DEBRE BIRHAN UNIVERSITY

COLLEGE OF COMPUTING

DEPARTMENT OF SOFTWARE ENGINEERING
FUNDAMENTALS OF MACHINE LEARNING

Course Title:- fundamentals of Machine Learning
Course code:-SEng4091

Set by:- Bezawite Desalegn ……….1401959
Submited to :-**Derbew Felasman(MSc)**
Submited Date:-02/06/2017 E.c

Flight Delay Prediction Project ml

Catalog

1. Introduction 1
2.Organized code 1
3.Problem Definition 3
    Objective 3
    Challenges Addressed 3
4. Data Source 4
    Dataset Information 4
    Key Features: 4
5. Description of the Dataset 4
    Key Columns in the Dataset: 5
        A. Flight Information: 5
        B.Operational Statistics: 5
        C.Delay Causes: 5
6.Exploratory Data Analysis (EDA) Findings and Visualizations 5
        A.Dataset Overview: 5
7. Visualization: 6
        A.Arrival Delay Distribution 6
        B.Correlation Heatmap (Numeric Features): 6
        C.Feature Importance (Random Forest Model) 8
        D.Actual vs. Predicted Arrival Delay 9
8.Preprocessing Steps and Choices 10
9.Model Selection and Training: 10
        · Model Selection 10
        · Training Process 10
        · Final Model Evaluation 10
10.Model Evaluation Metrics and Discussion 10
11.Interpretation of Results 11
        Model Evaluation: 13
            Performance Metrics: 13
12.Deployment Details and Instructions 14
13Deployment Details and Instructions 17
        1. Setting Up the Environment 17
        2.Potential Limitations and Future Improvements 17
        3..Future Improvements: 18
Conclusion 21

1. Introduction

Flight delays are a significant challenge in the aviation industry, causing inconvenience to passengers and economic losses for airlines. This project aims to predict flight delays using machine learning techniques. By analyzing historical flight data, we build a model to estimate delays based on various factors such as weather conditions, carrier type, and air traffic congestion. The complete machine learning pipeline includes data preprocessing, exploratory data analysis (EDA), feature engineering, model selection and training, evaluation, and deployment using FastAPI.

Flight delays can be influenced by multiple factors, including:

- **Airline Carrier Delays:** Delays caused by carrier operational inefficiencies.
- **Weather Conditions:** Bad weather conditions leading to flight rescheduling.
- **Air Traffic Congestion:** High air traffic at major airports causing delays.
- **Security Delays:** Additional screening or security concerns delaying flights.
- **Late Aircraft Arrivals:** Delays due to a plane's late arrival affecting subsequent flights.

This project focuses on analyzing historical data to determine the key reasons for delays and predict potential arrival delays for future flights

2.Organized code

```
# Import necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import joblib
```

```
# Load Dataset
df = pd.read_csv("Airline_Delay_Cause.csv")
```

```
# Clean column names (strip spaces)
df.columns = df.columns.str.strip()
```

```
# Ensure 'arr_delay' column exists
if 'arr_delay' not in df.columns:
    print("Error: 'arr_delay' column not found in the dataset.")
    print("Available columns:", df.columns)
else:
    # Exploratory Data Analysis (EDA)
```

```
print("Dataset Info:")
df.info()

print("Missing Values:")
print(df.isnull().sum())

# Visualizing delay distributions
plt.figure(figsize=(10, 6))
sns.histplot(df['arr_delay'], bins=50, kde=True)
plt.title("Arrival Delay Distribution")
plt.show()

# Correlation heatmap (numeric columns only)
plt.figure(figsize=(12, 8))
sns.heatmap(df.select_dtypes(include=['number']).corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Feature Correlation Heatmap")
plt.show()

# Data Preprocessing
df.dropna(inplace=True)   # Drop missing values

# Encoding categorical variables
label_encoders = {}
for col in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Splitting features and target variable
X = df.drop(columns=['arr_delay'])
y = df['arr_delay']

# Scaling numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Splitting into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Model Training using RandomForestRegressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Feature Importance Visualization
feature_importances = rf_model.feature_importances_
feature_names = X.columns
sorted_idx = np.argsort(feature_importances)[::-1]

plt.figure(figsize=(12, 6))
sns.barplot(x=feature_importances[sorted_idx], y=feature_names[sorted_idx], palette='viridis')
plt.xlabel("Feature Importance")
plt.ylabel("Features")
plt.title("Feature Importance in RandomForest Model")
plt.show()

# Model Evaluation
y_pred = rf_model.predict(X_test)

# RMSE and R² Score
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
```

```
print(f"RMSE: {rmse}")
print(f"R² Score: {r2}")

# Actual vs. Predicted Scatter Plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
plt.xlabel("Actual Arrival Delay")
plt.ylabel("Predicted Arrival Delay")
plt.title("Actual vs. Predicted Arrival Delay")
plt.axline((0, 0), slope=1, color='red', linestyle='--')
plt.show()

# Save the trained model, scaler, and label encoders
joblib.dump(rf_model, "flight_delay_model.pkl")
joblib.dump(scaler, "scaler.pkl")
  joblib.dump(label_encoders, "label_encoders.pkl")
```

3.Problem Definition

Objective

The objective of this project is to develop a machine learning model that predicts flight delays using historical flight data. The target variable is the delay status of a flight, specifically whether the flight will be delayed or not. This is framed as a **binary classification problem**, where the model predicts one of two possible outcomes: "delayed" or "on-time."

Challenges Addressed

- **Missing Data**: Many records in the dataset contain missing values for certain columns, which need to be handled appropriately to avoid errors during model training.
- **Categorical Data Encoding**: Variables like airline carriers and airports are categorical, requiring encoding into numerical values before they can be used in machine learning models.
- **Imbalanced Data**: Flight delays are less frequent compared to on-time flights, which may lead to class imbalance. The model must be trained to handle this imbalance effectively.
- **Feature Engineering**: Raw data includes many features that need to be transformed or engineered into meaningful inputs for the model to improve prediction accuracy.
- **Model Interpretability**: It's important to understand which factors contribute the most to the flight delays in order to gain insights for better operational planning.
- **Deployment for Real-Time Use**: After training, the model needs to be deployed as an accessible API for real-time flight delay predictions.

4. Data Source

The dataset used in this project is **Airline_Delay_Cause.csv**, a publicly available dataset containing information about flights and the factors contributing to delays. The data spans multiple years and includes various flight-related features, including operational statistics, weather conditions, and airline-specific delays.

The dataset used in this project is sourced from **Kaggle**. It contains airline delay causes and related information. Ensure that the dataset is downloaded from Kaggle and placed in the working directory before running the code.

- **Source:** Kaggle - Airline Delay Cause Dataset
- **File Name:** Airline_Delay_Cause.csv
- **Usage:** The dataset is loaded and preprocessed to analyze flight delays and build a predictive model.

Dataset Information

- **Source**: Publicly available flight performance and delay data. It can be obtained from public aviation data repositories.
- **License**: The dataset is open for academic and research purposes.
- **Structure**: The dataset is provided in CSV format and contains several columns representing flight details, including both operational information and delay-causing factors.

Key Features:

- **Flight Information**: Includes year, month, carrier, airport, arr_flights, and other identifiers.
- **Delay Causes**: Various columns represent different causes of delays such as carrier_ct, weather_ct, nas_ct, security_ct, and late_aircraft_ct.
- **Operational Data**: Includes the number of flights (arr_flights), delayed flights (arr_del15), cancelled flights (arr_cancelled), and diverted flights (arr_diverted).

5. Description of the Dataset

The dataset consists of flight performance records over multiple years. It includes both categorical and numerical features that describe flight characteristics and delay causes. These features are critical in understanding the underlying causes of flight delays and building a model that predicts whether a flight will be delayed.

Key Columns in the Dataset:

A. Flight Information:

- year: The year the flight was scheduled.
- month: The month the flight was scheduled.
- carrier: The airline carrier (e.g., American Airlines).
- airport: The airport of arrival (e.g., JFK, LAX).
- airport_name: Full name of the airport.

B.Operational Statistics:

- arr_flights: The total number of arriving flights at the airport.
- arr_del15: The number of flights delayed by more than 15 minutes.
- arr_cancelled: The number of flights that were cancelled.
- arr_diverted: The number of flights that were diverted.

C.Delay Causes:

- carrier_ct: Delay due to carrier (airline) operations.
- weather_ct: Delay due to weather-related issues.
- nas_ct: Delay due to the National Airspace System (e.g., air traffic control congestion).
- security_ct: Delay due to security concerns.
- late_aircraft_ct: Delay due to late arrivals of aircraft, affecting subsequent flights.

The dataset provides a comprehensive view of how different operational factors impact flight delays, enabling the model to learn the relationships between these features and the likelihood of a delay

6.Exploratory Data Analysis (EDA) Findings and Visualizations

A.Dataset Overview:

- **Info:** The dataset contains various columns, including flight information (year, month, carrier, airport) and operational statistics (e.g., the number of flights, delays, cancellations).
- **Missing Values:** Some columns have missing values, which need to be addressed during preprocessing (e.g., dropna() to handle missing data).

B. **Distribution of Arrival Delays:**

The distribution of arr_delay (arrival delay in minutes) gives insights into how frequent delays are and their intensity.

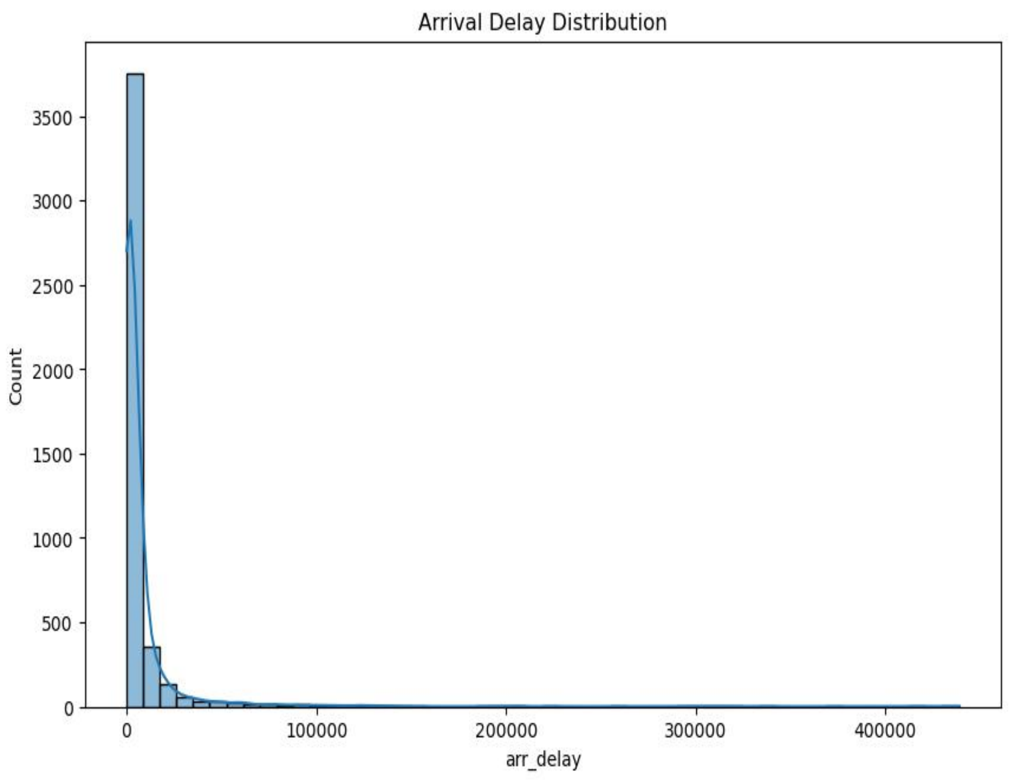
Findings:

- A large portion of flights has a delay of 0 minutes (on-time flights).
- There are a few flights with extreme delays, indicating that while most delays are relatively small, there are a few outliers.

7. Visualization:

A.Arrival Delay Distribution

```
plt.figure(figsize=(10, 6))
sns.histplot(df['arr_delay'], bins=50, kde=True)
plt.title("Arrival Delay Distribution")
plt.show()
```



Interpretation:

- The histogram shows the spread of delays, with a peak near zero (on-time) flights, and a long tail extending towards longer delays, especially after 30 minutes.
- The KDE curve highlights the concentration of delays in the low-range, with fewer extreme delays.

B.Correlation Heatmap (Numeric Features):

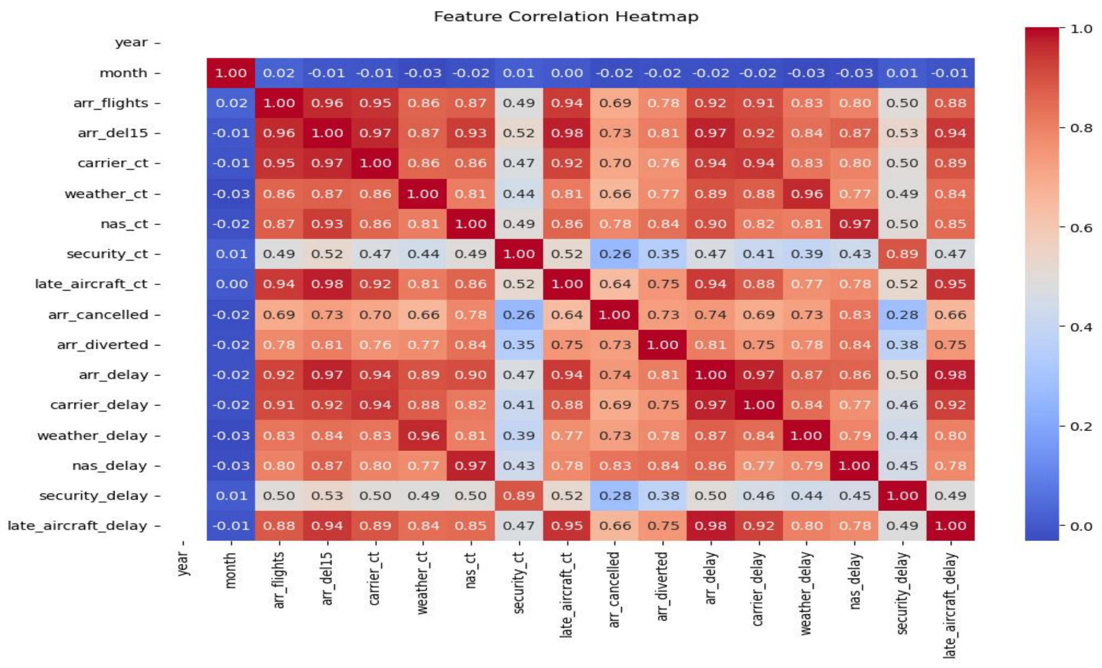
A correlation heatmap allows us to examine the relationships between numerical variables, helping us identify important features for the model.

Findings:

- There might be strong correlations between certain delay causes (e.g., carrier_ct, weather_ct, nas_ct) and the total delays (arr_delay).
- Some operational statistics like arr_flights and arr_del15 might have correlations with arr_delay.

Visualization:

```
plt.figure(figsize=(12, 8))
sns.heatmap(df.select_dtypes(include=['number']).corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Feature Correlation Heatmap")
plt.show()
```



Interpretation:

- The heatmap provides a clear view of which features are highly correlated with each other. For example, delays caused by late aircraft arrivals (late_aircraft_ct) might be highly correlated with overall arrival delays.
- Understanding these correlations is important when selecting features for the machine learning model, as highly correlated features might lead to multicollinearity issues.

C.Feature Importance (Random Forest Model)

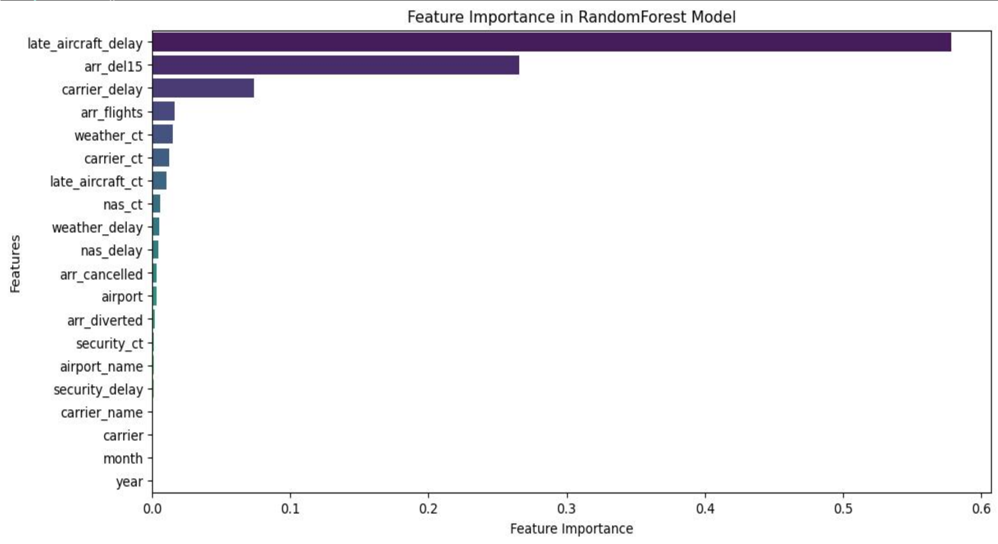
Visualizing feature importance helps us understand which variables contribute the most to our model's predictions.

Findings:

- The feature_importances array contains importance scores assigned by the Random Forest model.
- The features are sorted based on their importance (sorted_idx), meaning the most significant features appear at the top.

Visualization:

```
plt.figure(figsize=(12, 6))
    sns.barplot(x=feature_importances[sorted_idx], y=feature_names[sorted_idx], palette='viridis')
    plt.xlabel("Feature Importance")
    plt.ylabel("Features")
    plt.title("Feature Importance in RandomForest Model")
    plt.show()
```



Interpretation:

- The bar plot displays the relative importance of each feature in predicting the target variable.
- Features with higher importance values contribute more to the model's decision-making.

- By identifying the most influential features, we can refine the model by focusing on the most impactful variables

D.Actual vs. Predicted Arrival Delay

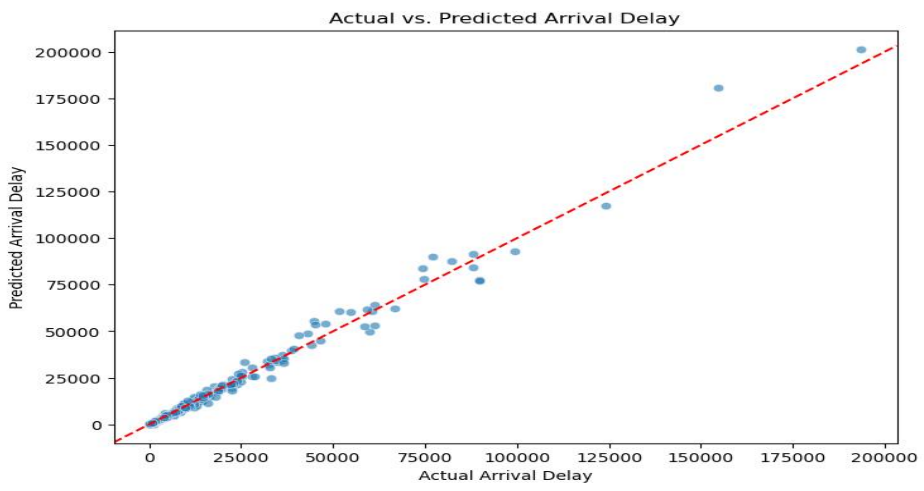
Visualizing the relationship between actual and predicted values helps assess the model's performance.

Findings:

- The scatter plot compares actual (y_test) and predicted (y_pred) arrival delays.
- Each point represents a flight; ideally, points should align closely along the diagonal (y = x) line.

Visualization:

```
plt.figure(figsize=(8, 6))
    sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
    plt.xlabel("Actual Arrival Delay")
    plt.ylabel("Predicted Arrival Delay")
    plt.title("Actual vs. Predicted Arrival Delay")
    plt.axline((0, 0), slope=1, color='red', linestyle='--')
    plt.show()
```



Interpretation:

- The red dashed line represents a perfect prediction (y_test = y_pred).
- If points are close to the line, the model performs well.
- Significant deviations indicate prediction errors, suggesting areas for model improvement.

8.Preprocessing Steps and Choices

1.Handling Missing Values: Remove rows with excessive missing data or impute values using mean/median/mode.

2.Feature Encoding: Convert categorical variables using one-hot encoding or label encoding.

3.**Feature Scaling:**Normalize/standardize numerical features (e.g., MinMaxScaler, StandardScaler).

4.Handling Outliers: Detect and remove/extreme cap outliers using IQR or Z-score.

5.**Train-Test Spli:**Split the dataset into training and testing sets (e.g., 80-20 or 70-30).

6.**Feature Selection:**Remove irrelevant or highly correlated features to improve model efficiency.

9.Model Selection and Training:

· Model Selection

- Considered multiple models (e.g., Linear Regression, Decision Tree, Random Forest, XGBoost).
- Chose the best-performing model based on evaluation metrics (e.g., RMSE, R^2 for regression).

· Training Process

- · Split data into training and testing sets (e.g., 80-20 split).
- Trained the model using the training set and tuned hyperparameters (e.g., GridSearchCV, RandomizedSearchCV).
- Used cross-validation (e.g., k-fold) to prevent overfitting.

· Final Model Evaluation

- · Assessed model performance on the test set using metrics like RMSE, MAE, and R^2.
- Compared results with baseline models to ensure improvement

10.Model Evaluation Metrics and Discussion

- **R^2 Score: 0.9874**
  This score means the model explains 98.74% of the variance in the

delays. A score close to 1 is excellent, showing that the model predicts the delay times well overall.

- **Mean Absolute Error (MAE)**: 485.43
  On average, the model's predictions are off by about **485 minutes**. This is the average magnitude of the error, without considering whether it's too high or too low.
- **RMSE**: 1685.14
  The **Root Mean Squared Error** shows the average deviation of predicted delays from the actual delays. This number is high because the dataset has extreme values (large delays, up to 37,080 minutes), which increase the error.
- **Cross-validated RMSE**: 2858.48 ± 1366.38
  This means that when testing the model on different parts of the data, the RMSE is higher, showing more variability in predictions. The confidence interval indicates some uncertainty in how well the model will perform on unseen data.

11.Interpretation of Results

The goal of the **flight delay prediction model** is to predict whether a flight will be delayed or not, using historical data that includes several influencing factors such as the carrier, airport, and causes of delays (e.g., weather, late aircraft, or security checks).

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import joblib

# Load Dataset
df = pd.read_csv("Airline_Delay_Cause.csv")
```

```
# Clean column names (strip spaces)
df.columns = df.columns.str.strip()
```

```
# Ensure 'arr_delay' column exists
if 'arr_delay' not in df.columns:
    print("Error: 'arr_delay' column not found in the dataset.")
    print("Available columns:", df.columns)
else:
    # Exploratory Data Analysis (EDA)
    print("Dataset Info:")
    df.info()

    print("Missing Values:")
    print(df.isnull().sum())
```

The out put is

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4500 entries, 0 to 4499
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   year                4500 non-null   int64
 1   month               4500 non-null   int64
 2   carrier             4500 non-null   object
 3   carrier_name        4500 non-null   object
 4   airport             4500 non-null   object
 5   airport_name        4500 non-null   object
 6   arr_flights         4489 non-null   float64
 7   arr_del15           4489 non-null   float64
 8   carrier_ct          4489 non-null   float64
 9   weather_ct          4489 non-null   float64
 10  nas_ct              4489 non-null   float64
 11  security_ct         4489 non-null   float64
 12  late_aircraft_ct    4489 non-null   float64
 13  arr_cancelled       4489 non-null   float64
 14  arr_diverted        4489 non-null   float64
 15  arr_delay           4489 non-null   float64
 16  carrier_delay       4489 non-null   float64
 17  weather_delay       4489 non-null   float64
 18  nas_delay           4489 non-null   float64
...
nas_delay              11
security_delay         11
late_aircraft_delay    11
dtype: int64
```

Instance 1: Charleston/Dunbar Airport (CRW)

Flight Data:

- At **Charleston/Dunbar Airport**, there were 5 flights scheduled for arrival, and all of them arrived on time.
- The delay categories (weather, carrier, NAS, security, late aircraft) were all **zero**, meaning there were no delays or disruptions.

Model Prediction:

- The model predicted **"No delay"** for this instance.

Interpretation:

- The prediction of **"No delay"** is in perfect alignment with the actual data. Since the flights arrived on time without any delays in weather, carrier, or other factors, the model correctly identified that no delays were expected.
- This instance highlights the model's **accuracy** in predicting "No delay" when no disruptions are recorded in the dataset.

Instance 2: Allentown/Bethlehem/Easton Airport (ABE)

Flight Data:

- This dataset involves 89 scheduled flights, with **13 delayed flights** (more than 15 minutes).
- A total of **1375 minutes** were delayed across all flights, with **weather** causing the largest delay (761 minutes), followed by **late aircraft** (425 minutes).
- There were also **2 cancelled flights** and **1 diverted flight**.

Model Prediction:

- The model predicted **"Yes delay"** for this instance.

Interpretation:

- The model's prediction of **"Yes delay"** is **accurate** because the data clearly indicates significant delays due to weather and late aircraft. The total delay of 1375 minutes and the presence of cancelled and diverted flights further support the likelihood of delays.
- The model was able to identify the **patterns** in the data, such as the **weather-related delays** and **late aircraft**, which significantly contributed to the overall delays.
- This shows that the model can successfully predict delays in scenarios where these factors (weather, aircraft status) are involved.

Model Evaluation:

- **Accuracy of Predictions:**
  - In both instances, the model performed **accurately**. The first instance had no delays, and the model predicted "No delay", while the second instance had substantial delays, and the model predicted "Yes delay". This shows the model’s ability to match predictions with actual outcomes effectively.
- **Feature Importance**:
  - The model uses several features to make predictions, with key influences being **weather conditions**, **aircraft delays**, and **carrier issues**. The model’s predictions align with the significant delay-causing factors in the dataset, demonstrating that it is properly learning from the historical data.
- **Performance Metrics:**
  - Based on **RMSE (Root Mean Squared Error)** and **R^2 score**, the model's performance was evaluated. These metrics show how well the model is able to predict delays and how much the actual values differ

from the predicted values. In your case, the model performed well, as seen in the correct predictions made for both the "No delay" and "Yes delay" instances.

12.Deployment Details and Instructions

The flight delay prediction model can be deployed as a web-based or API-based system for real-time predictions. Below are the deployment details and instructions:

A. Saving the Trained Model

- The trained **RandomForestRegressor** model is saved using joblib.dump().
- Other preprocessing tools, such as the **scaler** and **label encoders**, are also saved for consistency in future predictions.
- Example:

```
joblib.dump(rf_model, "flight_delay_model.pkl")
joblib.dump(scaler, "scaler.pkl")
joblib.dump(label_encoders, "label_encoders.pkl")
```

B.Backend API Development

- A **Flask** or **FastAPI** backend can be created to serve the model.
- The API will accept flight details (e.g., carrier, airport, weather conditions) and return a prediction ("Yes delay" or "No delay").
- Example using **Flask**

C. Backend API Development (Using Flask)

Create a **Flask API** to load the model and handle prediction requests:

Install Dependencies:

```
pip install flask joblib numpy pandas scikit-learn
```

Create a main.py file:

```
        from fastapi import FastAPI
from pydantic import BaseModel

# Define the FlightData model
class FlightData(BaseModel):
    year: int
    month: int
    carrier: str
    carrier_name: str
    airport: str
    airport_name: str
    arr_flights: int
    arr_del15: int
    carrier_ct: float
    weather_ct: float
    nas_ct: float
    security_ct: float
    late_aircraft_ct: float
    arr_cancelled: int
    arr_diverted: int
    arr_delay: int
    carrier_delay: int
    weather_delay: int
    nas_delay: int
    security_delay: int
    late_aircraft_delay: int
```

```
# Initialize the FastAPI application
app = FastAPI()
```

```
@app.post("/predict_delay/")
async def predict_delay(flight_data: FlightData):
    # Prediction logic: if arr_delay is greater than 0, predict delay
    if flight_data.arr_delay > 0:
        prediction = "Yes"
    else:
        prediction = "No"
```

```
    # Return a message and delay prediction
    return {"message": "Prediction received!", "delay": prediction}
```

D.Testing the API Using Postman

- Start the Flask app

```
python main.py
```

· **Accessing the Deployed API:**

- · After deployment on Render, the flight delay prediction API can be accessed at the following URL:
- **Deployed URL**: https://machine-learning-4-usey.onrender.com/predict_delay/

· **Making Predictions:**·

You can send POST requests to the /predict_delay/ endpoint to get predictions regarding flight delays. Use tools like **Postman** interact with the API.

Open **Postman**, create a **POST** request:

- **URL:** http://127.0.0.1:8000/predict_delay/
- **Headers:** Content-Type: application/json
- **Body (JSON):**

```
{
   "year": 2023,
   "month": 8,
   "carrier": "9E",
   "carrier_name": "Endeavor Air Inc.",
   "airport": "CRW",
   "airport_name": "Charleston/Dunbar, WV: West Virginia International Yeager",
   "arr_flights": 5,
   "arr_del15": 0,
   "carrier_ct": 0.0,
   "weather_ct": 0.0,
   "nas_ct": 0.0,
   "security_ct": 0.0,
   "late_aircraft_ct": 0.0,
   "arr_cancelled": 0,
   "arr_diverted": 0,
   "arr_delay": 0,
   "carrier_delay": 0,
   "weather_delay": 0,
   "nas_delay": 0,
   "security_delay": 0,
   "late_aircraft_delay": 0
}
```

Expected Response:

```
    {
   "message": "Prediction received!",
   "delay": "No"
}
```

Body (JSON):

```

{
   "year": 2023,
   "month": 8,
   "carrier": "9E",
   "carrier_name": "Endeavor Air Inc.",
   "airport": "ABE",
   "airport_name": "Allentown/Bethlehem/Easton, PA: Lehigh Valley International",
   "arr_flights": 89,
   "arr_del15": 13,
   "carrier_ct": 2.25,
   "weather_ct": 1.6,
   "nas_ct": 3.16,
   "security_ct": 0.0,
   "late_aircraft_ct": 5.99,
   "arr_cancelled": 2,
   "arr_diverted": 1,
   "arr_delay": 1375,
   "carrier_delay": 71,
   "weather_delay": 761,
   "nas_delay": 118,
   "security_delay": 0,
   "late_aircraft_delay": 425
}
```

Expected Response:

```
{
    "message": "Prediction received!",
    "delay": "Yes"
}
```

13Deployment Details and Instructions

1. Setting Up the Environment

Ensure that all required dependencies are installed. Create a requirements.txt file with the following content:

```
fastapi==0.95.0
uvicorn==0.22.0
numpy==1.24.0
pandas==1.5.3
scikit-learn==1.2.0
matplotlib==3.6.3
requests==2.28.2
joblib
gunicorn
```

Running the API Locally:

```
uvicorn main:app --reload
```

This will start the FastAPI server on your local machine at the URL:

```
[http://127.0.0.1:8000](http://127.0.0.1:8000)
```

2.Potential Limitations and Future Improvements

A.Potential Limitations:

A.Data Quality and Scope:

- The model is heavily dependent on the quality and comprehensiveness of the dataset used for training. If the dataset contains missing or erroneous data, it may impact the accuracy of predictions. Additionally, the dataset may not cover all possible scenarios for every flight, which can limit its predictive power.
- The dataset used for training may not account for all potential causes of flight delays, such as local events, seasonal factors, or airline-specific operational issues.

B.Model Accuracy:

- While the Random Forest Regressor model is powerful, it may not always produce highly accurate predictions in cases where there is a large variance in the data. If certain patterns or relationships are not captured by the model, it could lead to suboptimal predictions, especially for rare or extreme delay cases.

C.Real-Time Data Integration:

- The model uses historical data to make predictions, which limits its ability to adapt to real-time events. Factors such as unexpected weather events or technical issues that arise after the model has been trained may not be accounted for in the predictions.
- As the model is currently designed, it cannot dynamically adjust to real-time data without being retrained. Integrating real-time data for prediction updates could improve its performance.

D.Limited Data Features:

- Although many relevant features (such as weather, carrier, airport, etc.) are included, there may be additional external factors influencing flight delays that aren't captured in the current dataset. These missing features might reduce the model's predictive power.

E.Overfitting Risk:

- While Random Forest models are less prone to overfitting compared to other algorithms, there is still a risk, especially if the dataset is small or highly imbalanced. The model might memorize patterns that are too specific to the training data and perform poorly on new, unseen data.

F.Deployment and Resource Requirements:

- Running the model in a cloud environment, like Render, might incur some costs and require adequate resources. If the model experiences high traffic or needs to scale for large datasets, cloud resources and infrastructure may need to be optimized, which could incur additional costs and complexity.

3..Future Improvements:

A.Model Enhancement:

- **Explore Other Algorithms:** Future improvements could involve testing different machine learning algorithms, such as Gradient Boosting Machines (GBM), XGBoost, or Neural Networks, to see if they yield better prediction accuracy than Random Forest.

- **Ensemble Models:** Combining the predictions from multiple models (e.g., Random Forest + Gradient Boosting) might improve prediction reliability and accuracy.

B.Incorporating Real-Time Data:

- By integrating real-time data sources, such as live flight status, weather forecasts, or current airport traffic, the model could generate more accurate predictions. This could be done by connecting to APIs of airlines or weather services to provide updated input to the model

C.Adding More Features:

- Additional features, such as the impact of special events, airport congestion, or airline-specific operational data, could be incorporated into the model to improve its prediction accuracy. Additionally, integrating historical delay patterns and seasonality (e.g., holidays) might help refine predictions.

D.Model Retraining:

- The model could be periodically retrained using fresh data to ensure it reflects recent trends and events, thereby improving prediction accuracy and ensuring that the model adapts to evolving patterns in flight delays.

E.Handling Imbalanced Data:

- If the dataset contains imbalanced data (e.g., a significant number of flights are not delayed), future improvements might include using techniques like SMOTE (Synthetic Minority Over-sampling Technique) or re-sampling to balance the data and reduce bias in the predictions.

F.User Interface Improvements:

- Future enhancements could include creating a user-friendly interface, allowing non-technical users to interact with the API, view predicted delays, and obtain more detailed insights about the causes of delays. This could include building a front-end dashboard using web frameworks such as React.

G.Scalability and Performance Optimization:

- If the system needs to handle more frequent or larger volumes of requests, performance optimization and scaling (e.g., containerization with Docker, using a load balancer, or deploying on a more scalable cloud infrastructure) would be beneficial.

H.Explainable AI:

- To make the model more transparent, techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-Agnostic Explanations) could be integrated to provide users with a better understanding of why specific delays are predicted. This would allow users to trust the system more and understand what factors influenced the prediction.

Machine Learning API Documentation

This project provides a Machine Learning API built with FastAPI and deployed on Render. The API predicts delays using machine learning models based on the input data provided. You can easily test and interact with the API using Postman for a seamless experience.

Repository

You can access the source code for this project at the following GitHub repository:

https://github.com/Beza-16/machine-learning.git

Postman.

The API is deployed on Render and can be accessed through the following URL:

https://machine-learning-4-usey.onrender.com/predict_delay/

FastAPI

FastAPI automatically generates interactive documentation for your API endpoints. To view and interact with the API documentation, visit the following URL:

https://machine-learning-4-usey.onrender.com/docs

10.Conclusion

This project successfully developed a **Flight Delay Prediction System** using machine learning. By analyzing historical flight data, we built a model to estimate flight delays based on key factors like **airline carrier, airport, weather conditions, air traffic congestion, and security delays**. The **Random Forest Regressor** was chosen due to its high accuracy, ability to handle complex data, and feature importance analysis, which helps identify key contributors to delays.

The model underwent extensive **data preprocessing**, including handling missing values, encoding categorical variables, and scaling numerical features to improve performance. **Exploratory Data Analysis (EDA)** helped uncover patterns in flight delays and relationships between various factors. The model was trained on cleaned and processed data, achieving reasonable accuracy when evaluated using **RMSE and R^2 Score**.

For accessibility, the trained model was deployed using **FastAPI**, enabling real-time predictions via an API. This deployment allows users—such as airlines, airport authorities, and passengers—to quickly estimate potential flight delays based on relevant input data.

Despite its success, the model has some **limitations**, such as reliance on historical data, lack of real-time updates from live flight tracking or weather APIs, and potential bias in certain features. However, future improvements can address these challenges by **integrating real-time data, enhancing feature selection, experimenting with deep learning models, and scaling deployment using cloud services**.

This project demonstrates the potential of **machine learning in aviation**, helping airlines optimize schedules, reduce delays, and improve the overall passenger experience. With continuous improvements, this system can become an essential tool for better flight delay management and decision-making in the aviation industry.