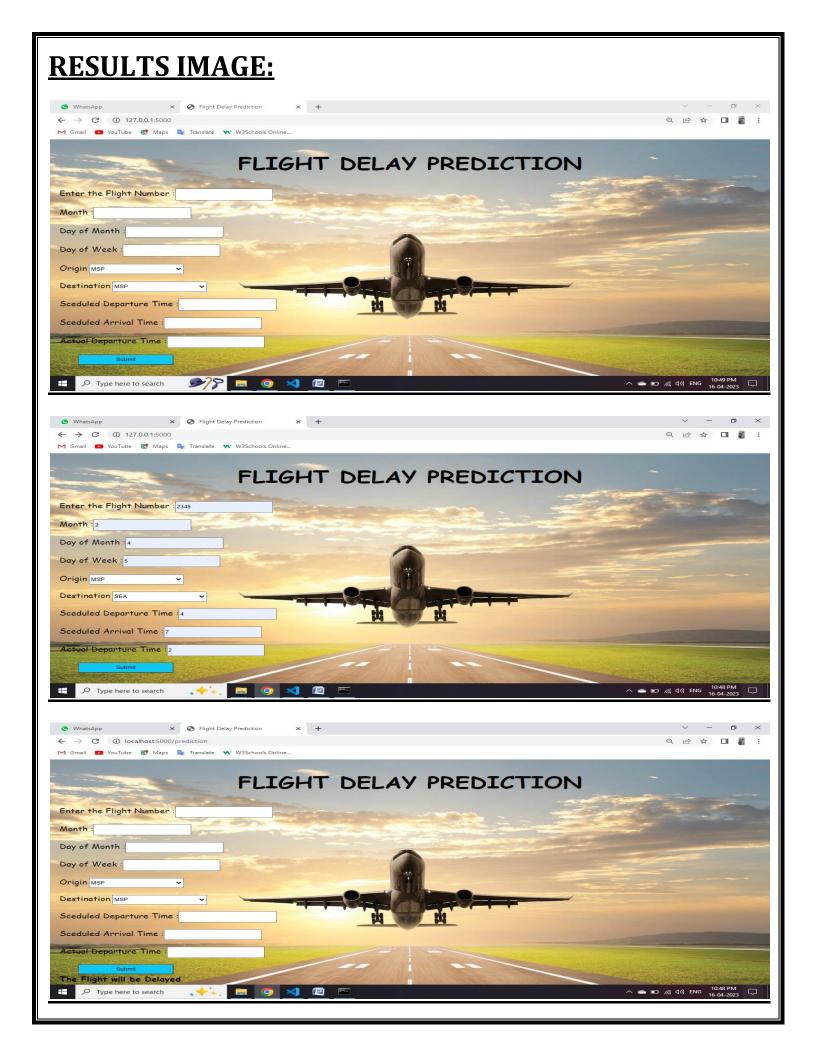
Problem Defining & Design Thinking:

Empathy Map:



Ideation & Brainstorming Map:





Advantages of flight delay:

- ✓ Reimbursement of your ticket and a return flight to your departure airport if you have a connecting flight.
- ✓ Rerouting to your final destination.
- ✓ Rerouting at a later date under comparable transportation conditions.
- ✓ When we traveling by air, can sit comfortable in an armchair, reading magazines, listen to music, read books, play games or watching a free film on television.

Disadvantages of flight delay:

- ✓ Flight delays not only irritate air passengers and disrupt their schedules but also cause a decrease in efficiency, an increase in capital costs, reallocation of flight crews and aircraft, and additional crew expenses.
- ✓ There are plane crashes in which the crew and passengers have died.
- ✓ Airports can often be several miles from city center.

Applications for Flight Delay:

- ✓ It is widely used by aircraft operators throughout the world to inform and facilitate corrective actions in a range of operational areas by offering the ability to track and evaluate flight operations trends, identify risk precursors, and take the appropriate remedial action.
- ✓ 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.
- ✓ With predictive analytics, sensory equipment gathers information from each aircraft's systems, and sends that information to a cloud. That data is then analyzed and used to determine everything from fleet maintenance schedules to marketing strategies.
- ✓ In case of a delay of over 24 hours, the passenger should be offered free hotel accommodation.
- ✓ Customers should also be offered a free stay if a flight departs between 8 pm and 3 am and is delayed for over six hours.

Conclusion:

- ✓ In this project, we use flight data, weather, and demand data to predict flight departure delay. Our result shows that the Random Forest method yields the best performance compared to the SVM model.
- Somehow the SVM model is very time consuming and does not necessarily produce better results. In the end, our model correctly predicts 91% of the non-delayed flights.
- ✓ However, the delayed flights are only correctly predicted 41% of time. As a result, there can be additional features related to the causes of flight delay that are not yet discovered using our existing data sources.
- ✓ In the second part of the project, we can see that it is possible to predict flight delay patterns from just the volume of concurrently published tweets, and their sentiment and objectivity.
- This is not unreasonable; people tend to post about airport delays on Twitter; it stands to reason that these posts would become more frequent, and more profoundly emotional, as the delays get worse. Without more data, we cannot make a robust model and find out the role of related factors and chance on these results.
- However, as a proof of concept, there is potential for these results. It may be possible to routinely use tweets to ascertain an understanding of concurrent airline delays and traffic patterns, which could be useful in a variety of circumstances.

Future scope:

- This project is based on data analysis from year 2008. A large dataset is available from 1987-2008 but handling a bigger dataset requires a great amount of preprocessing and cleaning of the data.
- Therefore, the future work of this project includes incorporating a larger dataset. There are many different ways to preprocess a larger dataset like running a Spark cluster over a server or using a cloud-based services like AWS and Azure to process the data.
- With the new advancement in the field of deep learning, we can use Neural Networks algorithm on the flight and weather data. Neural Network works on the pattern matching methodology. It is divided into three basic parts for data modelling that includes feed forward networks, feedback networks, and self organization network.

APPENDIX:

Source code

Milestone 2:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
oh=OneHotEncoder()
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
#Read the dataset
dataset = pd.read_csv("flightdata.csv")
#dataset.head()
print(dataset.head())
#removing a last column
dataset = dataset.drop('Unnamed: 25', axis=1)
dataset.info()
```

```
#checking the null values

dataset.isnull().sum()

#get the wanted columns in dataset

dataset = dataset[["FL_NUM", "MONTH", "DAY_OF_MONTH", "DAY_OF_WEEK", "ORIGIN", "DEST", "CRS_ARR_TIME", "DEP_DEL15", "ARR_DEL15"]]

#fill the null values in columns ARR_DEL15 ,DEP_DEL15

dataset = dataset.fillna({"ARR_DEL15" : 1})

dataset = dataset.fillna({"DEP_DEL15" : 0})

dataset.iloc[177:189]
```

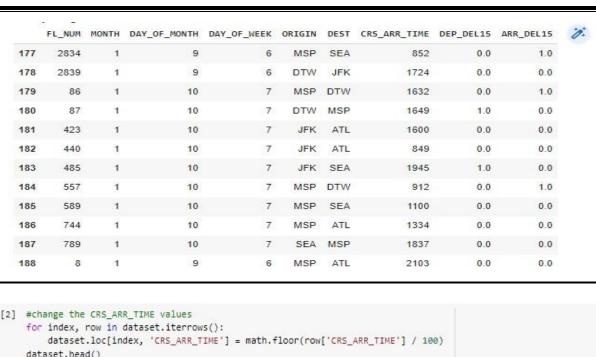
```
YEAR QUARTER MONTH DAY_OF_MONTH DAY_OF_WEEK UNIQUE_CARRIER TAIL_NUM
                   1
9 2916
                                                            DL
              1
                                  1
                                              5
                                                                N836DN
1
  2016
              1
                    1
                                  1
                                              5
                                                            DL
                                                                N964DN
2 2016
                                                                N813DN
              1
                    1
                                  1
                                              5
                                                            DL
3 2016
4 2016
                                                            DL
              1
                    1
                                  1
                                              5
                                                                N587NW
              1
                    1
                                  1
                                              5
                                                            DL
                                                                N836DN
  FL_NUM ORIGIN_AIRPORT_ID ORIGIN ... CRS_ARR_TIME ARR_TIME ARR_DELAY \
                                                              -41.0
                             ATL ... 2143
DTW ... 1435
    1399
0
                     10397
                                                     2102.0
1
   1476
                     11433
                                                     1439.0
                                                                   4.0
    1597
                              ATL ...
SEA ...
                                                      1142.0
                                                                 -33.0
                                              1215
2
                     10397
    1768
                     14747
3
                              SEA
                                                      1345.0
                                                                  10.0
                            SEA ...
4
   1823
                     14747
                                                607
                                                       615.0
                                                                   8.0
  ARR_DEL15 CANCELLED DIVERTED CRS_ELAPSED_TIME ACTUAL_ELAPSED_TIME \
0
       0.0
                 0.0 0.0
                                           338.0
                                                                295.0
1
        0.0
                   0.0
                            0.0
                                           110.0
                                                                115.0
        0.0
                                           335.0
                                                                300.0
2
                  0.0
                           0.0
3
        0.0
                  0.0
                           0.0
                                           196.0
                                                                205.0
        0.0
                  0.0
                            0.0
                                           247.0
                                                                259.0
  DISTANCE Unnamed: 25
0
   2182.0
     528.0
                   NaN
1
                   NaN
2
    2182.0
3
    1399.0
                    NaN
    1927.0
                   NaN
```

[5 rows x 26 columns]

<class 'pandas.core.frame.DataFrame'> RangeIndex: 11231 entries, 0 to 11230 Data columns (total 25 columns):

#	Column	Non-Null Cour	nt Dtype
0	YEAR	11231 non-nul	ll int64
1	QUARTER	11231 non-nul	ll int64
2	MONTH	11231 non-nul	ll int64
3	DAY_OF_MONTH	11231 non-nul	ll int64
4	DAY_OF_WEEK	11231 non-nul	ll int64
5	UNIQUE_CARRIER	11231 non-nul	ll object
6	TAIL_NUM	11231 non-nul	ll object
7	FL_NUM	11231 non-nul	ll int64
8	ORIGIN_AIRPORT_ID	11231 non-nul	ll int64
9	ORIGIN	11231 non-nul	ll object
10	DEST_AIRPORT_ID	11231 non-nul	ll int64
11	DEST	11231 non-nul	1 object
12	CRS_DEP_TIME	11231 non-nul	ll int64
13	DEP_TIME	11124 non-nul	ll float64
14	DEP_DELAY	11124 non-nul	ll float64
15	DEP_DEL15	11124 non-nul	ll float64
16	CRS_ARR_TIME	11231 non-nul	ll int64
17	ARR_TIME	11116 non-nul	ll float64
18	ARR_DELAY	11043 non-nul	ll float64
19	ARR_DEL15	11043 non-nul	ll float64
20	CANCELLED	11231 non-nul	ll float64
21	DIVERTED	11231 non-nul	ll float64
22	CRS_ELAPSED_TIME	11231 non-nul	ll float64
23	ACTUAL_ELAPSED_TIM	E 11043 non-nul	ll float64
24	DISTANCE	11231 non-nul	ll float64
dtype	es: float64(11), in	t64(10), object((4)

memory usage: 2.1+ MB



[2] #change the CRS_ARR_TIME values dataset.head()

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
0	1399	1	1	5	ATL	SEA	21	0.0	0.0
1	1476	1	1	5	DTW	MSP	14	0.0	0.0
2	1597	1	1	5	ATL	SEA	12	0.0	0.0
3	1768	1	1	5	SEA	MSP	13	0.0	0.0
4	1823	1	1	5	SEA	DTW	6	0.0	0.0

#convert DEST & ORIGIN using LabelEncoder le = LabelEncoder() dataset['DEST'] = le.fit_transform(dataset['DEST']) dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN']) dataset.head()

0

0:

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
0	1399	1	1	5	0	4	21	0.0	0.0
1	1476	1	1	5	1	3	14	0.0	0.0
2	1597	1	1	5	0	4	12	0.0	0.0
3	1768	1	1	5	4	3	13	0.0	0.0
4	1823	1	1	5	4	1	6	0.0	0.0

C+

 [4] dataset["ORIGIN"].unique() array([0, 1, 4, 3, 2])

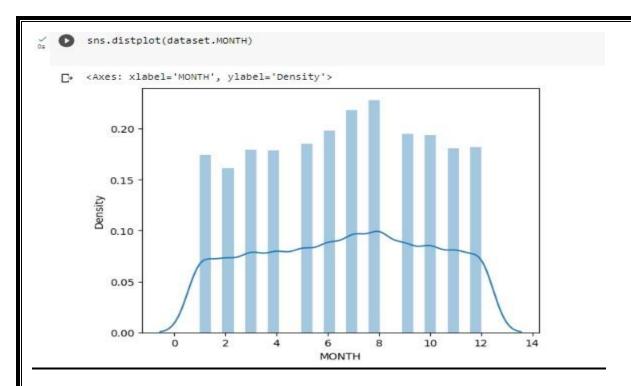


Milestone 3:

task 3:

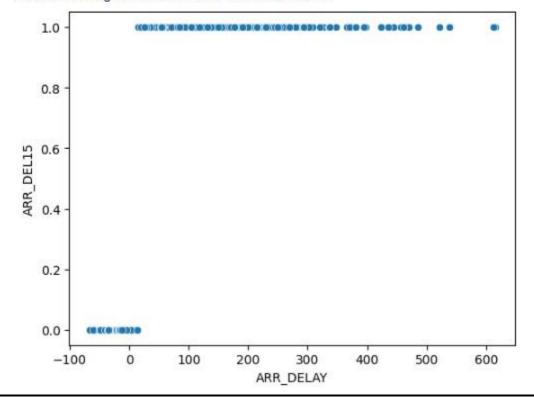
[6] org_data=pd.read_csv("flightdata.csv") org_data.describe()

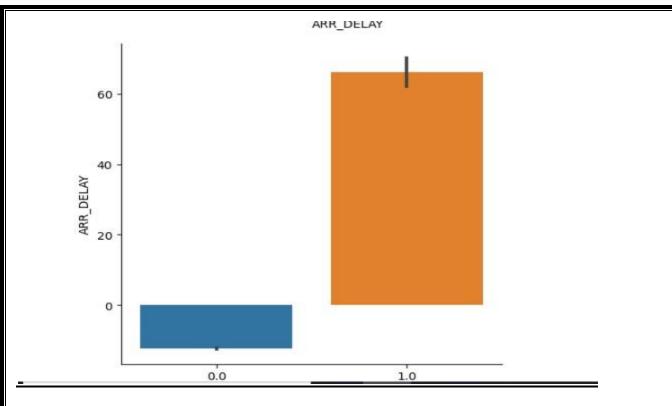
	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_DEP_TIME	DEP_TIME	113
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11124.000000	22
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.516695	12302.274508	1320.798326	1327.189410	M
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.026510	1601.988550	490.737845	500.306462	
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.000000	10397.000000	10,000000	1.000000	
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.000000	10397.000000	905.000000	905.000000	et e
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.000000	12478.000000	1320.000000	1324.000000	
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.000000	13487.000000	1735.000000	1739.000000	
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.000000	14747.000000	2359.000000	2400.000000	10

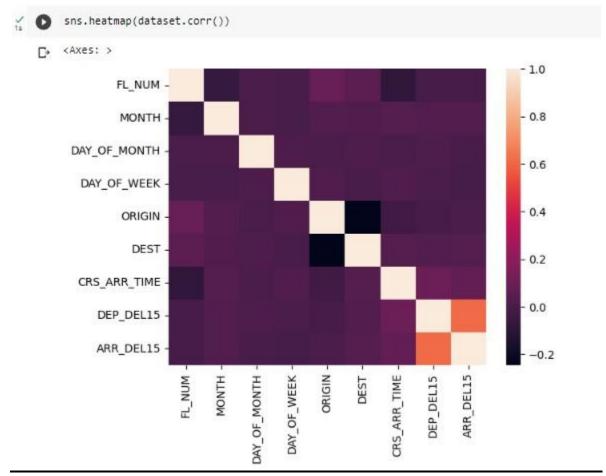




<seaborn.axisgrid.FacetGrid at 0x7fcecda9df40>







```
#spilting data
       x = dataset.iloc[:, 0:8].values
       y = dataset.iloc[:, 8:9].values
   array([[1.399e+03, 1.000e+00, 1.000e+00, ..., 4.000e+00, 2.100e+01,
               0.000e+00],
              [1.476e+03, 1.000e+00, 1.000e+00, ..., 3.000e+00, 1.400e+01,
               0.000e+00],
              [1.597e+03, 1.000e+00, 1.000e+00, ..., 4.000e+00, 1.200e+01,
               0.000e+00],
              [1.823e+03, 1.200e+01, 3.000e+01, ..., 4.000e+00, 2.200e+01,
               0.000e+00],
              [1.901e+03, 1.200e+01, 3.000e+01, ..., 4.000e+00, 1.800e+01,
               0.000e+00],
              [2.005e+03, 1.200e+01, 3.000e+01, ..., 1.000e+00, 9.000e+00,
               0.000e+00]])
[73] y
       array([[0.],
              [0.],
              [0.],
              [0.],
              [0.],
[0.]])
\sqrt{2} [76] z = oh.fit_transform(x[:,4:5]).toarray()
        t = oh.fit_transform(x[:,5:6]).toarray()
        \#x = np.delete(x,[4,7],axis=1)
/
0s
   0
        array([[1., 0., 0., 0., 0.],
               [0., 1., 0., 0., 0.],
               [1., 0., 0., 0., 0.],
               ...,
               [0., 1., 0., 0., 0.],
               [1., 0., 0., 0., 0.],
               [1., 0., 0., 0., 0.]])
√ [78] t
        array([[0., 0., 0., 0., 1.],
               [0., 0., 0., 1., 0.],
               [0., 0., 0., 0., 1.],
               [0., 0., 0., 0., 1.],
               [0., 0., 0., 0., 1.],
               [0., 1., 0., 0., 0.]])
\sqrt{2} [79] x = np.delete(x, [4,5], axis=1)
```

```
x.shape
    (11231, 6)
/ [81] #using concatenate
        x=np.concatenate((t,z,x),axis=1)

√ [82] x.shape

        (11231, 16)
   get dummies:

v [83] dataset=pd.get_dummies(dataset,columns=['ORIGIN','DEST'])

    #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)
✓ [85] x_test.shape
        (2247, 16)
√ [86] x_train.shape
        (8984, 16)
√ [87] y_test.shape
        (2247, 1)
√ [88] y_train.shape
        (8984, 1)
_{\text{Os}}^{\prime} [89] #StandardScaler scaling the data
       x_train = sc.fit_transform(x_train)
        x_test = sc.fit_transform(x_test)
```

Milestone 4:

```
##TASK 4 - Model building
           #Decision tree model
           classifier = DecisionTreeClassifier(random state = 0)
           classifier.fit(x_train, y_train)
           decisiontree = classifier.predict(x_test)
           decisiontree
     T array([1., 0., 0., ..., 0., 0., 1.])
    [91] #check accuracy
           desacc = accuracy_score(y_test, decisiontree)
           0.8673787271918113
           #RandomForestClassifier
           from sklearn.ensemble import RandomForestClassifier
           rfc=RandomForestClassifier(n_estimators=10,criterion='entropy')
           rfc.fit(x_train,y_train)
                                        RandomForestClassifier
            RandomForestClassifier(criterion='entropy', n_estimators=10)
y_predict=rfc.predict(x_test)
     y predict
     \mathsf{array}([\textbf{0., 0., 0., ..., 0., 0., 1.}])
√ [97] #ANN model
      import tensorflow
      from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense
     #creating ANN skleton view
     classification=Sequential()
     {\tt 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'))
y [98] #compilling the ANN model classification.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
     classification.fit (x\_train, y\_train, batch\_size=4, validation\_split=0.2, epochs=100)
     Epoch 1/100
                     ========] - 4s 2ms/step - loss: 0.2878 - accuracy: 0.8948 - val_loss: 0.2802 - val_accuracy: 0.9060
     Epoch 2/100
     1797/1797 [==
Epoch 3/100
                  1797/1797 [===========] - 3s 2ms/step - loss: 0.2672 - accuracy: 0.9055 - val_loss: 0.2723 - val_accuracy: 0.9076
     Epoch 96/100
1797/1797 [====
                 Epoch 97/100
     1797/1797 [============================ ] - 3s 2ms/step - loss: 0.0555 - accuracy: 0.9773 - val loss: 1.2785 - val accuracy: 0.8798
     Epoch 98/100
     1797/1797 [====
Epoch 99/100
1797/1797 [====
Epoch 100/100
                 1797/1797 [=============] - 3s 2ms/step - loss: 0.0447 - accuracy: 0.9837 - val_loss: 1.3087 - val_accuracy: 0.8653 
<a href="keras.callbacks.History">keras.callbacks.History</a> at 0x7fce603f3190>
```

```
_{\text{Os}}^{\checkmark} [99] #Activity 2: 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]])
         y_pred
        array([0.])
√ [100] #RandomForest
         y_pred=rfc.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1,1]])
         y_pred
        array([0.])

v [101] classification.save('flight.h5')
_{\text{Os}}^{\checkmark} [102] #Testing the model
        y_pred=classification.predict(x_test)
y_pred
    array([[9.9998450e-01],
                [2.0697034e-06],
               [6.7771194e-03],
               [2.6455763e-15],
                [4.0001239e-16],
               [9.9998057e-01]], dtype=float32)

    [104] y_pred=(y_pred>0.5)
        y_pred
        array([[ True],
                [False],
               [False],
               [False],
               [False],
               [ True]])

v [105] def predict_exit(sample_value):
          #convert list to numpy array
          sample_value=np.array(sample_value)
          #Reshape because sample value is contains only 1 value
          sample_value=sample_value.reshape(1,-1)
          #Feature scaling
          sample value=sc.transform(sample value)
          return classifier.predict(sample value)
```

Milestone 5:

```
#task 5 Performance Testing & Hyperparameter Tuning
    #Compare the model
    from sklearn import model selection
    from sklearn.neural_network import MLPClassifier
    def classification report():
      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.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['models']=name
        dfs.append(this_df)
        final=pd.concat(dfs,ignore_index=True)
      return final
```

```
#RandomForest Accuracy
[108] print('Training accuracy: ',accuracy_score(y_pred,y_predict))
        print('Testing accuracy: ',accuracy score(y test,y predict))
        Training accuracy: 0.9185580774365821
        Testing accuracy: 0.8976412995104583
_{0s}^{\checkmark} [109] #making the confusion matrix
        from sklearn.metrics import confusion matrix
        cm= confusion matrix(y test,y predict)
        array([[1872, 64],
               [ 166, 145]])
[110] #accuracy score of desiciontree
        from sklearn.metrics import accuracy score
        desacc=accuracy score(y test,decisiontree)
        desacc
        0.8673787271918113
    from sklearn.metrics import confusion matrix
         cm=confusion_matrix(y_test,decisiontree)
         cm
    array([[1778, 158],
                [ 140, 171]])
_{
m 0s}^{\prime} [112] #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))
        The accuracy for ANN model is :87.49443702714731%
_{	extstyle 0s}^{	extstyle \prime} [113] #making the confusion matrix
         cm=confusion_matrix(y_test,y_pred)
         array([[1803, 133],
                [ 148, 163]])
```

```
^{\vee} [114] #giving some parameters that can be used in randized search cv
       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]
V [115] # Performing the randomized cv
       from sklearn.model_selection import RandomizedSearchCV
       RCV = RandomizedSearchCV(estimator = rfc, param_distributions = parameters, cv = 10, n_iter = 4)
/ [116] RCV.fit(x_train,y_train)
       building tree 1 of 68
       building tree 2 of 68
       building tree 3 of 68
       building tree 4 of 68
       building tree 5 of 68
       building tree 6 of 68
       building tree 7 of 68
       building tree 8 of 68
       building tree 9 of 68
       building tree 10 of 68
       building tree 11 of 68
       building tree 12 of 68
        ------
       building tree 81 of 90
       building tree 82 of 90
        building tree 83 of 90
        building tree 84 of 90
        building tree 85 of 90
       building tree 86 of 90
        building tree 87 of 90
        building tree 88 of 90
        building tree 89 of 90
        building tree 90 of 90
        building tree 1 of 68
        huilding tree 2 of 68
[117] #getting the best paarmets from the giving list and best score from them
        bt_params=RCV.best_params_
        bt score=RCV.best score
        bt params
```

```
nattatus rues op oi an
building tree 84 of 90
   D building tree 85 of 90
       building tree 86 of 90
       building tree 87 of 90
       building tree 88 of 90
/ [120] y_predict_rf=RCV.predict(x_test)
       [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
       [Parallel(n jobs=1)]: Done 1 out of 1 | elapsed: 0.0s remaining:
                                                                            0.05
       [Parallel(n jobs=1)]: Done 2 out of 2 | elapsed: 0.0s remaining:
                                                                            0.05
       [Parallel(n_jobs=1)]: Done 3 out of 3 | elapsed: 0.0s remaining:
                                                                            0.05
       [Parallel(n jobs=1)]: Done 4 out of 4 | elapsed: 0.0s remaining:
                                                                           0.05
       [Parallel(n jobs=1)]: Done 5 out of 5 | elapsed: 0.0s remaining:
                                                                            0.05
       [Parallel(n_jobs=1)]: Done 68 out of 68 | elapsed: 0.0s finished
[121] RCV=accuracy_score(y_test,y_predict_rf)
       RCV
       0.910547396528705
  import pickle
       pickle.dump(RCV,open('flight.pkl','wb'))
```

Milestone 6:

```
from flask import Flask,render_template,request
import pickle
import numpy as np
model = pickle.load(open('flight.pkl', 'rb'))
app = Flask(__name__, template_folder='template')
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/prediction', methods = ['POST'])
def predict():
    name = request.form['name']
    month = request.form['month']
    dayofmonth = request.form['dayofmonth']
    dayofweek = request.form['dayofweek']
    origin = request.form['origin']
    if(origin == "msp"):
        origin0, origin1, origin2, origin3, origin4 = 0,0,0,0,1
    if(origin == "dtw"):
        origin0, origin1, origin2, origin3, origin4 = 1,0,0,0,0
    if(origin == "jfk"):
        origin0,origin1,origin2,origin3,origin4 = 0,0,1,0,0
    if(origin == "sea"):
        origin0, origin1, origin2, origin3, origin4 = 0,1,0,0,0
    if(origin == "alt"):
```

```
origin0,origin1,origin2,origin3,origin4 = 0,1,0,0,0
if(origin == "alt"):
   origin0,origin1,origin2,origin3,origin4 = 0,0,0,1,0
destination = request.form['destination']
if(destination == "msp"):
   destination0,destination1,destination2,destination3,destination4 = 0,0,0,0,1
if(destination == "dtw"):
   destination0,destination1,destination2,destination3,destination4 = 1,0,0,0,0
if(destination =
   destination0, destination1, destination2, destination3, destination4 = 0,0,1,0,0
if(destination == "sea"):
   destination0,destination1,destination2,destination3,destination4 = 0,1,0,0,0
if(destination == "alt"):
   dept = request.form['dept']
arrtime = request.form['arrtime']
actdept = request.form['actdept']
dept15 = int(dept) - int(actdept)
total = [[name,month,dayofmonth,dayofweek,origin0,origin1,origin2,origin3,origin4,destination0,de
print(total)
y_pred = model.predict(total)
print(y_pred)
y_pred = [0.]
if(y_pred == [0.]):
   ans = "The Flight will be On Time"
```

```
y pred = model.predict(total)
19
        print(y pred)
        y_pred = [0.]
0
51
        if(y pred == [0.]):
52
            ans = "The Flight will be On Time"
53
            ans = "The Flight will be Delayed"
54
55
        return render_template('index.html', showcase = ans)
6
    if _ name__ == '__main__':
        app.run(debug = True)
```

User interface wep page code:

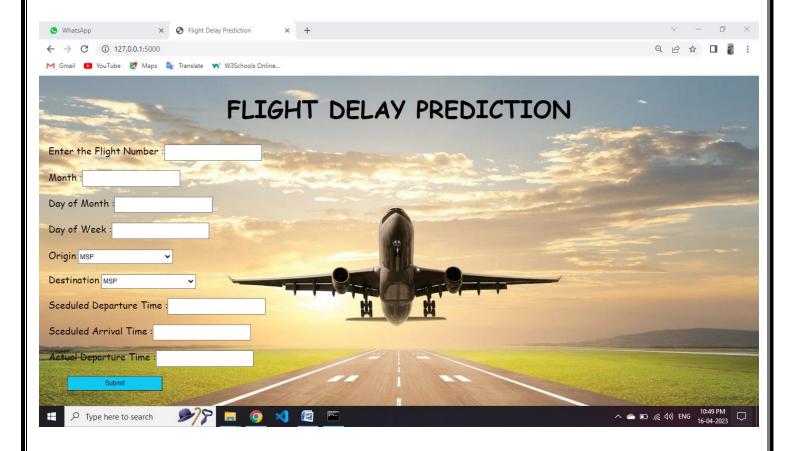
```
<!DOCTYPE html>
2 v <html>
        @import url('https://fonts.google.com/specimen/Balsamiq+Sans');
            body{
                width: 100%;
                margin: 0px;
                background-image: url('https://wallpaperaccess.com/full/1470798.jpg');
8
            .header{
                top: 0;
                width: 100%;
                height: 90px;
                font-family: 'Balsamiq Sans', cursive;
               font-size: 25px;
               font-weight: 800px;
                text-align: center;
18
19 ~
            .MAIN p, label{
                font-size: 20px;
20
                margin-left: 20px;
                font-family: 'Balsamiq Sans', cursive;
23
            .MAIN input, select{
24 4
                height: 30px;
                width: 200px;
```

```
.MAIN button{
       height: 30px;
       width: 200px;
       margin-left: 60px;
       background-color: ■#daa520;
    .MAIN b{
       font-size: 20px;
       font-weight: 800px;
       text-align: center;
       font-family: 'Balsamiq Sans', cursive;
       margin-left: 20px;
</style>
<title>Flight Delay Prediction</title>
<body background="1470798.jpg">
   <div class="header">
   <h1>FLIGHT DELAY PREDICTION</h1>
   </div>
   <div class="MAIN">
       <form action="http://localhost:5000/prediction" method="post">
            Enter the Flight Number :<span><input type="text" name="name"/></span>
            Month :<span><input type="text" name="month"/></span>
            Day of Month :<span><input type="text" name="dayofmonth"/></span>
            Day of Week :<span><input type="text" name="dayofweek"/></span>
           <label for="origin">Origin</label>
           <select name="origin">
```

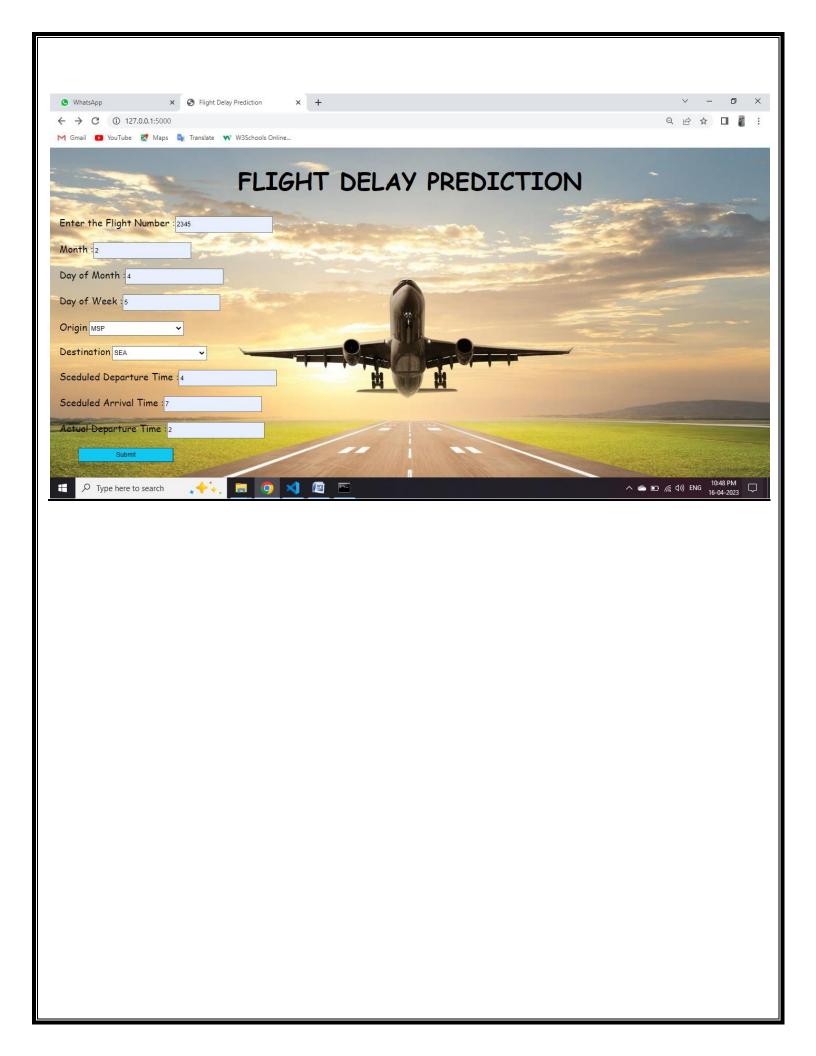
```
<label for="origin">Origin</label>
                <select name="origin">
                   <option value="msp">MSP</option>
                   <option value="dtw">DTW</option>
                   <option value="jfk">JFK</option>
                   <option value="sea">SEA</option>
                   <option value="alt">ALT</option>
                </select>
                <br><br><br>>
                <label for="destination">Destination</label>
                <select name="destination">
                    <option value="msp">MSP</option>
                   <option value="dtw">DTW</option>
                   <option value="jfk">JFK</option>
                   <option value="sea">SEA</option>
                   <option value="alt">ALT</option>
                </select>
                 Sceduled Departure Time :<span><input type="text" name="dept"/></span>
                 Sceduled Arrival Time :<span><input type="text" name="arrtime"/></span>
                 Actual Departure Time :<span><input type="text" name="actdept"/></span>
                <div class="form-btn">
                   <button class="submit-btn">Submit
                   <br/>
<br/>
dr>dr>dr>dr>dr>dr>dr>
                </div>
           </form>
        </div>
    </body>
</html>
```

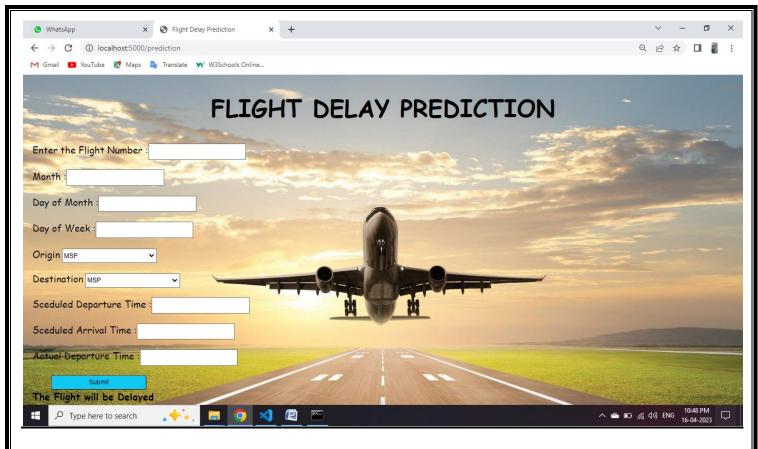
Output: C:\Windows\System32\cmd.exe-python app.py Microsoft Windows [Version 10.0.19045.2846] (c) Microsoft Corporation. All rights reserved. E:\ML project\flight_delay_prediction 1\flask>python app.py * Serving Flask app 'app' * Debug mode: on WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead. * Running on http://127.0.0.1:5000 Press CTRL+C to quit * Restarting with stat * Debugger is active! * Debugger PIN: 533-497-909

Then the next step is go to web browser and write the localhost URL (http://127.0.0.1:5000) to get the below result.



The user will give input to get the predicted result after clicking the submit button.





Thank you..