Final Project Report Template

1. Introduction
    1.1. Project overviews
    1.2. Objectives
2. Project Initialization and Planning Phase
    2.1. Define Problem Statement
    2.2. Project Proposal (Proposed Solution)
    2.3. Initial Project Planning
3. Data Collection and Preprocessing Phase
    3.1. Data Collection Plan and Raw Data Sources Identified
    3.2. Data Quality Report
    3.3. Data Exploration and Preprocessing
4. Model Development Phase
    4.1. Feature Selection Report
    4.2. Model Selection Report
    4.3. Initial Model Training Code, Model Validation and Evaluation Report
5. Model Optimization and Tuning Phase
    5.1. Hyperparameter Tuning Documentation
    5.2. Performance Metrics Comparison Report
    5.3. Final Model Selection Justification
6. Results
    6.1. Output Screenshots
7. Advantages & Disadvantages
8. Conclusion
9. Future Scope
10. Appendix
    10.1. Source Code
    10.2. GitHub & Project Demo Link

1. Introduction

1.1 Project Overview

- Flight delays are a common issue that disrupts travel schedules and cause significant inconvenience for passengers and airlines alike.
- The ability to predict these delays can help improve operational efficiency and enhance customer experience. This project aims to develop a machine learning model for predicting flight delays using a dataset of historical flight information.
- Several classification algorithms, including Decision Tree, K-Nearest Neighbors (KNN), Logistic Regression, and Naive Bayes, are applied to build the model.

❑ A flight delay occurs when an airline flight takes off and/or lands later than its scheduled time.
❑ The United States Federal Aviation Administration (FAA) considers a flight to be delayed when it is 15 minutes later than its scheduled time.
❑ Examples of occurrences:
a) aircraft cleaning
b) aircraft damage
c) awaiting the arrival of connecting passengers or crew
d) bird strike
e) crew legality (pilot or attendant rest).. etc,.

1.2 Objectives

- Analyze historical flight data through Exploratory Data Analysis (EDA) to identify key factors contributing to flight delays.
- Preprocess and prepare the dataset, addressing missing values and feature selection for optimal model performance.
- Evaluate the performance of various machine learning algorithms, including Decision Tree, KNN, Logistic Regression, and Naive Bayes, for accurate delay prediction.

2.Project Initialization and Planning Phase

2.1 Define Problem Statement

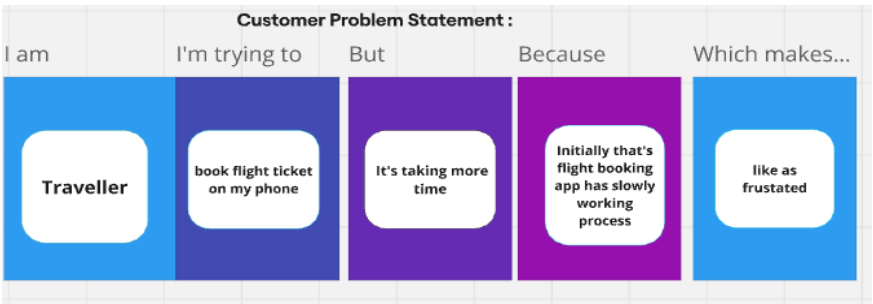
Date	01 October 2024
Team ID	LTVIP2024TMID24897
Project Name	Flight Delays Prediction using ML
Maximum Marks	3 Marks

Define Problem Statements (Customer Problem Statement Template):

Flight delays are quite frequent (19% of the US domestic flights arrive more than 15 minutes late), and are a major source of frustration and cost for the passengers. As we will see, some flights are more frequently delayed than others, and there is an interest in providing this information to travellers.
Flight prediction is crucial during the decision-making process for all players of commercial aviation. Moreover, the development of accurate prediction models for flight delays became cumbersome due to the complexity of air transportation system, the number of methods for prediction, and the deluge of flight data. Based on data, we would like to analyse what are the major cause for flight delays and assign a probability on whether a particular flight will be delayed.

Reference: https://miro.com/templates/customer-problem-statement/

Example:



Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A Travaller	Booking tickets in airport	So many travellers are in forming line	Delay for my flight booking process	Sad and angry about booking process

PS-2	Pilot	Departing the flight	15 minutes late	High winds and suddenly thunderstorms. That's why, the flight is landed on below airport	I was listening to approach control, and it was pretty clear there was a backup in traffic due to weather.

2.2 Project Proposal (Proposed Solution)

Date	01 October 2024
Team ID	LTVIP2024TMID24897
Project Title	Flight delay prediction using ML
Maximum Marks	3 Marks

Project Proposal (Proposed Solution) template

It can dramatically reduce the flight delays to saves huge amount of turnovers, using machine-learning algorithms.

machine learning models could accurately predict flight delays, with a prediction accuracy of up to 92%.

Project Overview	
Objective	Such a model may help both passengers as well as airline companies to predict future delays and minimize them.
Scope	This project can identify the wheather reports and passengers late
Problem Statement	
Description	A change in Flight schedule due to which You will not have sufficient time to change planes or other means of transportation for Flight Connection(s)
Impact	Delayed flights can be inconvenient, especially for business travelers who may miss appointments. Passengers may also experience anger, frustration, or air rage.

Proposed Solution	
Approach	statistical models and machine learning models
Key Features	low visibility, hail, high winds at takeoff, and thunderstorms

Resource Requirements

Resource Type	**Description**	**Specification/Allocation**
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	e.g., 2 x NVIDIA V100 GPUs
Memory	RAM specifications	e.g., 8 GB
Storage	Disk space for data, models, and logs	e.g., 1 TB SSD
Software		
Frameworks	Python frameworks	e.g., Flask
Libraries	Additional libraries	e.g., scikit-learn, pandas, numpy
Development Environment	IDE, version control	e.g., Jupyter Notebook, Git
Data		
Data	Source, size, format	e.g., Kaggle dataset, 10,000 images

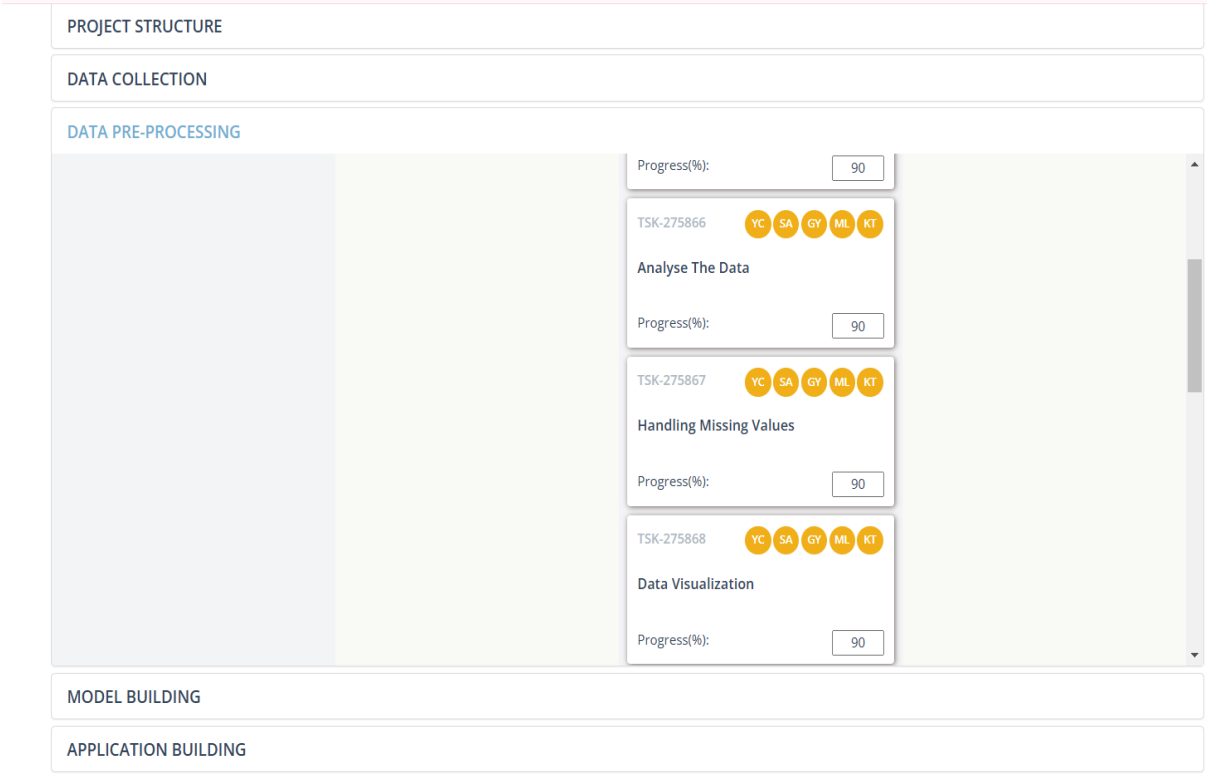
2.3 Initial Project Planning

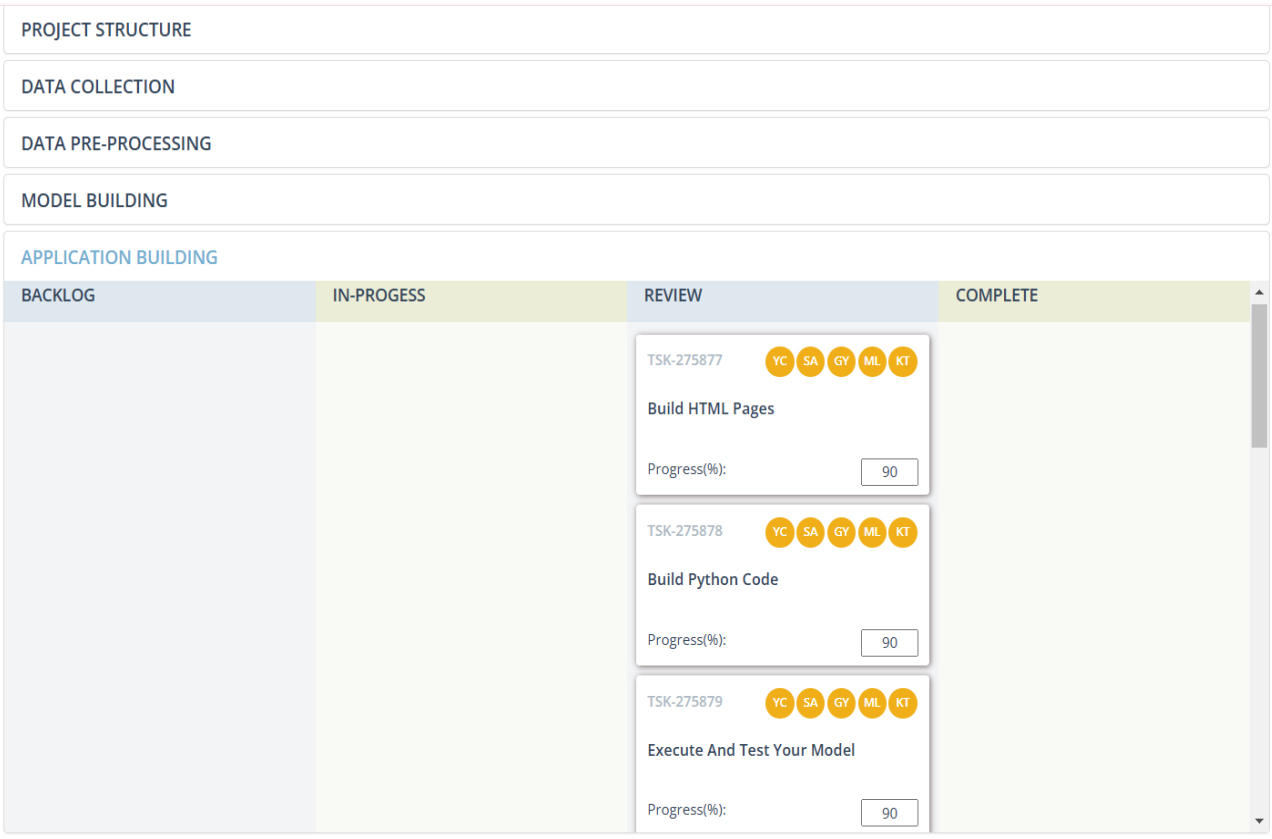
Date	28-09-2024
Team ID	LTVIP2024TMID24897
Project Name	Flight delays Prediction using ML
Maximum Marks	4 Marks

Product Backlog, Sprint Schedule, and Estimation :

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Data collecting	USN-1	Understanding and loading	2	High	Leelavathi	28-09-2024	29-09-2024
Sprint-1	Preprocessing	USN-2	Loading data	1	High	Taruna sree	01-10-2024	02-10-2024
Sprint-2	Model Development	USN-3	Training the model	2	Medium	Yasaswini	03-10-2024	04-10-2024
Sprint-3	Model tuning and testing	USN-4	Model tuning	2	Medium	Ajim , y.c.m.reddy	04-10-2024	05-10-2024
Sprint-4	Web integration and deployment	USN-5	Building HTML templates	1	High	Taruna sree	07-10-2024	09-10-2024
Sprint - 5	Web integration and deployment	USN-6	Local deployment	2	High	Tarunasree, yasaswini	09-10-2024	15-10-2024

Screenshots:





3.Data Collection and Preprocessing Phase

3.1 Data Collection Plan and Raw Data Sources Identified

Date	03-10-2024
Team ID	LTVIP2024TMID24897
Project Title	Flight delays prediction using ML
Maximum Marks	2 Marks

Data Collection Plan & Raw Data Sources Identification Template

A data collection plan for flight delays prediction involves gathering relevant data from various sources to identify the predict for flight delay. Raw data sources can include regression data, algoritms results. Additionally, data from wearable devices, mobile apps, and passenger-reported outcomes can also be used. The data collection plan should ensure that the data is accurate, complete, and relevant to the research question.

Data Collection Plan:

Section	Description
Project Overview	The objective of this machine learning project is to identify the persons who have flight delays based on the prediction records.
Data Collection Plan	The data will be searched from the kaggle and other websites and world airlines websites.
Raw Data Sources Identified	The data will be collected from the kaggle and other websites and world airlines websites.

Raw Data Source Report:

Source Name	Description	Location/URL	Format	Size	Access Permissions

Kaggle Dataset	The airlines details is valuable collections of data that includes 20 delays predict	https://www.kaggle.com/code/fabiendaniel/predicting-flight-delays-tutorial?scriptVersionId=1566193&cellId=1	CSV	200 MB	Public

3.2. Data Quality Report

Date	03-10- 2024
Team ID	LTVIP2024TMID24897
Project Title	Flight delays prediction using ML
Maximum Marks	2 Marks

Data Quality Report

The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

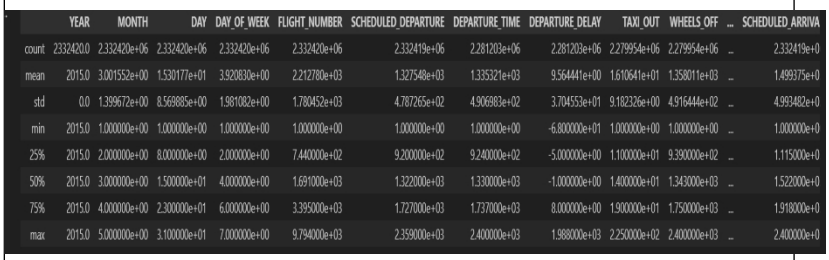
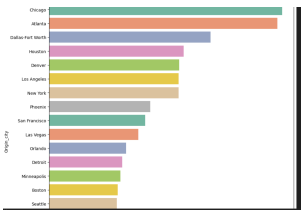
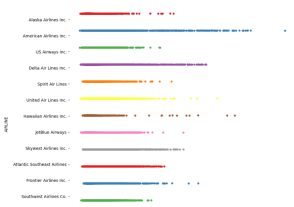
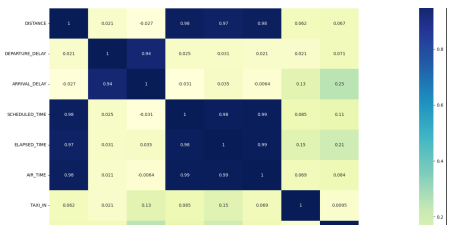
Data Source	Data Quality Issue	Severity	Resolution Plan
Kaggle Dataset	Missing values are in the [standard scalar]	Low	Used less imputation
Kaggle dataset	Categorical data in the dataset	Moderate	There is no need of Standardization and Normalization for the dataset, as we have used histogram techniques

Data Exploration and Preprocessing

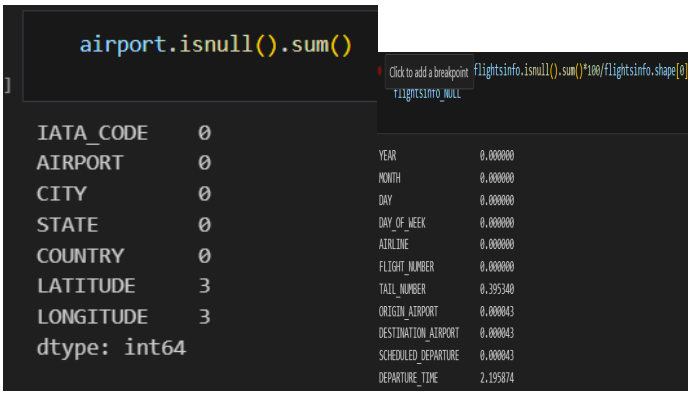
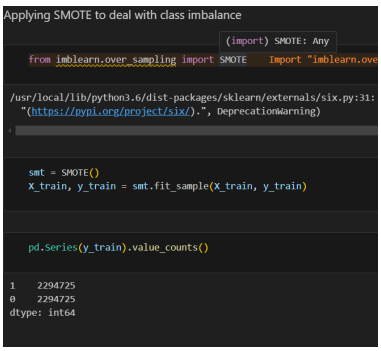
Date	03-10- 2024
Team ID	LTVIP2024TMID24897
Project Title	Flight delays prediction using ML
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	Description
Data Overview	dimension: 8rows X 26 columns Descriptive statistics: 
Univariate Analysis	
Bivariate Analysis	
Multivariate Analysis	

	YEAR	MONTH	DAY	DAY_OF_WEEK	FLIGHT_NUMBER	SCHEDULED_DEPARTURE	DEPARTURE_TIME	DEPARTURE_DELAY	TAXI_OUT	WHEELS_OFF	...	SCHEDULED_ARRIVA
count	2332420.0	2.332420e+06	2.332420e+06	2.332420e+06	2.332420e+06	2.332419e+06	2.281203e+06	2.281203e+06	2.279954e+06	2.279954e+06	...	2.332419e+0
mean	2015.0	3.001552e+00	1.530177e+01	3.920830e+00	2.212780e+03	1.327548e+03	1.335321e+03	9.564441e+00	1.610641e+01	1.358011e+03	...	1.499375e+0
std	0.0	1.399672e+00	8.569885e+00	1.981082e+00	1.780452e+03	4.787265e+02	4.906983e+02	3.704553e+01	9.182326e+00	4.916444e+02	...	4.993482e+0
min	2015.0	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	-6.800000e+01	1.000000e+00	1.000000e+00	...	1.000000e+0
25%	2015.0	2.000000e+00	8.000000e+00	2.000000e+00	7.440000e+02	9.200000e+02	9.240000e+02	-5.000000e+00	1.100000e+01	9.390000e+02	...	1.115000e+0
50%	2015.0	3.000000e+00	1.500000e+01	4.000000e+00	1.691000e+03	1.322000e+03	1.330000e+03	-1.000000e+00	1.400000e+01	1.343000e+03	...	1.522000e+0
75%	2015.0	4.000000e+00	2.300000e+01	6.000000e+00	3.395000e+03	1.727000e+03	1.737000e+03	8.000000e+00	1.900000e+01	1.750000e+03	...	1.918000e+0
max	2015.0	5.000000e+00	3.100000e+01	7.000000e+00	9.794000e+03	2.359000e+03	2.400000e+03	1.988000e+03	2.250000e+02	2.400000e+03	...	2.400000e+0

Outliers and Anomalies	There is no need of outliers ,as we using regressons and classifiers
Data Preprocessing Code Screenshots	
Loading Data	
Handling Missing Data	
Data Transformation	There is no need of standardization and normalization of our dataset , as we using histogram techniques.
Feature Engineering	
Save Processed Data	-

4. Model Development Phase

4.1 Feature Selection Report

Date	03-10-2024
Team ID	LTVIP2024TMID24897
Project Title	Flight Delaya prediction using ML
Maximum Marks	5 Marks

Feature Selection Report

Feature	Description	Selected (Yes/No)	Reasoning
Transportation	Flight delays has become a very important subject for air transportation all over the world because of the associated financial loses that the aviation industry is going through.	Yes	According to data from the Bureau of Transportation Statistics (BTS) of the United Stated, over 20% of US flights were delayed during 2018, which resulted in a severe economic impact equivalent to 41 billion US$.
Weather	Weather conditions are a significant factor in flight delays	Yes	According to the Bureau of Transportation Statistics, weather caused roughly a quarter of each month's delays in 2023
Impact	Impact of Flight Delays: Discuss the consequences of flight delays on passengers, airlines, and airports.	**No**	It seems like you are looking for something different than the features related to flight delays

Date	03-10- 2024
Team ID	LTVIP2024TMID24897
Project Title	Flight Delaya prediction using ML
Maximum Marks	6 Marks

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	A decision tree is a valuable tool for visualizing the factors that contribute to flight delays and the potential outcomes based on various conditions	**Weather Conditions:** Clear Inclement Weather **Aircraft Status:** On Time Delayed Arrival	Accuracy = 98%
KNN neighbors	The K-Nearest Neighbor (KNN) algorithm is a popular machine learning	Departure and Arrival Times Route Information Weather Conditions	82%

	approach for predicting flight delays	Air Traffic Control (ATC) Delays Aircraft Status (e.g., on-time, delayed arrival) Crew Availability	
Logistic regression	Logistic regression is a popular machine learning approach for predicting flight delays. In the context of flight delay prediction	Departure and Arrival Times Route Information (e.g., origin, destination, distance) Weather Conditions	**110%**

4.3. Initial Model Training Code, Model Validation and Evaluation Report

Date	03-10-2024
Team ID	LTVIP2024TMID24897
Project Title	Flight delay prediction using ML
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

Train Test split for all models:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state = 2)
```

+ Code + Markdown

```
y_train.value_counts()
```

```
0    2294725
1    1358745
Name: Is_Delayed, dtype: int64
```

Decision Tree:

```
from sklearn.tree import DecisionTreeClassifier
classifierDT = DecisionTreeClassifier(criterion = 'entropy', random_state = None)
classifierDT.fit(X_train_sc, y_train)
```

K nearest neighbors:

```
from sklearn.neighbors import KNeighborsClassifier
objClassifier=KNeighborsClassifier(n_neighbors=10,metric='minkowski',p=2)
objClassifier.fit(X_train_sc,y_train)
```

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)
```

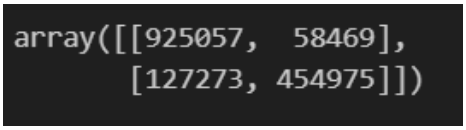
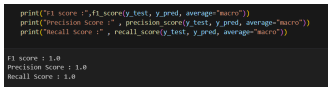
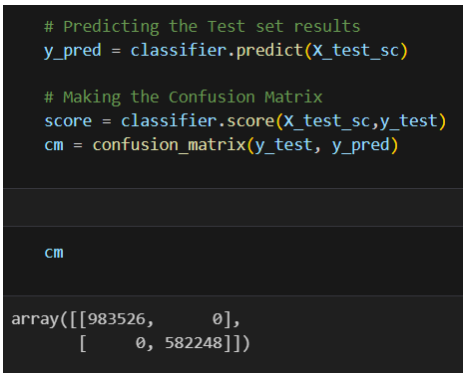
Model Validation and Evaluation Report:

Model	Classification report	Accuracy	Confusion matrix

Decision tree	score
	0.9826194584914554
	F1 score : 0.2725298912293622
	Precision Score : 0.6644134619299129
	Recall Score : 0.5006117468892067
		98%	# 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)
			array([[1303, 982223],
			[59, 582189]])

K nearest neighbours	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")) F1 score : 0.8696222002024336 Precision Score : 0.882589950527832 Recall Score : 0.8609813308550668	86%	from sklearn.metrics import confusion_matrix cm=confusion_matrix(y_test,y_pred) score=objClassifier.score(X_test,y_test)

Confusion Matrix

			
Logistic Regression		110%	

MODEL – Decision tree classifier ** ACCURACY – [0.9826]

5.Model Optimization and Tuning Phase

5.1 Hyperparameter Tuning Documentation

Date	03-10-2024
Team ID	LTVIP2024TMID24897
Project Title	Flight delay prediction using ML
Maximum Marks	10 Marks

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 (6 Marks):

Model	Tuned Hyperparameters	Optimal Values

Decision tree	`from sklearn.tree import DecisionTreeClassifier`
	`classifierDT = DecisionTreeClassifier(criterion = 'entropy', random_state = None)`
	`classifierDT.fit(X_train_sc, y_train)`
		`DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best')`

KNN	`from sklearn.neighbors import KNeighborsClassifier`
	`objClassifier=KNeighborsClassifier(n_neighbors=10,metric='minkowski',p=2)`
	`objClassifier.fit(X_train_sc,y_train)`
		`KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=None, n_neighbors=10, p=2, weights='uniform')`

Logistic regression	`from sklearn.linear_model import LogisticRegression`
	`classifier = LogisticRegression(random_state = 0)`
	`classifier.fit(X_train_sc, y_train)`
		`LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, l1_ratio=None, max_iter=100, multi_class='warn', n_jobs=None, penalty='l2', random_state=0, solver='warn', tol=0.0001, verbose=0, warm_start=False)`

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric, Optimized Metrics

Decision tree	`print("F1 score :",f1_score(y_test, y_pred, average="macro"))`
	`print("Precision Score :" , precision_score(y_test, y_pred, average="`
	`print("Recall Score :" , recall_score(y_test, y_pred, average="macro"`
	
	F1 score : 0.2725298912293622
	Precision Score : 0.6644134619299129
	Recall Score : 0.5006117468892067

KNN	`print("F1 score :",f1_score(y_test, y_pred, average="macro"))`
	`print("Precision Score :" , precision_score(y_test, y_pred, average="`
	`print("Recall Score :" , recall_score(y_test, y_pred, average="macro"`
	
	F1 score : 0.8696222002024336
	Precision Score : 0.8825899500527832
	Recall Score : 0.8609813308550668

Final Model Selection Justification (2 Marks):

Final Model	Reasoning

Decision Tree	`# 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)`
	
	`cm`
	
	array([[1303, 982223],
	[59, 582189]])
	
	`score`
	
	0.9826194584914554
	
	This model has been selected because it has the high accuracy and f1-score compared to the other model mentioned above.

Advantages & Disadvantages

Advantages---

- **Improved accuracy:** Machine learning algorithms can analyze large amounts of data and identify patterns that may not be apparent to human analysts, leading to more accurate predictions of flight delays.
- **Reduced costs:** By predicting flight delays, airlines and airports can reduce costs associated with delays, such as fuel consumption, crew expenses, and damaged reputations.
- **Enhanced customer experience**: Predicting flight delays can help airlines and airports to provide better customer service, such as informing passengers of delays in advance and offering alternative flights or compensation**.**
- **Increased efficiency:** Machine learning algorithms can help airlines and airports to optimize their operations, such as scheduling and resource allocation, to minimize the impact of delays.

Disadvantages---

- **Data quality issues**: Machine learning algorithms require high-quality data to make accurate predictions, but flight data can be noisy, incomplete, or biased, which can affect the accuracy of predictions.
- **Complexity**: Machine learning algorithms can be complex and require significant computational resources, which can be a challenge for airlines and airports with limited resources.
- **Interpretability**: Machine learning algorithms can be difficult to interpret, making it challenging to understand why a particular prediction was made, which can be a concern for airlines and airports that need to explain their decisions to customers and regulators.
- **Dependence on historical data**: Machine learning algorithms rely on historical data to make predictions, but flight delays can be affected by unexpected events, such as weather or air traffic control issues, which may not be reflected in historical data.

CONCLUSION

In conclusion, predicting flight delays using machine learning has both advantages and disadvantages. While machine learning algorithms can improve accuracy, reduce costs, enhance customer experience, and increase efficiency, they also have limitations, such as data quality issues, complexity, interpretability, and dependence on historical data.

By carefully considering these factors and implementing machine learning algorithms effectively, airlines and airports can harness the benefits of predicting flight delays and improve their operations, customer satisfaction, and overall performance.

APPENDIX :

“Source Code”---

“APP.PY”

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

# Load the model and encoders
model = joblib.load('flight_model.pkl')
encoders = joblib.load("encoders.pkl")
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
        flight_num = request.form['flight']
        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'])
        schedule_hr=int(request.form['shour'])
        arrival_hr=int(request.form['ahour'])

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

```
                                    'schedule_hr':[schedule_hr],
        'arrival_hr':[arrival_hr]
    })

    # Apply encoders on top of the input data
    input_data=input_data.iloc[:,1:8]
    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)
```

“ RESULT. 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/1.jpg');
    background-size: cover;
    background-position: center;
    color: #fff;
    text-align: center;
}

.container {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: rgba(0, 0, 0, 0.7); /* Dark overlay */
    padding: 20px;
    border-radius: 10px;
    width: 90%;
    max-width: 600px;
    margin: 0 auto;
}

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: 100%;
    margin-bottom: 10px;
}

button {
    background-color: #4CAF50;
    color: white;
    cursor: pointer;
    font-weight: bold;
}

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%;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Flight Delay Prediction</h1>
        <form method="POST">
            <div class="form-group">
                <label for="carrier">Flight Number:</label>
                    <input type="text" name="flight" id="flight" value="{{ flight
}}">
            </div>
            <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="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="day">Scheduled Departure Day (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="year">Scheduled Arrival Time (0-23):</label>
            <input type="number" name="shour" id="shour" value="{{ shour
}}" min="0" max="23">
        </div>
        <div class="form-group">
            <label for="day">Actual Departure Time (0-23):</label>
            <input type="number" name="ahour" id="ahour" value="{{ ahour
}}" min="0" max="23">
        </div>

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

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

pyModel Outputs:

+ Code + Text

```
import datetime, warnings, scipy
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

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

```
[ ] list(flightsinfo.columns)
```

```
['YEAR',
 'MONTH',
 'DAY',
 'DAY_OF_WEEK',
 'AIRLINE',
 'FLIGHT_NUMBER',
 'TAIL_NUMBER',
 'ORIGIN_AIRPORT',
 'DESTINATION_AIRPORT',
 'SCHEDULED_DEPARTURE',
 'DEPARTURE_TIME',
 'DEPARTURE_DELAY',
 'TAXI_OUT',
 'WHEELS_OFF',
 'SCHEDULED_TIME',
 'ELAPSED_TIME',
 'AIR_TIME',
 'DISTANCE',
 'WHEELS_ON',
 'TAXI_IN',
```

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

```
flightsinfo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2332420 entries, 0 to 2332419
Data columns (total 31 columns):
 #   Column               Dtype
---  ------               -----
 0   YEAR                 int64
 1   MONTH                int64
 2   DAY                  int64
 3   DAY_OF_WEEK          int64
 4   AIRLINE              object
 5   FLIGHT_NUMBER        int64
 6   TAIL_NUMBER          object
 7   ORIGIN_AIRPORT       object
 8   DESTINATION_AIRPORT  object
 9   SCHEDULED_DEPARTURE  float64
 10  DEPARTURE_TIME       float64
 11  DEPARTURE_DELAY      float64
 12  TAXI_OUT             float64
 13  WHEELS_OFF           float64
 14  SCHEDULED_TIME       float64
 15  ELAPSED_TIME         float64
 16  AIR_TIME             float64
 17  DISTANCE             float64
 18  WHEELS_ON            float64
 19  TAXI_IN              float64
 20  SCHEDULED_ARRIVAL    float64
 21  ARRIVAL_TIME         float64
```

+ Code + Text

```
flightsinfo.shape
```

```
(2332420, 31)
```

```
flightsinfo.describe()
```

	YEAR	MONTH	DAY	DAY_OF_WEEK	FLIGHT_NUMBER	SCHEDULE
count	2332420.0	2.332420e+06	2.332420e+06	2.332420e+06	2.332420e+06	2
mean	2015.0	3.001552e+00	1.530177e+01	3.920830e+00	2.212780e+03	1
std	0.0	1.399672e+00	8.569885e+00	1.981082e+00	1.780452e+03	4
min	2015.0	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1
25%	2015.0	2.000000e+00	8.000000e+00	2.000000e+00	7.440000e+02	9
50%	2015.0	3.000000e+00	1.500000e+01	4.000000e+00	1.691000e+03	1
75%	2015.0	4.000000e+00	2.300000e+01	6.000000e+00	3.395000e+03	1
max	2015.0	5.000000e+00	3.100000e+01	7.000000e+00	9.794000e+03	2

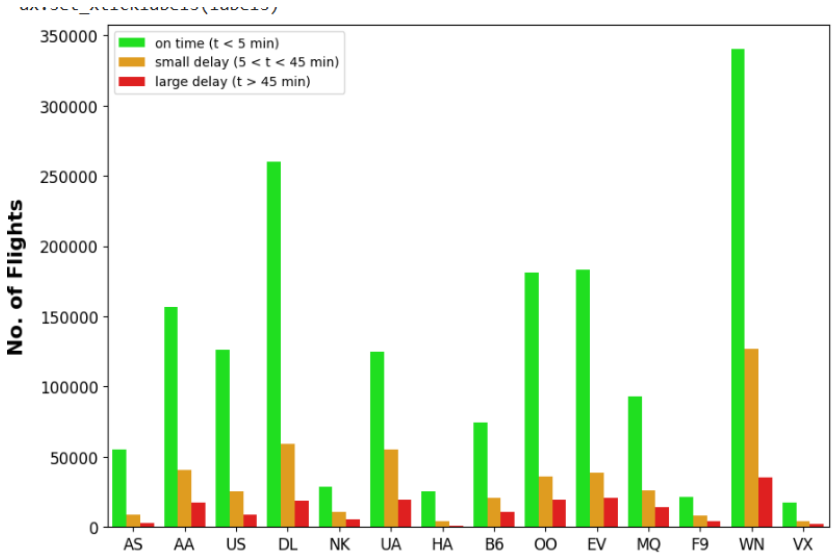
8 rows × 26 columns

```
[ ] airlinecompanies = airlines.set_index('IATA_CODE')['AIRLINE'].to_dict()
```

```
[ ] airlinecompanies
```

```
{'UA': 'United Air Lines Inc.',
 'AA': 'American Airlines Inc.',
 'US': 'US Airways Inc.',
 'F9': 'Frontier Airlines Inc.',
 'B6': 'JetBlue Airways',
 'OO': 'Skywest Airlines Inc.',
 'AS': 'Alaska Airlines Inc.',
 'NK': 'Spirit Air Lines',
 'WN': 'Southwest Airlines Co.',
 'DL': 'Delta Air Lines Inc.',
 'EV': 'Atlantic Southeast Airlines',
 'HA': 'Hawaiian Airlines Inc.',
 'MQ': 'American Eagle Airlines Inc.',
 'VX': 'Virgin America'}
```

```
[ ] 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=(11,6))
ax = sns.barplot(x="AIRLINE", y="DEPARTURE_DELAY", data=flightsinfo, color="orange"
ax = sns.barplot(x="AIRLINE",y="ARRIVAL_DELAY",  data=flightsinfo, color="b", hatch
                 alpha = 0.0, ci=None)
labels = ax.get_xticklabels()
ax.set_xticklabels(labels)
ax.xaxis.label.set_visible(False)
plt.ylabel('Mean delay - Yellow (Departure), X (Arrival)',
           fontsize=14, weight = 'bold', labelpad=10);
```

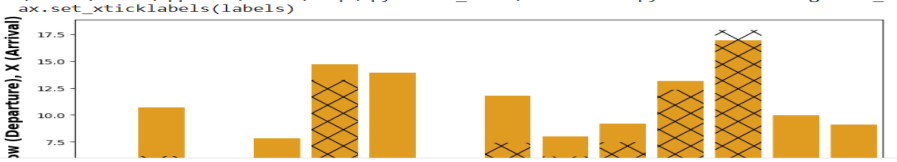
```
C:\Users\tharu\AppData\Local\Temp\ipykernel_8292\3270147513.py:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  ax = sns.barplot(x="AIRLINE", y="DEPARTURE_DELAY", data=flightsinfo, color="orange
C:\Users\tharu\AppData\Local\Temp\ipykernel_8292\3270147513.py:3: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  ax = sns.barplot(x="AIRLINE",y="ARRIVAL_DELAY",  data=flightsinfo, color="b", hatc
C:\Users\tharu\AppData\Local\Temp\ipykernel_8292\3270147513.py:6: UserWarning: set_t
  ax.set_xticklabels(labels)
```




```
airport.isnull().sum()
```

```
IATA_CODE    0
AIRPORT      0
CITY         0
STATE        0
COUNTRY      0
LATITUDE     3
LONGITUDE    3
dtype: int64
```

```
[ ] airport = airport.dropna(subset = ['LATITUDE','LONGITUDE'])
```

```
[ ] airport.head(10)
```

	IATA_CODE	AIRPORT	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
0	ABE	Lehigh Valley International Airport	Allentown	PA	USA	40.65236	-75.44040
1	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
2	ABQ	Albuquerque International Sunport	Albuquerque	NM	USA	35.04022	-106.60919
3	ABR	Aberdeen Regional Airport	Aberdeen	SD	USA	45.44906	-98.42183
4	ABY	Southwest Georgia Regional Airport	Albany	GA	USA	31.53552	-84.19447
5	ACK	Nantucket Memorial	Nantucket	MA	USA	41.25305	-70.06018

```
[ ]  airlines
```

	IATA_CODE	AIRLINE
0	UA	United Air Lines Inc.
1	AA	American Airlines Inc.
2	US	US Airways Inc.
3	F9	Frontier Airlines Inc.
4	B6	JetBlue Airways
5	OO	Skywest Airlines Inc.
6	AS	Alaska Airlines Inc.
7	NK	Spirit Air Lines
8	WN	Southwest Airlines Co.
9	DL	Delta Air Lines Inc.
10	EV	Atlantic Southeast Airlines
11	HA	Hawaiian Airlines Inc.
12	MQ	American Eagle Airlines Inc.
13	VX	Virgin America

```
[ ]  flightsinfo_NULL = flightsinfo.isnull().sum()*100/flightsinfo.shape[0]
     flightsinfo_NULL
```

```
YEAR                 0.000000
MONTH                0.000000
DAY                  0.000000
DAY_OF_WEEK          0.000000
AIRLINE              0.000000
FLIGHT_NUMBER        0.000000
TAIL_NUMBER          0.395340
ORIGIN_AIRPORT       0.000043
DESTINATION_AIRPORT  0.000043
SCHEDULED_DEPARTURE  0.000043
DEPARTURE_TIME       2.195874
DEPARTURE_DELAY      2.195874
TAXI_OUT             2.249423
WHEELS_OFF           2.249423
SCHEDULED_TIME       0.000300
ELAPSED_TIME         2.522530
AIR_TIME             2.522530
DISTANCE             0.000043
WHEELS_ON            2.320637
TAXI_IN              2.320637
SCHEDULED_ARRIVAL    0.000043
ARRIVAL_TIME         2.320637
ARRIVAL_DELAY        2.522530
DIVERTED             0.000043
CANCELLED            0.000043
```

Decision trees

```
[ ] from sklearn.tree import DecisionTreeClassifier
    classifierDT = DecisionTreeClassifier(criterion = 'entropy', random_state = None)
    classifierDT.fit(X_train_sc, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

```
[ ] #from sklearn.externals.six import StringIO
    #from IPython.display import Image
    #from sklearn.tree import export_graphviz
    #import pydotplus
    #dot_data = StringIO()
    #export_graphviz(classifierDT, out_file=dot_data,
     #                filled=True, rounded=True,
     #                special_characters=True)
    #graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    #Image(graph.create_png())
```

```
[ ] from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
```

```
[ ] 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)
```

```
[ ] cm
```

```
array([[  1303, 982223],
       [    59, 582189]])
```

```
[ ] score
```

```
0.9826194584914554
```

```
[ ] 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"))
```

```
F1 score : 0.2725298912293622
Precision Score : 0.6644134619299129
Recall Score : 0.5006117468892067
```

```
[ ] fig = plt.figure()
    ax = fig.add_subplot(111)
    cax = ax.matshow(cm)
    plt.title('Confusion matrix of the Decision tree classifier')
```

K nearest neighbours classifier

```
[ ] from sklearn.neighbors import KNeighborsClassifier
    objClassifier=KNeighborsClassifier(n_neighbors=10,metric='minkowski',p=2)
    objClassifier.fit(X_train_sc,y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                     weights='uniform')
```

```
[ ] 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)
```

```
[ ] cm
```

```
array([[925057,  58469],
       [127273, 454975]])
```

```
[ ] score
```

```
0.8719942980276847
```

+ Code + Text Connect GPU High-RAM Gemini

Logistic regression

```
[ ] from sklearn.linear_model import LogisticRegression
    classifier = LogisticRegression(random_state = 0)
    classifier.fit(X_train_sc, y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:432: FutureW
  FutureWarning)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
[ ] # 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)
```

```
[ ] cm
```

```
array([[983526,      0],
       [     0, 582248]])
```

```
[ ] score
```

```
1.0
```

Applying Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
objclassifierGNB=GaussianNB()
objclassifierGNB.fit(X_train_sc,y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
# 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)
```

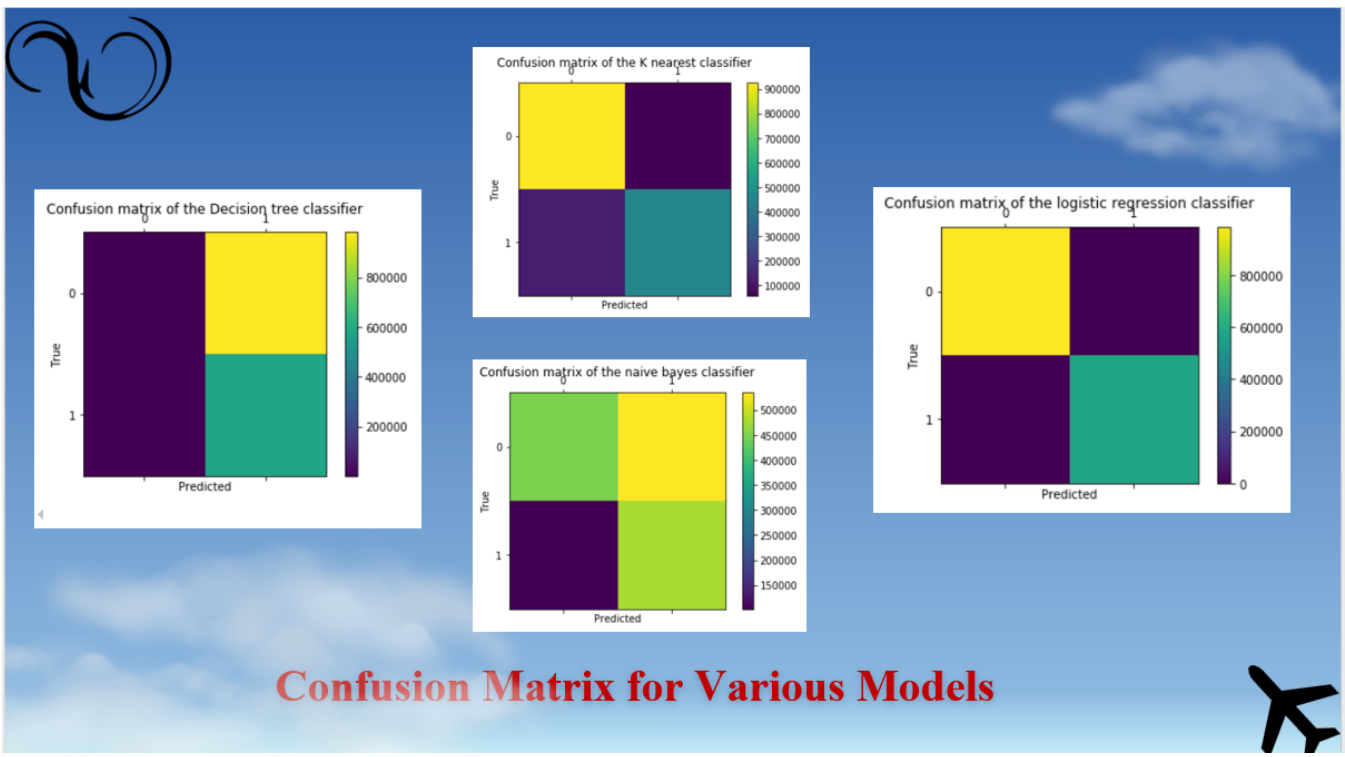
```
cm
```

```
array([[448999, 534527],
       [102122, 480126]])
```

```
score
```

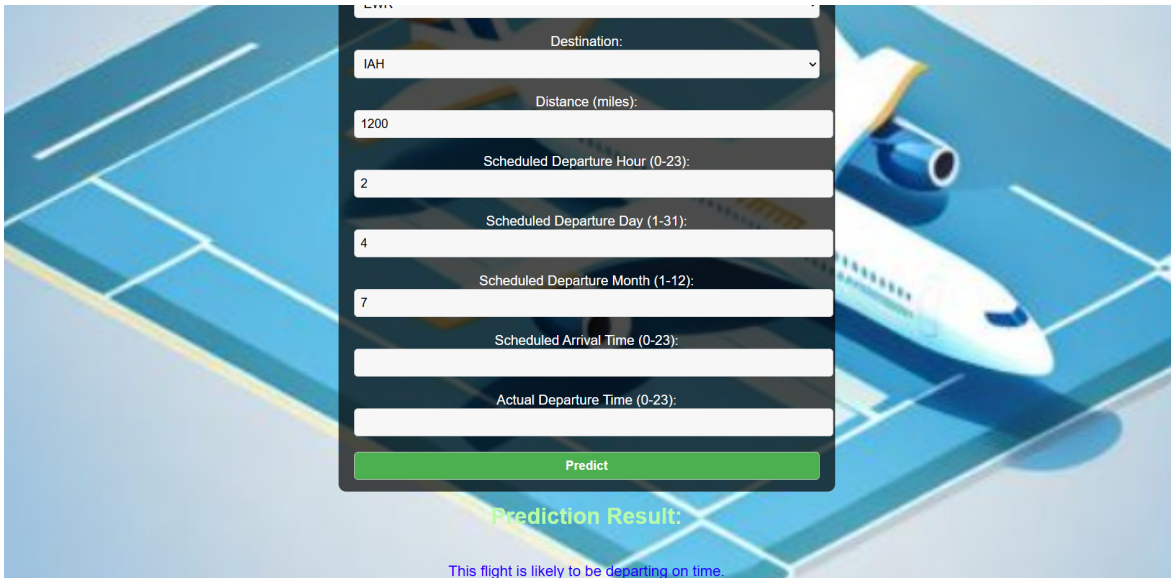
```
0.8399379476220706
```

```
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"))
```

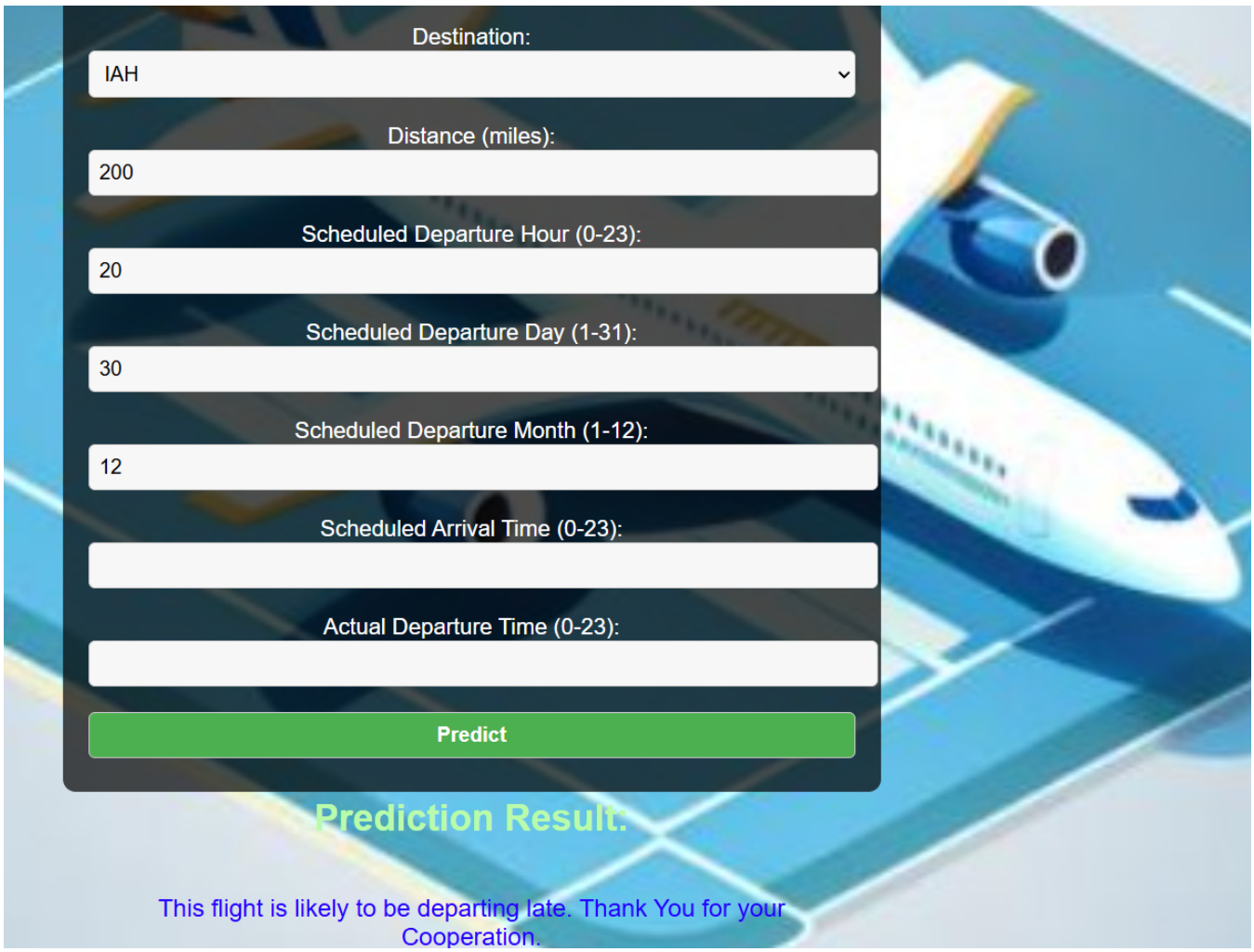


Outputs:

Departing on Time:



Departing Delay:



GitHub & Demo-link:

Github: https://github.com/Pandu-123-sree/Flight-Delays-Prediction-Using-Machine-Learning

Demo-Link: https://drive.google.com/file/d/1V-lRX7VV02cY8NGFIcAr2_enyj3GzjEj/view?usp=drive_link