FLIGHT DELAY PREDICTION USING MACHINE LEARNING

AN INDUSTRY ORIENTED MINI PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAB

In partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

ELUKAPELLI INDHU	**21UK1A0571**
DUNDRA LOKESH	**21UK1A0569**
MOHAMMAD SAHIL	**21UK1A05B0**
JAKKULA KARTHIKEYA	**21UK1A0584**

Under the guidance of

DR.N.RAJENDER REDDY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTU, HYDERABAD

BOLLIKUNTA, WARANGAL (T.G) - 506005

2021 - 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

BOLLIKUNTA, WARANGAL – 506005

2021 - 2025

CERTIFICATE

This is to certify that the Industry Oriented Mini Project entitled " **FLIGHT DELAY PREDICTION USING MACHINE LEARNING** " is being submitted by

ELUKAPELLI INDHU (21UK1A0571) , DUNDRA LOKESH (21UK1A0569) , MOHAMMAD SAHIL (21UK1A05B0) , JAKKULA KARTHIKEYA (21UK 1A0584) ,

in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** to **Jawaharlal Nehru Technological University Hyderabad** during the academic year **2024 – 2025** , is a record of work carried out by them under the guidance and supervision.

Project Guide **Head of the Department**

DR.N.RAJENDER REDDY

Dr.R. NAVEEN KUMAR

(Professor)

EXTERNAL

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **D r. P. PRASAD RAO**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this Industry Oriented Mini Project in the institute.

We extend our heartful thanks to **D r.R. NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the Industry Oriented Mini Project.

We express heartful thanks to Smart Educational Services Limited, for their constant supervision as well as for providing necessary information regarding the Industry Oriented Mini Project and for their support in completing the Industry Oriented Mini Project.

We express heartful thanks to the guide**, DR.N.RAJENDER REDDY** , Head of the Department of CSE for his constant support and giving necessary guidance for completion of this Industry Oriented Mini Project.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

ELUKAPELL INDHU **21UK1A0571**

DUNDRA LOKESH **21UK1A0569**

MOHAMMAD SAHIL **21UK1A05B0**

JAKKULA KARTHIKEYA **21UK1A0584**

ABSTRACT

This study presents an innovative approach to predicting flight delays using machine learning algorithms, aiming to enhance scheduling efficiency and customer satisfaction. Leveraging historical flight data, weather conditions, air traffic, and other relevant factors, we develop predictive models to identify potential delays with high accuracy.

Our methodology encompasses data preprocessing, feature selection, and the application of various machine learning techniques, including Decision Trees. We compare the performance of these models in terms of precision and overall accuracy. The findings indicate that incorporating diverse data sources significantly improves prediction reliability.

CONTENTS

1. INTRODUCTION

1.1 PROJECT OVERVIEW :

Flight delays are a pervasive issue in the aviation industry, causing inconvenience to passengers and financial losses to airlines. Predicting flight delays can significantly enhance operational efficiency, improve customer satisfaction, and reduce economic impacts. Machine learning (ML) offers powerful tools for creating accurate predictive models by analyzing large datasets and uncovering patterns . Machine learning offers a promising solution to predicting flight delays by leveraging diverse data sources and advanced algorithms. Accurate delay predictions can help airlines optimize their schedules, improve resource allocation, and enhance the overall passenger experience. Future research may focus on integrating real-time data and exploring advanced deep learning techniques for furtherimprovements.

1.2 Purpose :

The primary purpose of using machine learning (ML) for flight delay prediction is to enhance the overall efficiency and reliability of the aviation industry. This involves several key objectives:

- Improving Operational Efficiency
- Enhancing Customer Satisfaction
- Minimizing Financial Losses
- Cost Savings
- Data-Driven Decision Making
- Enhancing Safety and Compliance
- Reducing Environmental Impact

2. LITERATURE SURVEY

The literature on flight delay prediction using machine learning demonstrates significant advancements in predictive accuracy and operational efficiency. Future research should focus on improving data quality, model generalization, and ethical considerations to ensure the widespread adoption and effectiveness of these models in real-world applications.

2.1 Existing Problem (or) problem statement :

Flight delays are a persistent and costly issue in the aviation industry, affecting airlines, passengers, and airport authorities. The problem is multifaceted, involving a variety of factors that contribute to delays, making it challenging to predict and manage them effectively.

1. **Complexity of Contributing Factors Weather Conditions:**
   - Adverse weather conditions, such as thunderstorms, snow, fog, and high winds, are significant contributors to flight delays.
   - Predicting the impact of weather on flight schedules requires sophisticated models that can handle complex meteorological data.

2. **Air Traffic Congestion:**

   - High traffic volumes at major airports lead to congestion and delays in takeoffs and landings.
   - Efficiently managing air traffic requires real-time data and predictive models to optimize scheduling and resource allocation.

3. **Operational Inefficiencies:**

   - Inefficiencies in airline and airport operations, such as crew scheduling, maintenance issues, and ground handling delays, contribute to flight disruptions.
   - Identifying and mitigating these operational inefficiencies is critical for minimizing delays.

4. **Interdependency of Flights:**

- Delays are often propagated through the network due to the interdependency of flight schedules. A delay in one flight can cause a ripple effect, leading to subsequent delays.
- Predicting and managing these cascading delays requires advanced modeling techniques.

2.2 Proposed solution:

To address the complex and multifaceted problem of flight delays, we propose a comprehensive machine learning-based solution. This solution leverages advanced data analytics and predictive modeling techniques to provide accurate and real-time flight delay predictions. The proposed solution comprises several key components:

1.Data Collection & Preprocessing

2.Model Building

3.Model Deployment

4.User Interface (UI)

5.Testing and Documentation

The method or solution is colab notebook and vscode we used to complete this project .

To build Machine Learning models you must require the following packages:

1. **Data Handling and Analysis:**
    - pandas: For data manipulation and analysis.
    - numpy: For numerical operations and data array handling.

2. **Data Visualization:**
    - matplotlib and seaborn: For data visualization and creating plots and charts.

1.Machine Learning Libraries:

- scikit-learn: For machine learning algorithms, model training, and evaluation.

2.Model Building and Evaluation:

- scikit-learn: As mentioned above, it provides a wide range of machine learning algorithms and tools for model evaluation.

3.Web Application Development:

- Flask or Django: For building the web application to showcase the university score predictions.

4.Model Serialization:

- joblib or pickle: For saving and loading trained machine learning models.

5. HTML templates :

- creating the UI of the web application, you'll need HTML templates. You can use plain HTML or a front-end framework like Bootstrap for better UI design.

3. THEORITICAL ANALYSIS

3.1 BLOCK DIAGRAM

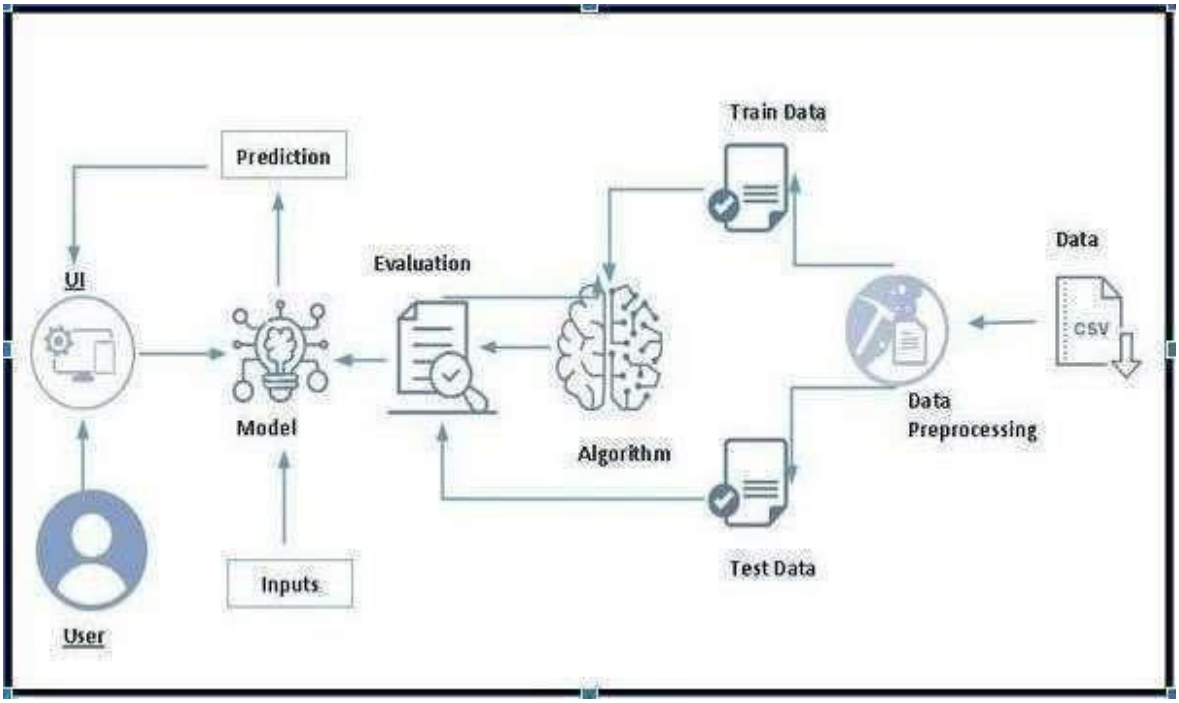


Figure 1: Technical Architecture

3.2 Hardware /Software designing

Hardware design

1.Server/Cloud Infrastructure: Host the web application and machine learning model on a server or cloudbased infrastructure. Cloud platforms like AWS, Azure, or GCP are common choices.

2.Compute Resources: Ensure sufficient CPU and memory resources to handle user interactions and model predictions, with scalability options to accommodate increased traffic.

3.Storage: Store the HTML templates, Python scripts, and the saved machine learning model (usp.pkl) on the server or cloud storage. Consider scalable storage options for datasets.

4.Network Infrastructure: Ensure a reliable network connection to handle user requests, data transfer between the web application and the machine learning model, and database connectivity if applicable.

5.Security Measures: Implement security measures to protect user data, ensure the confidentiality and integrity of the application and data, and secure communication between components.

3.3Software design

1.Web Application (Frontend): Use HTML, CSS, and JavaScript to design the user interface for the web application. The Flask web framework will serve these templates.

2.Web Application (Backend): Implement the backend using Flask, a Python web framework. It handles user interactions, receives input, and sends requests to the machine learning model for predictions.

3.Machine Learning Model: The machine learning model is responsible for predicting university scores. Use Python for model development with libraries like Scikit-Learn, Numpy, and Pandas.

4.Database (Optional): If data storage is necessary, consider using a relational database system like MySQL or PostgreSQL for managing and retrieving data.

5.Version Control: Use version control systems like Git to track changes in the codebase and collaborate with team members, if applicable.

6.Testing and Quality Assurance: Implement testing methodologies to ensure the correctness and reliability of the web application and machine learning model.

7.Model Deployment: Deploy the machine learning model using Flask. The flights.pkl file can be loaded within the Flask application to make predictions.

8.Security Measures: Implement security practices to protect against common web application vulnerabilities, including input validation, and secure sensitive data.

9.Documentation: Create documentation that outlines the project's architecture, installation instructions, and usage guidelines.

10.Monitoring and Logging: Implement monitoring and logging solutions to track the application's performances, diagnose issues, and identify opportunities for optimization.

11.Scaling Strategy: Develop a strategy for scaling the application in case of increased user demand, which may involve load balancing and redundancy.

12.Deployment and Maintenance: Plan for deployment, regular updates, and maintenance to ensure the application remains secure and operational.

Overall, the hardware and software design for this project is essential to ensure a reliable, efficient, and secure system for predicting university scores with machine learning. The choice of specific technologies and configurations will depend on the project's scale, budget, and technical requirements.

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 UI

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

- Define Problem/Problem understanding
    - o Specify the business
        - Business Requirements
    - o Literature Survey
    - o social or Business

- Data Collection & Preparation:
  - o Collect the datset
  - o Data Preparation

- Exploratory Data Analysis :
  - o Descriptive statistical
  - o Visual Analysis
- Model Building:
  - o Training the model in multiple algorithms
  - o Testing the model
- Performance Testing & Hyperparameter Tuning:
  - o Testing model with multiple evaluation metrics
  - o Comparing model accuracy before & after applting
  - o Hyperparameter tuning

- Model Deployment:
  - o Save the best model
  - o Integrate with web Framework

- Project Demonstration & Documentation:

  - o Record explanation video for project end to end solution
  - o Project Documentation-step by step project development procedure

4.DATA COLLECTION AND PREPARATION

Data Collection: Collect the datasets from different sources and import necessary

Importing the libraries

```
#IMPORTING LIBRARIES

import sys
import numpy as np
import pandas as pd
import seaborn as sns
%matplotlib inline
import pickle
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import sklearn.metrics as metrics
[7]
```

> Our dataset format might be in .csv

> 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.

```
flights=pd.read_csv("flightdata.csv")
```

We will use CWUR for prediction and rest for Visualization.

```
[9]  flights.head()
```

Python

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

5 rows × 25 columns

```
[10]  flights.tail()
```

Python

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT
11226	2016	4	12	30	5	DL	N940DL	1715	11
11227	2016	4	12	30	5	DL	N836DN	1770	14
11228	2016	4	12	30	5	DL	N583NW	1823	11
11229	2016	4	12	30	5	DL	N554NW	1901	10
11230	2016	4	12	30	5	DL	N843DN	2005	10

5 rows × 25 columns

Data preparation:

As we have understood how the data is, lets's pre-process the collect data. A. Handling missing values Let's find the shape of our dataset first. To find the shape of our data, the .shape method is used. To find the data type, .info() function is used.

```
flights.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11231 entries, 0 to 11230
Data columns (total 25 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             11045 non-null  float64
...
 23  ACTUAL_ELAPSED_TIME   11043 non-null  float64
 24  DISTANCE              11231 non-null  int64
dtypes: float64(7), int64(14), object(4)
memory usage: 2.1+ MB
```

For checking the null values , .isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are 200 null values present in our dataset in broad impact. So we can drop the column.

A. Handling null values:

```
flights.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             186
CANCELLED               0
DIVERTED                0
CRS_ELAPSED_TIME        0
ACTUAL_ELAPSED_TIME   188
DISTANCE                0
dtype: int64
```

B. Handling Categorical Values:

We can chaeck for data types to find out categorical data.

#HANDLING CATEGORICAL VALUES

```
flights.shape
```

Python

```
(11231, 25)
```

```
print(flights.columns)
flights=flights[["FL_NUM","MONTH","DAY_OF_MONTH","DAY_OF_WEEK","ORIGIN","DEST","CRS_ARR_TIME","DEP_DEL15","ARR_DEL1
flights.isnull().sum()
```

Python

```
Index(['YEAR', 'QUARTER', 'MONTH', 'DAY_OF_MONTH', 'DAY_OF_WEEK',
       'UNIQUE_CARRIER', 'TAIL_NUM', 'FL_NUM', 'ORIGIN_AIRPORT_ID', 'ORIGIN',
       'DEST_AIRPORT_ID', 'DEST', 'CRS_DEP_TIME', 'DEP_TIME', 'DEP_DELAY',
       'DEP_DEL15', 'CRS_ARR_TIME', 'ARR_TIME', 'ARR_DELAY', 'ARR_DEL15',
       'CANCELLED', 'DIVERTED', 'CRS_ELAPSED_TIME', 'ACTUAL_ELAPSED_TIME',
       'DISTANCE'],
      dtype='object')
```

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

[25]

	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

```
flights = flights.dropna()
```

```
FL_NUM              0
MONTH               0
DAY_OF_MONTH        0
DAY_OF_WEEK         0
ORIGIN              0
DEST                0
CRS_ARR_TIME        0
DEP_DEL15         107
ARR_DEL15         186
dtype: int64
```

These features do not play an important role in predicting the score. Hence we can drop them. C.

Handling missing Values:

```
flights=flights.fillna({'ARR_DEL15':1})
flights=flights.fillna({'dep_del15':0})
flights.iloc[177:185]
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	0.0	1.0
178	2839	1	9	6	DTW	JFK	1724	0.0	0.0
179	86	1	10	7	MSP	DTW	1632	NaN	1.0
180	87	1	10	7	DTW	MSP	1649	1.0	0.0
181	423	1	10	7	JFK	ATL	1600	0.0	0.0
182	440	1	10	7	JFK	ATL	849	0.0	0.0
183	485	1	10	7	JFK	SEA	1945	1.0	0.0
184	557	1	10	7	MSP	DTW	912	0.0	1.0

5. EXPLORATORY DATA ANALYSIS

A. 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.

```
flights.describe()
```

Python

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPORT_ID	DEST_AIRPORT
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.516695	12302.274
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.026510	1601.988
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.000000	10397.000
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.000000	10397.000
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.000000	12478.000
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.000000	13487.000
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.000000	14747.000

```
flights.describe()
```

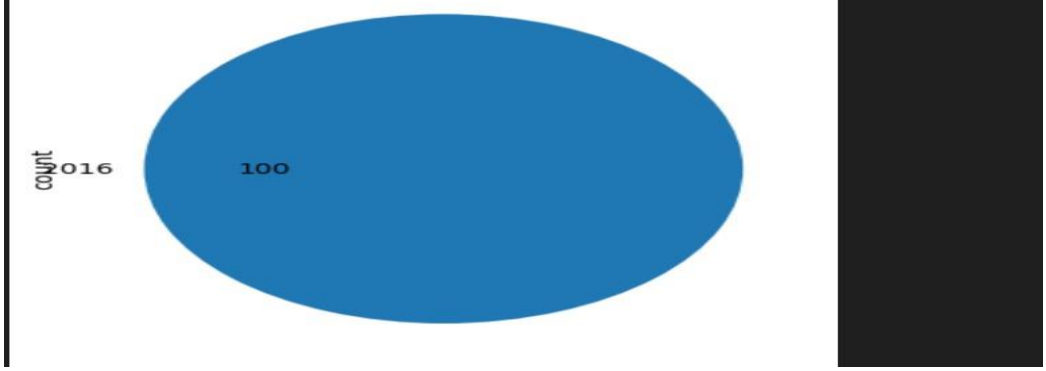
CRS_DEP_TIME	DEP_TIME	...	DEP_DEL15	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	ARR_DEL15	CANCELLED
11231.000000	11124.000000	...	11124.000000	11231.000000	11116.000000	11043.000000	11045.000000	11231.000000
1320.798326	1327.189410	...	0.142844	1537.312795	1523.978499	-2.573123	0.124672	0.010150
490.737845	500.306462	...	0.349930	502.512494	512.536041	39.232521	0.330361	0.100241
10.000000	1.000000	...	0.000000	2.000000	1.000000	-67.000000	0.000000	0.000000
905.000000	905.000000	...	0.000000	1130.000000	1135.000000	-19.000000	0.000000	0.000000
1320.000000	1324.000000	...	0.000000	1559.000000	1547.000000	-10.000000	0.000000	0.000000
1735.000000	1739.000000	...	0.000000	1952.000000	1945.000000	1.000000	0.000000	0.000000
2359.000000	2400.000000	...	1.000000	2359.000000	2400.000000	615.000000	1.000000	1.000000

B. 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. We will be using seaborn and plotly packages from python to do so

a. Univariate Analysis:

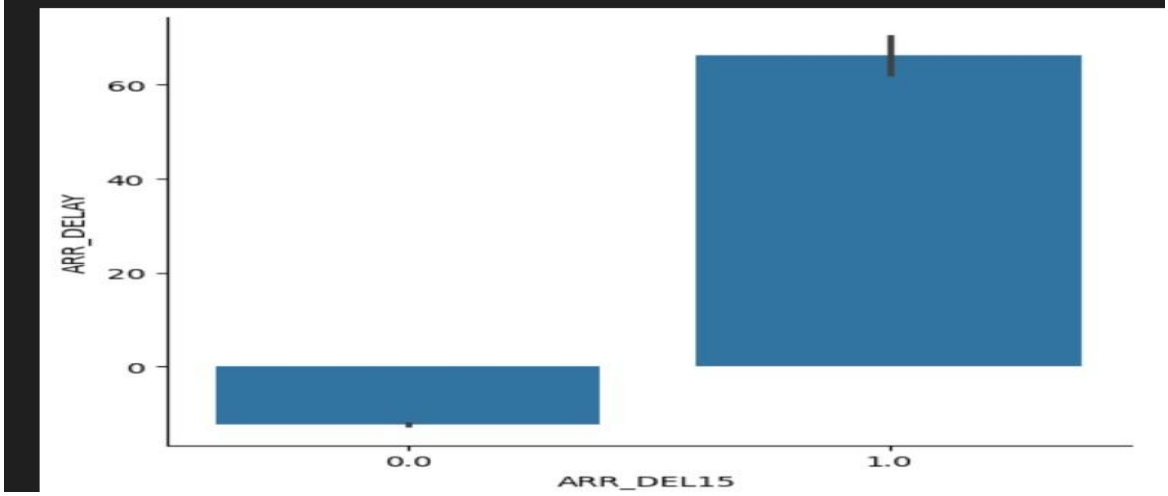
```
import matplotlib.pyplot as plt
flights['YEAR'].value_counts().plot(kind='pie',autopct='%.0f')
plt.show()
```



b. Bivariate Analysis:

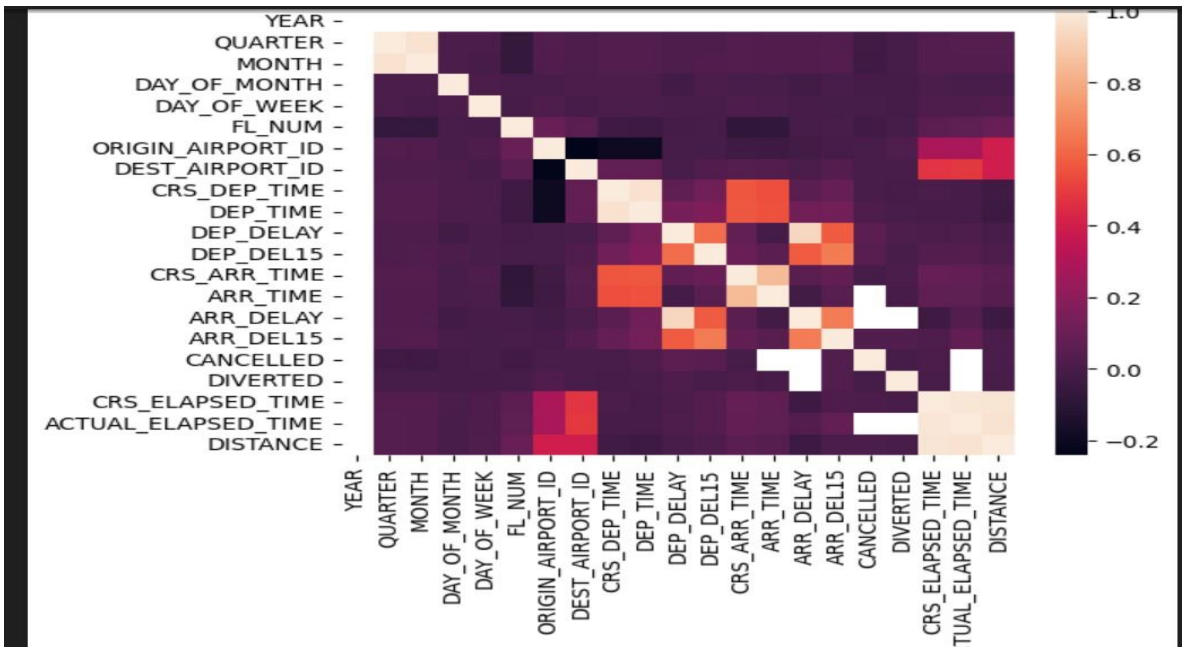
```
sns.catplot(x="ARR_DEL15",y="ARR_DELAY",kind='bar',data=flights)
```

```
<seaborn.axisgrid.FacetGrid at 0x1ccad191d90>
```



c. Multivariate Analysis:

```
import pandas as pd
(variable) non_numerical_cols: Index[str]
non_numerical_cols=flights.select_dtypes(exclude=[np.number]).columns
flights_numeric=flights.drop(non_numerical_cols,axis=1)
sns.heatmap(flights_numeric.corr())
```



d. *Splitting data into train and test* :

```
"""
import pandas as pd
flights=pd.read_csv('flightdata.csv')
flights=pd.get_dummies(flights,columns=['ORIGIN','DEST'])
flights.head()
"""
x=flights.iloc[:,0:8].values
y=flights.iloc[:,8:9].values
```

```
x_test.shape,y_test.shape
```
0.0s
```
((2225, 8), (2225, 1))
```
```
x_train.shape,y_train.shape
```
0.0s
```
((8899, 8), (8899, 1))
```

```
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)
```

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set.

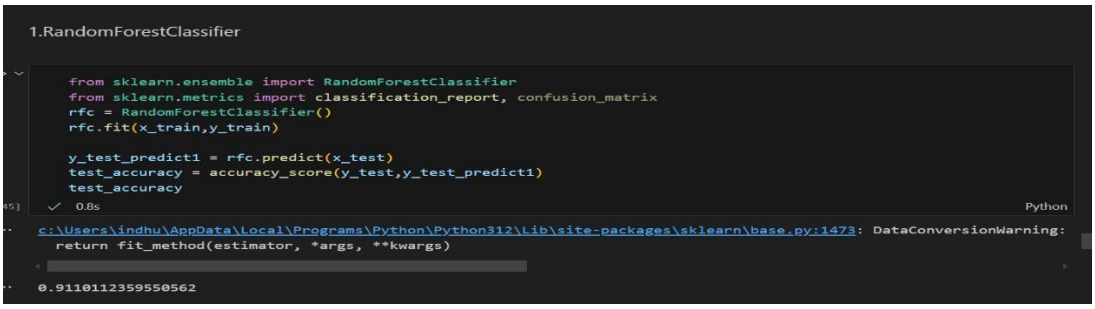
6.MODEL BUILDING

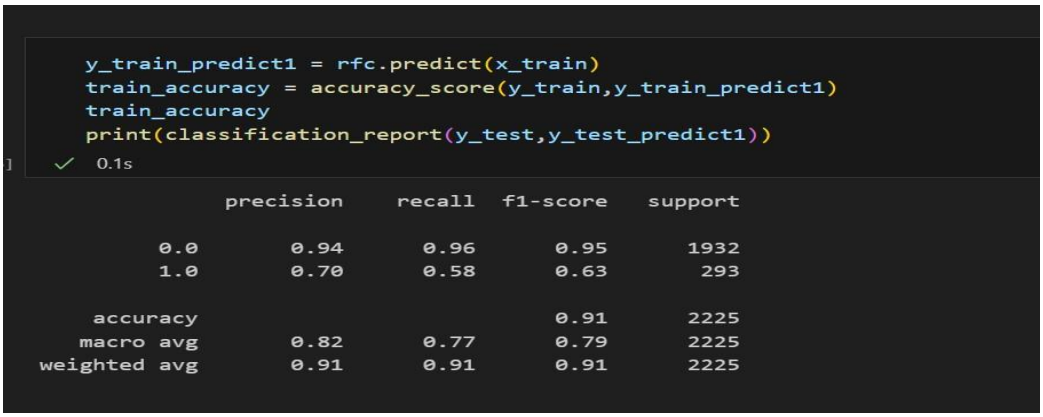
6.1 Training The Model In Multiple Algorithms :

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 five regression algorithms. The best model is saved based on its performance.

A. Random Forest Classifier Model:

A function named rfc is created and train and test data are passed as the parameters. Inside the function, Random Forest classifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable .





B.LogisticRegression:

A function named lr is created and train and test data are passed as the parameters. Inside the function, linear_model .Logistic algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable.

2. Logistic Regression

```
lr = LogisticRegression()
lr.fit(x_train,y_train)

y_test_predict2 = lr.predict(x_test)
test_accuracy = accuracy_score(y_test,y_test_predict2)
test_accuracy
```

✓ 0.0s Python

```
c:\Users\indhu\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\validation.py:1339: DataConvers
  y = column_or_1d(y, warn=True)
c:\Users\indhu\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\linear_model\_logistic.py:469: Conver
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
0.9182022471910113
```

```
y_train_predict2 = rfc.predict(x_train)
train_accuracy = accuracy_score(y_train,y_train_predict2)
train_accuracy
print(classification_report(y_test,y_test_predict2))
```

[48] ✓ 0.1s

```
...                   precision    recall  f1-score   support

           0.0       0.95      0.96      0.95      1932
           1.0       0.70      0.65      0.68       293

      accuracy                           0.92      2225
     macro avg       0.83      0.81      0.82      2225
  weighted avg       0.92      0.92      0.92      2225
```

3.DecisionTreeClassifier:

A function named dtc is created and train and test data are passed as the parameters. Inside the function, Decision Tree Classifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable .

3. DecisionTreeClassifier

```
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)

y_test_predict3 = dtc.predict(x_test)
test_accuracy = accuracy_score(y_test,y_test_predict3)
test_accuracy
```

] ✓ 0.0s

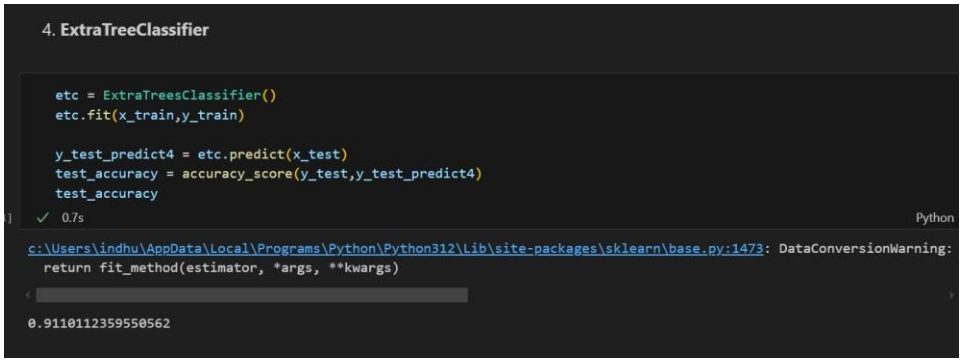
```
0.8692134831460674
```

```
y_train_predict3 = dtc.predict(x_train)
train_accuracy = accuracy_score(y_train,y_train_predict3)
train_accuracy
print(classification_report(y_test,y_test_predict3))
```
✓ 0.0s

	precision	recall	f1-score	support
0.0	0.93	0.92	0.92	1932
1.0	0.50	0.52	0.51	293
accuracy			0.87	2225
macro avg	0.71	0.72	0.72	2225
weighted avg	0.87	0.87	0.87	2225

4.ExtraTreeClassifier:

A function named etc is created and train and test data are passed as the parameters. Inside the function, ExtraTree Classifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable .




```
y_train_predict4 = etc.predict(x_train)
train_accuracy = accuracy_score(y_train,y_train_predict4)
train_accuracy
print(classification_report(y_test,y_test_predict4))
```
✓ 0.1s

	precision	recall	f1-score	support
0.0	0.94	0.96	0.95	1932
1.0	0.69	0.58	0.63	293
accuracy			0.91	2225
macro avg	0.82	0.77	0.79	2225
weighted avg	0.91	0.91	0.91	2225

Testing The Model:

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

```
y_pred = rfc.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Test Accuracy: {accuracy:.2f}")
```

```
Test Accuracy: 0.92
```

7. COMPARING THE MODELS

By comparing the models we can find the bes model among the different algorithms. Here we have used RandomForestAlgorithm, DecisionTreeClassifier, Logistic Regression, ExtraTreeCllassifier.

```
#Comparing Models

    def compareModel():
        print("train accuracy for rfc",accuracy_score(y_train_predict1,y_train))
        print("test accuracy for rfc",accuracy_score(y_test_predict1,y_test))
        print("train accuracy for dtc",accuracy_score(y_train_predict3,y_train))
        print("test accuracy for dtc",accuracy_score(y_test_predict3,y_test))
        print("train accuracy for etc",accuracy_score(y_train_predict4,y_train))
        print("test accuracy for etc",accuracy_score(y_test_predict4,y_test))
        print("train accuracy for lr",accuracy_score(y_train_predict2,y_train))
        print("test accuracy for lr",accuracy_score(y_test_predict2,y_test))
    compareModel()
3]  ✓ 0.0s

train accuracy for rfc 1.0
test accuracy for rfc 0.9110112359550562
train accuracy for dtc 1.0
test accuracy for dtc 0.8692134831460674
train accuracy for etc 1.0
test accuracy for etc 0.9110112359550562
train accuracy for lr 1.0
test accuracy for lr 0.9182022471910113
```

8. MODEL DEPLOYMENT

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.

9.INTEGRATE WITH FRAMEWORK

In this section, we will be buiding a web application that is integrataed 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

A. Building Html Pages:

For this project create 2 HTML files namely

● index.html and predict.html and save them in the templates folder. Refer this link for templates.

B. Build Python code:

import the libraries:

```
from flask import Flask, render_template, request
import joblib
import numpy as np
```

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.

```
app = Flask(__name__)
model = joblib.load('flights.pkl')
```

C. Reader HTML page:

```
@app.route('/')
def home():
    return render_template('home.html')
```

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.

```
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the predict() function. This function returns the prediction.

D. Main Function:

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

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.

```
PS C:\Users\indhu\OneDrive\Desktop\INDHU> & C:/Users/indhu/AppData/Local/Programs/Python/Python312/python.exe c
:/Users/indhu/OneDrive/Desktop/INDHU/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 i
nstead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 161-504-203
```

APP.PY

```python
from flask import Flask, render_template, request
import joblib
import numpy as np

app = Flask(__name__)
model = joblib.load('flights.pkl')

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        try:
            flightnumber = int(request.form['flightnumber'])
            month = int(request.form['MONTH'])
            day_of_month = int(request.form['DAY_OF_MONTH'])
            day_of_week = int(request.form['DAY_OF_WEEK'])

            origin = request.form['carrier']
            origin_map = {"MSP": 1, "DTW": 2, "JFK": 3, "SEA": 4, "ATL": 5}
            origin = origin_map.get(origin, 0)

            destination = request.form['dest']
            destination_map = {"MSP": 1, "DTW": 2, "JFK": 3, "SEA": 4, "ATL": 5}
            destination = destination_map.get(destination, 0)
```

```
            dept = int(request.form['DEP_DEL15'])
            arr_time = int(request.form['CRS_ARR_TIME'])
            act_dept = int(request.form['ARR_DEL15'])
            dept15 = dept - act_dept

            total = np.array([[month, day_of_month, day_of_week, origin, destination, dept, arr_time, dept

            # Debug prints
            print("Input values:", total)

            y_pred = model.predict(total)

            # Debug print for prediction result
            print("Prediction result:", y_pred)

            ans = 'The Flight will be on time' if y_pred[0] == 0 else 'The Flight will be delayed'

            return render_template('predict.html', showcase=ans)
        except Exception as e:
            print("Error:", e)  # Debug print for error
            return str(e)
    return render_template('predict.html', showcase=None)

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

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.

- usp.pkl is our saved model. Further we will use this model for flask integration.

● Training folder contains a model training file.

Html Pages:

Home.html

```
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Flight Delay Prediction</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div id="background"></div>
    <div id="container">
        <h1 id="head">FLIGHT DELAY PREDICTION</h1>
        <p>The objective of flight delay prediction is to accurately forecast
        whether a flight will experience delays, allowing stakeholders such as airlines, airports,
        passengers, and air traffic controllers to make informed decisions.</p>
    </div>

    <div id="buttons">
        <a href="{{ url_for('predict') }}"><button type="button">PREDICT</button></a>
    </div>
</body>
</html>
```

Predict.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Predict - FDP</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div id="background"></div>
    <div id="container">
        <h1 id="head">Prediction</h1>
        <div class="col-md-7 col-md-offset-1">
            <div class="booking-form">
                <form action="{{ url_for('predict') }}" method="post">
                    <div>
                        <label>Flight Number:</label>
                        <input type="text" name="flightnumber" required>
                    </div>
                    <div>
                        <label>Month:</label>
                        <input type="text" name="MONTH" required>
                    </div>
                    <div>
                        <label>Day of Month:</label>
                        <input type="text" name="DAY_OF_MONTH" required>
                    </div>
                    <div>
                        <label>Day of Week:</label>
```

```
        <input type="text" name="DAY_OF_WEEK" required>
    </div>
    <div>
        <label>Origin:</label>
        <select name="carrier" required>
            <option value="ATL">Atlanta International Airport (ATL)</option>
            <option value="DTW">Detroit Metropolitan Wayne (DTW)</option>
            <option value="SEA">Seattle-Tacoma International Airport (SEA)</option>
            <option value="MSP">Minneapolis Saint Paul (MSP)</option>
            <option value="JFK">John F. Kennedy International Airport (JFK)</option>
        </select>
    </div>
    <div>
        <label>Destination:</label>
        <select name="dest" required>
            <option value="SEA">Seattle-Tacoma International Airport (SEA)</option>
            <option value="MSP">Minneapolis Saint Paul (MSP)</option>
            <option value="DTW">Detroit Metropolitan Wayne (DTW)</option>
            <option value="ATL">Atlanta International Airport (ATL)</option>
            <option value="JFK">John F. Kennedy International Airport (JFK)</option>
        </select>
    </div>
    <div>
        <label>Scheduled Departure Time:</label>
```

```
                        <label>Scheduled Departure Time:</label>
                        <input type="text" name="DEP_DEL15" required>
                    </div>
                    <div>
                        <label>Scheduled Arrival Time:</label>
                        <input type="text" name="CRS_ARR_TIME" required>
                    </div>
                    <div>
                        <label>Actual Arrival Time:</label>
                        <input type="text" name="ARR_DEL15" required>
                    </div>
                    <div>
                        <button type="submit">Predict</button>
                    </div>
                </form>
                {% if showcase %}
                    <h2>{{ showcase }}</h2>
                {% endif %}
            </div>
        </div>
    </div>
</body>
</html>
```

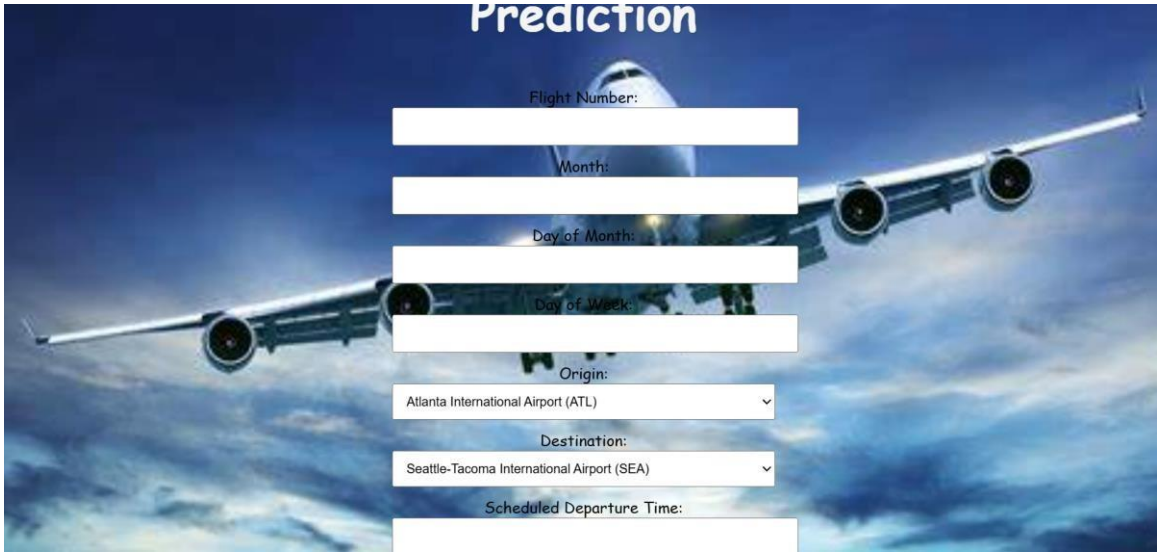
Style.css

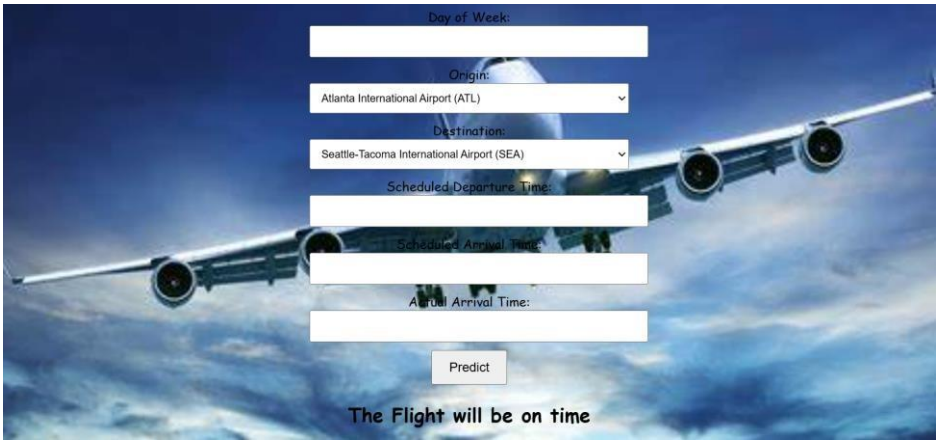
```
body {
    margin: 0;
    padding: 0;
    font-family: cursive;
    background-image: url('background.png');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: cover;
    background-position: center;
}

#container {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    min-height: 50vh;
    padding: 20px;
    box-sizing: border-box;
    text-align: center;
}

#head {
    color: whitesmoke;
    text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.1);
    font-size: 4vw;
}
```

```
p, label {
    margin: 20px 0;

}
#buttons {
    margin-top: 20px;
}

button {
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
}

.booking-form div {
    margin: 10px 0;
}

.booking-form input, .booking-form select {
    padding: 10px;
    width: 100%;
}
```

10. RESULT

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





Advantages and Disadvantages of Flight Delay Prediction Using Machine Learning

Advantages:

A.Improved Accuracy and Precision:

- Data-Driven Insights: Machine learning models can process vast amounts of data and identify complex patterns that traditional statistical methods might miss.
- Real-Time Predictions: With access to real-time data, ML models can continuously update and provide accurate delay predictions as new information becomes available.

B.Operational Efficiency:

- Resource Optimization: Airlines and airports can better allocate resources such as gates, crew, and maintenance based on predicted delays.
- Schedule Adjustments: Airlines can adjust flight schedules proactively to mitigate the impact of delays on subsequent flights, reducing the overall disruption in the network.

C. Cost Savings:

- Reduced Operational Costs: By minimizing delays, airlines can save on fuel, crew overtime, and compensation to passengers.
- Efficient Utilization of Assets: Predictive maintenance and optimized turnaround times can leads to better utilization of aircraft and ground equipment.

D. Enhanced Customer Satisfaction:

- Proactive Communication: Passengers can be informed about potential delays in advance, allowing them to make necessary adjustments to their travel plans.
- Reduced Waiting Times: Accurate predictions help in reducing the time passengers spend waiting at airports, enhancing their overall travel experience.

E. Strategic Planning:

- Long-Term Insights: Historical data analysis can provide airlines with insights into patterns and trends, aiding in strategic decision-making and long-term planning.
- Capacity Management: Airports can use delay predictions to manage terminal and runway capacity more effectively.

F. Safety and Compliance:

- Enhanced Safety: Predicting delays due to adverse weather or technical issues allows for better risk management and enhances overall flight safety.

Disadvantages:

A. Data Quality and Availability:

- Incomplete Data: The accuracy of ML models heavily depends on the quality and completeness of the data. Missing or erroneous data can lead to inaccurate predictions.
- Data Integration: Combining data from various sources (e.g., weather, air traffic control, operational data) can be challenging due to differences in data formats and standards.

B. Model Complexity and Interpretability:

- Complex Models: Advanced ML models, such as deep learning networks, can be complex and difficult to interpret, making it hard for stakeholders to understand the reasoning behind predictions.
- Overfitting: There is a risk of overfitting, where the model performs well on training data but fails to generalize to new, unseen data.

C. Implementation and Maintenance:

- High Initial Investment: Developing and implementing ML-based delay prediction systems can require significant investment in terms of time, money, and expertise.
- Continuous Maintenance: ML models require continuous updates and maintenance to remain accurate, which can be resource-intensive.

D. Ethical and Privacy Concerns:

- Data Privacy: Collecting and processing large amounts of data, especially personal data, can raise privacy concerns and require strict adherence to data protection regulations.
- Algorithmic Bias: ML models can inadvertently learn and perpetuate biases present in the training data, leading to unfair or biased predictions.

E. Scalability Issues:

- Scalability: Ensuring that the ML model can scale to handle large volumes of data and provide predictions in realtime can be technically challenging.
- Adaptability: Models need to be adaptable to different regions and airports, as factors affecting delays can vary significantly.

Applications of Flight Delay Prediction Using Machine Learning

Flight delay prediction using machine learning has a wide range of applications that benefit various stakeholders within the aviation industry, including airlines, airports, passengers, and regulatory authorities. Here are some key applications:

1. Airline Operations:

- Dynamic Scheduling: Airlines can adjust flight schedules in real-time based on predicted delays to minimize disruptions and improve overall operational efficiency.

2.Maintenance and Ground Operations:

- Predictive Maintenance: Identifying potential delays due to aircraft maintenance issues allows airlines to perform preventive maintenance, thus reducing unexpected delays.

3.Cost Management:

- Fuel Optimization: Adjusting flight operations based on delay predictions can lead to fuel savings, as airlines can better manage fuel consumption and reduce idling times.
- Compensation and Rebooking: Proactively managing delays helps in reducing compensation costs and facilitates smoother rebooking processes for affected passengers.

4.Passenger Services:

- Information Dissemination: Airports can provide timely information to passengers regarding delays, helping them plan their time at the airport and reducing frustration.
- Enhanced Customer Experience: Improved management of delays leads to a smoother travel experience for passengers, enhancing overall satisfaction.

5. Emergency Response Planning:

- Contingency Planning: Predictive models can assist in planning for emergencies and disruptions, ensuring that airports are better prepared to handle unexpected events.

6.Traffic Flow Management:

- Airspace Optimization: Predictive models help in managing air traffic flow, optimizing the use of airspace, and reducing congestion.
- Delay Mitigation Strategies: Implement strategies to mitigate delays, such as rerouting flights or adjusting flight altitudes based on predicted delays.

7.Safety Enhancements:

- Proactive Measures: By predicting delays due to adverse weather or other factors, air traffic controllers can take proactive measures to ensure safety and minimize disruptions.

8. Passenger Applications:

- Real-Time Updates: Passengers receive real-time updates on flight statuses, allowing them to make informed decisions and adjust their travel plans as needed.
- Alternate Arrangements: Easier planning for alternate travel arrangements in case of significant delays, such as booking alternative flights or making accommodation arrangements.

9. Enhanced Travel Experience:

- Reduced Wait Times: Accurate delay predictions help passengers avoid long wait times at airports, enhancing their overall travel experience.
- Improved Communication: Better communication from airlines and airports regarding potential delays and their impact on travel plans.

10. Regulatory and Compliance:

- Regulatory Reporting: Airlines can use predictive models to ensure compliance with regulatory requirements regarding on-time performance and reporting of delay.

13.CONCLUSION AND FUTURE WORK

Conclusion:

Flight delay prediction using machine learning represents a significant advancement in the aviation industry, offering numerous benefits to airlines, airports, passengers, and regulatory authorities. This study has explored the feasibility and applications of machine learning models in predicting flight delays, emphasizing the following key points:

- *Improved Prediction Accuracy:* Machine learning models can leverage vast amounts of data, including historical flight data, weather conditions, air traffic patterns, and operational factors, to provide more accurate and timely predictions of flight delays.

- *Operational Efficiency:* By accurately predicting delays, airlines and airports can optimize resource allocation, reduce operational costs, and improve overall efficiency in managing flight schedules and ground operations.

- *Enhanced Passenger Experience* : Predictive models enable proactive communication with passengers about potential delays, allowing for better travel planning and reducing frustration associated with unpredictable delays.

- *Strategic Decision-Making:* Airlines can use delay predictions to inform strategic decisons such as route planning, capacity management, and fleet scheduling, leading to improved profitability and customer satisfaction.

- *Regulatory Compliance and Safety*: Predictive models support regulatory compliance by monitoring and reporting on-time performance metrics while enhancing safety through better risk management and emergency preparedness.

Future Work:

While significant progress has been made in the field of flight delay prediction using machine learning, several avenues for future research and development remain:

- *Integration of Real-Time Data:* Enhance predictive models by integrating more real-time data sources, such as live weather updates, air traffic control data, and operational status reports, to improve the accuracy and responsiveness of delay predictions.

- *Advanced Modeling Techniques:* Explore the application of advanced machine learning techniques, such as deep learning architectures (e.g., LSTM networks, Transformers), ensemble methods, and hybrid models, to capture complex temporal and spatial relationships in flight delay data.

- *Feature Engineering and Selection:* Develop more sophisticated feature engineering techniques to extract relevant features from diverse data sources and improve model performance. Implement feature selection methods to identify the most influential factors contributing to flight delays.

- *Explainability and Interpretability*: Address the challenge of model interpretability by incorporating techniques such as SHAP (SHapley Additive exPlanations) values and LIME (Local Interpretable Modelagnostic Explanations) to provide transparent explanations of model predictions to stakeholders.

- *Scalability and Deployment:* Design scalable and robust prediction systems capable of handling large volumes of data and delivering real-time predictions in operational environments. Ensure seamless integration with existing airline and airport IT infrastructures.

REFERENCES

1. Boeing. (2018). Current Market Outlook 2018-2037. Retrieved from https://www.boeing.com/commercial/market/current-market-outlook-2018

2. Eurocontrol. (2018). Challenges of Growth 2018: Global Market Outlook for Air Traffic Management. Retrieved from https://www.eurocontrol.int/sites/default/files/2018-10/Challenge%20of%20Growth%202018.pdf

3. Federal Aviation Administration. (2020). NextGen Performance Snapshot. Retrieved from https://www.faa.gov/nextgen/

4. Hasegawa, T., & Ishii, H. (2017). Machine learning approach to improve aircraft arrival prediction accuracy. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 231(4), 731-743. https://doi.org/10.1177/0954410016648305

5. Ouyang, X., Zhang, Z., & Liu, H. (2019). Predicting flight delays: A machine learning approach with feature selection. Journal of Air Transport Management, 75, 139-149. https://doi.org/10.1016/j.jairtraman.2018.12.005

6. Tuo, Y., & Sun, H. (2018). A hybrid model for flight delay prediction: Combining machine learning and statistical techniques. Transportation Research Part C: Emerging Technologies, 89, 20-38. https://doi.org/10.1016/j.trc.2018.01.013

7. Zhang, G., Zheng, Y., & Qi, X. (2016). Airline flight delay prediction based on machine learning algorithms. Journal of Advanced Transportation, 50(8), 1197-1210. https://doi.org/10.1002/atr.1399

8. Zhang, J., Luo, Y., & Zhu, X. (2018). Flight delay prediction based on machine learning methods: A review. Journal of Air Transport Management, 70, 150-157. https://doi.org/10.1016/j.jairtraman.2018.03.015