CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This study aims to develop a machine learning-based predictive model that will allow airlines to forecast flight delays in the future. There are two approaches involved in this methodology, which are data-driven techniques involving deep learning models as well as classical machine learning techniques. In this paper, we will look into historical flight data in order to identify patterns and correlations that can be used to predict potential delays more accurately. During the process, data are collected, preprocessed, models are developed, models are trained, evaluated, and a comparative analysis is performed on three different models, which are Random Forest, XGBoost, and an Attention-Based Bidirectional Long Short-Term Memory (ATT-BI-LSTM) network.

3.2 Data Collection

In this study, the dataset was sourced from Kaggle (Flight Take Off Prediction). There are more than 20,000 records in the data between November 2019 and December 2020. There are a number of key variables that should be considered, including flight number, airline, scheduled and actual departure/arrival times, flight distance, weather conditions, and day of week. To enhance the performance of the model, additional data such as weather conditions corresponding to JFK Airport during the departure period were also incorporated in order to enhance the accuracy of the model.

3.3 Data Preprocessing

In order to ensure the reliability and accuracy of a model, effective preprocessing is essential. To carry out the following steps, we took the following steps:

3.3.1 Convert Numeric Columns and Handling Missing Data

This dataset includes several key numeric columns, which are revelant to weather conditions and flight operations, such as, for instance, ‘Temperature’, ‘Dew Point’, ‘Humidity’, ‘Wind Speed’, ‘Wind Gust’, ‘Pressure’, and ‘TAXI_OUT’. To ensure consistent data formatting, each of these columns is explicitly converted to a numeric type, especially if any values were initially stored as strings, so that the data can be formatted consistently. For example, when a value is missing or invalid in a column, then the column's mean is used to fill that value so that data does not get lost, but statistical integrity is maintained. Missing values are filled with the string ‘Unknown’ in the categorical column ‘Condition’. This likely represents the weather conditions at a given time. By using this method, the information in the dataset will still be retained, while a clear indication will be provided of records that originally had missing data in them, rather than removing the records and losing the data as a result.

3.3.2 Remove Irrelevant Column.

In addition to this, the dataset is emptied of columns such as ‘TAIL_NUM’, ‘sch_dep’, and ‘sch_arr’. Depending on their relevance to the prediction task, these features either do not contribute to the prediction task or have a very high number of unique values, which could introduce noise in the prediction task or lead to overfitting in machine learning models. By dropping them, the dataset is streamlined for analysis, making it easier for the analyst to analyze.

3.3.3 Creating Categorical Variables

In order to encode categorical variables such as ‘OP_UNIQUE_CARRIER’, ‘DEST’, and ‘Condition’, label encoding is used. Using this method, each unique text category can be converted into an integer, which makes it numerically suitable for input into machine learning algorithms that require numeric input features to work properly.

3.3.4 Create a Binary Delay Label

The ‘DEP_DELAY’ column is used to create a new binary column called ‘is_delayed’ that contains the values for both delays. If the departure time of a flight has been delayed by more than 15 minutes, then that flight will be labeled with 1 (indicating a delay), whereas all other flights will be labeled with 0. As a result of this step, continuous delay data are transformed into binary classification labels, which enable the development of classification models to be developed.

3.3.5 Remove The Remaining Missing Values From The Table

The last step in the preprocessing step is to remove from the dataset any rows that still contain missing values after all other preprocessing steps have been completed, so if there are any remaining rows in the dataset with missing values, those rows are deleted. In this way, you can make sure the final dataset is clean and does not contain any null values that may interfere with the training or evaluation of the model.

3.4 Feature Engineering

In order to improve the model’s ability to detect complex relationships within the data, a series of engineered features was added to the dataset to enhance its detection capabilities. As a result of these transformations, we have been able to elevate the representation of temporal, operational, and weather-related factors that have been shown to influence delays to a more accurate form.

3.4.1 Time Patterns

A number of temporal features have been designed to capture typical traffic patterns and weekly behavior patterns. There was introduced a binary variable was introduced for the purpose of indicating peak operational hours, specifically the period between 6:00 a.m. to 10:00 a.m. and 4:00 p.m. to 7:00 p.m., when airports often experience higher congestion and a higher risk of delays. Furthermore, a weekend indicator was created in an effort to flag flights that are scheduled to take place on Saturdays and Sundays, due to the fact that these days tend to have different traffic patterns and staffing patterns compared to weekdays.

3.4.2 Operational Complexity

Taking into account the airport's operational load and complexity, a flight volume per hour feature was added by counting departures per hour. By using this metric, we can be able to determine how congested an airport is. In addition, flight distances were categorized into categories like short-haul, medium-haul, and long-haul. As a result of categorization, it is possible to identify patterns in delays that may be related to the length or type of the route that has cause delays.

3.4.3 Climate-Based Features

The raw weather descriptions were used as the basis for many of the features in order to better reflect adverse conditions in the weather. It is possible to create binary flags for fog, rain, and snow by parsing keywords from textual weather data in order to allow the model to be able to immediately recognize the presence of disruptive weather events. In addition, a temperature range variable was introduced, based on the difference between maximum and minimum temperatures. As an indicator of weather instability, this feature can significantly impact flight schedules.

3.5 Machine Learning Models

In this study, three predictive models were developed and evaluated in order to identify flights that are likely to encounter delays in takeoff, and to identify them before they begin flying. The models include both traditional ensemble methods and advanced deep learning architectures integrating tubular and sequential features. The purpose of this study was to asses the predictive accuracy and robustness of different modelling approaches.

3.5.1 Random Forest Classifier

Random Forest classifiers were used to benchmark the performance of deep learning models against a classical machine learning baseline. This method constructs multiple decision trees by bootstrapping subsets of data and randomly selecting features at each split. Using this technique, it is much easier to enhance robustness as well as mitigate overfitting, especially when the dataset contains noise or missing values that may adversely affect the model.

In addition to being effective for tabular data, Random Forest is also highly interpretable. An advantage of this platform is its ability to calculate feature importance scores, which can be used to determine which factors contribute to flight delays. With its robustness, the Random Forest model often serves as an appropriate baseline, especially when dealing with high-dimensional datasets.

With the help of the scikit-learn Python library, the Random Forest classifier was implemented. The performance of the model was further improved by tuning hyperparameters such as the number of estimators and the maximum depth of trees using grid search and cross-validation. As a result of this systematic tuning, the model’s generalization ability was optimized while computational efficiency was maintained.

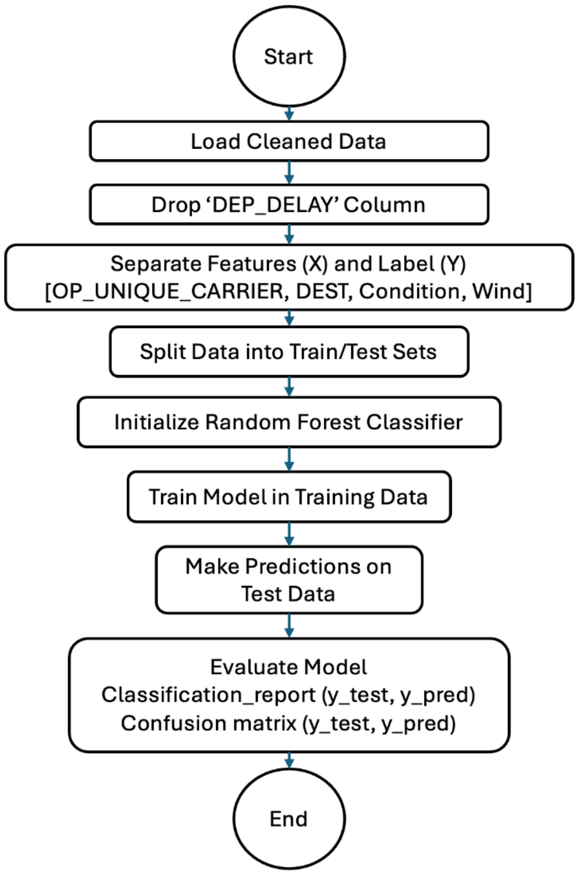

Start
Load Cleaned Data
Drop 'DEP_DELAY' Column
Separate Features (X) and Label (Y)
[OP_UNIQUE_CARRIER, DEST, Condition, Wind]
Split Data into Train/Test Sets
Initialize Random Forest Classifier
Train Model in Training Data
Make Predictions on Test Data
Evaluate Model
Classification_report (y_test, y_pred)
Confusion matrix (y_test, y_pred)
End

Figure 3.1 Flowchart Random Forest Classifier

3.5.2 XGBoost

It was discovered that the XGBoost algorithm performs well in structured data tasks due to its ability to boost gradients and its high accuracy. This approach was implemented by using the official XGBoost library, and parameters such as learning rate, maximum depth, and subsampling ratio were tuned based on the library and then evaluated using cross-validation to ensure proper performance. Based on the results, it was found that XGBoost was more accurate and faster than other algorithms, making it an excellent choice for structured data tasks. The library's flexibility improved model performance significantly through hyperparameter tuning. Furthermore, XGBoost has the ability to handle large datasets due to its scalability and robustness. The algorithm was also able to perform better on structured data tasks since it was able to handle missing data and reduce overfitting.

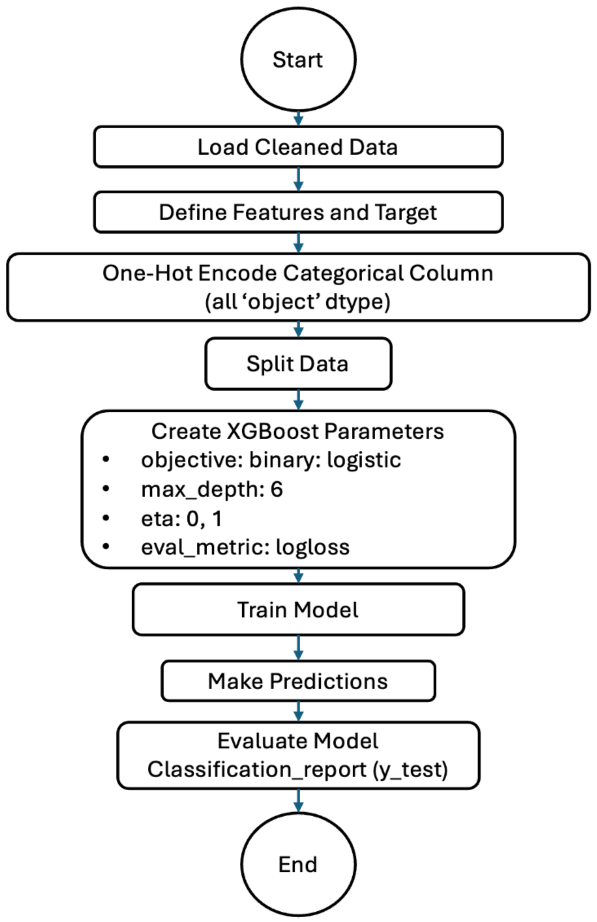

Start
Load Cleaned Data
Define Features and Target
One-Hot Encode Categorical Column (all 'object' dtype)
Split Data
Create XGBoost Parameters
objective: binary: logistic
max_depth: 6
eta: 0, 1
eval_metric: logloss
Train Model
Make Predictions
Evaluate Model Classification_report (y_test)
End

Figure 3.2 Flowchart XGBoost Classifier

3.5.3 ATT-BI-LSTM

In this study, a third model was developed called the Attention-Based Bidirectional Long Short-Term Memory (ATT-BI-LSTM). The deep learning model architecture comprises of embedding layer, a bidirectional LSTM layer, and a layer of attention that can be trained on a given dataset. Due to the structure of the model, it was able to capture both forward and backward time-dependent events, in addition to using the attention mechanism to weigh the importance of important events. Using TensorFlow/Keras, the model was implemented to prevent overfitting by using early stopping and dropout techniques. The model achieved promising results by effectively learning temporal dependencies and focusing on critical events. Aside from improving prediction accuracy, this method also demonstrated its ability to handle complex sequential data in a wide range of applications.

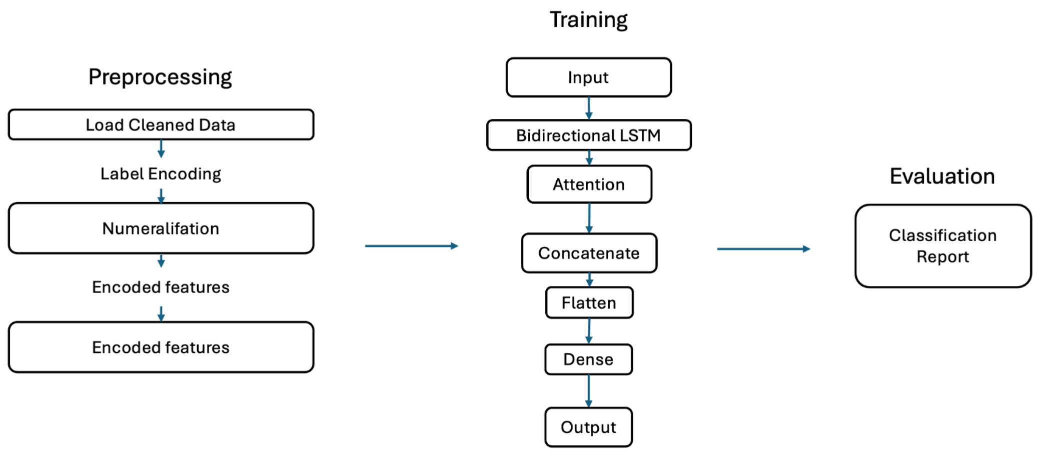


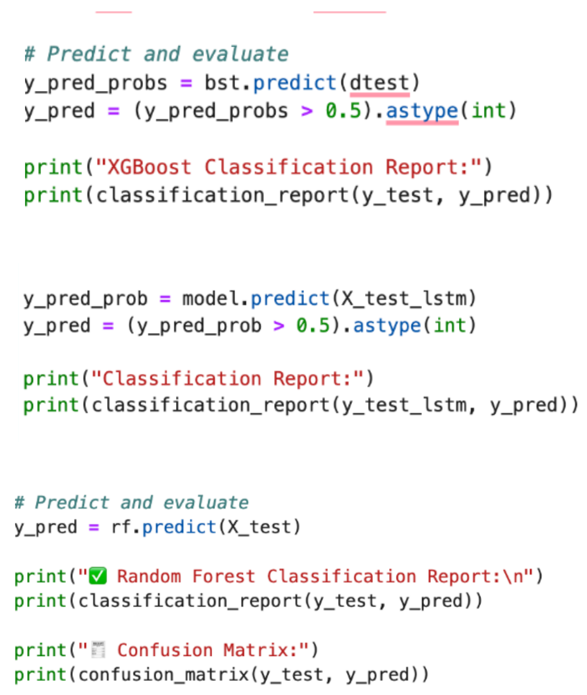
Figure 3.3 Flowchart ATT-BI-LSTM

3.6 Model Evaluation

The predictive models developed for flight take-off delay classification were evaluated comprehensively to ensure their reliability and effectiveness. In support of training, testing, and validation, multiple evaluation metrics were used as well as comparisons between models and robust tools for training, testing and validation. The evaluation process was created to understand both the practical implications of false positives and false negatives in operational contexts with significant consequences for both False Positives and False Negatives.

3.6.1 Evaluation Metrics

Model performance was evaluated using several metrics. A measure of accuracy can be misleading when data are imbalanced, since it reflects the overall number of correct predictions. Precision indicates the percentage of correctly predicted delays among all predicted delays, minimizing false alarms. Model recall measures the ability of the model to detect actual delays, reducing missed delays. F1 scores, which are the harmonic means of precision and recall, balance imbalanced datasets. By using ROC-AUC, the model was assessed for its ability to differentiate between delayed and non-delayed flights. To visualize classification errors, confusion matrices were used. Several cross-validation tests were performed to ensure robustness and generalization.



```
# Predict and evaluate
y_pred_probs = bst.predict(dtest)
y_pred = (y_pred_probs > 0.5).astype(int)

print("XGBoost Classification Report:")
print(classification_report(y_test, y_pred))


y_pred_prob = model.predict(X_test_lstm)
y_pred = (y_pred_prob > 0.5).astype(int)

print("Classification Report:")
print(classification_report(y_test_lstm, y_pred))


# Predict and evaluate
y_pred = rf.predict(X_test)

print("✅ Random Forest Classification Report:\n")
print(classification_report(y_test, y_pred))

print("🧾 Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

Figure 4: Model Prediction and Evaluation using Machine Learning

3.7 Summary

The methodology of the study is explained in detail in this chapter, from data collection to model evaluation. Using this process, a flight delay predictive model using machine learning can be measured systematically.