Real-Time Flight Delay Prediction Using Machine Learning

Ali El Masri

**Abstract**

Flight delays represent a significant challenge within the aviation industry, leading to considerable operational inefficiencies and customer dissatisfaction. This paper introduces a comprehensive machine learning approach to predict flight delays and estimate delay durations in real time. The study focuses on flights departing from JFK Airport, utilizing data from the Aviation Edge API for flight information and the Open-Weather API for real-time weather conditions. After experimenting with several models, an ensemble approach combining XGBoost and Random Forest was selected for its superior performance. This ensemble model leverages the strengths of both algorithms to capture complex patterns in the data, providing accurate delay predictions. While the initial scope included additional data sources such as aircraft details and on-time performance data, these were excluded due to inconsistencies and insufficient coverage. Preliminary results show that the model is effective in predicting delays, and the developed pipeline enables real-time predictions by integrating current data from multiple sources.

**1. Introduction**

Flight delays are a pervasive issue that impacts both airline operations and passenger experiences. Delays can lead to a cascade of operational disruptions, increased costs, and decreased customer satisfaction. The ability to accurately predict flight delays in real time allows airlines to better manage resources, improve scheduling, and enhance the overall passenger experience. This study aims to develop a robust machine learning model capable of predicting the duration of flight delays using both historical and real-time data.

**Objective:**

* Develop a regression model that predicts flight delay durations in real time.
* Utilize comprehensive historical flight data and integrate real-time updates from the Aviation Edge and Open-Weather APIs.
* Ensure the model's accuracy and reliability through evaluation and testing.

**2. Data Sources and Preparation**

**The data for this study is sourced from two primary APIs:**

* Aviation Edge API: Provides detailed historical and real-time data on flights, including information on departures, arrivals, delays, and various other flight-related parameters.
* Open-Weather API: Supplies real-time weather data, which is crucial for understanding how current weather conditions may impact flight delays.

**Initial Data Collection and Merging Attempts:**

* Aviation Edge Data: Historical flight data was collected for JFK Airport, focusing on a one-year period to capture a wide range of flight patterns and delay scenarios. Given the API limitations, data was retrieved in three-day increments to ensure completeness.
* Bureau of Transportation Statistics (BTS) and Federal Aviation Database: In an effort to enrich the dataset, on-time performance data from BTS and aircraft data from the Federal Aviation Database were collected. The goal was to merge these datasets with the Aviation Edge data to provide a more comprehensive understanding of factors influencing delays.
* Weather Data: Real-time weather data was integrated with flight data to provide contextual information on conditions that could influence delays.

**Challenges in Data Merging:**

During the data preparation phase, several challenges were encountered:

* Inconsistent Coverage: The BTS and Federal Aviation Database provided valuable information; however, their coverage was inconsistent, with many flights lacking the corresponding data. This inconsistency led to significant gaps in the dataset, reducing the overall quality and reliability of the model.
* Noise Introduction: The inclusion of aircraft-specific data, such as tail numbers, introduced noise into the model rather than improving accuracy. The variability in this data made it difficult to establish strong correlations with delay outcomes.

**Final Decision:**

Given these challenges, the decision was made to exclude the BTS and aircraft data from the final model. Instead, the focus was placed on the more consistent and reliable data provided by the Aviation Edge and Open-Weather APIs. This streamlined approach allowed for a cleaner dataset and more robust predictions.

**Data Cleaning and Processing:**

* The initial dataset contained over 400 features, which were systematically reduced to 22 relevant features. This reduction was guided by domain knowledge and exploratory data analysis (EDA).
* Features related to post-takeoff events and non-JFK arrivals were excluded to maintain focus on pre-departure and departure-related factors.
* Outliers were handled by removing observations below the 20th percentile and above the 80th percentile, which helped to improve the quality of the data and the robustness of the model.

*Image 1: Post Reduction Outliers Boxplot*

A group of blue and white graphs

Description automatically generated

**Exploratory Data Analysis (EDA)** was conducted to uncover patterns, correlations, and outliers within the data. The analysis revealed that most delays were concentrated at the lower end of the spectrum, with a significant number of flights experiencing minimal delays. Additionally, the count of delayed versus non-delayed flights indicated more delays. The top 10 airlines by flight count were identified, showing key players in the airline industry. Summary statistics provided further insights into the distribution and characteristics of the dataset.

*Image 2: Top 10 Airlines by Flight Count*

A graph of blue bars

Description automatically generated

*Image 3: Distribution of Departure Delays*A graph of a flight delay

Description automatically generated

**3. Modeling**

Given that the majority of flights experience delays in this dataset, a classification model to predict whether a flight will be delayed was deemed unnecessary. Instead, the focus is solely on developing a regression model to estimate the duration of delays.

*Image 4: Count of Delayed vs. Non-Delayed Flights*

A graph of flight status

Description automatically generated

**Model Selection:**

The choice of model is critical in capturing the complex interactions between different factors that contribute to flight delays. After experimenting with various machine learning algorithms, including decision trees, random forests, and gradient boosting, an ensemble model was selected. The ensemble model combines:

* **XGBoost:** Known for its ability to handle large datasets and model complex non-linear relationships efficiently.
* **Random Forest:** Adds stability and reduces the risk of overfitting by aggregating predictions from multiple decision trees.

This ensemble approach was chosen because it leverages the strengths of both algorithms, balancing bias and variance, and improving overall model performance.

**Preprocessing Techniques:**

To prepare the data for the ensemble model, several preprocessing steps were implemented:

* **Cyclical Encoding:** Time-based features, such as the hour of the day, day of the week, and month, exhibit cyclical patterns. Traditional encoding methods would not capture this, so cyclical encoding was applied to preserve the temporal relationships.
* **Label Encoding:** Categorical features with a large number of unique values (e.g., airline codes, flight numbers) were label encoded to convert them into numerical values while maintaining their ordinal relationships.
* **One-Hot Encoding:** For categorical variables with fewer unique values, one-hot encoding was used to avoid introducing any ordinal bias.
* **Standard Scaling:** To ensure that features are on a similar scale and prevent any single feature from disproportionately influencing the model, standard scaling was applied to all numerical features.

**Model Training:**

* The dataset was split into training and testing sets, and the ensemble model was trained using the preprocessed data.
* **GridSearchCV** was employed to fine-tune the hyperparameters of the model, ensuring optimal performance.
* The training process was accelerated using GPU support for XGBoost, enabling faster iteration over different hyperparameter configurations.

**Model Performance:**

The ensemble approach combining XGBoost and Random Forest, demonstrated robust performance as indicated by the evaluation metrics. The model achieved a Mean Absolute Error (MAE) of 1.72 minutes, which suggests that, on average, the model's predictions deviate from the actual delay times by just under two minutes. This low MAE highlights the model's precision in predicting flight delays. The Mean Squared Error (MSE) was recorded at 13.54, indicating the presence of some larger errors, though these are relatively few given the complexity of the data. The Root Mean Squared Error (RMSE) of 3.68 further supports the model's effectiveness, showing that most predictions are within a 3.68-minute margin of error. Overall, these results confirm that the ensemble model is well-calibrated and capable of making accurate predictions, making it a reliable tool for real-time flight delay prediction.

**4. Real-Time Predictions:** A dedicated pipeline was developed to handle real-time data processing and prediction. This pipeline integrates live data from the Aviation Edge and Open-Weather APIs, ensuring that the predictions are based on the most current information available.

**Real-Time Prediction Pipeline:**

* The pipeline replicates the preprocessing steps applied during model training, including cyclical encoding, label encoding, one-hot encoding, and standard scaling.
* Once the data is preprocessed, it is fed into the ensemble model to generate real-time delay predictions.
* **Placeholder for Real-Time Prediction DataFrame:** A table showcasing the real-time predictions made by the model will be included here, demonstrating its practical application in a live environment.

**5. Conclusion**

The study successfully demonstrates the development of a machine learning model capable of predicting flight delays in real time. The choice of an ensemble model combining XGBoost and Random Forest proved effective in handling the complexity of the data and providing accurate predictions. By focusing on the most relevant features and applying rigorous preprocessing techniques, the model shows strong potential for operational use.

Future work will focus on refining the model further, incorporating additional data sources where possible, and testing the model's generalizability to other airports beyond JFK

**6. References**

1. Ding, Y. (2017). Airline flight delay prediction using machine learning models. *IOP Conference Series: Earth and Environmental Science, 81*(1), 012198. https://doi.org/10.1088/1755-1315/81/1/012198
2. Tang, Y. (2021, October 15). Airline flight delay prediction using machine learning models. *ACM Digital Library*. https://dl.acm.org/doi/fullHtml/10.1145/3497701.3497725

**7. Appendix**

A. Libraries and Tools:

1. Python Libraries:
   * Pandas: Used for data manipulation and analysis, including data cleaning, feature engineering, and handling large datasets.
     + import pandas as pd
   * NumPy: Used for numerical computations, particularly in feature engineering and handling cyclical encoding.
     + import numpy as np
   * Scikit-Learn (sklearn):
     + Preprocessing:
       - StandardScaler: Used for scaling features to ensure they are on a similar scale.
         * from sklearn.preprocessing import StandardScaler
       - OneHotEncoder: Used for one-hot encoding categorical variables with a limited number of unique values.
         * from sklearn.preprocessing import OneHotEncoder
       - LabelEncoder: Used for label encoding categorical variables with many unique values.
         * from sklearn.preprocessing import LabelEncoder
     + Model Selection and Training:
       - GridSearchCV: Used for hyperparameter tuning to find the best-performing model configuration.
         * from sklearn.model\_selection import GridSearchCV
   * XGBoost: Used for training the XGBoost model, which was part of the ensemble model.
     + import xgboost as xgb
   * Joblib: Used for saving and loading the trained model, scalers, and encoders to and from disk.
     + import joblib
   * Requests: Used for making HTTP requests to the APIs (Aviation Edge and Open-Weather).
     + import requests
   * Datetime: Used for handling and manipulating date and time information, particularly in feature engineering.
     + from datetime import datetime, timedelta
2. Development Environment:
   * Jupyter Notebook: Used as the primary development environment for code experimentation, model training, and visualization.
   * VSCode/PyCharm: (Optional) Integrated Development Environments (IDEs) that may have been used for more extensive coding and debugging.

B. Data Sources:

1. Aviation Edge API:
   * Provides historical and real-time flight data, including departure and arrival times, delays, and flight statuses.
   * The primary source for collecting flight-related data is used in model training and real-time predictions.
   * API Documentation: <https://aviation-edge.com/premium-api/>
2. Open-Weather API:
   * Supplies real-time weather data, which is critical for understanding environmental factors impacting flight delays.
   * Used to enrich the dataset with contextual information about weather conditions at JFK Airport.
   * API Documentation: https://openweathermap.org/api
3. Bureau of Transportation Statistics (BTS):
   * Initially used to gather on-time performance data to merge with the Aviation Edge data. However, it was excluded from the final model due to inconsistent coverage.
   * Data Source: <https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGJ&QO_fu146_anzr=b0-gvzr>
4. Federal Aviation Database:
   * Provided aircraft data, such as tail numbers, that was initially considered for the model. This data was also excluded due to introducing noise and inconsistencies.
   * Data Source: <https://www.faa.gov/licenses_certificates/aircraft_certification/aircraft_registry/releasable_aircraft_download>
5. Weather.gov:
   * Used for historical weather data, particularly to supplement the dataset with past weather conditions for exploratory data analysis (EDA).
   * Data Source: <https://www.weather.gov/wrh/Climate?wfo=okx>

C. Model Training and Evaluation:

1. XGBoost:
   * A gradient boosting framework that uses decision trees and was used in the ensemble model for its ability to handle large datasets and capture complex patterns.
   * Official Documentation: <https://xgboost.readthedocs.io/>
2. Random Forest:
   * An ensemble learning method used alongside XGBoost to provide stability and reduce overfitting by aggregating predictions from multiple decision trees.
   * Scikit-Learn Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
3. GridSearchCV:
   * Used to perform hyperparameter tuning by exhaustively searching through a specified parameter grid to find the best combination of parameters for the model.
   * Scikit-Learn Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html

D. Data Preprocessing Techniques:

1. Cyclical Encoding:
   * Applied to time-based features to preserve the cyclical nature of hours, days, and months in the model.
   * Purpose: To allow the model to better understand temporal relationships.
2. Label Encoding:
   * Used for categorical features with a large number of unique values, converting them into a format suitable for machine learning algorithms.
   * Purpose: To maintain the ordinal relationship between categorical values.
3. One-Hot Encoding:
   * Applied to categorical features with fewer unique values to prevent any ordinal assumptions and create binary features.
   * Purpose: To ensure that the model does not infer incorrect relationships between categories.
4. Standard Scaling:
   * Used to standardize numerical features by removing the mean and scaling to unit variance.
   * Purpose: To ensure that all features contribute equally to the model's learning process, avoiding domination by features with largerscales.

**8. Questions from the Audience:**

1. **How does the model handle real-time data updates?**
   * The model is integrated with a real-time data pipeline that continuously fetches the latest flight and weather information from the Aviation Edge and Open-Weather APIs. This data is preprocessed using the same steps as the training data, ensuring consistency. The model then makes predictions based on this real-time data, providing up-to-date delay estimates.
2. **What specific features were most influential in predicting delays?**
   * Time-related features, such as the hour of departure and day of the week, were among the most influential. Weather conditions, including temperature, wind speed, and visibility, also significantly impacted the delay predictions. The cyclical encoding of these temporal features allowed the model to better capture patterns related to time and seasonality.
3. **How do weather conditions impact flight delays?**
   * Weather conditions play a crucial role in flight delays. Adverse weather, such as low visibility, high winds, and precipitation, can cause delays due to safety concerns and operational constraints. The model accounts for these factors by incorporating real-time weather data, which helps in predicting delays more accurately.
4. **Can the model be generalized to other airports besides JFK?**
   * While the model is currently tailored for JFK Airport, the underlying methodology is generalizable. By retraining the model with data from other airports and adjusting for local weather patterns and operational characteristics, the model can be adapted to predict delays at different locations.
5. **What are the limitations of the current model?**
   * The model's performance is limited by the quality and consistency of the input data. The exclusion of certain features, such as aircraft-specific data, due to inconsistencies may reduce the model's accuracy. Additionally, the model is primarily trained on data from JFK, so its predictions might not be as accurate when applied to other airports without retraining.
6. **How frequently should the model be retrained with new data?**
   * The model should be retrained periodically, ideally every few months, to incorporate the latest trends in flight delays and weather patterns. Additionally, retraining should be considered after significant operational changes, such as updates to airport infrastructure or changes in flight scheduling.
7. **What were the biggest challenges in data preprocessing?**
   * One of the biggest challenges was handling inconsistencies and missing values across different data sources. The initial attempt to merge data from multiple databases, including the BTS and Federal Aviation Database, introduced noise and inconsistencies, leading to the exclusion of certain features. Another challenge was ensuring that cyclical features, such as time and seasonality, were properly encoded to preserve their natural order.
8. **How does the model's performance compare to existing delay prediction systems?**
   * While specific comparisons to existing systems are ongoing, preliminary results indicate that the ensemble model provides competitive accuracy in predicting delays, especially when considering real-time data. The integration of weather data and advanced preprocessing techniques like cyclical encoding contribute to its robust performance.
9. **What are the implications of inaccurate delay predictions for airlines?**
   * Inaccurate delay predictions can lead to suboptimal resource allocation, increased operational costs, and diminished passenger satisfaction. For instance, overestimating delays may result in unnecessary gate hold times, while underestimating delays can cause cascading disruptions across the flight schedule. Accurate predictions are crucial for maintaining operational efficiency and customer satisfaction.
10. **How can the model be integrated into an airline's operational system?**
    * The model can be integrated into an airline's operational system via an API that allows real-time data to be fed into the model, which then returns delay predictions. These predictions can be used by flight operations teams for decision-making and by customer service teams to provide timely information to passengers. Additionally, integration with existing systems for resource management and scheduling could further enhance operational efficiency.