Predicting Flight Delays: How Weather and Seasons Affect Air Travel?

MSc Research Project
Data Analytics

Aniket Guru
Student ID: X22119914
x22119914@student.ncirl.ie

School of Computing
National College of Ireland

Supervisor: Bharat Agarwal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aniket Guru
Student ID: x22119914@student.ncirl.ie	X22119914
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Bharat Agarwal
Submission Due Date:	14/12/2023
Project Title:	Predicting Flight Delays: How Weather and Seasons Affect Air Travel?
Word Count:	1211
Page Count:	9

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:	Aniket Guru
Date:	14th December 2023

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):	

Predicting Flight Delays: How Weather and Seasons Affect Air Travel?

Aniket Guru
X22119914
x22119914@student.ncirl.ie

1 Introduction

In this configuration manual a detailed procedure used in achieving “Predicting Flight Delays: The effect of weather and seasons on air travel, including a description for how we ask it.” It involves detailed guidelines regarding hardware and software needs, data origin, environment description and modeling methods employed.

2 System Specification

For modelling and evaluation in this project, Google Colab is used with its powerful compute power and collaboration features. However, Jupyter Notebook remains the most important tool for data preparation and exploration that involves manual data manipulation and plotting. A system specification is essentially a detailed guide that

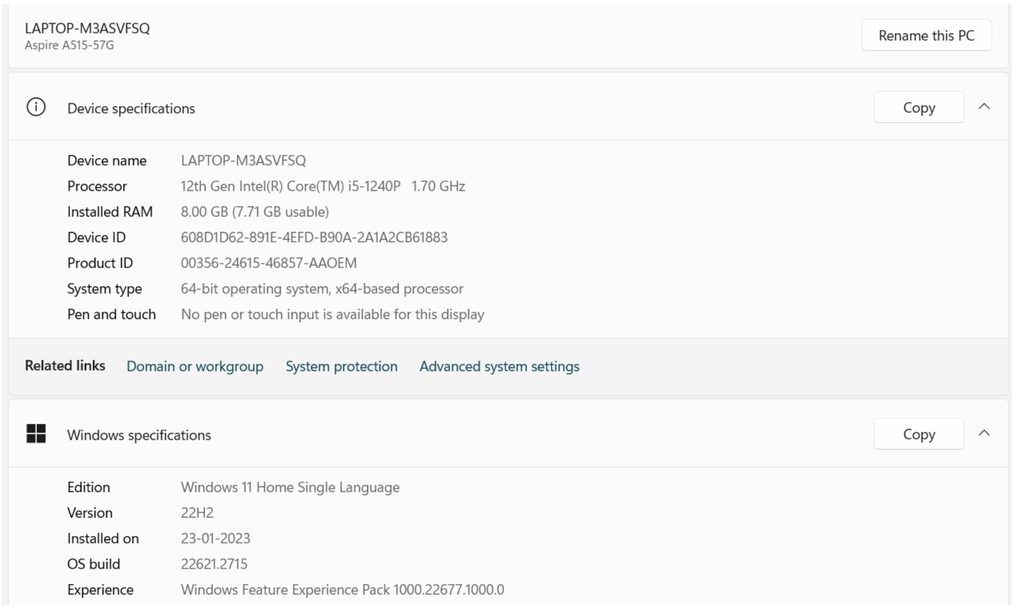


Figure 1: System Configuration

spells out the technical aspects and requirements of a system. It typically includes details about the components of the system, how it operates, its design, and various other technical details. Figure 1 illustrates the setup of the system that was employed for this project, and Tab 1shows the specifics of the Google Colab setup we used.

Resource	**Total Available**
System RAM	12.7 GB
GPU RAM	15.0 GB
Disk	78.2 GB

Table 1: Google Colab System Configuration

3 Data Collection

The project incorporates data from three distinct sources :

1. The "On-Time Flight" dataset, detailing departures from JFK airport, is referenced from Bureau of Transportation Statistics (2023).
2. Hourly meteorological data is obtained from Open-Meteo (2023), providing insights into weather conditions.
3. Information on public holidays is sourced from a Kaggle dataset, as cited in Kaggle (2023).

4 Software Used

The project leveraged the following software tools, each chosen for their specific capabilities in handling different aspects of data management and analysis:

- **Microsoft Excel**: Utilized for the preliminary data exploration to understand the basic structure and contents of the datasets.
- **Google Colab**: Used for modeling and evaluation due to its cloud-based environment and high computation power.
- **Jupyter Notebook**:Used for initial pre-processing

5 Section 5

Before starting to program, the Python language must be installed on the system. For optimal compatibility and features, installing the latest release is recommended. For this endeavor, Python version 3.10.2 was installed on a Windows 11 machine, which was the latest available version at that time. Following the installation of Python, a development environment is required for writing and executing code. Among the most accessible and widely-used environments is the Jupyter Notebook, which comes included with the Anaconda Python distribution. Depending on the user's operating system, an appropriate version of Anaconda can be downloaded from this link. The dashboard of Anaconda, depicted in Figure 2, conveniently showcases the pre-installed packages such as the Jupyter Notebook. To commence coding in Python, one initiates the Jupyter Notebook to create a new Python script.

Using Google Collab is straightforward. By signing in with a Google account, users can easily upload files to the drive. Google Collab offers complimentary access to computational resources such as GPUs and TPUs, which are particularly beneficial for running tasks that require intensive computation. More information on Google Collab can be found at this link.

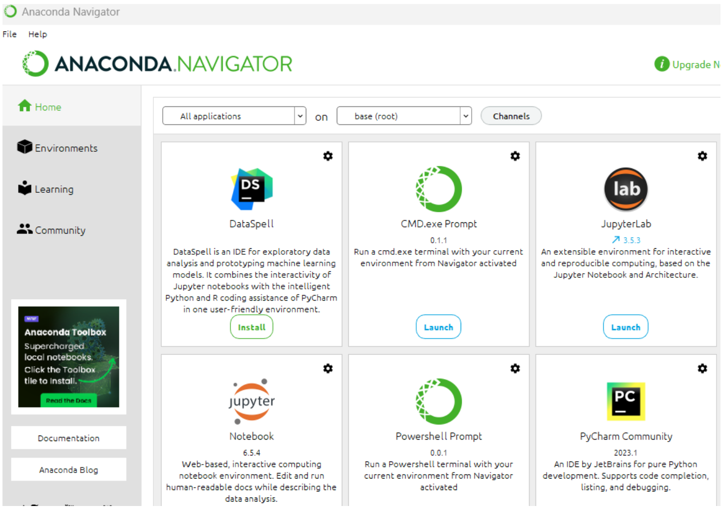


Figure 2: Anaconda User Interface

6 Project Development

When you have done the setup, you can start on Jupyter notebooks or Google colabs opening the program and starting new files. Next, you call your script through the code reference. That means you'll have an option of running all the script or just executing a part of it at one time. In case you discover that you have to create a new package; then you can install the package using the command **"'pip install package-name'"**.

6.1 Importing Library

In the scope of this project, the utilized packages are showcased in Figure 3. Our cloud platform, chosen for this project, conveniently provides several fundamental libraries pre-installed. Should there be a requirement for additional libraries, they can be imported as needed. Furthermore, it is imperative to pay attention to Figure 5, which delineates the versions of TensorFlow and Python being utilized. Adhering to the appropriate library versions is essential for the successful execution of the project's code.

6.2 Important function

Throughout the Jupyter Notebook, several important functions are employed to facilitate data preprossessing and analysis.Like drop columns , Missing values , numerical feature extraction from data set , categorical feature.

```
[5]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import chi2_contingency
from sklearn.preprocessing import OneHotEncoder
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from itertools import cycle
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import GridSearchCV
from scikeras.wrappers import KerasClassifie
```

Figure 3: Overview of the Packages Used in the Project

```
In [3]: import tensorflow as tf
import sys
import scikeras
print("Tesorflow Version",tf.__version__)
print("Python Version:", sys.version)
print("scikeras Version:", scikeras.__version__)

Tesorflow Version 2.12.0
Python Version: 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
scikeras Version: 0.12.0
```

Figure 4: Versions of TensorFlow and Python

6.3 Data Storage

The data-set has been stored on GitHub and made publicly available for use.

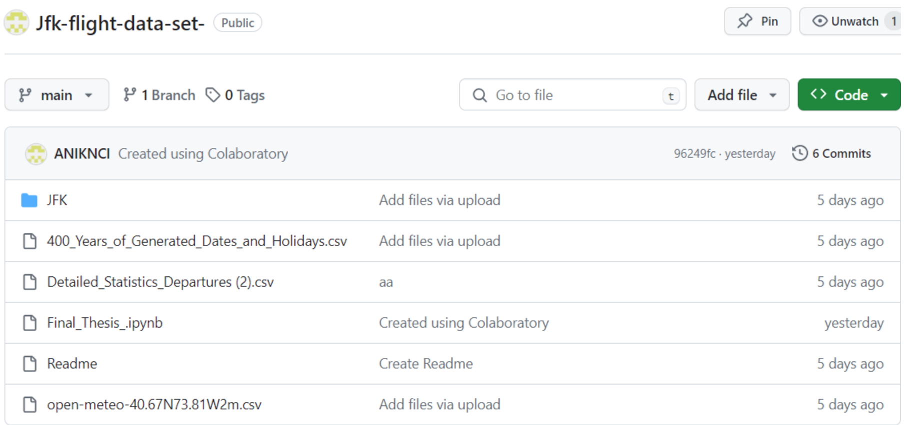
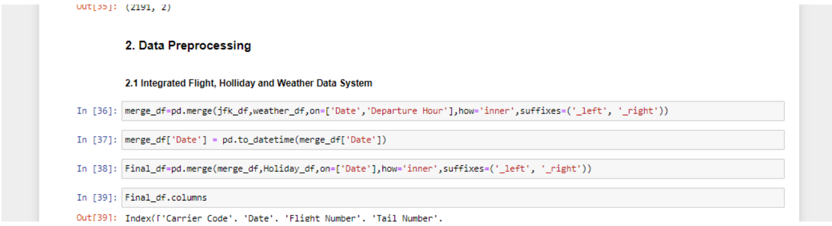


Figure 5: Github

6.4 Data integration and Feature Extraction

Data integration:- There are three distinct datasets, specifically flight data, weather data, and holiday data. All three datasets are merged by the date and departure hour using an inner join. The figure illustrates the code used for this merging process.

Feature extraction:- is crucial because it enhances the model's accuracy and expedites the training process. Identifying the correct set of features is essential for the model to make accurate predictions. The goal is to streamline the dataset by capturing important information and eliminating what's not relevant, thereby reducing the number of features. Fig 7shows the some of the extracted features on final data set.



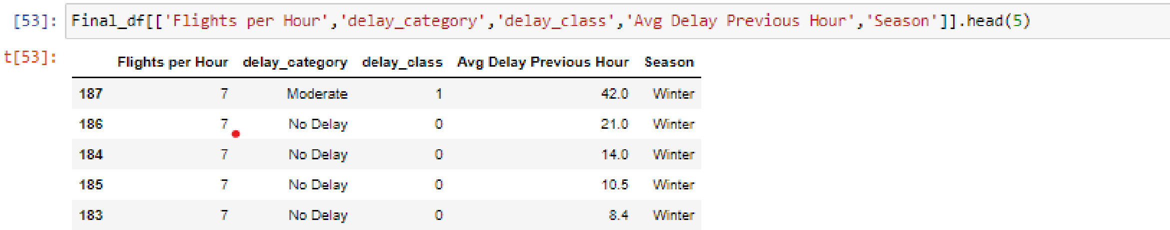
```
Out[35]: (2191, 2)

2. Data Preprocessing

2.1 Integrated Flight, Holliday and Weather Data System

In [36]: merge_df=pd.merge(jfk_df,weather_df,on=['Date','Departure Hour'],how='inner',suffixes=('_left', '_right'))
In [37]: merge_df['Date'] = pd.to_datetime(merge_df['Date'])
In [38]: Final_df=pd.merge(merge_df,Holiday_df,on=['Date'],how='inner',suffixes=('_left', '_right'))
In [39]: Final_df.columns
Out[39]: Index(['Carrier Code', 'Date', 'Flight Number', 'Tail Number',
```

Figure 6: Data Integration



```
[53]: Final_df[['Flights per Hour','delay_category','delay_class','Avg Delay Previous Hour','Season']].head(5)
```

t[53]:

	Flights per Hour	delay_category	delay_class	Avg Delay Previous Hour	Season
187	7	Moderate	1	42.0	Winter
186	7	No Delay	0	21.0	Winter
184	7	No Delay	0	14.0	Winter
185	7	No Delay	0	10.5	Winter
183	7	No Delay	0	8.4	Winter

Figure 7: Feature Extraction

6.5 Data Exploration and Visualization

The project includes exploratory data analysis (EDA) on the combined dataset using the libraries Matplotlib and Seaborn, which enhance several visualizations listed below.
Fig 8 Shows the distribution of the departure delay amongst all classes.
Fig 9 shows the seasonal impact on delays.

6.6 Feature selection And Encoding

Feature selection: Project used Select K best method for feature selection.Fig 7 shows the implementation in code.

Encoding: is done by the one-hot encoder,Fig 11 shows the implementation in code.

6.7 Modeling

Prior to the modeling phase, significant predictors are identified using recursive feature elimination and are then loaded into 'x-data' and 'y-data' functions, with 'y-data' capturing the target variable. To address data imbalances, the SMOTE algorithm is utilized to normalize the distribution. The dataset is further segmented into training, test, and validation sets based on the departure year, ensuring that the model training and validation are robust and comprehensive. The code implementation of this dataset segmentation is detailed in the figure 12 that follows.

To conclude this study, two algorithms—Long Short-Term Memory (LSTM) and Random Forest—were utilized to analyze both oversampled data (to address imbalances) and the actual, unmodified dataset.

3.2 Distribution of delay class

```
In [59]: delay_counts = Final_df['delay_category'].value_counts(ascending=False)
plt.figure(figsize=(6, 4))
bars = plt.bar(delay_counts.index, delay_counts.values)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, int(yval), va='bottom', ha='center')
plt.xlabel('Delay Category')
plt.ylabel('Count')
plt.title('Count of Flights by Delay Category')
plt.show()
```

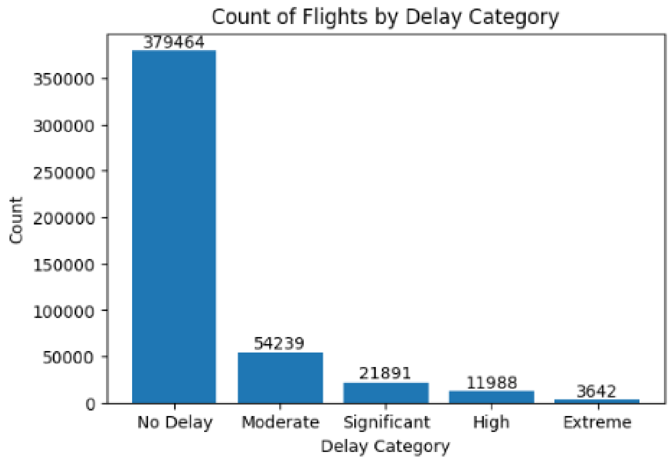


Figure 8: Departure delay distribution

6.7.1 LSTM implementation

In this study, we applied the Long Short-Term Memory (LSTM) algorithm using the TensorFlow library to both the oversampled dataset and the original dataset. We conducted two sets of experiments: one with hyperparameter tuning and the other without. Additionally, we evaluated various performance metrics using the necessary libraries to assess the model's effectiveness. The fig 13 below illustrates the code implementation of these experiments.

6.7.2 Random forest implementation

In this study, we employed the Random Forest algorithm utilizing the required libraries on both the oversampled dataset and the unmodified dataset. We conducted two sets of experiments: one involving hyperparameter tuning and the other without. Furthermore, we assessed the model's performance by evaluating multiple performance metrics. The figure shown in Fig. 14 below presents the code implementation for these experiments.

References

Bureau of Transportation Statistics (2023). Bureau of transportation statistics - on-time departure data, `https://www.transtats.bts.gov/ONTIME/Departures.aspx`. Accessed: December 1, 2023.

Kaggle (2023). Us federal pay and leave holidays, `https://www.kaggle.com/datasets/donnetew/us-holiday-dates-2004-2021`. Kaggle dataset.

Figure 9: Seasonal impact on Departure delay

Figure 10: Feature selection

Open-Meteo (2023). Historical weather api documentation, `https://open-meteo.com/en/docs/historical-weather-api`. Accessed: December 1, 2023.

one Hot encoder

```
100]: import pandas as pd
      from sklearn.preprocessing import OneHotEncoder

      # Assuming Main_df is already loaded into the environment, if not, it would have to be loaded from a file or defined.

      # Initialize the OneHotEncoder
      one_hot_encoder = OneHotEncoder()

      # Select categorical data only for encoding
      categorical_cols = Main_df.select_dtypes(include=['object', 'category']).columns

      # Apply OneHotEncoder to the selected columns and convert it to a dataframe
      encoded_data = one_hot_encoder.fit_transform(Main_df[categorical_cols]).toarray()

      # Here's the updated line with the correct method
      encoded_df = pd.DataFrame(encoded_data, columns=one_hot_encoder.get_feature_names_out(categorical_cols))

      # Drop the original categorical columns from Main_df
      Main_df = Main_df.drop(categorical_cols, axis=1)
      Main_df = pd.concat([Main_df, encoded_df], axis=1)

      # Verifying the data frame
      Main_df.head()
```

100]:

	Scheduled elapsed time (Minutes)	Taxi-Out time (Minutes)	Delay Carrier (Minutes)	Delay Weather (Minutes)	Delay National Aviation System (Minutes)	Delay Security (Minutes)	Delay Late Aircraft Arrival (Minutes)	Departure Month	Departure Day	Departure Hour	...	Destination Airport_SJC	Destination Airport_SJU	Destination Airport_SLC
187	166.0	13.0	34.0	0.0	0.0	0.0	0.0	1	1	5	...	0.0	0.0	0.0
186	220.0	21.0	0.0	0.0	0.0	0.0	0.0	1	1	5	...	0.0	1.0	0.0
184	185.0	11.0	0.0	0.0	0.0	0.0	0.0	1	1	5	...	0.0	0.0	0.0

Figure 11: Encoding

and robustness across different time frames, assuming 'Departure Year' is a critical feature in the dataset.

```
In [103]: train_df_x=train_df.drop(columns=['delay_class','Departure Year'],axis=1)
          train_df_y=train_df['delay_class']

          print("Shape of train_df_x:", train_df_x.shape)
          print("Shape of train_df_y:", train_df_y.shape)

          Shape of train_df_x: (259098, 100)
          Shape of train_df_y: (259098,)

In [104]: Validation_df_x=Validation_df.drop(columns=['delay_class','Departure Year'],axis=1)
          Validation_df_y=Validation_df['delay_class']

          print("Shape of Validation_df_x:", Validation_df_x.shape)
          print("Shape of Validation_df_y:", Validation_df_y.shape)

          Shape of Validation_df_x: (39056, 100)
          Shape of Validation_df_y: (39056,)

In [105]: test_df_x=test_df.drop(columns=['delay_class','Departure Year'],axis=1)
          test_df_y=test_df['delay_class']

          print("Shape of test_df_x:", test_df_x.shape)
          print("Shape of test_df_y:", test_df_y.shape)

          Shape of test_df_x: (173070, 100)
          Shape of test_df_y: (173070,)
```

The 'Departure Year' feature was used exclusively for segmenting the dataset into training, validation, and testing sets. So dropped it along with 'delay_class'

Figure 12: Code Implementation of Data split

7.1 LSTM

```
In [108]: import numpy as np
          from tensorflow.keras.models import Sequential
          from tensorflow.keras.layers import LSTM, Dense, Dropout
          from tensorflow.keras.utils import to_categorical
          from tensorflow.keras.optimizers import Adam

          # Convert the feature DataFrames to numpy arrays
          train_df_x_1 = np.array(train_df_x)
          Validation_df_x_1 = np.array(Validation_df_x)
          test_df_x_1 = np.array(test_df_x)

          # Convert the target DataFrames to numpy arrays and use one-hot encoding
          train_df_y_1 = to_categorical(train_df_y, num_classes=5)
          Validation_df_y_1 = to_categorical(Validation_df_y, num_classes=5)
          test_df_y_1 = to_categorical(test_df_y, num_classes=5)

          # Reshape the input data for LSTM [samples, time steps, features]
          train_df_x_1 = train_df_x_1.reshape((train_df_x_1.shape[0], 1, train_df_x_1.shape[1]))
          Validation_df_x_1 = Validation_df_x_1.reshape((Validation_df_x_1.shape[0], 1, Validation_df_x_1.shape[1]))
          test_df_x_1 = test_df_x_1.reshape((test_df_x_1.shape[0], 1, test_df_x_1.shape[1]))

          # Define the LSTM model architecture
          model = Sequential()
          model.add(LSTM(units=50, return_sequences=True, input_shape=(1, train_df_x_1.shape[2])))
          model.add(Dropout(0.2))
          model.add(LSTM(units=50))
          model.add(Dropout(0.2))
          model.add(Dense(units=5, activation='softmax'))  # Output layer with 5 units for 5 classes

          # Compile the model
          model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

          # Train the model
          history = model.fit(train_df_x_1, train_df_y_1,
                              epochs=10,
                              batch_size=32,
                              validation_data=(Validation_df_x_1, Validation_df_y_1),  # Use reshaped validation data
                              verbose=2)

          # Evaluate the model on the test set
          test_loss, test_accuracy = model.evaluate(test_df_x_1, test_df_y_1, verbose=0)

          # Print the test loss and accuracy
          print(f'Test Loss: {test_loss:.4f}, Test Accuracy: {test_accuracy:.4f}')

          Epoch 1/10
          8097/8097 - 53s - loss: 0.2943 - accuracy: 0.9076 - val_loss: 0.1336 - val_accuracy: 0.9650 - 53s/epoch - 7ms/step
          Epoch 2/10
          8097/8097 - 43s - loss: 0.2603 - accuracy: 0.9188 - val_loss: 0.1315 - val_accuracy: 0.9630 - 43s/epoch - 5ms/step
          Epoch 3/10
```

Figure 13: LSTM

7.2 Random Forest

In [121]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

train_df_x_2 = np.array(train_df_x)
Validation_df_x_2 = np.array(Validation_df_x)
test_df_x_2 = np.array(test_df_x)
train_df_y_2 = np.array(train_df_y)
Validation_df_y_2 = np.array(Validation_df_y)
test_df_y_2 = np.array(test_df_y)

# Define the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100,
                                  random_state=42)

# Train the model
rf_model.fit(train_df_x_2, train_df_y_2)

# Evaluate the model on the validation set
validation_predictions = rf_model.predict(Validation_df_x_2)
validation_accuracy = accuracy_score(Validation_df_y_2, validation_predictions)
print(f'Validation Accuracy: {validation_accuracy:.4f}')

# Evaluate the model on the test set
test_predictions = rf_model.predict(test_df_x_2)
test_accuracy = accuracy_score(test_df_y_2, test_predictions)
print(f'Test Accuracy: {test_accuracy:.4f}')
```

```
Validation Accuracy: 0.9686
Test Accuracy: 0.9236
```

In [122]:

```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20]
}
grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5)
grid_search.fit(train_df_x_2, train_df_y_2)
best_params = grid_search.best_params_
print("Best Parameters:", best_params)
```

```
Best Parameters: {'max_depth': None, 'n_estimators': 200}
```

Figure 14: Random Forest