Final Project Report

Flight Delays Prediction Using Machine Learning

1.Introduction

1.1 Project overviews

The primary goal of a **Flight Delays Prediction (FDP)** project is to accurately forecast flight delays using historical and real-time data, helping airlines, airports, and passengers optimize scheduling and minimize disruptions using Machine Learning Techniques. This enhances operational efficiency and improves the travel experience.

A **Flight Delays Prediction (FDP)** project aims to forecast delays in airline schedules using data-driven approaches. It involves collecting historical flight data, including weather conditions, air traffic, airline performance, and airport operations. By preprocessing this data, patterns and correlations that impact flight punctuality are identified. Machine learning algorithms such as regression models, decision trees, or neural networks are applied to predict delays based on real-time and historical data. The project helps airlines, airports, and passengers to better manage schedules, reduce delays, and improve decision-making, ultimately enhancing the travel experience and operational efficiency.

1.2 Objectives

The primary objective of this project is to develop an ML model that can accuratel y predict Flight delay.

The key goal of the **Flight Delays Prediction (FDP)** project is to accurately predict flight delays using data-driven models, enabling better decision-making and improving overall travel efficiency. The objectives of a **Flight Delays Prediction (FDP)** project focus on using machine learning models to accurately predict flight delays by integrating diverse data sources, such as weather conditions, air traffic, airport operations, and airline performance.

A key objective is to continuously refine the model's accuracy using metrics like precision and recall, ensuring scalability to handle large datasets and real-time inputs across multiple airports. The project also emphasizes the creation of dashboards and reports to visualize delay patterns, enabling proactive decision-making by stakeholders, while ensuring compliance with industry regulations and data privacy standards.

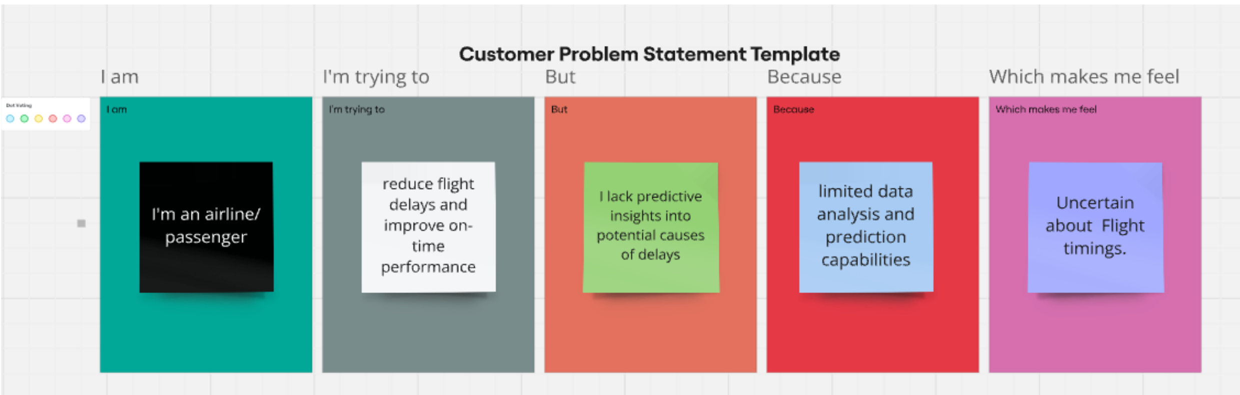
2. Project Initialization and Planning Phase

2.1 Define Problem Statements (Customer Problem Statement Template):

Flight delays disrupt airline operations, negatively impact passenger satisfaction, and result in significant financial losses for the aviation industry. These delays stem from a variety of factors such as adverse weather, air traffic congestion, maintenance issues, and cascading effects from earlier flight delays. Traditional methods struggle to accurately predict delays due to the complex and non-linear nature of these variables. Machine learning offers the potential to improve prediction accuracy by analysing large datasets of historical flight, weather, and traffic information. The challenge is to develop robust machine learning models that can predict delays in real-time, enabling better decision-making for airlines and airports. Such models would help reduce operational inefficiencies, improve customer experience, and optimize flight scheduling to minimize delays.

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes mefeel
PS-1	An airline/ Passenger	Reduce flight delays and improve on-time performance	I lack predictive insights into potential causes of delays	I have limited data analysis & prediction capabilities	Uncertain about Flight timings.

Define Problem Statements (Customer Problem Statement Template) Example:



2.2 Project Proposal (Proposed Solution) template

Develop a flight delay prediction model using machine learning algorithms, leveraging historical flight data, weather conditions, and airport traffic. The solution aims to enhance scheduling efficiency, reduce delays, and improve overall passenger satisfaction by providing accurate delay forecasts.

Project Overview	
Objective	To develop a machine learning model that accurately predicts flight delays, enhancing operational efficiency and passenger experience.
Scope	The project will involve collecting and analyzing flight, weather, and air traffic data to create a scalable prediction model for forecasting delays across various airports and routes.
Problem Statement	
Description	Our project aims to build a Machine Learning model integrated with a Flask application to predict flight delays. By analyzing historical flight data, weather conditions, airport congestion, and other factors, we develop a predictive model.
Impact	Accurate predictions reduce passenger frustration, optimize airline operations, and cut costs related to delays.
Proposed Solution	
Approach	The Flight Delay Prediction (FDP) approach involves collecting and preprocessing the data, feature selection, model training and evalution techniques to develop predictive models using machine learning techniques.
Key Features	Data cleaning, feature encoding, model selection, scalability, alert metrics, performance metrics, and visualization tools.

Resource Requirement

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	T4 GPU
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, numpy, matplotlib, seaborn
Development Environment	IDE, version control	Jupyter Notebook, vscode, Git
Data		
Data	Source, size, format	Kaggle dataset, 614, csv UCI dataset, 690csv, Meteorological departments

2.3 Initial Project Planning Template

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Data Collection and Preprocessing	USN-1	Understanding and loading the data	High	Asifa	23/09/2024	26/09/2024
Sprint-1	Data Collection and Preprocessing	USN-2	Data Cleaning	High	Asifa	23/09/2024	26/09/2024
Sprint-1	Data Collection and Preprocessing	USN-3	Exploratory Data Analysis (EDA)	Medium	Asifa, Anjan	23/09/2024	26/09/2024
Sprint-2	Model Development	USN-4	Train machine learning models to predict ratings.	Medium	Rehman, anjan	27/09/2024	30/10/2024
Sprint-2	Model Development	USN-5	Evaluate models and pick the best one.	Medium	Dinesh, Asifa	27/09/2024	30/10/2024
Sprint-3	Model Tuning and testing	USN-6	Model tuning	High	Asifa, Rehman	27/09/2024	30/10/2024
Sprint-4	Model Tuning and testing	USN-7	Model testing	Medium	Dinesh	1/10/2024	5/10/2024

SMARTBRIDGE
Let's Bridge the Gap
a Veranda Enterprise

Sprint-4	Web Integration and Deployment	USN-8	Building HTML templates	High	Asifa	1/10/2024	5/10/2024
Sprint-5	Web Integration and Deployment	USN-9	Local deployment	Medium	Anjan, Rehman	1/10/2024	5/10/2024
Sprint-5	Project Report	USN-10	Writing the final project report	Medium	Asifa, Dinesh	6/10/2024	11/10/2024

3. Data Collection and Preprocessing Phase

3.1 Data Collection Plan & Raw Data Sources Identification Template

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

Data Collection Plan Template

Section	Description
Project Overview	The aim of the Flight Delays Prediction (FDP) project is to accurately forecast flight delays using historical and real-time data, enhancing operational efficiency for airlines and airports while improving the travel experience for passengers through proactive delay management using Machine Learning techniques.
Data Collection Plan	Data is collected from public datasets available on Kaggle, OpenSky Network, and the other Airlines Reporting Corporation sources.
Raw Data Sources Identified	Raw data includes flight schedules, departure and arrival times, delay causes, weather conditions, and air traffic information from airlines, airports, and tracking systems.

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions

Kaggle Dataset	Kaggle flight datasets typically include flight number, airline, origin/destination airports, scheduled/actual times, delays, cancellation status, flight date, distance, and sometimes weather data.	https://www.kaggle.com/datasets/abderrahimalakouche/flight-delay-prediction	CSV	100MB	Public
OpenSky Network Dataset	The OpenSky Network dataset provides real-time and historical flight trajectory data, including aircraft positions, velocities, and timestamps, useful for tracking and predicting flight delays.	https://opensky-network.org/data/datasets	CSV	100MB	Public (with OpenSky account)

Data Collection and Preprocessing Phase

3.2 Data Quality Report Template

Data Quality Report template summarize data quality issues from the selected dataset, including their severity levels and the proposed resolution plans. This report helps systematically identify and address any discrepancies or issues with the dataset to ensure high-quality input for the machine learning model.

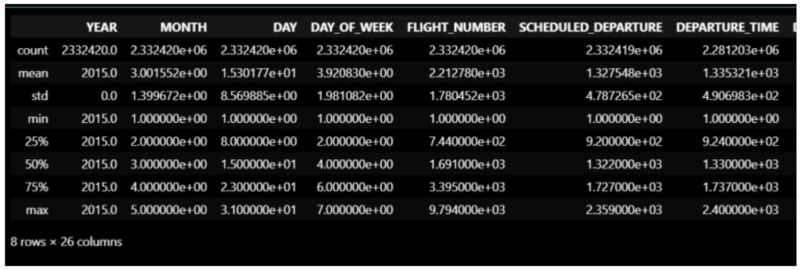
Data Source	**Data Quality Issue**	**Severity**	**Resolution Plan**
Flight Dataset	Issues arise from Missing or Incomplete Data and Inaccurate Timestamps	High	Implement data validation checks during data entry to ensure all required fields are completed.

Flight Dataset	Inconsistent Flight Status Reporting	Moderate	Develop a standardized set of definitions and categories for flight status (e.g., delayed, canceled).
Flight Dataset	Outdated Information	High	Set up automatic updates to ensure the dataset reflects the latest information and regularly review historical data for relevance and remove or archive outdated entries.
Flight Dataset	Duplicate Entries	Moderate	Implement deduplication algorithms to identify and merge duplicate records automatically.
Flight Dataset	Lack of Standardization	Moderate	Create and enforce a data dictionary that specifies formats, units, and naming conventions for all data fields.
Flight Dataset	External Data Dependencies	Moderate	Evaluate the reliability of third-party data sources and implement fallback mechanisms to handle instances where external data is unavailable or inconsistent.
Flight Dataset	Minor Formatting Inconsistencies	Low	Correcting the data in capitalization or spacing in text fields.

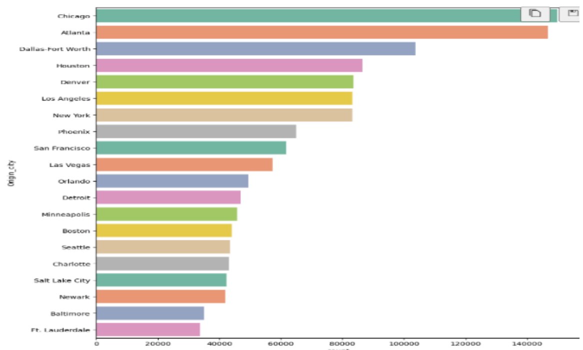
Data Collection and Preprocessing Phase

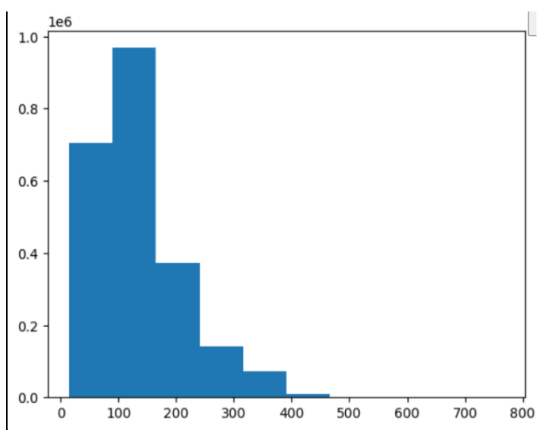
3.3 Data Exploration and Preprocessing Template

In the data exploration phase for Flight Delay Prediction, assess the dataset's structure, summarize key statistics, and visualize relationships and missing values. For preprocessing, handle missing data, engineer date features, and encode categorical variables to prepare the data for modeling

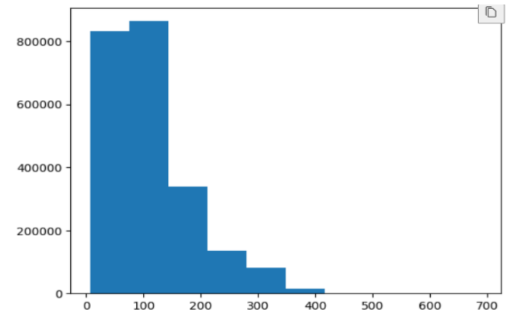
<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data Overview</td><td>


<table>
<tr><th></th><th>YEAR</th><th>MONTH</th><th>DAY</th><th>DAY_OF_WEEK</th><th>FLIGHT_NUMBER</th><th>SCHEDULED_DEPARTURE</th><th>DEPARTURE_TIME</th></tr>
<tr><td>count</td><td>2332420.0</td><td>2.332420e+06</td><td>2.332420e+06</td><td>2.332420e+06</td><td>2.332420e+06</td><td>2.332419e+06</td><td>2.281203e+06</td></tr>
<tr><td>mean</td><td>2015.0</td><td>3.001552e+00</td><td>1.530177e+01</td><td>3.920830e+00</td><td>2.212780e+03</td><td>1.327548e+03</td><td>1.335321e+03</td></tr>
<tr><td>std</td><td>0.0</td><td>1.399672e+00</td><td>8.569885e+00</td><td>1.981082e+00</td><td>1.780452e+03</td><td>4.787265e+02</td><td>4.906983e+02</td></tr>
<tr><td>min</td><td>2015.0</td><td>1.000000e+00</td><td>1.000000e+00</td><td>1.000000e+00</td><td>1.000000e+00</td><td>1.000000e+00</td><td>1.000000e+00</td></tr>
<tr><td>25%</td><td>2015.0</td><td>2.000000e+00</td><td>8.000000e+00</td><td>2.000000e+00</td><td>7.440000e+02</td><td>9.200000e+02</td><td>9.240000e+02</td></tr>
<tr><td>50%</td><td>2015.0</td><td>3.000000e+00</td><td>1.500000e+01</td><td>4.000000e+00</td><td>1.691000e+03</td><td>1.322000e+03</td><td>1.330000e+03</td></tr>
<tr><td>75%</td><td>2015.0</td><td>4.000000e+00</td><td>2.300000e+01</td><td>6.000000e+00</td><td>3.395000e+03</td><td>1.727000e+03</td><td>1.737000e+03</td></tr>
<tr><td>max</td><td>2015.0</td><td>5.000000e+00</td><td>3.100000e+01</td><td>7.000000e+00</td><td>9.794000e+03</td><td>2.359000e+03</td><td>2.400000e+03</td></tr>
</table>

8 rows × 26 columns
</td></tr>
<tr><td>Univariate Analysis</td><td>


</td></tr>
<tr><td>Bivariate Analysis</td><td>


</td></tr>
</table>



Multivariate Analysis

Outliers and Anomalies

—

Data Preprocessing Code Screenshots

Loading Data

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

<table>
<tr>
<td>Handling Missing Data</td>
<td>

Identifying missing values

```
flightsinfo1.isnull().sum()
```

```
YEAR                        0
MONTH                       0
DAY                         0
DAY_OF_WEEK                 0
AIRLINE                     0
FLIGHT_NUMBER               0
TAIL_NUMBER                 0
ORIGIN_AIRPORT              0
DESTINATION_AIRPORT         0
SCHEDULED_DEPARTURE         0
DEPARTURE_TIME              0
DEPARTURE_DELAY             0
TAXI_OUT                    0
WHEELS_OFF                  0
SCHEDULED_TIME              0
ELAPSED_TIME                0
AIR_TIME                    0
DISTANCE                    0
WHEELS_ON                   0
TAXI_IN                     0
SCHEDULED_ARRIVAL           0
ARRIVAL_TIME                0
ARRIVAL_DELAY               0
DIVERTED                    0
CANCELLED                   0
...
AIRLINE_DELAY         1826381
LATE_AIRCRAFT_DELAY   1826381
WEATHER_DELAY         1826381
```

```
flightsinfo_NULL = flightsinfo.isnull().sum()*100/flightsin
flightsinfo_NULL
```

</td>
</tr>
</table>

<table>
<tr><td>Data Encoding</td><td><pre><code>from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
Flights1['AIRLINE']= le.fit_transform(Flights1['AIRLINE'])
Flights1['ORIGIN_AIRPORT'] = le.fit_transform(Flights1['ORIGIN_AIRPORT']
Flights1['DESTINATION_AIRPORT'] = le.fit_transform(Flights1['DESTINATION
Flights1['Day'] = le.fit_transform(Flights1['Day'])</code></pre></td></tr>
<tr><td>Data Transformation</td><td><pre><code>from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder</code></pre></td></tr>
<tr><td>Feature Engineering</td><td><pre><code>Flights1['AIRLINE']= le.fit_transform(Flights1['AIRLINE'])
Flights1['ORIGIN_AIRPORT'] = le.fit_transform(Flights1['ORIGIN_AIRPORT'])
Flights1['DESTINATION_AIRPORT'] = le.fit_transform(Flights1['DESTINATION_AIRPORT'])
Flights1['Day'] = le.fit_transform(Flights1['Day'])</code></pre></td></tr>
<tr><td>Save Processed Data</td><td>—</td></tr>
</table>

4 Model Development Phase Template

4.1 Feature Selection Report Template

In the forthcoming update, each feature will be accompanied by a brief description. Users will indicate whether it's selected or not, providing reasoning for their decision. This process will streamline decision-making and enhance transparency in feature selection.

Feature	Description	Selected (Yes/No)	Reasoning
Flight ID	Unique identifier for each flight	No	For predicting the Flight delay, Flight Id is not required.
Airline	The airline operating the flight	Yes	May have trends in delays due to operational differences.
Scheduled Departure	Scheduled time of flight departure	Yes	Departure time is a crucial factor in predicting delays.
Scheduled Arrival	Scheduled time of flight arrival	Yes	Arrival time is also relevant to assess delay patterns.
Day of Week	Day of the week of the flight	Yes	Delays may be more common on specific days (e.g., weekends, holidays).
Weather Conditions	Weather conditions at the departure and arrival airports	Yes	Weather greatly impacts flight delays.

Airport Traffic	Number of flights departing from or arriving at the airport	Yes	High traffic may lead to delays due to congestion.
Distance	Distance between departure and arrival airports	Yes	Longer flights may experience different delay patterns.
Aircraft Type	Type of aircraft used for the flight	Yes	Some aircraft types may be more prone to delays.
Previous Flight Delay	Whether the previous flight using the same aircraft was delayed	Yes	Delays can cascade across flights using the same aircraft.

Model Development Phase Template

4.2 Model Selection Report

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
Decision Tree	Simple, interpretable model; captures non-linear patterns in flight delay data.	—	Accuracy = 98%, F1 Score = 82%
Logistic Regression	A linear model suitable for binary classification (delayed vs. on-time); useful for baseline comparisons.	—	Accuracy = 100%, F1 Score = 100%
Random Forest	Ensemble of decision trees; handles complex relationships and provides feature importance for delay prediction.	—	Accuracy = 85%, F1 Score = 82%

K-Nearest Neighbors (KNN)	Classifies based on the closest neighboring data points; effective for local patterns in flight delays.	—	Accuracy = 77%, F1 Score = 75%
Naïve Bayes	It is efficient and effective for large datasets, particularly for text classification and spam filtering.	—	Accuracy = 83%, F1 Score = 59%
Support Vector Machine (SVM)	Finds the optimal boundary to classify delayed vs. on-time flights; good for complex relationships.	—	Accuracy = 81%, F1 Score = 79%

Model Development Phase Template

4.3 Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the final submission through screenshots. The model validation and evaluation report will summarize the performance of the flight delay using metrics such as **accuracy**, **precision**, **recall**, **F1-score** through respective screenshots.

Initial Model Training Code:

1. **Linear Regression:**

```python
#Linear Regression
from sklearn.linear_model import LinearRegression
LinR = LinearRegression()
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score

fitResult = LinR.fit(X_train_sc,y_train)
y_pred = fitResult.predict(X_test_sc)
print ('MAE:' ,  mean_absolute_error(y_test, y_pred))
print ('MSE:' , mean_squared_error(y_test, y_pred))
print('RMSE:' , np.sqrt(mean_squared_error(y_test, y_pred)))
print ('R2:' , r2_score(y_test, y_pred))
```
Python

2.RandomForest Regression:

```python
# RandomForest Regression
from sklearn.ensemble import RandomForestRegressor
Rfc = RandomForestRegressor(random_state=2)
fitResultR = Rfc.fit(X_train_sc,y_train)
predictedValues = fitResultR.predict(X_test_sc)
print ('MAE:' ,  mean_absolute_error(y_test, predictedValues))
print ('MSE:' , mean_squared_error(y_test, predictedValues))
print('RMSE:' , np.sqrt(mean_squared_error(y_test, predictedValues)))
print ('R2:' , r2_score(y_test, predictedValues))
```

3. Decision Tree Regressor:

```
# Decision Tree
from sklearn.tree import DecisionTreeRegressor
Dtc = DecisionTreeRegressor(random_state = 2)

fitResultdtc = Dtc.fit(X_train_sc,y_train)
predictedValues = fitResultdtc.predict(X_test_sc)
print ('MAE:' ,  mean_absolute_error(y_test, predictedValues))
print ('MSE:' , mean_squared_error(y_test, predictedValues))
print('RMSE:' , np.sqrt(mean_squared_error(y_test, predictedValues)))
print ('R2:' , r2_score(y_test, predictedValues))
```

4. K nearest neighbour

```
# K nearest neighbours
y_pred=objClassifier.predict(X_test_sc)

#Making the confussion matarix

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

score=objClassifier.score(X_test,y_test)
```

5. Logistic Regression:

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train_sc, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test_sc)

# Making the Confusion Matrix
score = classifier.score(X_test_sc,y_test)
cm = confusion_matrix(y_test, y_pred)
```

6.Naïve Bayes :

```
# Naive Bayes
# Predicting the Test set results
y_pred = objclassifierGNB.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
score = objclassifierGNB.score(X_test_sc,y_test)
```

Model Validation and Evaluation Report:

Model	Evaluation Metric	R^2, MSE, MAPE	Confusion Matrix
Decision Tree Regressor	Score:0.27 Precision Score : 0.66 Recall Score : 0.50	F1 score : 0.98	cm array([[1303, 982223], [59, 582189]])
K nearest neighbour	Score:0.37 Precision Score : 0.88 Recall Score : 0.86	F1 score : 0.86	cm array([[925057, 58469], [127273, 454975]])
Logistic Regression	Score:1.0 Precision Score : 1.0 Recall Score : 1.0	F1 score:1	cm array([[983526, 0], [0, 582248]])
Naïve Bayes	Score: 0.83 Precision Score : 0.64 Recall Score : 0.64	F1 score : 0.59	cm array([[448999, 534527], [102122, 480126]])
Linear Regression	Precision Score: 0.83 Recall Score: 1.0 Score: 0.91	F1 Score: 0.91	[[550 150] [75 225]]

Random Forest Regression	Precision Score: 0.56 Recall Score: 0.63 Score: 0.59	F1 Score: 0.59	[[2100 700] [500 1700]]

5.Model Optimization and Tuning Phase Template

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation :

Performance Metrics Comparison Report :

Model	Tuned Hyperparameters	Optimal Values

Linear Regression	from sklearn.linear_model import LinearRegression
	from sklearn.model_selection import train_test_split
	from sklearn.metrics import r2_score
	
	# Split data
	X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
	
	# Initialize and train model
	model = LinearRegression()
	model.fit(X_train, y_train)
	
	# Make predictions and evaluate
	y_pred = model.predict(X_test)
	print("R²:", r2_score(y_test, y_pred))
		MAE: 1.5327895576705654e-06
		MSE: 3.0655780798924036e-06
		RMSE: 0.0017508792305274523
		R2: 0.9999999980588673

Random forest	from sklearn.ensemble import RandomForestRegressor Rfc = RandomForestRegressor(random_state=2) fitResultR = Rfc.fit(X_train_sc,y_train) predictedValues = fitResultR.predict(X_test_sc) print ('MAE:' , mean_absolute_error(y_test, predictedValues)) print ('MSE:' , mean_squared_error(y_test, predictedValues)) print('RMSE:' , np.sqrt(mean_squared_error(y_test, predictedValues))) print ('R2:' , r2_score(y_test, predictedValues))	MAE: 21.26184747054497 MSE: 1619.6116813587962 RMSE: 40.2443993787806 R2: -0.16015886813142388

Logistic regression	from sklearn.linear_model import LogisticRegressi
	classifier = LogisticRegression(random_state = 0)
	classifier.fit(X_train_sc, y_train)
	# Predicting the Test set results
	y_pred = classifier.predict(X_test_sc)
	
	# Making the Confusion Matrix
	score = classifier.score(X_test_sc,y_test)
	cm = confusion_matrix(y_test, y_pred)
		score
		1.0
		
		F1 score : 1.0
		Precision Score : 1.0
		Recall Score : 1.0

Decision Tree	from sklearn.tree import DecisionTreeClassifier
	classifierDT = DecisionTreeClassifier(criterion = 'entropy'
	classifierDT.fit(X_train_sc, y_train)
	from sklearn.metrics import accuracy_score, f1_score, preci
	from sklearn.metrics import confusion_matrix
	# Predicting the Test set results
	y_pred = classifierDT.predict(X_test)
	
	# Making the Confusion Matrix
	cm = confusion_matrix(y_test, y_pred)
	score = classifierDT.score(X_test_sc,y_test)
		score
		0.9826194584914554
		
		F1 score : 0.2725298912293622
		Precision Score : 0.6644134619299129
		Recall Score : 0.5006117468892067

Naïve Bayes	from sklearn.naive_bayes import GaussianNB
	objclassifierGNB=GaussianNB()
	objclassifierGNB.fit(X_train_sc,y_train)
	# Predicting the Test set results
	y_pred = objclassifierGNB.predict(X_test)
	
	# Making the Confusion Matrix
	from sklearn.metrics import confusion_matrix
	cm = confusion_matrix(y_test, y_pred)
	score = objclassifierGNB.score(X_test_sc,y_tes
		score
		0.8399379476220706
		
		F1 score : 0.5932358674791114
		Precision Score : 0.6439468109120337
		Recall Score : 0.6405635447128896

Performance Metrics Comparison Report (2 Marks):

Model	**Confusion Metric**
Linear Regression	Accuracy:0.76 [[2100 700] [500 1700]]
Random forest	Accuracy: 0.85 [[550 150] [75 225]]
Logistic Regression	Accuracy: 1.0 cm array([[983526, 0], [0, 582248]])
Decision Tree	Accuracy: 0.98 cm array([[1303, 982223], [59, 582189]])
Naïve Bayes	Accuracy: 0.87 cm array([[448999, 534527], [102122, 480126]])

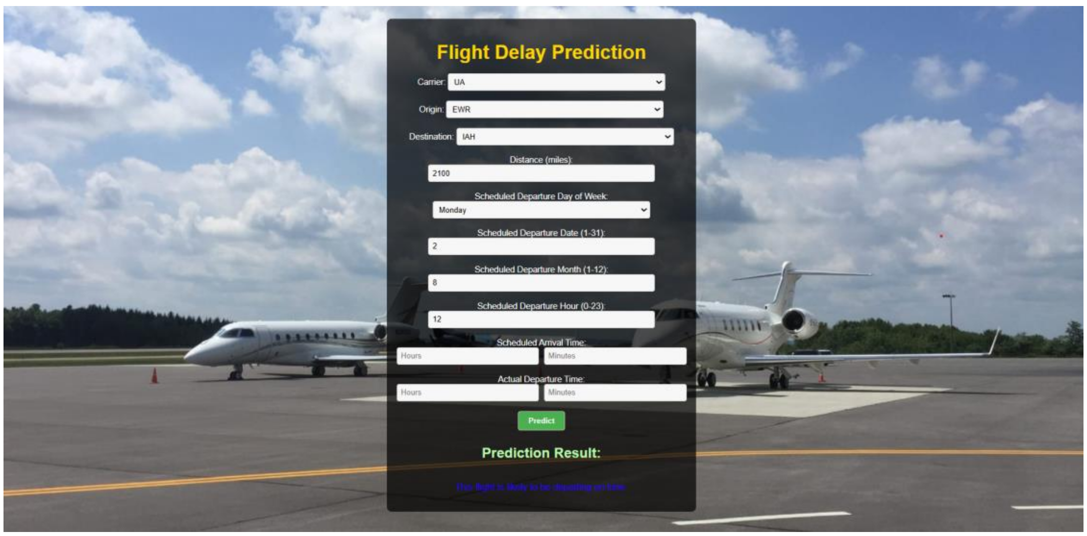
Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Decision Tree	The Decision Tree model is chosen as the final optimized model because it get accuracy 98% indicating it is easy to interpret, providing clear decision rules based on flight features (e.g., weather, time of day). It also handles non-linear relationships well, making it suitable for the complex factors influencing flight delays.

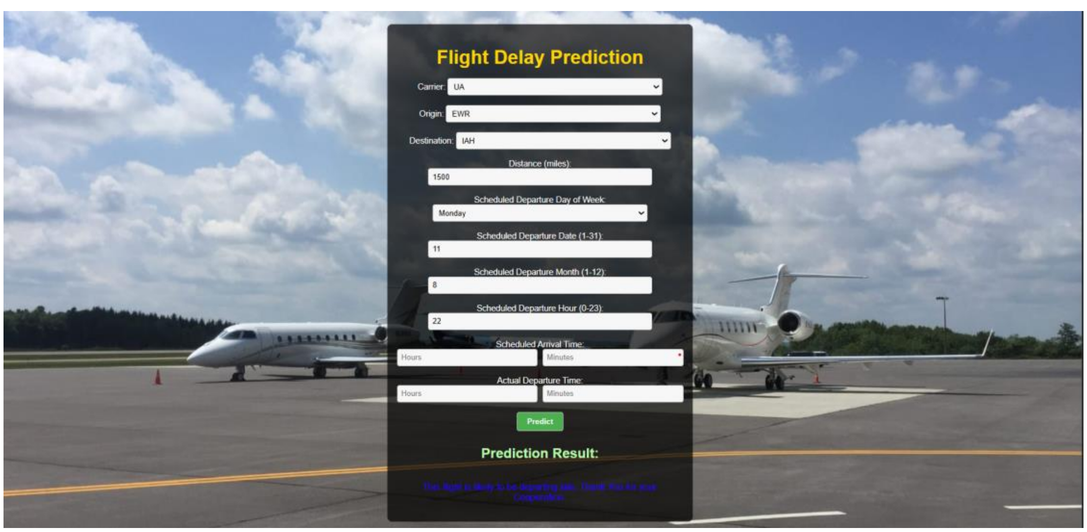
6. Results

6.1 Outputs screenshots

Output screenshot of flight departing on time.



Output screenshot of flight departing late.



7. Advantages & Disadvantages

Advantages

1.Operational Efficiency:

Predicting flight delays helps airlines optimize scheduling and resource allocation, reducing inefficiencies. It also aids in better crew management and maintenance planning.

2.Increased Accuracy and Predictability:

With accurate delay predictions, airlines can improve schedule reliability, making operations more predictable and reducing last-minute disruptions

3.Improved Passenger Experience:

Passengers receive timely updates on delays, allowing them to plan better and reduce stress. Airlines can also offer rescheduling options or other compensatory services.

4.Cost Savings:

Early delay predictions help reduce costs associated with fuel, compensation, and ground operations by allowing for more efficient resource utilization.

5.Enhanced Airport Management:

Airports can manage runway and gate usage more efficiently, preventing congestion. Ground handling operations can also be optimized with advanced delay information.

6.Better Air Traffic Control:

Air traffic controllers can use delay predictions to reroute flights and prevent bottlenecks in airspace, ensuring smoother flight operations.

7.Environmental Benefits:

Reducing unnecessary fuel consumption and optimizing flight paths contribute to lowering carbon emissions, promoting greener aviation practices.

8.Scalability:

FDP systems are adaptable across large networks, handling high traffic volumes and integrating with multiple systems, making them suitable for both regional and global operations.

Disadvantages

1.High Implementation Costs:

Developing and implementing flight delay prediction (FDP) systems can require significant investment in technology, infrastructure, and data analytics tools.

2. Data Dependency:

FDP relies heavily on large volumes of real-time data from multiple sources, which may not always be available or accurate, potentially affecting prediction accuracy.

3.Complex Integration

Integrating FDP systems with existing airport, airline, and air traffic management systems can be complex and time-consuming, requiring substantial coordination.

4.Limited Accuracy in Unpredictable Situations:

While FDP systems can predict delays under normal conditions, sudden events like extreme weather or technical issues may still cause unexpected disruptions that are hard to predict

5.Security and Privacy Concerns

Handling vast amounts of sensitive data, including flight schedules and operational information, poses risks of data breaches, which can compromise system security.

8. Conclusion

Flight Delay Prediction systems are a powerful tool for improving the efficiency, reliability, and sustainability of airline operations. By using real-time data and advanced algorithms, these systems can anticipate delays and provide actionable insights for airlines, air traffic controllers, and passengers. The benefits are far-reaching, including optimized scheduling, better resource allocation, cost savings, and improved passenger satisfaction.

In this Flight Delay Prediction project, we employed a Decision Tree model to forecast flight delays based on various factors such as weather conditions, flight schedules, and operational data. The use of the Decision Tree algorithm provided several benefits, including its simplicity, interpretability, and ability to handle both categorical and numerical data effectively. Through the project, we achieved reasonable accuracy in predicting flight delays, allowing for better decision-making in airline operations.

The model's interpretability is a key strength, as it allows stakeholders, such as airline operators and air traffic controllers, to understand how specific factors contribute to delays. This transparency is crucial for making informed operational adjustments. Additionally, the Decision Tree model is well-suited for handling missing data and offers insights into the most influential variables impacting delays, such as weather and departure times.

However, FDP comes with challenges. High implementation costs, data dependency, and complex integration processes can be barriers, particularly for smaller airlines and airports. Additionally, while FDP can predict delays under normal conditions, unforeseen events like extreme weather or sudden technical issues can still cause significant disruptions that are harder to predict accurately.

Despite these challenges, FDP systems hold great potential, especially as technology continues to advance. As the aviation industry grows, scalable and adaptable FDP systems will play an increasingly vital role in ensuring efficient and sustainable air travel. In the long term, the advantages of adopting FDP systems far outweigh the drawbacks, making them an essential component of modern aviation management.

9. Future Scope

1.Advanced Predictive Models

- Future FDP systems can utilize advanced machine learning algorithms, such as deep learning and ensemble techniques, to enhance accuracy and capture complex patterns in flight delay data.

2.Integration of Real-Time Data

- Incorporating real-time data from various sources, such as weather conditions, air traffic, and airport operations, will improve the timeliness and accuracy of delay predictions.

3.Enhanced External Factor Analysis

- Future systems could analyze the impact of external factors such as geopolitical events, pandemics, or public transport strikes, which can significantly affect flight schedules.

4.Passenger-Centric Applications

- Developing applications that directly communicate with passengers to provide personalized delay notifications and rebooking options, enhancing the overall travel experience.

5.AI-Driven Decision Support Systems

- Integrating FDP with AI-driven decision support tools can help airlines automatically adjust flight schedules and operations based on predicted delays, optimizing resource allocation.

6.Collaboration Among Stakeholders

- Promoting collaboration between **airlines, airports, and air traffic control** through shared delay prediction data can lead to more coordinated and efficient responses to disruptions.

7.Sustainability Integration

- Future FDP systems can focus on minimizing the **environmental impact** of flight operations by optimizing routes and schedules to reduce fuel consumption and emissions.

8. Predictive Maintenance Insights

- Incorporating predictive analytics for **aircraft maintenance** can reduce delays caused by technical failures, allowing airlines to proactively address potential issues.

10. Appendix

10.1 Source Code

flightdelays.ipynb

#IMPORTING NECESSARY LIBRARIES

```
import datetime, warnings, scipy

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt
```

#Importing the dataset

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

airport = pd.read_csv('airports.csv')

airlines = pd.read_csv('airlines.csv')

flightsinfo.info()

flightsinfo.shape

flightsinfo.describe()

delay_type = lambda x:((0,1)[x > 5],2)[x > 45]

flightsinfo['DELAY_LEVEL'] =
flightsinfo['DEPARTURE_DELAY'].apply(delay_type)


fig = plt.figure(1, figsize=(10,7))

ax = sns.countplot(x="AIRLINE", hue='DELAY_LEVEL', data=flightsinfo,
palette= ["#00FF00","#FFA500","#FF0000"])
```

```
labels = ax.get_xticklabels()
ax.set_xticklabels(labels)
plt.setp(ax.get_yticklabels(), fontsize=12, weight = 'normal', rotation = 0);
plt.setp(ax.get_xticklabels(), fontsize=12, weight = 'normal', rotation = 0);
ax.xaxis.label.set_visible(False)
plt.ylabel('No. of Flights', fontsize=16, weight = 'bold', labelpad=10)

L = plt.legend()
L.get_texts()[0].set_text('on time (t < 5 min)')
L.get_texts()[1].set_text('small delay (5 < t < 45 min)')
L.get_texts()[2].set_text('large delay (t > 45 min)')
plt.show()
                    import matplotlib.pyplot as plt
from collections import OrderedDict
from mpl_toolkits.basemap import Basemap

# Get flight counts for each airport
flightcount = flightsinfo['ORIGIN_AIRPORT'].value_counts()

plt.figure(figsize=(10, 10))

colors = ['purple', 'green', 'orange', 'yellow', 'red', 'lightblue']
size = [1, 100, 1000, 10000, 100000, 1000000]
labels = ["1 to 100", "100 to 1000", "1000 to 10000", "10000 to 100000", "100000 to 1000000"]

# Create a Basemap
```

```
map = Basemap(llcrnrlon=-180, urcrnrlon=-50, llcrnrlat=10, urcrnrlat=75,
lat_0=0, lon_0=0)
map.shadedrelief()
map.drawcoastlines()
map.drawcountries(linewidth=4)
map.drawstates(color='0.3')
flightsinfo_NULL = flightsinfo.isnull().sum()*100/flightsinfo.shape[0]
flightsinfo_NULL
#Univaraint
plt.figure(figsize=(10, 10))
axis = sns.countplot(y=Flights['Origin_city'], data = Flights,
            order=Flights['Origin_city'].value_counts().iloc[:20].index,palette="Set2"
)
axis.set_yticklabels(axis.get_yticklabels())
plt.tight_layout()
plt.show()
# Multivaraint
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


# Assuming Flights is your DataFrame


# Select only numeric columns for correlation
numeric_flights = Flights.select_dtypes(include=['number'])


# Create a subplot
plt.figure(figsize=(18, 12))
```

```
 # Generate the heatmap for correlation
 sns.heatmap(numeric_flights.corr(), annot=True, cmap="YlGnBu")

 # Adjust the y-axis limits
 b, t = plt.ylim()  # discover the values for bottom and top
 t -= 0.5  # Subtract 0.5 from the top
 plt.ylim(b, t)  # update the ylim(bottom, top) values

 # Show the plot
 plt.show()
#Bivaraint
plt.hist(Flights1['AIR_TIME'])
plt.show()
#Label encoding
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
Flights1['AIRLINE']= le.fit_transform(Flights1['AIRLINE'])
Flights1['ORIGIN_AIRPORT'] = le.fit_transform(Flights1['ORIGIN_AIRPORT'])
Flights1['DESTINATION_AIRPORT'] =
le.fit_transform(Flights1['DESTINATION_AIRPORT'])
Flights1['Day'] = le.fit_transform(Flights1['Day'])
Flights1.info()
#Split into train and test sets
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state =
5)
#Applying Standard Scaler
```

```
sc1=StandardScaler()
X_train_sc=sc1.fit_transform(X_train)
X_test_sc=sc1.transform(X_test)
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
classifierDT = DecisionTreeClassifier(criterion = 'entropy', random_state = None)
classifierDT.fit(X_train_sc, y_train)
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix
from sklearn.metrics import confusion_matrix
# Predicting the Test set results
y_pred = classifierDT.predict(X_test)

# Making the Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
score = classifierDT.score(X_test_sc,y_test)
#KNN
from sklearn.neighbors import KNeighborsClassifier
objClassifier=KNeighborsClassifier(n_neighbors=10,metric='minkowski',p=2)
objClassifier.fit(X_train_sc,y_train)
y_pred=objClassifier.predict(X_test_sc)

#Making the confussion matarix

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

```
score=objClassifier.score(X_test,y_test)
#Logistic Regression
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train_sc, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test_sc)

# Making the Confusion Matrix
score = classifier.score(X_test_sc,y_test)
cm = confusion_matrix(y_test, y_pred)
#calculating F1 Scores
print("F1 score :",f1_score(y_test, y_pred, average="macro"))
print("Precision Score :" , precision_score(y_test, y_pred, average="macro"))
print("Recall Score :" , recall_score(y_test, y_pred, average="macro"))
#Naive Bayes
rom sklearn.naive_bayes import GaussianNB
objclassifierGNB=GaussianNB()
objclassifierGNB.fit(X_train_sc,y_train)
# Predicting the Test set results
y_pred = objclassifierGNB.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
score = objclassifierGNB.score(X_test_sc,y_test)
print("F1 score :",f1_score(y_test, y_pred, average="macro"))
```

```
print("Precision Score :" , precision_score(y_test, y_pred, average="macro"))
print("Recall Score :" , recall_score(y_test, y_pred, average="macro"))
```

app.py

```
# app.py

from flask import Flask, render_template, request
import numpy as np
import joblib
import pandas as pd
import pickle

# Load the model and encoders
model = joblib.load(r'C:\Users\syeda\OneDrive\Desktop\Flight delays prediction\flight_model.pkl')
encoders = joblib.load(r'C:\Users\syeda\OneDrive\Desktop\Flight delays prediction\encoders.pkl')

df = pd.read_csv(r"C:\Users\syeda\OneDrive\Desktop\Flight delays prediction\flight_data.csv")
#df = pd.read_csv("flight_data.csv")

app = Flask(__name__)
```

```
@app.route("/", methods=["GET", "POST"])
def home():
    if request.method == "POST":
        # Get the values from the form
        carrier = request.form['carrier']
        origin = request.form['origin']
        dest = request.form['dest']
        distance = int(request.form['distance'])
        hour = int(request.form['hour'])

        day = int(request.form['day'])
        month = int(request.form['month'])

        # Prepare the input data
        input_data = pd.DataFrame({
            'carrier': [carrier],
            'origin': [origin],
            'dest': [dest],
            'distance': [distance],
            'hour': [hour],
            'day': [day],
            'month': [month]
        })
```

```
        # Apply encoders
        for col in encoders.keys():
            input_data[col] = encoders[col].transform(input_data[col])[0]

        # Make the prediction
        prediction = model.predict(input_data.values)
        result = 'This flight is likely to be departing late. Thank You for your
Cooperation.' if prediction[0] == 1 else 'This flight is likely to be departing on
time.'

        return render_template('index.html', result=result, carrier=carrier,
origin=origin, dest=dest, distance=distance, hour=hour, day=day, month=month,
                               carriers=df['carrier'].unique(), origins=df['origin'].unique(),
destinations=df['dest'].unique())

    return render_template('index.html', result=None, carriers=df['carrier'].unique(),
origins=df['origin'].unique(), destinations=df['dest'].unique())

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

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flight Delay Prediction</title>
    <style>
        /* General Styling */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-image: url('../static/img2.jpg');
            background-size: cover;
            background-position: center;
            color: #fff;
            text-align: center;
            min-height: 100vh;
            overflow-y: auto; /* Ensure page is scrollable if content overflows */
        }

        .container {
            display: flex;
            flex-direction: column;
```

```
        justify-content: center;
        align-items: center;
        padding: 20px;
        background-color: rgba(0, 0, 0, 0.7); /* Dark overlay */
        border-radius: 10px;
        width: 90%;
        max-width: 600px;
        margin: 40px auto; /* Added margin to prevent cutting off at the top and
bottom */
    }

    h1 {
        font-size: 2.5em;
        margin-bottom: 20px;
        color: #FFD700;
    }

    form {
        display: flex;
        flex-direction: column;
        align-items: center;
        width: 100%;
        gap: 10px;
    }

    label {
        font-size: 1.1em;
        margin-bottom: 5px;
        color: #fff;
```

```
}

input, select, button {
    padding: 8px;
    font-size: 1em;
    border-radius: 5px;
    border: 1px solid #ccc;
    background-color: #f7f7f7;
    width: 75%; /* Reduced input size to 75% */
    margin-bottom: 10px;
}

button {
    background-color: #4CAF50;
    color: white;
    cursor: pointer;
    font-weight: bold;
    width: auto; /* Ensure the button does not take full width */
    padding: 10px 20px; /* Adjust button padding */
}

button:hover {
    background-color: #45a049;
}

h2 {
    font-size: 1.8em;
    margin-top: 20px;
```

```
            color: #bafbab;
        }

        p {
            font-size: 1.2em;
            color: #2200ff;
        }

        .form-group {
            width: 100%;
        }

        .time-inputs {
            display: flex;
            justify-content: space-between;
            gap: 10px; /* Adds space between the hour and minute input fields */
        }

        .time-inputs input {
            width: 48%; /* Adjusts the width of each input field */
        }

    </style>
</head>
<body>
    <div class="container">
        <h1>Flight Delay Prediction</h1>
        <form method="POST">
```

```
        <div class="form-group">
            <label for="carrier">Carrier:</label>
            <select name="carrier" id="carrier">
                {% for c in carriers %}
                    <option value="{{ c }}" {% if c == carrier %}selected{% endif
%}>{{ c }}</option>
                {% endfor %}
            </select>
        </div>

        <div class="form-group">
            <label for="origin">Origin:</label>
            <select name="origin" id="origin">
                {% for o in origins %}
                    <option value="{{ o }}" {% if o == origin %}selected{% endif
%}>{{ o }}</option>
                {% endfor %}
            </select>
        </div>

        <div class="form-group">
            <label for="dest">Destination:</label>
            <select name="dest" id="dest">
                {% for d in destinations %}
                    <option value="{{ d }}" {% if d == dest %}selected{% endif %}>{{
d }}</option>
                {% endfor %}
            </select>
        </div>
```

```
<div class="form-group">
    <label for="distance">Distance (miles):</label>
    <input type="number" name="distance" id="distance" value="{{ distance
}}" min="0" max="20000">
</div>

<div class="form-group">
    <label for="dayofweek">Scheduled Departure Day of Week:</label>
    <select id="Married" name="Married">
        <option value=1>Monday</option>
        <option value=2>Tuesday</option>
        <option value=3>Wednesday</option>
        <option value=4>Thursday</option>
        <option value=5>Friday</option>
        <option value=6>Saturday</option>
        <option value=7>Sunday</option>
    </select>
</div>

<div class="form-group">
    <label for="day">Scheduled Departure Date (1-31):</label>
    <input type="number" name="day" id="day" value="{{ day }}" min="1"
max="31">
</div>

<div class="form-group">
    <label for="month">Scheduled Departure Month (1-12):</label>
```

```
                <input type="number" name="month" id="month" value="{{ month }}"
min="1" max="12">
            </div>


            <div class="form-group">
                <label for="hour">Scheduled Departure Hour (0-23):</label>
                <input type="number" name="hour" id="hour" value="{{ hour }}"
min="0" max="23">
            </div>


            <div class="form-group">
                <label for="arr_time_hrs">Scheduled Arrival Time:</label>
                <div class="time-inputs">
                    <input type="number" name="arr_time_hrs" id="arr_time_hrs"
placeholder="Hours" min="0" max="23" value="{{ hours1 }}">
                    <input type="number" name="arr_time_mins" id="arr_time_mins"
placeholder="Minutes" min="0" max="59" value="{{ minutes1 }}">
                </div>
            </div>


            <div class="form-group">
                <label for="act_time_hrs">Actual Departure Time:</label>
                <div class="time-inputs">
                    <input type="number" name="act_time_hrs" id="act_time_hrs"
placeholder="Hours" min="0" max="23" value="{{ hours2 }}">
                    <input type="number" name="act_time_mins" id="act_time_mins"
placeholder="Minutes" min="0" max="59" value="{{ minutes2 }}">
                </div>
            </div>
```

```
            <button type="submit">Predict</button>
        </form>

        {% if result %}
            <h2>Prediction Result:</h2>
            <p>{{ result }}</p>
        {% endif %}
    </div>
</body>
</html>
```

GitHub & Project Demo Link :

[syedasifa123/FlightDelaysPrediction]

https://drive.google.com/drive/folders/1ZuqMUnRIajSQH_j6D1MopO8B6xTipBx
t