Machine Learning Flight Prediction

Vikas Papana, Ajay Hemanshu Desai, Hemil Anip Shah, Rangel Anselm Koli
Syracuse University

Introduction

There are a number of factors that can cause flight delays, including weather, late-arriving aircraft, and airport congestion. Congestion at airports is caused by a high volume of air traffic and airlines overscheduling flights. Airlines schedule flights knowing that there will be delays, in order to meet passenger demand.

In this machine learning project, the goal is to predict the arrival status of flights into Syracuse (SYR) from various airports such as Chicago (ORD), New York (JFK), and Orlando (MCO) based on historical data. Specifically, we have information on multiple flights from each origin airport. The objective of the project is to predict if the given flight was Early, On-Time or Late and also predict if the next flight would be Early, On-Time or Late. The project was divided into 2 parts Initial Prediction of flight Arrival and Final Prediction of flight Arrival.

Data Collection

To compile comprehensive data for our analysis, we utilized government websites to procure arrival and departure information for all flights. This data was meticulously matched with our main dataset, ensuring accuracy and completeness. Additionally, we integrated weather data obtained from Weatherbit.io using the Postman API. By merging this weather data into our main dataset based on the corresponding dates, we enriched our dataset with crucial environmental variables, providing a deeper context for our analysis. This meticulous approach to data collection and integration ensures the robustness and reliability of our findings.

Our data collection methodology involved a systematic approach to gather and merge relevant information from multiple sources. By leveraging government websites for flight arrival and departure data, we ensured the inclusion of comprehensive aviation records. To enhance the accuracy of our flight delay predictions, we integrated historical weather data, acknowledging its significant impact on flight statuses. For this purpose, we utilized the Weatherbit.io API, which offers comprehensive historical weather data. The process involved the following steps:

1. **API Setup**: We registered on Weatherbit.io to obtain an API key, which is essential for accessing the API services. This key was securely stored and used in subsequent requests to authenticate our access.
2. **Postman Configuration**: We used Postman, a popular tool for API exploration and testing, to construct and test our API requests. Postman's user-friendly interface allowed us to easily configure the request parameters and headers.
3. **Parameter Selection**: The API request was configured with specific parameters to retrieve historical weather data. The primary parameters included:
    - **key**: Our API key for authentication.
    - **city_id**: The identifier for the location of interest (e.g., airport locations such as JFK, ORD).
    - **start_date** and **end_date**: The date range for which historical data was required. We aligned these dates with the flight dates from our datasets.
    - **units**: We chose the metric system for consistency in data measurement.
4. **Data Retrieval**: We executed the API requests via Postman, which returned the weather data in JSON format. The data included various weather attributes such as temperature, precipitation, wind speed, and humidity—factors that can significantly impact flight schedules.
5. **Data Integration**: The retrieved weather data was parsed and integrated into our main dataset using Python. This involved mapping the weather conditions to corresponding flight dates and locations, ensuring that each flight record in our dataset was enriched with accurate historical weather conditions for both departure and arrival airports.
6. **Quality Checks**: To ensure the reliability of our data, we conducted several checks for consistency and completeness. Any anomalies or missing data points were addressed by re-fetching the data for the affected dates.

The integration of weather data, acquired through Weatherbit.io's API, further enhanced the richness of our dataset by incorporating crucial meteorological factors. This meticulous process of merging data based on corresponding dates ensures the alignment of flight and weather data, facilitating more accurate analysis and insightful conclusions.

Data Preprocessing

In the initial phase of data preprocessing, we initiated by filtering the dataset according to the specific flight numbers, departure and arrival airports as per our requirements. Subsequently, we categorized the arrival status of flights into distinct categories: early, late, and on time, enabling a more granular analysis of punctuality. Following this, we conducted a thorough cleanup process, removing extraneous columns from the dataset to streamline further analysis. Finally, leveraging the power of pandas' get_dummies function, we transformed categorical variables into integer columns, facilitating seamless integration into machine learning models.

This meticulous data preprocessing journey not only ensured the extraction of relevant flight information but also optimized the dataset for subsequent analytical tasks. By strategically filtering flight data based on specific criteria and categorizing arrival statuses, we enhanced the dataset's utility and interpretability. The removal of unnecessary columns further refined the dataset, eliminating redundancy and reducing computational overhead. Finally, the conversion of categorical variables into integer columns via "get_dummies" enriched the dataset with numerical representations, paving the way for robust modeling and insightful analysis.

Model Training

We divided the data into training and testing sets during the model-training phase, assessing both scaled and unscaled versions. To ascertain the maximum accuracy, four machine learning algorithms—Logistic Regression, Gradient Boosting Classifier, Random Forest, and XGBoost—were used. After testing, the results of each algorithm were compared to the training set of data.

Logistic Regression

Logistic Regression was utilized as one of the algorithms in our evaluation process. Its simplicity and interpretability make it a popular choice for binary classification tasks.

Gradient Boosting Classifier

The Gradient Boosting Classifier, known for its ensemble learning technique, was another algorithm assessed. It builds sequential trees, iteratively correcting the errors of the previous models, often resulting in high predictive accuracy.

Random Forest

Random Forest, a versatile ensemble learning method, was also included in our analysis. By constructing multiple decision trees and combining their outputs, Random Forest mitigates overfitting and delivers robust performance.

XGBoost Classifier

XGBoost, an optimized implementation of gradient boosting, was the fourth algorithm examined. Its superior performance and efficiency make it a preferred choice for various machine learning tasks.

Scores of Each Model for Whole Dataset

1. **Random Forest Classifier**

   Accuracy: 0.5053191489361702
   Score:1.0

2. **Gradient Boosting Classifier**

   Accuracy: 0.5053191489361702
   Score: 0.928

3. **XGBoost**

   Accuracy:0.43617021276595747
   Score: 1.0

4. **Logistic Regression**

   Accuracy: 0.48936170212765956
   Score: 0.5426666666666666

Latter Flights Data

In the latter flights dataset, we initially filtered the data to distinguish between first and second flights. By segmenting the dataset into these categories, we ensured that each flight record was properly labeled. Subsequently, we merged the filtered data to establish a clear connection between each second flight and its corresponding first flight. This merging process enabled us to provide context for each second flight by linking it to its preceding flight. Once this relational structure was established, we proceeded with preprocessing the data. This involved encoding categorical variables and conducting other necessary preprocessing steps to prepare the dataset for further analysis or modeling

After preparing the dataset appropriately, we used all four models for training. The model that demonstrated the highest accuracy among these was XGBoost, indicating that it was effective in interpreting the subtleties of the flight data and producing precise predictions. This result emphasizes how crucial careful model selection and data preparation are to attaining peak performance. Our method proved to be effective as we were able to extract significant insights and obtain improved predicted accuracy by iteratively refining the dataset and utilizing advanced modeling approaches like XGBoost.

Scores of Each Model for Latter Flights Dataset

1. **Random Forest Classifier**

   Accuracy:0.3181818181818182
   Score:1.0

2. **Gradient Boosting Classifier**

   Accuracy:0.36363636363636365
   Score: 0.928

3. **XGBoost**

   Accuracy: 0.4090909090909091
   Score: 1.0

4. **Logistic Regression**

   Accuracy: 0.3181818181818182
   Score: 0.5426666666666666

Initial Prediction

Predictions for First Flights

Step 1: **Data Extraction and Pre-processing**
Data Extraction: We began by extracting relevant features from our testing dataset, which included scheduled departure times, historical delay patterns, and current and historical weather conditions at both the departure and arrival airports.
Pre-processing: The data underwent a series of pre-processing steps to ensure compatibility with our model. This included encoding categorical variables, handling missing values, and normalizing or scaling numerical inputs to enhance model performance.

Step 2: **Model Selection**
Based on extensive testing and validation on a separate development dataset, the Random Forest Classifier was selected for its superior performance, particularly its ability to handle diverse data types and complex interactions between variables.

Step 3: **Predictions Using Pre-trained Model**
We loaded the pre-trained Random Forest Classifier, which had been optimized on historical data to predict flight statuses accurately. Using this model, we predicted whether each flight would arrive early, on time, or be delayed, based on the pre-processed test data.

Predicting initial test data using pre-trained model - Random Forest Classifier

```
y1_pred = rf_classifier.predict(test_initials)
#y1_pred = gb.predict(test_initials)
#y1_pred = xgb_classifier.predict(test_initials)
#y1_pred = log_reg.predict(test_initials)
y1_pred
```

```
59]: array([0, 0, 2, 2, 0, 2, 0, 0, 2, 2, 2, 2, 0, 0, 2, 0, 2, 2, 0, 0, 2, 0,
            2], dtype=int64)
```

```
label_mapping = {
    0: "Early",
    1: "On Time",
    2: "Late"
}
```

```
predictions = np.vectorize(label_mapping.get)(y1_pred)
predictions
```

```
61]: array(['Early', 'Early', 'Late', 'Late', 'Early', 'Late', 'Early',
            'Early', 'Late', 'Late', 'Late', 'Late', 'Early', 'Early', 'Late',
            'Early', 'Late', 'Late', 'Early', 'Early', 'Late', 'Early', 'Late'],
           dtype='<U5')
```

Predictions for Latter Flights

Step 1: **Data Extraction and Pre-processing**
Data Extraction: For the latter flights, we extracted data similar to the first flights but added additional features, such as the outcomes of the corresponding earlier flights, considering their potential influence on subsequent flight statuses.
Pre-processing: The dataset for the latter flights received similar pre-processing treatments as the first flights, with particular attention given to integrating the status of the first flights effectively into the model input features.

Step 2: **Model Selection**
The XGBoost Classifier was chosen for predicting the latter flights due to its demonstrated efficacy in handling sequential data inputs and its robustness in higher-dimensional spaces, crucial for incorporating the status of earlier flights.

Step 3: **Predictions Using Pre-trained Model**
We deployed the pre-trained XGBoost Classifier, previously trained and validated against a subset of historical flight data. This model utilized the prepared dataset to forecast the arrival status (early, on-time, or delayed) of the latter flights, taking into account both external conditions and the outcomes of the first flights.

Predicting initial test data of latter flights using pre-trained model - XGBoost classifier

```
Latter_Flight_weather['Arr_Status1'] = 0
y1_pred0 = xgb_classifier.predict(Latter_Flight_weather)

Latter_Flight_weather['Arr_Status1'] = 1
y1_pred1 = xgb_classifier.predict(Latter_Flight_weather)

Latter_Flight_weather['Arr_Status1'] = 2
y1_pred2 = xgb_classifier.predict(Latter_Flight_weather)

y1_pred = np.column_stack((y1_pred0, y1_pred1, y1_pred2))
y1_pred
```

```
]: array([[1, 1, 1],
          [1, 1, 1],
          [1, 1, 2],
          [2, 2, 2],
          [2, 2, 2],
          [2, 2, 2],
          [1, 1, 1],
          [1, 1, 1],
          [2, 2, 2],
          [0, 0, 0],
          [0, 0, 0]])
```

```
predictions = np.vectorize(label_mapping.get)(y1_pred)
predictions
```

```
array([['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'Late'],
       ['Late', 'Late', 'Late'],
       ['Late', 'Late', 'Late'],
       ['Late', 'Late', 'Late'],
       ['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'On Time'],
       ['Late', 'Late', 'Late'],
       ['Early', 'Early', 'Early'],
       ['Early', 'Early', 'Early']], dtype='<U7')
```

Final Prediction

Predictions for First Flights

Step 1: **Data Extraction and Pre-processing**

Data Extraction: We re-extracted the relevant data from our testing dataset to ensure we captured any updates or corrections to the data fields.

Pre-processing: The pre-processing steps were re-applied, this time incorporating adjustments based on insights gained from the initial prediction outcomes, such as recalibrating thresholds for weather-related features and tweaking the handling of outlier values.

Step 2: **Model Re-evaluation and Modification**

After reviewing the performance of the initial predictions, we made targeted modifications to the Random Forest Classifier to better align with the actual outcomes observed. Adjustments included refining the feature selection and tuning the model parameters to optimize prediction accuracy.

Step 3: **Final Predictions Using Modified Model**

With the refined model, we conducted the final round of predictions. The updated Random Forest Classifier was employed to predict the arrival status of the first flights using the latest pre-processed data, aiming to leverage the enhanced model capabilities for improved accuracy.

Predicting final test data using pre-trained model - Random Forest Classifier

```
y2_pred = rf_classifier.predict(test_finals)
#y2_pred = gb.predict(test_finals)
#y2_pred = xgb_classifier.predict(test_finals)
#y2_pred = Log_reg.predict(test_finals)

y2_pred
```

```
array([0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 2, 0, 2, 2, 2, 2, 2,
       2], dtype=int64)
```

```
predictions = np.vectorize(label_mapping.get)(y2_pred)
predictions
```

```
array(['Early', 'Early', 'Early', 'Late', 'Early', 'Early', 'Early',
       'Early', 'Early', 'Early', 'Late', 'Early', 'Early', 'Late',
       'Late', 'Early', 'Late', 'Late', 'Late', 'Late', 'Late', 'Late',
       'Late'], dtype='<U5')
```

Predictions for Latter Flights

Step 1: **Data Extraction and Pre-processing**

Data Extraction: Similarly, for the latter flights, we re-extracted all relevant features, ensuring the inclusion of the actual statuses of the corresponding earlier flights.

Pre-processing: The pre-processing adjustments were particularly focused on accurately integrating the outcomes of the earlier flights into the dataset, which is crucial for the predictive accuracy of the latter flights.

Step 2: **Model Re-evaluation and Modification**

The XGBoost Classifier was reassessed and modified based on the actual values and performance metrics from the initial predictions. Modifications were focused on enhancing the model's sensitivity to the sequence of events, especially how earlier flight statuses impact the latter flight predictions.

Step 3: **Final Predictions Using Modified Model**

Employing the modified XGBoost Classifier, the final predictions for the latter flights were generated. The refined model used the updated dataset, which now included the adjusted features and model parameters, to predict whether each flight would be early, on-time, or delayed, with an emphasis on accuracy and reliability.

Predicting final test data of latter flights using pre-trained model - XGBoost classifier

```
Latter_Flight_weather['Arr_Status1'] = 0
y2_pred0 = xgb_classifier.predict(Latter_Flight_weather)

Latter_Flight_weather['Arr_Status1'] = 1
y2_pred1 = xgb_classifier.predict(Latter_Flight_weather)

Latter_Flight_weather['Arr_Status1'] = 2
y2_pred2 = xgb_classifier.predict(Latter_Flight_weather)

y2_pred = np.column_stack((y1_pred0, y1_pred1, y1_pred2))
y2_pred
```

```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 2],
       [2, 2, 2],
       [2, 2, 2],
       [2, 2, 2],
       [1, 1, 1],
       [1, 1, 1],
       [2, 2, 2],
       [0, 0, 0],
       [0, 0, 0]])
```

```
predictions = np.vectorize(label_mapping.get)(y2_pred)
predictions
```

```
array([['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'Late'],
       ['Late', 'Late', 'Late'],
       ['Late', 'Late', 'Late'],
       ['Late', 'Late', 'Late'],
       ['On Time', 'On Time', 'On Time'],
       ['On Time', 'On Time', 'On Time'],
       ['Late', 'Late', 'Late'],
       ['Early', 'Early', 'Early'],
       ['Early', 'Early', 'Early']], dtype='<U7')
```

Conclusion

In conclusion, this machine learning project aimed to predict the arrival status of flights into Syracuse based on historical data, initially focusing solely on flight data and subsequently incorporating historical weather data for enhanced accuracy. Through meticulous data preprocessing, feature engineering, and model training, we developed a robust predictive model capable of categorizing flights as "Early," "On Time," or "Late." While our initial predictions provided valuable insights, the integration of weather data significantly improved the model's predictive capabilities. By leveraging both flight and weather information, we gained a more comprehensive understanding of the factors influencing flight schedules, allowing for more precise predictions and proactive measures to mitigate potential delays.

References

[1] On-Time Index page. (n.d.). https://www.transtats.bts.gov/ontime/

[2] Postman https://learning.postman.com/docs/introduction/overview/

[3] Weatherbit.io https://www.weatherbit.io/api/historical-weather-daily

[4] T. Chen and C. Guestrin, "XGBoost: a Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pp. 785–794, 2016, doi: https://doi.org/10.1145/2939672.2939785.

[5] "(PDF) Random Forests and Decision Trees," *ResearchGate*. https://www.researchgate.net/publication/259235118_Random_Forests_and_Decision_Trees