National College of Ireland

Exploring the Impact of Weather on Flight Delays

MSc Research Project

Research Practicum part-2

Vasanth Babu Kassa

Student ID: 23408936
School of Computing

National College of Ireland
Supervisor: Hicham Rifai

National College of Ireland

MSc Project Submission

Sheet School of Computing

Student Name: Vasanth Babu Kassa …….
……
……

Student ID: ……23408936…… **Programme:**

……MSC_DAD_B_JAN_25…… **Year:** …Jan 2025……..

Module: Research Practicum part-2……

Lecturer: ……

Submission Due Date: ……

Project Title: ……Exploring the Impact of Weather on Flight Delays……

Word Count: ……602…… **Page Count:** ……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: **vasanth k**……

Date: **10-12-25**……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Exploring the Impact of Weather on Flight Delays

Vasanth Kassa

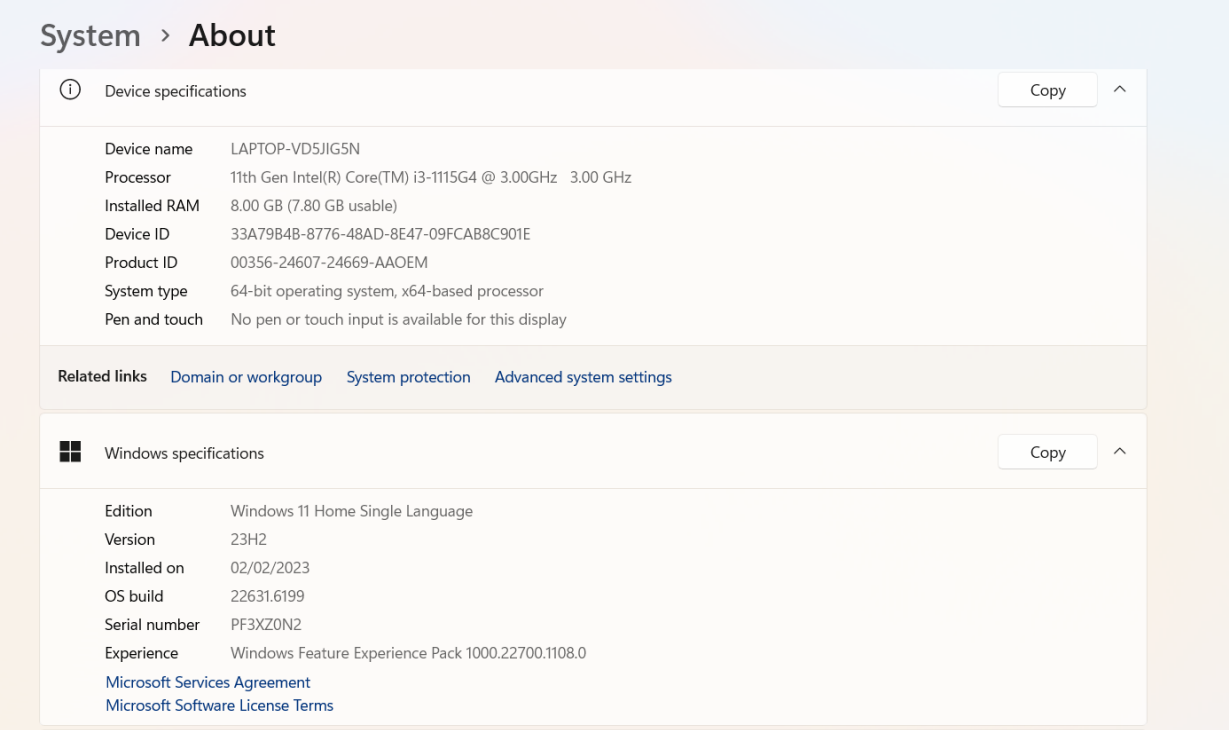
23408936

1. Introduction

The paper is a configuration manual that provides technical guidelines on the implementation of the flight delay prediction system that was developed in the dissertation, Exploring the Impact of Weather on Flight Delays. It specifies the software requirements, data preprocessing, model configuration settings, and deployment guidelines of the XGBoost-based prediction system with 98.45% accuracy.

2. System Configuration

It is running on Windows 11 Home (Build 22631.6199) and 11 th -11th-generation Intel Core i3-1115G4 processor (3.00GHz, 8.00GB RAM). The 64-bit architecture is efficient in processing huge data sets. The model execution and the data handling require Python 3.8 and the libraries scikit-learn, XGBoost, pandas, and NumPy.



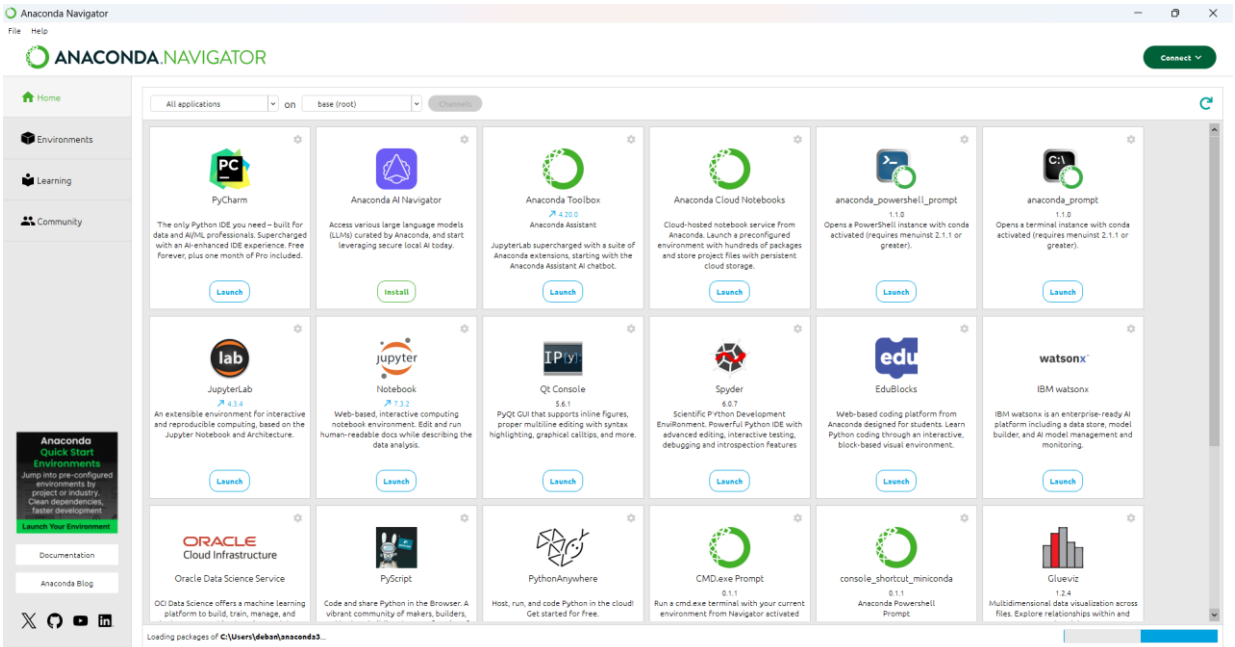
2.1 Recommended Hardware

Component	Recommended
GPU	NVIDIA GeForce GTX 1650 or higher (4GB VRAM) for accelerated model training
RAM	16GB DDR4 (minimum 8GB for basic operations)
CPU	Intel Core i5 11th Gen or AMD Ryzen 5 (3.0GHz+, quad-core minimum)
Storage	256GB SSD with at least 50GB free space for datasets and model files

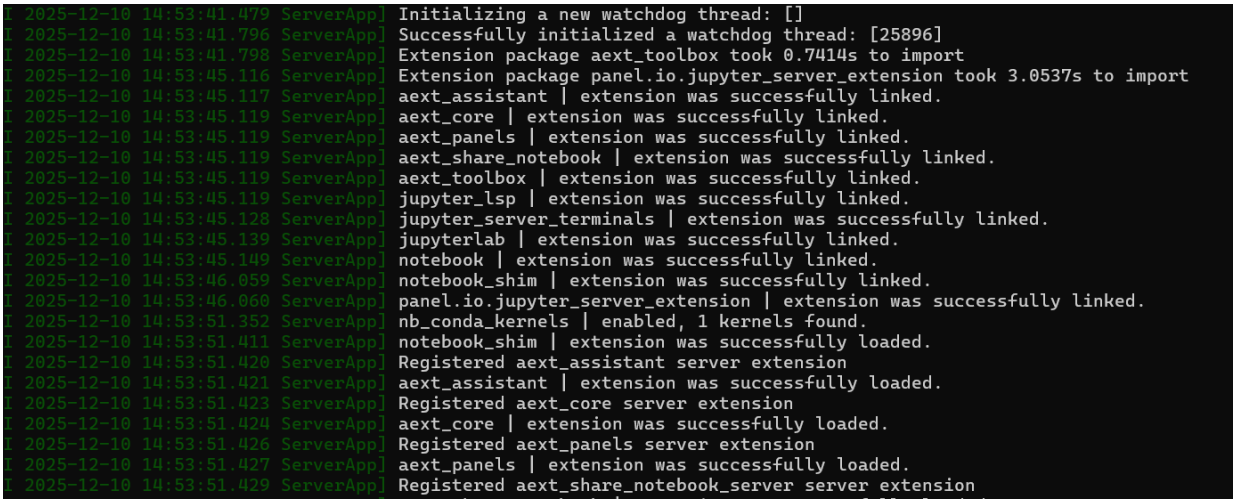
2.2 Software Requirements

Operating System

- Windows 10/11
- Anaconda Navigator
- Jupyter Notebook 7.2.2
- macOS
- Linux (Ubuntu highly recommended)



2.3 Python Version



Python 3.10 **or** 3.11 recommended

2.4 Required Libraries

The project uses the following major libraries:

Category	Libraries Used
Data Manipulation	pandas 1.5.3, NumPy 1.24.3
Machine Learning	scikit-learn 1.2.2, XGBoost 1.7.5, imbalanced-learn 0.10.1
Visualization	Matplotlib 3.7.1, Seaborn 0.12.2
Data Processing	SciPy 1.10.1
Development Environment	Jupyter Notebook 6.5.4, Python 3.8+

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    accuracy_score, classification_report, confusion_matrix,
    roc_auc_score, RocCurveDisplay, ConfusionMatrixDisplay
)
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

3. Data Sources & Linking

The data exploration stage demonstrates the structure of the data set, which consists of 1,936,758 flight records over 30 running features, providing the basis of the analytical process of predicting weather delays. The data is selected from Kaggle

```
# Inserting Dataset
df = pd.read_csv('DelayedFlights.csv')
```

Exploring Insights of Data

```
# First Few Rows
df.head(5)
```

	Unnamed: 0	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	...	TaxiIn	TaxiOut	Cancelled	CancellationCode
0	0	2008	1	3	4	2003.0	1955	2211.0	2225	WN	...	4.0	8.0	0	N
1	1	2008	1	3	4	754.0	735	1002.0	1000	WN	...	5.0	10.0	0	N
2	2	2008	1	3	4	628.0	620	804.0	750	WN	...	3.0	17.0	0	N
3	4	2008	1	3	4	1829.0	1755	1959.0	1925	WN	...	3.0	10.0	0	N
4	5	2008	1	3	4	1940.0	1915	2121.0	2110	WN	...	4.0	10.0	0	N

5 rows × 30 columns

4. Data Cleaning & Feature Engineering

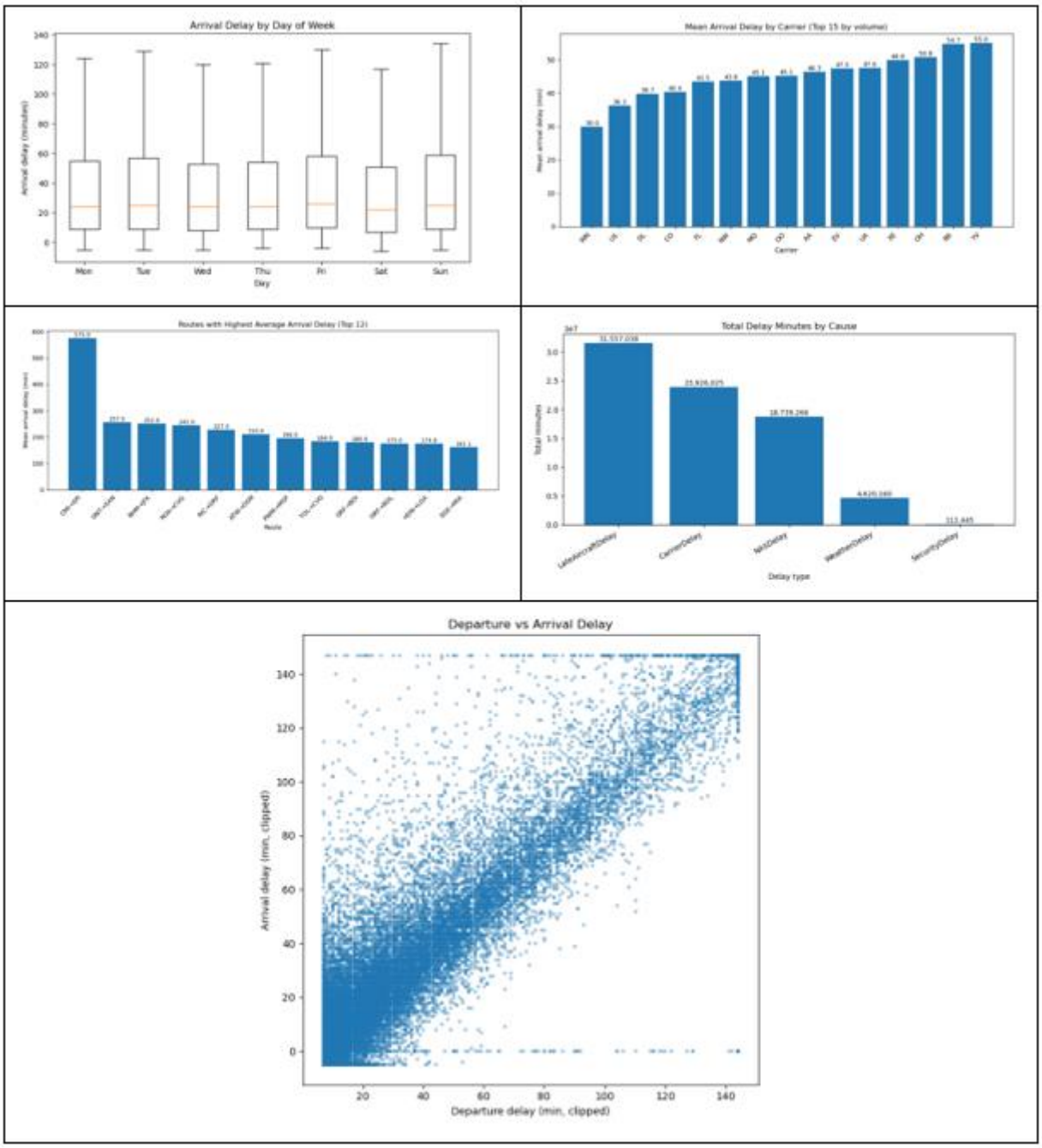
The missing value analysis shows that there is a great existing gap in delay-related columns (689,270 nulls). The systematic strategies in dealing with the data are the dropping of the unnamed index column, filling the gaps of missing data with suitable statistical indicators, and the elimination of duplicates in order to have 1,936,756 valid cases.

```
import pandas as pd

# Drop the 'Unnamed' column
df = df.drop(columns=['Unnamed: 0'], errors='ignore')

# Handling Null Values

# 1. TailNum → very few missing (5) → fill with 'UNKNOWN'
df['TailNum'] = df['TailNum'].fillna('UNKNOWN')

# 2. ArrTime, TaxiIn → missing values likely mean cancelled/diverted → fill with 0
df['ArrTime'] = df['ArrTime'].fillna(0)
df['TaxiIn'] = df['TaxiIn'].fillna(0)

# 3. ActualElapsedTime, AirTime, ArrDelay → missing due to cancellation/diversion → fill with 0
df['ActualElapsedTime'] = df['ActualElapsedTime'].fillna(0)
df['AirTime'] = df['AirTime'].fillna(0)
df['ArrDelay'] = df['ArrDelay'].fillna(0)

# 4. CRSElapsedTime → small number missing (198) → impute with mean
df['CRSElapsedTime'] = df['CRSElapsedTime'].fillna(df['CRSElapsedTime'].mean())

# 5. TaxiOut → some missing → fill with median (robust to outliers)
df['TaxiOut'] = df['TaxiOut'].fillna(df['TaxiOut'].median())

# 6. Delay columns (CarrierDelay, WeatherDelay, NASDelay, SecurityDelay, LateAircraftDelay)
delay_cols = ['CarrierDelay','WeatherDelay','NASDelay','SecurityDelay','LateAircraftDelay']
df[delay_cols] = df[delay_cols].fillna(0)
```

5. Data Visualization

Box plots illustrate distributions of arrival delays on weekdays with consistent median

distributions, whereas bar charts illustrate the delay hours and the main causes of the delays, with carrier delays as the largest. The scatter plot indicates that there is a positive correlation between the delays on the departure and arrival sides and which is high, and this means there is delay propagation



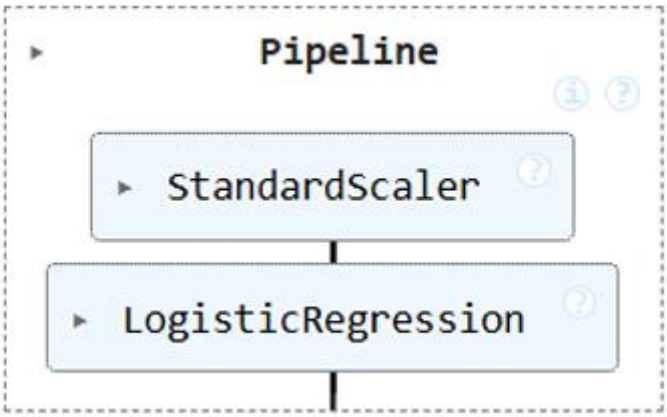
6. Data Preprocessing

Label encoding transforms the categorical variables (UniqueCarrier, Origin, Dest, CancellationCode) into numerical form, which allows the algorithm to process the set of categorical relationships. The step displays a dichotomous target variable, such as WeatherDelayIndicator, which transforms continuous WeatherDelay scores into classification categories (no delay 0 and delay 1).

```
# Identify categorical columns (object or category dtype)
cat_cols = df.select_dtypes(include=['object', 'category']).columns

# Initialize label encoder
le = LabelEncoder()

# Apply label encoding column by column
for col in cat_cols:
    df[col] = le.fit_transform(df[col].astype(str))

print("Categorical columns encoded:\n", cat_cols.tolist())
print(df.head())
```

7. Logistic Regression

The Logistic Regression has been found to have an accuracy of 93.70% and ROC-AUC of 0.96, which implies its good discriminative performance. The confusion matrix shows that there are 3469 true negatives and 161 true positives, 207 false positives, and 37 false negatives.



8. Evaluation Metrics

Accuracy percentage, precision, recall, F1-score are used as evaluation matrices for each ML models.

```
# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred_xgb, digits=3))
```

```
Classification Report:
              precision    recall  f1-score   support
```

9. Train–Test Split

The optimal technique is split into an 80-20 train-test split with data divided into training (80% samples) and a testing set (20%) randomly with random_state=42

```
# Features (X) and Target (y)
X = df_subsample.drop(columns=["WeatherDelayIndicator"])
y = df_subsample["WeatherDelayIndicator"]
```

```
# Train-Test Split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print("Train shape:", X_train.shape, y_train.shape)
print("Test shape:", X_test.shape, y_test.shape)
```

10. Model Comparison

The high number of measures explained by XGBoost as a better choice makes the effective decisions of flight delays by weather prediction, which presents practitioners with a solid and stable predictive tool to improve airline services.

```python
# Evaluate on the untouched test set
rows = []
for name, model in models.items():
    if not hasattr(model, "predict"):
        raise ValueError(f"Model {name} is not a fitted estimator.")
    y_pred = model.predict(X_test)

    acc  = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, average="binary", zero_division=0)
    rec  = recall_score(y_test, y_pred, average="binary", zero_division=0)
    f1   = f1_score(y_test, y_pred, average="binary", zero_division=0)

    rows.append({
        "Model": name,
        "Accuracy (%)": 100*acc,
        "Precision (%)": 100*prec,
        "Recall (%)": 100*rec,
        "F1-score (%)": 100*f1
    })

# Rank by Accuracy (%)
summary_df = (
    pd.DataFrame(rows)
    .sort_values(by=["Accuracy (%)"], ascending=False)
    .reset_index(drop=True)
)

# Display the table
print(summary_df.to_string(index=False, formatters={
    "Accuracy (%)":  lambda x: f"{x:.2f}",
    "Precision (%)": lambda x: f"{x:.2f}",
    "Recall (%)":    lambda x: f"{x:.2f}",
    "F1-score (%)":  lambda x: f"{x:.2f}"
}))
```

LR:

```
# Pipeline: Scale -> Logistic Regression
logreg_pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("logreg", LogisticRegression(max_iter=2000, solver="lbfgs"))
])

# Fit on SMOTE-resampled training data
logreg_pipe.fit(X_train_res, y_train_res)
```

SVC:

```
# Pipeline: Scale -> SVM (RBF)
svm_pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("svm", SVC(kernel="rbf", C=1.0, gamma="scale", probability=True, random_state=42))
])

# Fit on SMOTE-resampled training data
svm_pipe.fit(X_train_res, y_train_res)
```

RF Classifier:

```
#  Pipeline:-> Random Forest
rf_pipe = Pipeline([
    ("rf", RandomForestClassifier(
        n_estimators=300,
        max_depth=None,
        min_samples_split=2,
        min_samples_leaf=1,
        n_jobs=-1,
        random_state=42
    ))
])

# Train on SMOTE-resampled data
rf_pipe.fit(X_train_res, y_train_res)
```

XGB Classifier:

```
# Pipeline: (no scaler needed) -> XGBoost
xgb_pipe = Pipeline([
    ("xgb", XGBClassifier(
        n_estimators=400,
        max_depth=6,
        learning_rate=0.08,
        subsample=0.9,
        colsample_bytree=0.9,
        reg_lambda=1.0,
        objective="binary:logistic",
        eval_metric="logloss",
        n_jobs=-1,
        random_state=42,
        tree_method="hist"
    ))
])

# Train on SMOTE-resampled data
xgb_pipe.fit(X_train_res, y_train_res)
```

Repositories:

Kaggle:

Data Source: https://www.kaggle.com/

Dataset Link: https://www.kaggle.com/datasets/giovamata/airlinedelaycauses/data

References

Python: https://www.python.org