FLIGHT DELAY PREDICTION FOR AVIATION INDUSTRY USING MACHINE LEARNING

PRESENTED BY:
S.KIRUBA MONIKA MARY(20326ER052)
S.LAVANYA(20326ER053)
V.MAHALAKSHMI(20326ER054)
S.MEENA(20326ER055)

CONTENT

- ABSTRACT
- INTRODUCTION
- PROBLEM DEFINITION & DESIGN THINKING
- RESULT
- APPENDIX
- ADVANTAGE
- APPLICATION
- CONCLUSION
- FUTURE ENHANCEMENT

ABSTRACT

- OVER the last twenty years, air travel has been increasingly preferred among traveller , 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, 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.

INTRODUCTION

- over the last twenty years, air travel has been increasingly preferred among travellers, mainly because of its speed and in some cases comfort.
- 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

OVER VIEW

- Flight delays can cause significant disruptions to travel plans and can have a negative impact on airline operations.
- In this project, we aim to build a machine learning model that can predict the flight delays based on historical data.
- The model will be trained on a dataset containing information such as flight schedules, weather conditions, and other factors that can influence flight delays.
- To achieve this, we will use various machine learning algorithms such as decision trees, random forests, and neural networks.
- Overall, this project aims to develop a useful tool for the aviation industry that can improve the accuracy of flight delay predictions, ultimately benefiting both airlines and traveller alike.

PURPOSE

- ❖The purpose of analyzing and predicting flight delays to help airlines and passengers better plan and manage their travel.
- ❖This could involve collecting and analyzing data on factors that contribute to flight delays, such as weather, air traffic, and mechanical issues.
- ❖The project may also involve developing algorithms or machine learning models to predict delays and provide real-time updates to passengers.
- ❖Ultimately, the purpose of such a project would be to improve the efficiency and reliability of air travel and reduce the negative impact of flight delays on passengers and the industry.

PROBLEM DEFINITION & DESIGN THINKING

- Specify the Business Problem

The impact of flight delay can be a risk and this risk represents financial losses, the dissatisfaction of passengers, time losses, loss of reputation and bad business relations.

- Business Requirements

Business requirements, also known as stakeholder requirements specification from the viewpoint of the system's end user like a CONOPS.

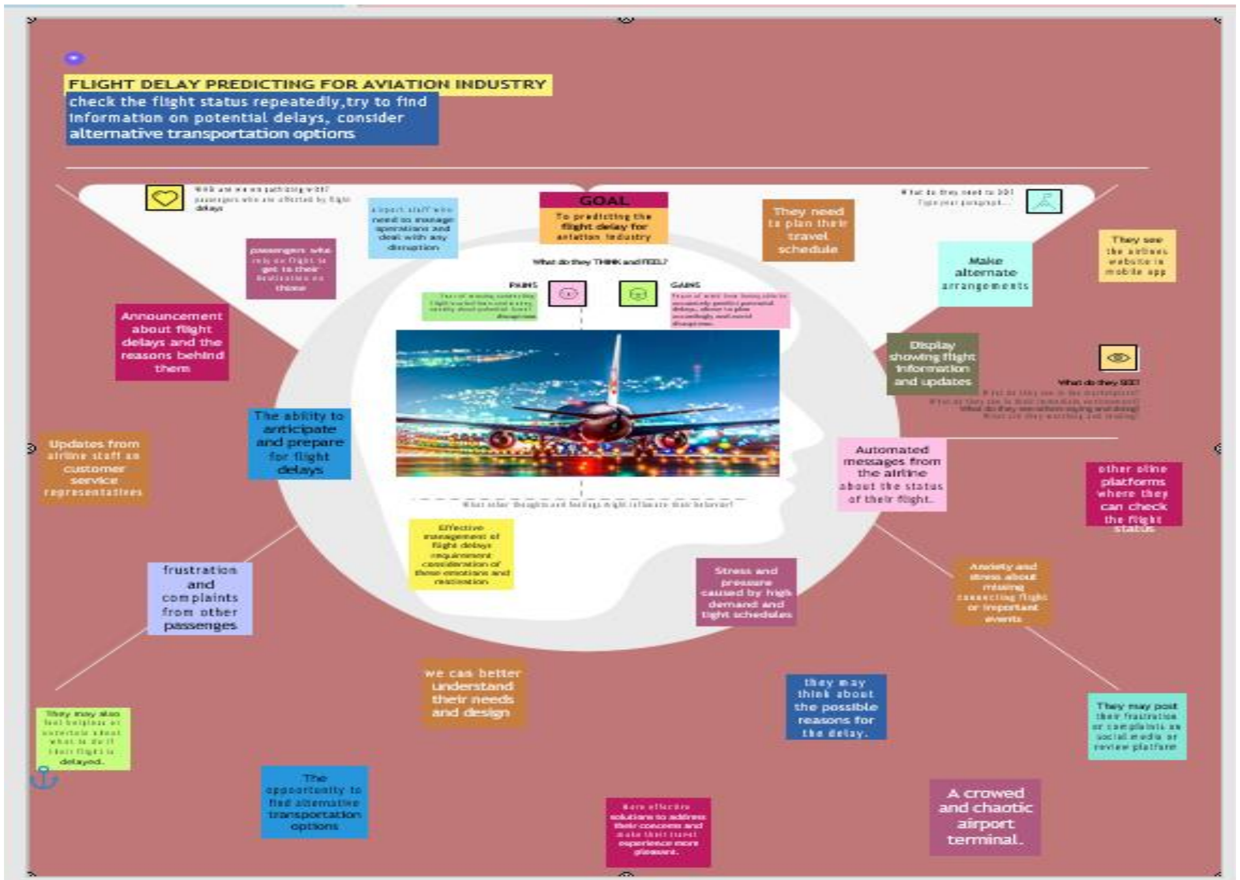
- Literature Survey

It is a systematic method for identifying, evaluating and interpreting thee work produced by researchers, scholars and practitioners.

- Social Or Business Impact

Flight delay not only irritate air passengers and disrupt their schedules but also cause a decrease in efficiency, an increase capital costs, reallocation of flight crews aircraft.

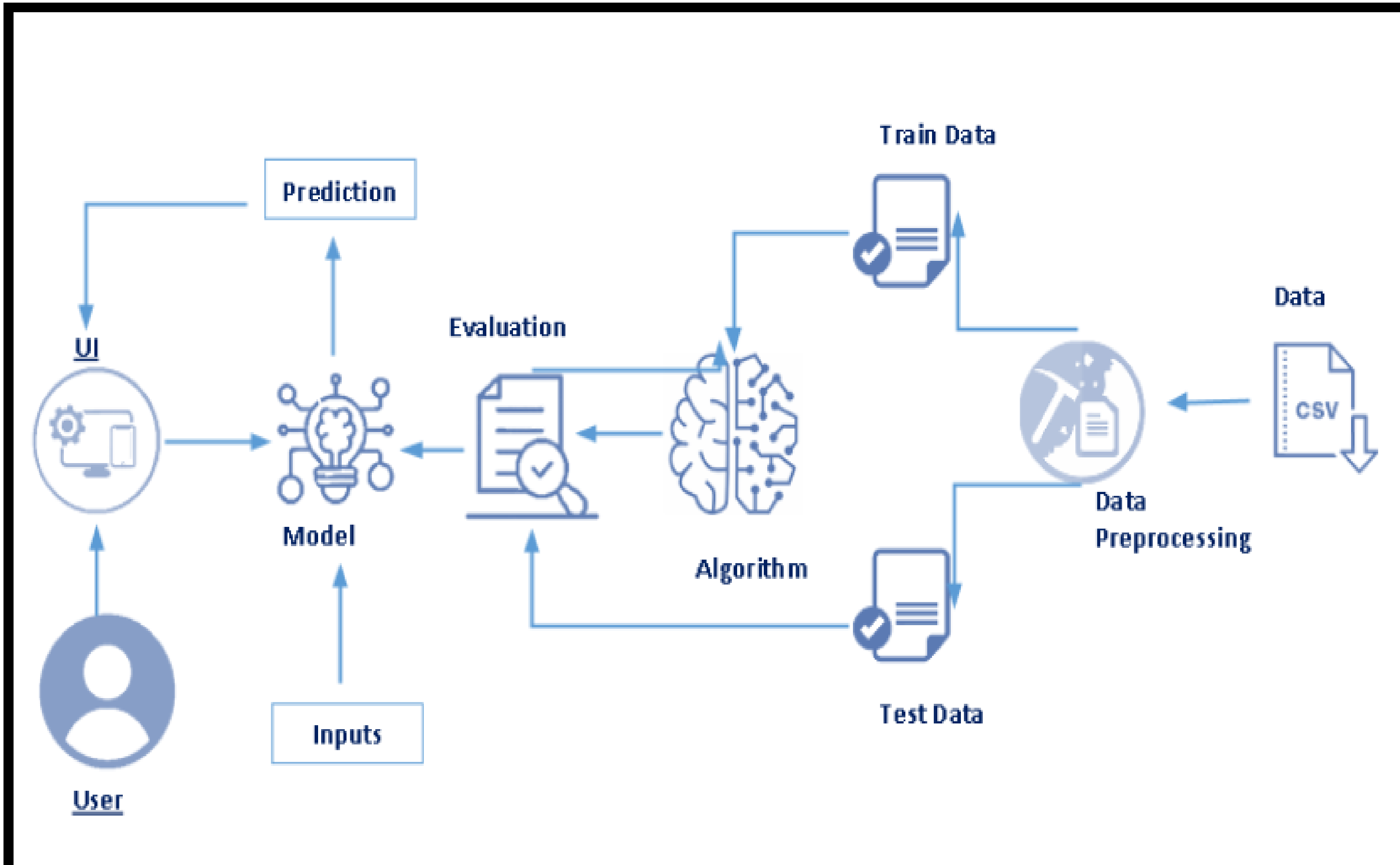
EMPATHY MAP



FLIGHT DELAY PREDICTING FOR AVIATION INDUSTRY
check the flight status repeatedly,try to find information on potential delays, consider alternative transportation options
GOAL
To predicting the flight delay for aviation industry
What do they THINK and FEEL?
PAINS
GAINS
They need to plan their travel schedule
Make alternate arrangements
They see the airlines website in mobile app
Announcement about flight delays and the reasons behind them
Updates from airline staff as customer service representatives
The ability to anticipate and prepare for flight delays
Display showing flight information and updates
What do they SEE?
Automated messages from the airline about the status of their flight.
other oline platforms where they can check the flight status
Effective management of flight delays requirement consideration of these emotions and restoration
frustration and complaints from other passenges
Stress and pressure caused by high demand and tight schedules
Anxiety and stress about missing connecting flight or important events
we can better understand their needs and design
they may think about the possible reasons for the delay.
They may post their frustration or complaints on social media or review platform
They may also feel helpless or uncertain about what to do if their flight is delayed.
The opportunity to find alternative transportation options
A crowed and chaotic airport terminal.

IDEATION & BRAINSTROMING MAP

TECHNIQUAL ARCHITECTURE



RESULT


127.0.0.1:5000
Prediction of Flight Delay
Enter the Flight Number :
Month :
Day of Month :
Day of Week :
origin MSP
destination MSP
Scheduled Departure Time :
Scheduled Arrival Time :
Actual Departure Time :
SUBMIT


127.0.0.1:5000
Prediction of Flight Delay
Enter the Flight Number : 1399
Month : 2
Day of Month : 4
Day of Week : 5
origin JFK
destination SEA
Scheduled Departure Time : 1
Scheduled Arrival Time : 22
Actual Departure Time : 1
SUBMIT


localhost:5000/prediction
Prediction of Flight Delay
Enter the Flight Number :
Month :
Day of Month :
Day of Week :
origin MSP
destination MSP
Scheduled Departure Time :
Scheduled Arrival Time :
Actual Departure Time :
SUBMIT
The Flight will be on time

APPENDIX

\+ Code + Text

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plot
%matplotlib inline
import seaborn as sns
import sklearn
import pandas
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
```

```
dataset=pd.read_csv("/content/flightdata.csv")
dataset.head()
```

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	CRS_AR
0	2016	1	1	1	5	DL	N836DN	1399	10397	ATL	...	
1	2016	1	1	1	5	DL	N964DN	1476	11433	DTW	...	
2	2016	1	1	1	5	DL	N813DN	1597	10397	ATL	...	
3	2016	1	1	1	5	DL	N587NW	1768	14747	SEA	...	
4	2016	1	1	1	5	DL	N836DN	1823	14747	SEA	...	

5 rows × 26 columns

+ Code + Text

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11231 entries, 0 to 11230
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   YEAR                 11231 non-null  int64
 1   QUARTER              11231 non-null  int64
 2   MONTH                11231 non-null  int64
 3   DAY_OF_MONTH         11231 non-null  int64
 4   DAY_OF_WEEK          11231 non-null  int64
 5   UNIQUE_CARRIER       11231 non-null  object
 6   TAIL_NUM             11231 non-null  object
 7   FL_NUM               11231 non-null  int64
 8   ORIGIN_AIRPORT_ID    11231 non-null  int64
 9   ORIGIN               11231 non-null  object
 10  DEST_AIRPORT_ID      11231 non-null  int64
 11  DEST                 11231 non-null  object
 12  CRS_DEP_TIME         11231 non-null  int64
 13  DEP_TIME             11124 non-null  float64
 14  DEP_DELAY            11124 non-null  float64
 15  DEP_DEL15            11124 non-null  float64
 16  CRS_ARR_TIME         11231 non-null  int64
 17  ARR_TIME             11116 non-null  float64
 18  ARR_DELAY            11043 non-null  float64
 19  ARR_DEL15            11043 non-null  float64
 20  CANCELLED            11231 non-null  float64
 21  DIVERTED             11231 non-null  float64
 22  CRS_ELAPSED_TIME     11231 non-null  float64
 23  ACTUAL_ELAPSED_TIME  11043 non-null  float64
 24  DISTANCE             11231 non-null  float64
 25  Unnamed: 25          0 non-null      float64
dtypes: float64(12), int64(10), object(4)
memory usage: 2.2+ MB
```

+ Code + Text

```python
dataset = dataset.drop('Unnamed: 25', axis=1)
dataset.isnull().sum()
```

```
YEAR                     0
QUARTER                  0
MONTH                    0
DAY_OF_MONTH             0
DAY_OF_WEEK              0
UNIQUE_CARRIER           0
TAIL_NUM                 0
FL_NUM                   0
ORIGIN_AIRPORT_ID        0
ORIGIN                   0
DEST_AIRPORT_ID          0
DEST                     0
CRS_DEP_TIME             0
DEP_TIME               107
DEP_DELAY              107
DEP_DEL15              107
CRS_ARR_TIME             0
ARR_TIME               115
ARR_DELAY              188
ARR_DEL15              188
CANCELLED                0
DIVERTED                 0
CRS_ELAPSED_TIME         0
ACTUAL_ELAPSED_TIME    188
DISTANCE                 0
dtype: int64
```

+ Code + Text

```
#filter the dataset to clininate columns that aren't relevant to a predictive model.
dataset = dataset[["FL_NUM", "MONTH", "DAY_OF_MONTH", "DAY_OF_WEEK", "ORIGIN", "DEST", "CRS_ARR_TIME", "DEP_DELAY","DEP_DEL15", "ARR_DELAY","ARR_DEL15"]]
dataset.isnull().sum()
```

```
FL_NUM           0
MONTH            0
DAY_OF_MONTH     0
DAY_OF_WEEK      0
ORIGIN           0
DEST             0
CRS_ARR_TIME     0
DEP_DELAY      107
DEP_DEL15      107
ARR_DELAY      188
ARR_DEL15      188
dtype: int64
```

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

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	-2.0	0.0	NaN	NaN
179	86	1	10	7	MSP	DTW	1632	NaN	NaN	NaN	NaN
184	557	1	10	7	MSP	DTW	912	-5.0	0.0	NaN	NaN
210	1096	1	10	7	DTW	MSP	1303	NaN	NaN	NaN	NaN
478	1542	1	22	5	SEA	JFK	723	NaN	NaN	NaN	NaN
481	1795	1	22	5	ATL	JFK	2014	NaN	NaN	NaN	NaN
491	2312	1	22	5	MSP	JFK	2149	NaN	NaN	NaN	NaN
499	423	1	23	6	JFK	ATL	1600	NaN	NaN	NaN	NaN
500	425	1	23	6	JFK	ATL	1827	NaN	NaN	NaN	NaN

✓ 0s completed at 9:08 PM

+ Code + Text

500	425	1	23	6	JFK	ATL	1827	NaN	NaN	NaN	NaN
501	427	1	23	6	JFK	SEA	1053	NaN	NaN	NaN	NaN

[11]

```
dataset['DEP_DEL15'].mode()
```

```
0    0.0
Name: DEP_DEL15, dtype: float64
```

[12]

```
dataset=dataset.fillna({'ARR_DEL15': 1})
dataset=dataset.fillna({'DEP_DEL15': 0})
dataset=dataset.fillna({'ARR_DELAY': 1})
dataset=dataset.fillna({'DEP_DELAY': 0})
dataset.iloc[177:185]
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	-2.0	0.0	1.0	1.0
178	2839	1	9	6	DTW	JFK	1724	-4.0	0.0	-15.0	0.0
179	86	1	10	7	MSP	DTW	1632	0.0	0.0	1.0	1.0
180	87	1	10	7	DTW	MSP	1649	24.0	1.0	14.0	0.0
181	423	1	10	7	JFK	ATL	1600	11.0	0.0	7.0	0.0
182	440	1	10	7	JFK	ATL	849	-1.0	0.0	-14.0	0.0
183	485	1	10	7	JFK	SEA	1945	65.0	1.0	10.0	0.0
184	557	1	10	7	MSP	DTW	912	-5.0	0.0	1.0	1.0

```
[13] import math

     for index, row in dataset.iterrows():
         dataset.loc[index, 'CRS_ARR_TIME'] = math.floor(row['CRS_ARR_TIME'] / 100)
     dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15
0	1399	1	1	5	ATL	SEA	21	2.0	0.0	-41.0	0.0
1	1476	1	1	5	DTW	MSP	14	-1.0	0.0	4.0	0.0
2	1597	1	1	5	ATL	SEA	12	2.0	0.0	-33.0	0.0
3	1768	1	1	5	SEA	MSP	13	1.0	0.0	10.0	0.0
4	1823	1	1	5	SEA	DTW	6	-4.0	0.0	8.0	0.0

```
[14] from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     dataset['DEST'] = le.fit_transform(dataset['DEST'])
     dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```

s Help All changes saved

+ Code + Text RAM Disk

```
[14] dataset['DEST'] = le.fit_transform(dataset['DEST'])
     dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```

```
[16] dataset.head(5)
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15
0	1399	1	1	5	0	4	21	2.0	0.0	-41.0	0.0
1	1476	1	1	5	1	3	14	-1.0	0.0	4.0	0.0
2	1597	1	1	5	0	4	12	2.0	0.0	-33.0	0.0
3	1768	1	1	5	4	3	13	1.0	0.0	10.0	0.0
4	1823	1	1	5	4	1	6	-4.0	0.0	8.0	0.0

```
dataset['ORIGIN'].unique()

array([0, 1, 4, 3, 2])
```

```
[ ] dataset = pd.get_dummies(dataset, columns=['ORIGIN', 'DEST'])
    dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2	ORIGIN_3
0	1399	1	1	5	21	2.0	0.0	-41.0	0.0	1	0	0	0
1	1476	1	1	5	14	-1.0	0.0	4.0	0.0	0	1	0	0

0s completed at 9:24PM

\+ Code + Text

```
[17] dataset['ORIGIN'].unique()

array([0, 1, 4, 3, 2])
```

```
[18] dataset = pd.get_dummies(dataset, columns=['ORIGIN', 'DEST'])
     dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2	ORIGIN_3	O
0	1399	1	1	5	21	2.0	0.0	-41.0	0.0	1	0	0	0	
1	1476	1	1	5	14	-1.0	0.0	4.0	0.0	0	1	0	0	
2	1597	1	1	5	12	2.0	0.0	-33.0	0.0	1	0	0	0	
3	1768	1	1	5	13	1.0	0.0	10.0	0.0	0	0	0	0	
4	1823	1	1	5	6	-4.0	0.0	8.0	0.0	0	0	0	0	

```
[19] x = dataset.iloc[:, 0:8].values
     y = dataset.iloc[:, 8:9].values
```

\+ Code + Text

```
x

array([[ 1.399e+03,  1.000e+00,  1.000e+00, ...,  2.000e+00,  0.000e+00,
        -4.100e+01],
       [ 1.476e+03,  1.000e+00,  1.000e+00, ..., -1.000e+00,  0.000e+00,
         4.000e+00],
       [ 1.597e+03,  1.000e+00,  1.000e+00, ...,  2.000e+00,  0.000e+00,
        -3.300e+01],
       ...,
       [ 1.823e+03,  1.200e+01,  3.000e+01, ...,  0.000e+00,  0.000e+00,
        -1.600e+01],
       [ 1.901e+03,  1.200e+01,  3.000e+01, ..., -1.000e+00,  0.000e+00,
        -5.000e+00],
       [ 2.005e+03,  1.200e+01,  3.000e+01, ..., -2.000e+00,  0.000e+00,
        -1.200e+01]])
```

```
[21] from sklearn.preprocessing import OneHotEncoder
     oh = OneHotEncoder()
     z=oh.fit_transform(x[:,4:5]).toarray()
     t=oh.fit_transform(x[:,5:6]).toarray()
```

```
z

array([[0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Help

+ Code + Text

RAM
Disk

```
t
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

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

```
dataset.describe()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DELAY	DEP_DEL15	ARR_DELAY	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2
count	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000
mean	1334.325617	6.628973	15.790758	3.960199	15.067314	8.379663	0.141483	-2.513311	0.139168	0.276022	0.195975	0.122340
std	811.875227	3.354678	8.782056	1.995257	5.023534	36.596640	0.348535	38.905443	0.346138	0.447048	0.396967	0.327693
min	7.000000	1.000000	1.000000	1.000000	0.000000	-16.000000	0.000000	-67.000000	0.000000	0.000000	0.000000	0.000000
25%	624.000000	4.000000	8.000000	2.000000	11.000000	-3.000000	0.000000	-19.000000	0.000000	0.000000	0.000000	0.000000
50%	1267.000000	7.000000	16.000000	4.000000	15.000000	-1.000000	0.000000	-10.000000	0.000000	0.000000	0.000000	0.000000
75%	2032.000000	9.000000	23.000000	6.000000	19.000000	4.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000
max	2853.000000	12.000000	31.000000	7.000000	23.000000	645.000000	1.000000	615.000000	1.000000	1.000000	1.000000	1.000000

+ Code + Text

```
[30] 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)
```

```
[31] 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)
```

```
[32] x_test.shape
```

```
(2247, 6)
```

```
x_train.shape
```

```
(8984, 6)
```

```
[34] y_test.shape
```

```
(2247, 1)
```

```
[35] y_train.shape
```

```
(8984, 1)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(random_state=0)
classifier.fit(x_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

```
[38] decisiontree = classifier.predict(x_test)
```

```
[39] decisiontree
```

```
array([1., 0., 0., ..., 0., 0., 1.])
```

```
[40] from sklearn.metrics import accuracy_score
desacc = accuracy_score(y_test,decisiontree)
```

```
[41] from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=10,criterion='entropy')
```

```
rfc.fit(x_train,y_train)
```

```
<ipython-input-42-b87bb2ba9825>:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for
  rfc.fit(x_train,y_train)
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

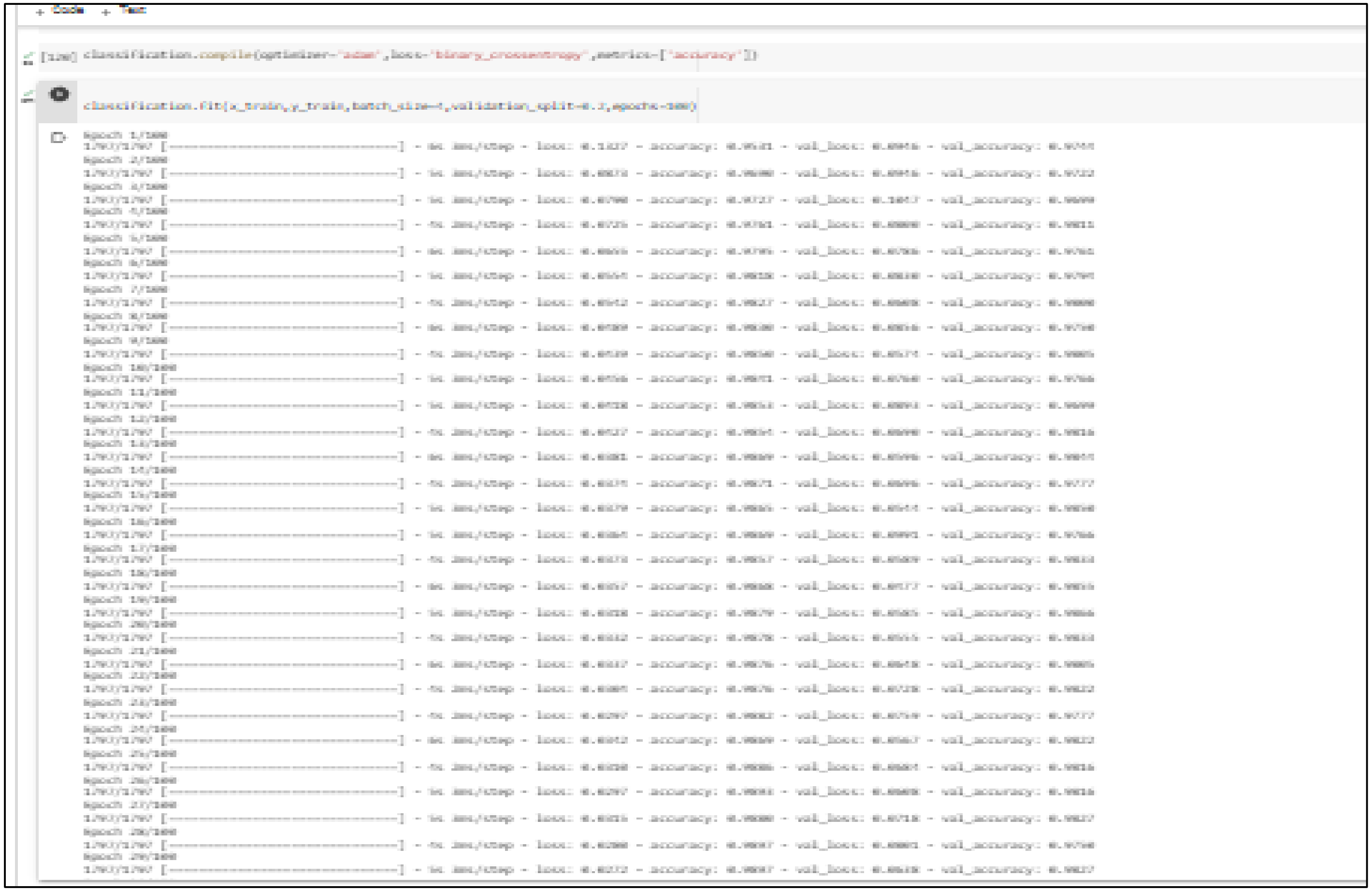
```
<ipython-input-116-b87bb2ba9825>:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape
  rfc.fit(x_train,y_train)
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

```
[117] y_predict = rfc.predict(x_test)
```

```
[118] 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'))
```

```
[120] classification.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```


+ Code + Text
[120] classification.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

ols Help Save failed

+ Code + Text

```
1797/1797 [==============================] - 4s 2ms/step - loss: 0.0264 - accuracy: 0.9903 - val_loss:
```

```
y_pred = classifier.predict([[129,99,1,0,0,0]])
print(y_pred)
(y_pred)

[0.]
array([0.])
```

```
[123] y_pred=rfc.predict([[129,99,1,0,0,0]])
      print(y_pred)
      (y_pred)

[0.]
array([0.])
```

```
[124] classification.save('flight.h5')
```

```
[125] y_pred=classification.predict(x_test)

71/71 [==============================] - 0s 2ms/step
```

```
[126] y_pred

array([[1.0000000e+00],
       [0.0000000e+00],
       [1.6780123e-20],
       ...,
       [3.2196211e-19],
       [1.4602942e-25],
       [1.0000000e+00]], dtype=float32)
```

+ Code + Text

```
y_pred=(y_pred>0.5)
y_pred

array([[ True],
       [False],
       [False],
       ...,
       [False],
       [False],
       [ True]])
```

+ Code + Text

```
[128] 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)
```

```
[129] test = classification.predict([[1, 1, 121.000000, 36.0, 0, 0 ]])
      if test==1:
          print('Prediction: Chance of delay')
      else:
          print('Prediction: No chance of delay')

1/1 [==============================] - 0s 40ms/step
Prediction: No chance of delay
```

```
[130] from sklearn import model_selection
      from sklearn.neural_network import MLPClassifier
```

+ Code + Text

```
[130] from sklearn.neural_network import MLPClassifier
```

```
    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['model']=name
        dfs.append(this_df)
final=pd.concat(dfs,ignore_index=True)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,
  estimator.fit(X_train, y_train, **fit_params)
<ipython-input-131-4982c366b9c7>:14: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  clf=model.fit(x_train,y_train)
RF
              precision    recall  f1-score   support

    no delay       0.96      1.00      0.98      1936
       delay       0.99      0.77      0.86       311

    accuracy                           0.97      2247
   macro avg       0.98      0.88      0.92      2247
weighted avg       0.97      0.97      0.96      2247

DecisionTree
              precision    recall  f1-score   support
```

ols Help Save failed

\+ Code + Text

```
[213] y_pred = y_pred.ravel()
```

```
from sklearn.metrics import accuracy_score
print ('Testing accuracy:', accuracy_score(y_test,y_predict))
```

```
Testing accuracy: 0.9666221628838452
```

```
[133] from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_predict)
cm
```

```
array([[1934,    2],
       [  73,  238]])
```

```
from sklearn.metrics import accuracy_score
desacc= (y_test,decisiontree)
```

```
[137] desacc
```

```
0.9643969737427681
```

```
[137] desacc
```

```
0.9643969737427681
```

```
[139] from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,decisiontree)
```

```
[140] cm
```

```
array([[1929,    7],
       [  73,  238]])
```

```
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:96.70672007120605%
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
cm
```

Tools Help Save failed

\+ Code \+ Text

```
[143] 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]

      }
```

```
[144] RCV = RandomizedSearchCV(estimator=rfc, param_distributions=parameters, cv=10, n_iter=4)
```

```
[145] RCV.fit(x_train,y_train)
```

```
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    0.0s remaining:    0.0s
```

ools Help Save failed

Comment Share

+ Code + Text

RAM
Disk

```
[146] bt_params=RCV.best_params_
      bt_score=RCV.best_score_
```

```
[147] bt_params
```

```
{'verbose': 10,
 'n_estimators': 30,
 'max_features': 'sqrt',
 'max_depth': 10,
 'criterion': 'entropy'}
```

```
[148] bt_score
```

```
0.9916513275081691
```

```
[149] model=RandomForestClassifier(verbose=10,n_estimators=120,max_features='log2',max_depth=10,criterion='entropy')
      RCV.fit(x_train,y_train)
```

Help Save failed

+ Code + Text

```
[150]
      [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
      [Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.0s finished
```

```
[151] RFC=accuracy_score(y_test,y_predict_rfc)
      RFC
```

```
0.9666221628838452
```

```
[152] import pickle
      pickle.dump(RCV,open('flight.pkl','wb'))
```

```
[153] from flask import Flask,request,render_template
      import numpy as np
      import pandas as pd
      import pickle
      import os
```

```
[154] model=pickle.load(open('flight.pkl','rb'))
```

fflightdata.ipynb

File Edit View Insert Runtime Tools Help Last saved at 11:22 PM

+ Code + Text

```
[221] app.route('/')
      def home():
          return render_template("index.html")
      app.route('/prediction',methods=['POST'])

      <function flask.scaffold.Scaffold.route.<locals>.decorator(f: ~T_route) -> ~T_route>
```

```
[223] 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"):
              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
```

```
194] destination=request.form['destination']
    if(destination=="msp"):
        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,destination2,destination3,destination4,destination5],
        [name2,month2,dayofmonth2,dayofweek2,origin1_2,origin2_2,origin3_2,origin4_2,origin5_2,destination1_2,destination2_2,destination3_2,destination4_2,destination5_2],
    ]
    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)
```

fflightdata.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect

```
destination = request.form['destination']
if(destination == "msp"):
    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,destination2,destination3,destination4,destination5]]
y_pred= model.predict(total)
print(y_pred)s
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)
```

fflightdata.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment

Code + Text

[ ]

```python
app = Flask(__name__)
if __name__ == '__main__':
    app.run(debug=True)
```

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug: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
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```

ADVANTAGE

- **Improved customer satisfaction:** By predicting flight delays, airlines can proactively inform passengers and provide them with alternative options, reducing frustration and improving overall customer satisfaction.

- **Efficient resource management:** Airlines can optimize their resources, such as ground staff and aircraft, based on the predicted delays and avoid overbooking, unnecessary waiting times, or other issues that can arise due to unexpected delays.

- **Increased safety:** Flight delays can cause a ripple effect across the airline's network, potentially leading to cascading delays, overworked staff, and other safety concerns. By predicting delays, airlines can take proactive measures to ensure the safety of their passengers and crew.

APPLICATION

- **Proactive communication:** Airlines can use the predictions to proactively communicate with passengers regarding delays and offer alternative options, such as rebooking or providing compensation, improving customer satisfaction.
- **Resource optimization:** Airlines can use the predictions to optimize their resources, such as reducing ground staff, aircraft, and flight crew, based on the predicted delays, reducing costs and increasing efficiency.
- **Safety measures:** Airlines can use the predictions to take proactive measures to ensure passenger and crew safety, such as adjusting schedules or changing routes based on weather conditions.
- **Operational planning:** The predictions can help airlines in planning their operations better by providing insights into the potential risks and opportunities related to the delays, enabling them to take appropriate measures in advance.
- **Performance analysis:** The predictions can help airlines to monitor and analyze their performance and identify areas for improvement, such as identifying recurring delays and addressing underlying issues.

CONCLUSION

- The flight delay prediction project aims to build a machine learning model that can accurately predict the likelihood of flight delays based on historical flight data.
- The project involves various steps such as data preprocessing, feature engineering, model selection, and evaluation.
- By predicting the likelihood of flight delays, the model can be used by airlines and airports to better plan and manage their operations.
- This can help airlines adjust their schedules in advance, minimize the impact of delays, and improve the travel experience for passengers.
- The project has used various machine learning algorithms such as decision trees, random forests, and neural networks, along with feature engineering and data preprocessing techniques.
- The performance of the model has been evaluated using various metrics, and the best performing model can be deployed for real-time prediction of flight delays.
- Overall, the project has the potential to make a significant impact on the aviation industry, improving airline operations, reducing passenger frustration, and enhancing the overall travel experience.

FUTURE ENHANCEMENT

- There are several possible future enhancements that can be considered for the Flight Delay Prediction project, including:
- Using ensemble learning:
- Ensemble learning is a technique where multiple models are combined to produce a more accurate prediction.
- Implementing ensemble learning techniques such as stacking or bagging can help improve the overall accuracy of the model.
- While the project already includes several factors that can affect flight delays, other factors such as the airline's safety record, the aircraft's maintenance history, and flight crew availability can also be considered to improve the accuracy of the model.
- Feature engineering plays a crucial role in building accurate machine learning modelsOverall, the Flight Delay Prediction project offers several opportunities for future enhancements that can improve the accuracy and usability of the model in real-world scenarios.