Introduction to Machine Learning
CIS 662

Flight Delay Prediction

Niranjan Balsubramani	nibalasu@syr.edu
Preet Karia	pkaria@syr.edu
Spoorthi Jayaprakash Malgund	smalgund@syr.edu
Parth Pramod Kulkarni	pkulka11@syr.edu

Table of Contents

Abstract

This project develops a predictive model to determine the timeliness—categorized as early, on-time, or late—of flights arriving in Syracuse (SYR) from Chicago (ORD), New York (JFK), and Orlando (MCO). Utilizing a dataset comprising approximately 7,475 records from the Bureau of Transportation Statistics (BTS) for the period from January 2022 to December 2023, this study integrates essential flight information with hourly weather data sourced from visualcrossing.com. The inclusion of weather data is crucial, as it allows for an in-depth examination of environmental impacts on flight schedules.

Extensive data preprocessing was performed to ready the dataset for analysis, which included handling missing values, applying one-hot encoding to categorical variables, and employing Principal Component Analysis (PCA) for dimensionality reduction. PCA was specifically implemented to explore whether a reduction in feature space could lead to improved model accuracy.

The predictive modeling involved advanced machine learning algorithms such as Gradient Boosting, XGBoost, and Random Forests, with a focus on using regularization techniques to prevent overfitting. This project employs a classifier model, and performance was rigorously evaluated based on accuracy, alongside precision, recall, and F1-score to ensure a comprehensive assessment. This study demonstrates a robust analytical framework for predicting flight arrival times by integrating comprehensive flight and hourly weather data, laying a solid foundation for future enhancements to boost prediction reliability and applicability.

Data Collection

For our predictive modeling project, we meticulously selected two primary sources of data to ensure the accuracy and relevance of our predictions for flight arrival times.

Flight Data Collection

The flight data was sourced from the Bureau of Transportation Statistics (BTS)[1] through their comprehensive online database. We specifically accessed data spanning from January 2022 to December 2023, aligning with our objective to utilize the most recent and relevant information for our analysis. This timeline was chosen to avoid the atypical travel patterns affected by the COVID-19 pandemic during 2020 and 2021, ensuring our model is trained on data reflective of normal travel conditions. From the BTS, we extracted detailed flight-specific variables which are critical in predicting flight timeliness. These fields include:

- **Date and Time Identifiers**: MONTH, DAY_OF_WEEK, and FL_DATE
- **Airline and Flight Identifiers**: MKT_UNIQUE_CARRIER, MKT_CARRIER_FL_NUM, OP_UNIQUE_CARRIER, and OP_CARRIER_FL_NUM
- **Departure and Arrival Specifics**: ORIGIN, DEST, CRS_DEP_TIME, CRS_ARR_TIME
- **Flight Status Indicators**: ARR_DELAY, CANCELED, and DIVERTED
- **Delay Reasons and Operational Details**: WEATHER_DELAY, NAS_DELAY, LATE_AIRCRAFT_DELAY, and DIV_AIRPORT_LANDINGS

Fields such as DEP_TIME, ARR_TIME, TAXI_OUT, TAXI_IN, WHEELS_OFF, and WHEELS_ON were excluded since these would not be available for future flights.This selective approach ensures we focus on variables that directly impact flight delays and operational efficiencies.

Weather Data Collection

Given the significant impact of weather conditions on flight delays, we integrated hourly weather data from Visual Crossing[2]. This platform provided comprehensive historical and forecast weather data which we collected for the same period, January 2022 to December 2023, covering the airports in Syracuse, Orlando, Chicago, and New York. Our focus was on ensuring that the weather data corresponded accurately with the flight data in terms of timing and location, enhancing the predictive accuracy of our model by accounting for environmental variables. The collected flight and weather data were meticulously integrated using common keys such as date, time, and location. This integration enabled us to create a robust dataset where each flight record is enhanced with corresponding weather conditions, providing a holistic view of the factors influencing each flight's timeliness.

Data Preprocessing

In this section, we detail the preprocessing steps undertaken to format and filter the flight data for effective analysis. The dataset, spanning from January 2022 to December 2023, was loaded and preprocessed.

Filtering Flight Data:

Filtered the flight data to include only flights by the marketing airlines (AA, B6, DL, UA, WN), originating from (MCO, ORD, JFK) and destined for (SYR).

Now we had **7475** rows of flight data.

Flight Date and Time Processing:

Dates were converted to *datetime* format. Created **2** *datetime* columns `SCH_ARR_TIME` and `SCH_DEP_TIME` derived from `FL_DATE` (had only Date), `CRS_DEP_TIME` (had only departure time 24 hour format), `CRS_ARR_TIME` (had only arrival time in 24 hour format).

Then dropped `FL_DATE, CRS_DEP_TIME, CRS_ARR_TIME.`

Filtering Late-Night Flights (Flights arriving next day):

To focus the analysis on relevant flights, late-night flights arriving at Syracuse Airport on the next day were filtered out.

This left us with **6969** rows of flight data.

Merging Flight and Hourly Weather Data:

We are combining weather and airline data, needing both origin and destination weather information.

Steps for Merging:

- **Load Weather Data:** Load two data frames for weather data, one for origin and one for destination.
- **Column Renaming:** Rename origin columns with prefix `ORGIN_WTH_` and destination columns with prefix `DEST_WTH_`.
- **Merge Origin and Destination Weather:** Merge origin and destination weather data one by one.
- **Create Join Columns:** Add join columns in both datasets.
- **For Origin:**

    - Flight data `ORGIN_WTH_JOIN` = `SCH_DEP_TIME` (rounded to nearest hour) + `ORIGIN`
    - Origin Weather data `ORGIN_WTH_JOIN` = `ORGIN_WTH_datetime` + `name`
- **For Destination:**
    - Flight data `DEST_WTH_JOIN` = `SCH_ARR_TIME` (rounded to nearest hour) + `DEST`
    - Destination Weather data `DEST_WTH_JOIN` = `DEST_WTH_datetime` + `name`
- **Final Merge:** Join origin and destination data sequentially. Drop the join columns added previously.

Shape of the merged dataset: **(6969, 45)**

More Data Filtering and Dropping Columns:

Filtered the dataset on following condition:

- No diversions (`DIV_AIRPORT_LANDINGS` = 0)
- No cancellations (`CANCELLED` = 0)
- No duplicated flights (`DUP` = 'N')
- No actual diversions (`DIVERTED` = 0)

Removed irrelevant or redundant columns:
['OP_CARRIER_FL_NUM', 'CANCELLED', 'DUP', 'OP_CARRIER_FL_NUM',
'DIV_AIRPORT_LANDINGS', 'DISTANCE', 'WEATHER_DELAY', 'NAS_DELAY',
'LATE_AIRCRAFT_DELAY', 'CRS_ELAPSED_TIME', 'ORGIN_WTH_feelslike',
'ORGIN_WTH_windgust', 'DEST_WTH_feelslike', 'DEST_WTH_windgust', 'DEST',
'DEST_WTH_preciptype', 'ORGIN_WTH_preciptype', 'DIVERTED', 'DEST_WTH_conditions',
'ORGIN_WTH_conditions', 'ARR_TIME']

Final shape of merged data set **(6773, 25)**

Handling Missing Values

Checked for missing values in the filtered dataset. Filled missing values in the `ORGIN_WTH_severerisk` and `DEST_WTH_severerisk` columns with their respective minimum values.

Now the `former_flight_data.csv` is generated.

These are the columns

['DAY_OF_WEEK', 'MKT_UNIQUE_CARRIER', 'OP_UNIQUE_CARRIER', 'ORIGIN',
'ARR_DELAY', 'SCH_DEP_TIME', 'SCH_ARR_TIME', 'ORGIN_WTH_temp',

'ORGIN_WTH_precip', 'ORGIN_WTH_precipprob', 'ORGIN_WTH_snow',
'ORGIN_WTH_windspeed', 'ORGIN_WTH_winddir', 'ORGIN_WTH_cloudcover',
'ORGIN_WTH_visibility', 'ORGIN_WTH_severerisk', 'DEST_WTH_temp',
'DEST_WTH_precip', 'DEST_WTH_precipprob', 'DEST_WTH_snow',
'DEST_WTH_windspeed', 'DEST_WTH_winddir', 'DEST_WTH_cloudcover',
'DEST_WTH_visibility', 'DEST_WTH_severerisk']

`ARR_DELAY` is our **y.**

Data for Latter Flight Prediction model

For latter flights prediction model, we need one additional feature -- status of the former flight

Steps to add column – `FORMER_FLIGHT_STATUS`

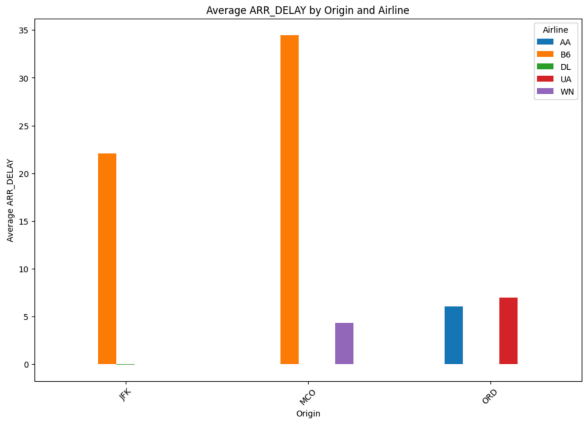
- Things we consider:
  - For any given flight -- `FORMER_FLIGHT_STATUS` = Status of the preceding flight **just before the given flight** on the **same day** and **same origin - destination**.
- We first sort the data, according to the scheduled arrival time. We reset the index after that.
- Then for every given row, we figure it's `FORMER_FLIGHT_STATUS` based on the above consideration.

Now `latter_flight_data.csv` is generated

These are the columns:

['DAY_OF_WEEK', 'MKT_UNIQUE_CARRIER', 'OP_UNIQUE_CARRIER', 'ORIGIN',
'ARR_DELAY', 'SCH_DEP_TIME', 'SCH_ARR_TIME', 'ORGIN_WTH_temp',
'ORGIN_WTH_precip', 'ORGIN_WTH_precipprob', 'ORGIN_WTH_snow',
'ORGIN_WTH_windspeed', 'ORGIN_WTH_winddir', 'ORGIN_WTH_cloudcover',
'ORGIN_WTH_visibility', 'ORGIN_WTH_severerisk', 'DEST_WTH_temp',
'DEST_WTH_precip', 'DEST_WTH_precipprob', 'DEST_WTH_snow',
'DEST_WTH_windspeed', 'DEST_WTH_winddir', 'DEST_WTH_cloudcover',
'DEST_WTH_visibility', 'DEST_WTH_severerisk', 'FORMER_FLIGHT_STATUS']

EDA



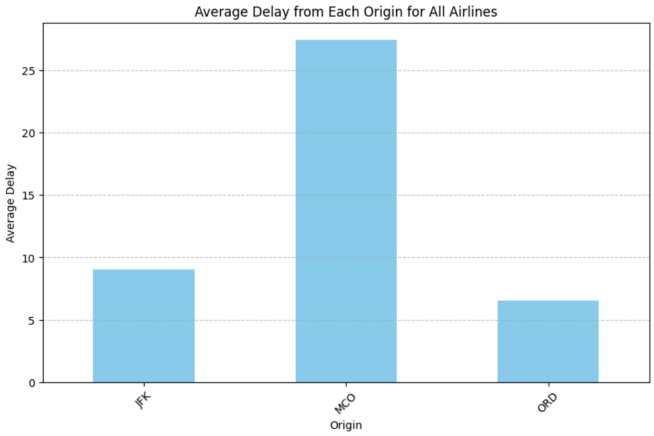


Fig 1.1 Average ARR_DELAY by Origin and Airline from all airlines

Fig 1.2 Average Delay from Each origin

In Fig 1.1, the bar chart shows the average arrival delay (ARR_DELAY) for flights from three different airports (represented as "Origin": JFK, MCO, and ORD), and involving different airlines (represented by the colors and legend: AA, B6, DL, UA, WN). Each bar represents the average delay for an airline at a specific airport.

- **JFK Airport:** The orange bar shows that airline B6 has a very high average delay, around 25 minutes and the green bar is hardly visible i.e. the airline DL has average delay of around 2 mins.
- **MCO Airport:** Airline B6 is represented by the orange bar, indicating an average delay of about 35 minutes whereas the purple bar (WN airline) shows an average delay of 5 mins.
- **ORD Airport:** The blue and red bars represent airlines AA and UA respectively, with average delays of about 6 minutes for AA and 8 minutes for UA.

The chart effectively highlights differences in delay times by both airline and airport origin. Notably, airline B6 at airport MCO has the highest average delay. Additionally, the lack of bars for some airline-airport combinations suggests there are no available data points.

We utilized this graph to determine which airport and airline combination experiences the highest average arrival delay (ARR_DELAY). By visually comparing the delay times represented by the bars, we can easily identify the specific airline and airport where delays are most pronounced. This analysis is crucial for understanding performance differences across airline services at various airports and helps in pinpointing areas where improvements might be necessary

In Fig 1.2, the graph displays the average delay for all airlines combined at three different airport origins: JFK, MCO, and ORD. Each bar represents the aggregated average delay in minutes for flights departing from these airports.

- **JFK**: The bar for JFK shows an average delay of around 10 minutes.

- **MCO**: This airport has a significantly higher average delay, depicted by a tall bar that extends to about 25 minutes.
- **ORD**: The bar for ORD indicates an average delay of approximately 10 minutes, similar to JFK.

The purpose of this graph is to illustrate the comparative performance of different airports regarding flight departure delays, offering a clear view of which airport experiences the most significant delays on average across all airlines. Here, MCO stands out as having the highest average delay

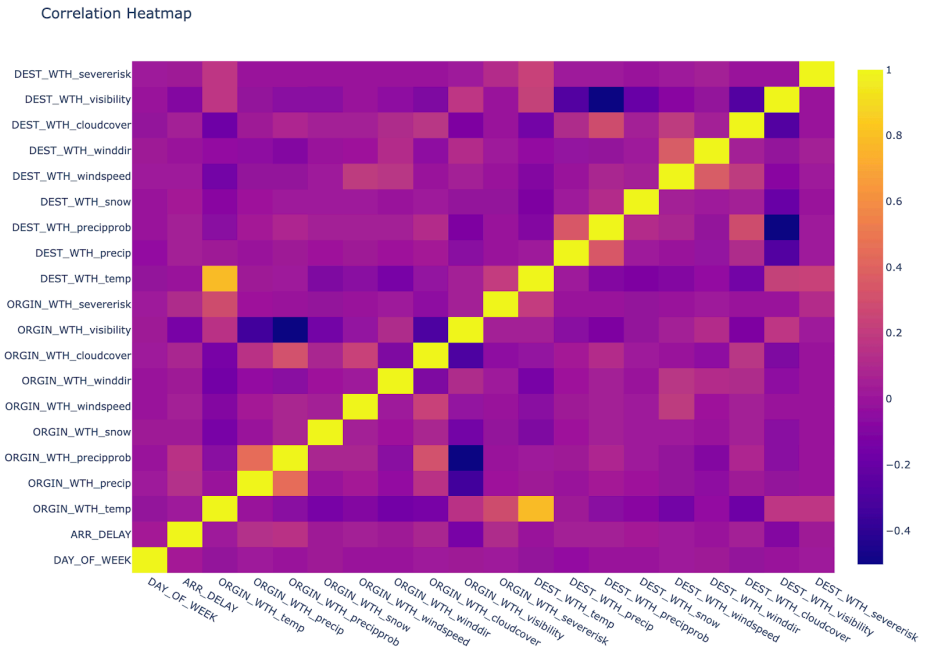


Fig 1.3: Heatmap (Correlation map with 25 columns)

In Fig 1.3, The correlation heatmap is used to represent the correlation coefficients between various variables. Each cell in the heatmap shows the correlation between two variables, indicated by the row and column headings. The colors range from purple (indicating a negative correlation) to yellow (indicating a positive correlation), with the strength of the correlation reflected by the color intensity.

Key observations in the heatmap:

1. **Strong Positive Correlations:**

- Variables related to the same weather condition at the origin and destination airports tend to show strong positive correlations. For instance, wind speed at the origin (`ORGIN_WTH_windspeed`) is positively correlated with wind speed at the destination (`DEST_WTH_windspeed`), as reflected by the yellow cells.

- Snow and temperature conditions between origin and destination also display similar correlations, suggesting that weather conditions tend to be similar across the route of the flight.

2. **Strong Negative Correlations:**

- Some variables show less intense negative correlations (darker purple cells), but there are no extreme negative correlations observed in this heatmap. This suggests that there are no direct inversely proportional relationships between the measured variables that are very strong.

3. **Impact on Flight Delays (`ARR_DELAY`):**

- `ARR_DELAY` has some visible correlations with weather conditions at both the origin and the destination, such as temperature and snow, though these correlations are not very strong. This could imply a moderate influence of adverse weather conditions on flight delays.

4. **Day of the Week (`DAY_OF_WEEK`):**

- The `DAY_OF_WEEK` seems to have very little correlation with weather conditions and `ARR_DELAY`, which is shown by the predominantly purple color in its row and column. This indicates that the day of the week does not significantly impact these variables.

Considering some other potentially significant correlations observed in the heatmap:

1. **Visibility Correlations:**

- The correlation between visibility at the origin (`ORGIN_WTH_visibility`) and the destination (`DEST_WTH_visibility`) is notably positive. High visibility in similar routes can contribute to smoother operations and potentially fewer delays due to fewer navigation or landing complications.

2. **Cloud Cover Correlations:**

- There is a clear correlation between cloud cover at the origin and destination (`ORGIN_WTH_cloudcover` and `DEST_WTH_cloudcover`). This could be important for flight operations, as extensive cloud cover can affect flight schedules and routing.

3. **Wind-related Factors:**

- Wind direction correlations (`ORGIN_WTH_winddir` with `DEST_WTH_winddir`) are also visible. Consistent wind directions along a flight route can impact fuel efficiency and flight time.

- Wind speed shows a strong positive correlation between origin and destination, which is crucial for flight planning, especially in terms of estimating travel time and potential turbulence.

4. **Temperature Variations:**

- Temperature at the origin and destination (`ORGIN_WTH_temp` and `DEST_WTH_temp`) has a strong correlation. Similar temperatures can indicate stable weather conditions along the route, affecting the overall ease of operations and passenger comfort.

5. **Precipitation and Severe Weather Risks:**

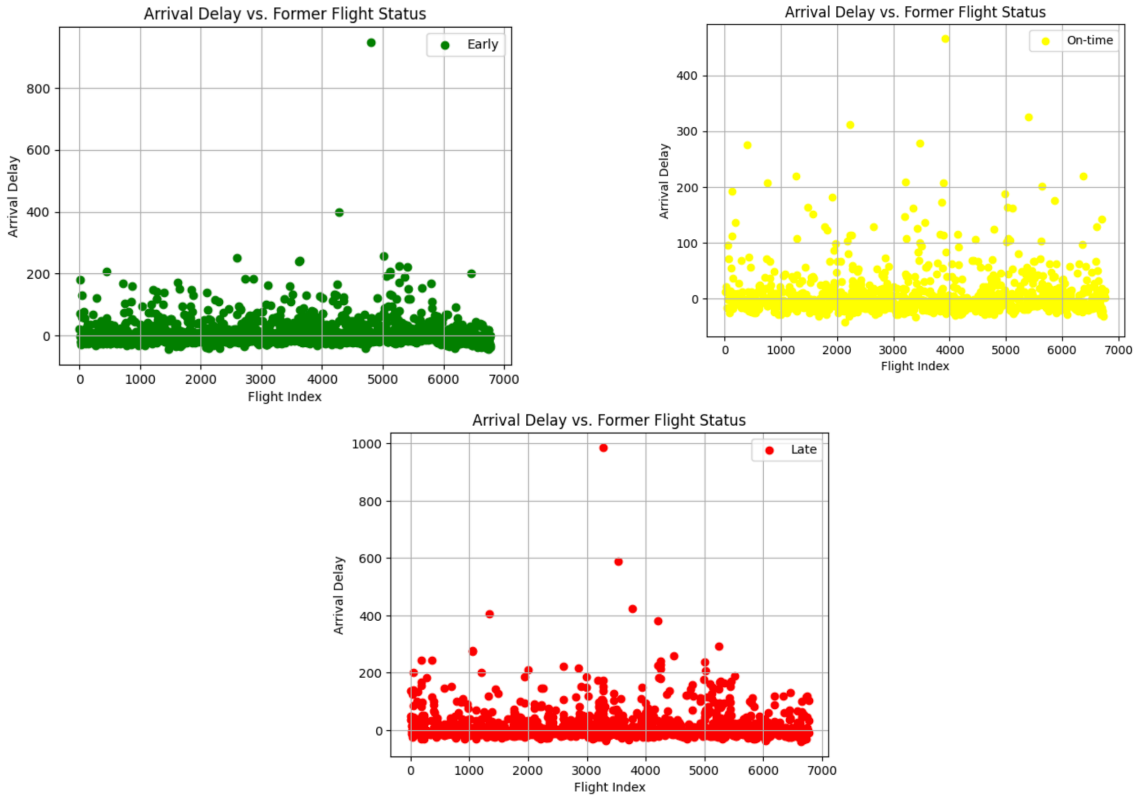
- Precipitation probability (`ORGIN_WTH_precipprob` with `DEST_WTH_precipprob`) and actual precipitation (`ORGIN_WTH_precip` with `DEST_WTH_precip`) show significant correlations.

These correlations are essential for understanding risks associated with wet runways and visibility issues.

- Severe risk factors (`ORGIN_WTH_severerisk` with `DEST_WTH_severerisk`) also correlate well. This metric is crucial for anticipating severe weather disruptions that could lead to major delays or rerouting of flights.

These correlations can provide airlines and airport operations teams with critical insights into planning and managing flights, particularly under varying weather conditions.

Fig 1.4: Arrival Delay vs. Former Flight Status

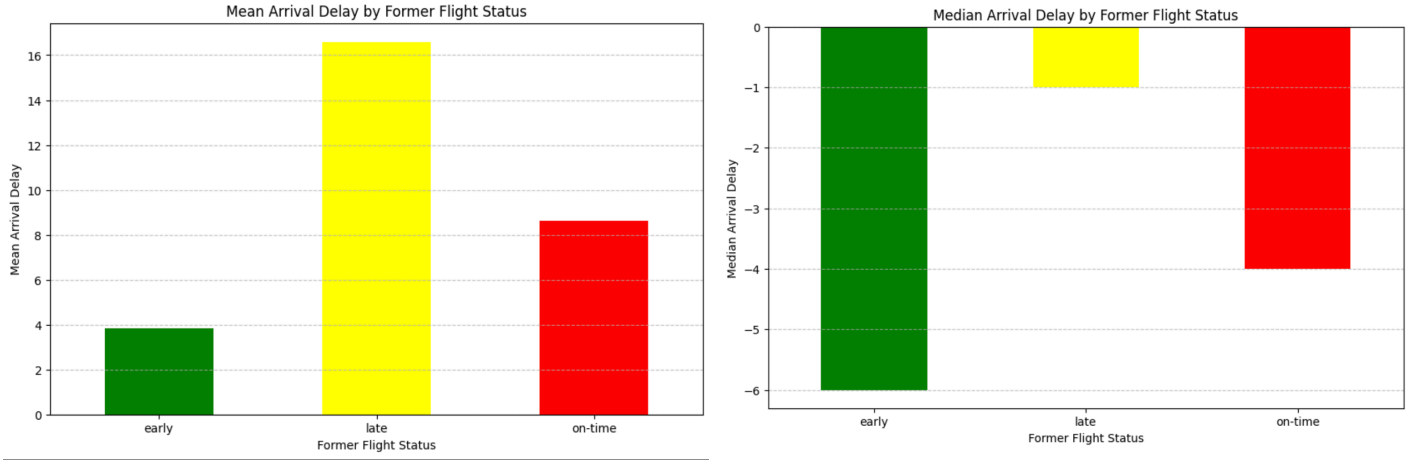


The bar chart titled "Mean Arrival Delay by Former Flight Status" in Figure 1.5(a) shows:

- **Early Arrivals:** Flights categorized as 'early' have the lowest mean arrival delay, which is depicted by a short green bar, indicating minimal delay or possibly arriving ahead of schedule.
- **Late Arrivals:** The 'late' category, represented by a yellow bar, unsurprisingly shows a significantly higher mean arrival delay. This aligns with the expectation that these flights were late, leading to longer average delays.
- **On-Time Arrivals:** The 'on-time' status shows an interesting trend with a high mean delay depicted by a red bar. Although these flights were considered 'on-time' for departure, they still accumulated a considerable amount of delay by the time they arrived.

Median Delays Bar chart: The chart in Figure 1.5(b) displays three bars, each corresponding to a different flight status:

- **Green Bar (Early)**: Represents flights that left early. Interestingly, the median delay is negative, around -6 minutes, indicating these flights generally arrived earlier than scheduled.
- **Yellow Bar (Late)**: Corresponds to late departures with a median delay around -1 minutes, which suggests that despite being late, these flights managed to make up time en route.
- **Red Bar (On-Time)**: Shows a median delay of nearly -4 minutes for flights that departed on schedule, indicating they typically arrived ahead of their scheduled time.



(a) (b)

Figure 1.5: Mean and Median Arrival Delay by Former Flight Status

Feature Selection

In the development of our predictive model for flight arrival times, effective feature selection was crucial to optimize model accuracy and efficiency. This section outlines our approach to selecting, modifying, and discarding features based on extensive Exploratory Data Analysis (EDA) and the specific requirements of our modeling techniques.

Initial Data Considerations

Our initial dataset included a broad range of features, sourced from BTS and integrated with hourly weather data from Visual Crossing. From January 2022 to December 2023, we collected variables that we identified as potentially influential to flight timeliness. Certain features such as **ORGIN_WTH_severerisk** and **DEST_WTH_severerisk** were initially considered but dropped post-EDA, as they showed minimal impact on our target variable, **ARR_DELAY**.

Similarly, real-time flight specifics like **DEP_TIME** and **WHEELS_OFF** were excluded because they would not be available for predictive scenarios of test data.

Categorical Variables Identification

Some features, even though they look quantitative are categorical – eg. Month. In determining which variables to treat as categorical, we used a systematic approach based on the number of unique values in each column from the train data. For this, we iteratively checked the uniqueness of data within each column — fewer than 12 unique values. These will be potentially categorical variables. This threshold was established after noticing that columns with limited unique entries typically represent discrete categories or have a restricted range of ordinal values, which are best managed through categorical encoding techniques. Variables such as **DAY_OF_WEEK**, airline identifiers like **MKT_UNIQUE_CARRIER** and **OP_UNIQUE_CARRIER**, airport codes (**ORIGIN**), and transformed weather probabilities (**ORGIN_WTH_precipprob** and **DEST_WTH_precipprob**) were thus encoded categorically to allow the model to effectively interpret these as discrete data points, enhancing the accuracy and efficiency of our predictions.

```
categorical_vars = {}
if True:
    for col in flight_data.columns:
        unique_col_vals = flight_data[col].unique()
        if(len(unique_col_vals) < 12):
            categorical_vars[col] = unique_col_vals
categorical_vars
```

```
{'DAY_OF_WEEK': array([1, 2, 3, 4, 5, 6, 7]),
 'MKT_UNIQUE_CARRIER': array(['AA', 'UA', 'B6', 'DL', 'WN'],
dtype=object),
 'OP_UNIQUE_CARRIER': array(['MQ', 'G7', 'B6', '9E', 'OO', 'PT', 'WN',
'UA', 'YX', 'ZW'],
```

```
      dtype=object),
 'ORIGIN': array(['ORD', 'JFK', 'MCO'], dtype=object),
 'ORGIN_WTH_precipprob': array([  0, 100]),
 'DEST_WTH_precipprob': array([  0, 100])}
```

Correlation Analysis and Final Adjustments

A correlation matrix was generated to identify and eliminate features that provided overlapping or irrelevant information. This included dropping **ORGIN_WTH_temp**, **DEST_WTH_temp**, and weather probability indicators post-binarization, which simplified the model without sacrificing predictive power. We chose **pd.get_dummies()** for encoding categorical variables, specifically because it effectively handles the categorical data present in our dataset and ensures that our model can be applied reliably to future data without the risk of encountering new, unseen categories.

Model Training

Model Preparation

The final feature set was used to train a series of **GradientBoostingRegressor** models, initially keeping ARR_DELAY as a numerical target to establish baseline model performance. The model's effectiveness was assessed through metrics such as Mean Absolute Error, which stood at 25.61, and the error ratio, at 2.78. The sMAPE score for this model was 72.10, indicating room for improvement in subsequent iterations. These predictions were way off and the MAE of the model was not that promising. But with this, we had a fundamental idea about the features we have to use going further while building the model. The initial insights guide the subsequent iterations of feature engineering and model refinement to enhance predictive accuracy and reliability.

Transition from Regression to Classification

Initially, our predictive efforts utilized a GradientBoostingRegressor model to predict **ARR_DELAY** as a continuous variable. However, after an initial evaluation, we decided to reframe the problem into a classification task. This decision was based on the intuition that classification models could provide more actionable insights into flight status (early, on-time, or late) and might be more robust against outliers which are common in delay data. To facilitate this, ARR_DELAY was transformed into categorical labels ('early', 'on-time', 'late') using a custom function that categorized delays based on a threshold of five minutes.

Classifier Training

After preprocessing and encoding the data as described in the "Feature Selection" section, we proceeded with the classification approach. We first predicted only for the former flight status, and once we had a base model, we then proceeded with predicting for latter flights. The latter flight dataset had an extra feature called “FORMER_FLIGHT_STATUS” which has the predicted values from the first model.

To enhance the performance of our XGBoost classifier and ensure its robustness, we employed a systematic approach to hyperparameter tuning using GridSearchCV, a powerful tool provided by the scikit-learn library.

Hyperparameter Tuning with GridSearchCV

The primary objective of using GridSearchCV was to explore a range of combinations across several hyperparameters that influence the behavior and performance of the XGBoost model. The parameters we focused on included:

- min_child_weight: Helps control overfitting

- learning_rate: Shrinks the feature weights to make the boosting process more conservative
- max_depth: Limits the maximum depth of the trees and controls overfitting

For our grid search, we defined a grid of possible values for each parameter:
- min_child_weight: [1, 2, 3]
- learning_rate: [0.1, 0.01, 0.005]
- max_depth: [1, 2, 3, 4]

Additionally, we set n_estimators to 600 and reg_lambda to 0.007, configuring the model for a substantial number of boosting rounds and applying L2 regularization to improve model performance and reduce overfitting.

GridSearchCV was configured to use a 7-fold cross-validation, ensuring that each combination of parameters was adequately evaluated against different subsets of the data. This approach not only helps in assessing the effectiveness of different parameter settings but also mitigates the risk of model bias to specific data peculiarities. We also set the scoring parameter to 'accuracy' to focus on overall prediction correctness, which is sufficient for our dataset as opposed to medical datasets where metrics might differ and importance must be given to False Positives / False Negatives. Other metrics like 'f1_macro' could also be considered to better handle class imbalances. GridSearchCV was used only to get an initial set of parameters that tend to give better results, but later on tried with different values manually.

We employed a GradientBoostingClassifier, initially tuning it with broad hyperparameters:
- min_samples_split: 2
- min_samples_leaf: 6
- max_depth: 1
- n_estimators: 200

The training process involved fitting the GradientBoostingClassifier on the training set and validating its performance on the test set using accuracy as the primary metric and feature importances to understand which variables most influenced the predictions.

We used a prediction counter script to compare predicted flight timings against actual flight timings from the past 10 days, which helped in validating and selecting the final model and its parameters.

Post-training, the model's performance was evaluated based on its accuracy on the test set, which provided a straightforward measure of its effectiveness in classifying flight statuses correctly. Additionally, we analyzed the feature importances generated by the model, which helped in identifying the most influential factors affecting flight delays. Features like weather

conditions, time of departure, and airline-specific variables were found to be significant predictors.

Regularization and Pruning

To further refine the model, regularization techniques were employed during the training phase. These techniques helped in pruning the decision trees within the GradientBoostingClassifier, reducing the complexity of the model, and enhancing its performance by eliminating overfitting.

Eventually we chose the XGBClassifier model with the following hyperparameters:

- eta: 0.006
- Min_child_weight: 4
- max_depth: 2
- n_estimators: 1000
- reg_alpha: 0.001

In our project, we explored a variety of models including **GradientBoostingRegressor**, **BaggingClassifier**, **RandomForestClassifier**, **GradientBoostingClassifier**, and **XGBClassifier**, to identify the most effective approach for predicting flight delays. Ultimately, the **XGBClassifier** was chosen for its superior ability to handle imbalanced data, a common challenge in flight delay datasets. We also experimented with applying **Principal Component Analysis (PCA)** to the **XGBClassifier** to reduce dimensionality and potentially enhance accuracy. However, this did not yield significant improvements, indicating that the original features were crucial for maintaining predictive accuracy. This exploration helped us confirm that while PCA can be beneficial for simplifying models, its applicability may vary depending on the dataset's characteristics and the information content of the features.

Making Final Predictions

In the final stage of our project, we focused on applying our predictive model to make final predictions using new data:

- **Data Integration**: We utilized the test dataset provided on Blackboard, complementing it with the latest weather forecast data obtained from Visual Crossing.
- **Data Transformation and Preprocessing**: To ensure consistency between training and test datasets, we transformed the test data to match the format of our training model. This involved applying the same preprocessing steps to the test data that we had used for our training data, ensuring that both datasets had identical features and structures.
- **Final Predictions and Output**: Once the test data was appropriately formatted and the model parameters were optimized, we performed the final predictions. The results were then compiled into a CSV file, presenting a clear outcome of our predictive analysis.

Results

Model Used	**Former Flight Model Accuracy %**		**Latter Flight Model Accuracy %**	
	Train	**Test**	**Train**	**Test**
BaggingClassifier	100	52.69	100	49.20
RandomForestClassifier	73.14	53.57	72.76	52.69
GradientBoostingClassifier	57.58	51.36	60	51.84
XGBClassifier	**57.43**	**51.95**	**59.70**	**52.05**
XGBClassifier with PCA	43.90	-	44.37	-

* XGBClassifier with PCA was tested on the entire dataset, test/train split was not made.

Based on the above results, Bagging and RandomForestClassifier seems to have the high accuracy but the difference between train and test accuracy was drastically different compared to the other models which depicts that it might have overfitted the data.
Both GradientBoosting and XGBClassifier had similar accuracies but XGBClassifier was chosen because XGB classifier's feature importances seemed to be more accurate and representative of the dataset that we are working with.

Conclusion

In this project, we decided to use the XGB Classifier because it handles imbalanced data well and avoids overfitting, making it ideal for predicting flight timeliness. Despite trying PCA to simplify our model, it did not improve our results, likely because it removed important details that help predict flight delays accurately. Interestingly, incorporating weather data did not significantly impact our predictions, suggesting that operational factors might be more influential in causing flight delays. This insight is crucial for directing future improvements in our predictive modeling efforts.

References

1. "Data Download Tool." *Bureau of Transportation Statistics*, U.S. Department of Transportation, Accessed 2 May 2024, https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGK&QO_fu146_anzr=b0-gvzr.

2. "Weather History." *Visual Crossing*, Visual Crossing Corporation, Accessed 2 May 2024, https://www.visualcrossing.com/weather-history/.

3. Zhang, J., Peng, Z., Yang, C., Wang, B.: Data-driven flight time prediction for arrival aircraft within the terminal area. *IET Intell. Transp. Syst.* 16, 263–275 (2022). https://doi.org/10.1049/itr2.12142

4. Meshal Alfarhood, Rakan Alotaibi, Bassam Abdulrahim, Ahmad Einieh, Mohammed Almousa, Abdulrhman Alkhanifer, "Predicting Flight Delays with Machine Learning: A Case Study from Saudi Arabian Airlines", *International Journal of Aerospace Engineering*, vol. 2024, Article ID 3385463, 12 pages, 2024. https://doi.org/10.1155/2024/3385463