**MIE368 Final Report**


December 4, 2024

Group 12

| Name | Student ID |
|---|---|
| Keunjoo Oh | 1005935749 |
| Jovan Cui | 1006717634 |
| Berkin Kadayifci | 1006551269 |
| Reezwan-Us Sami | 1007767141 |

## 1. Introduction

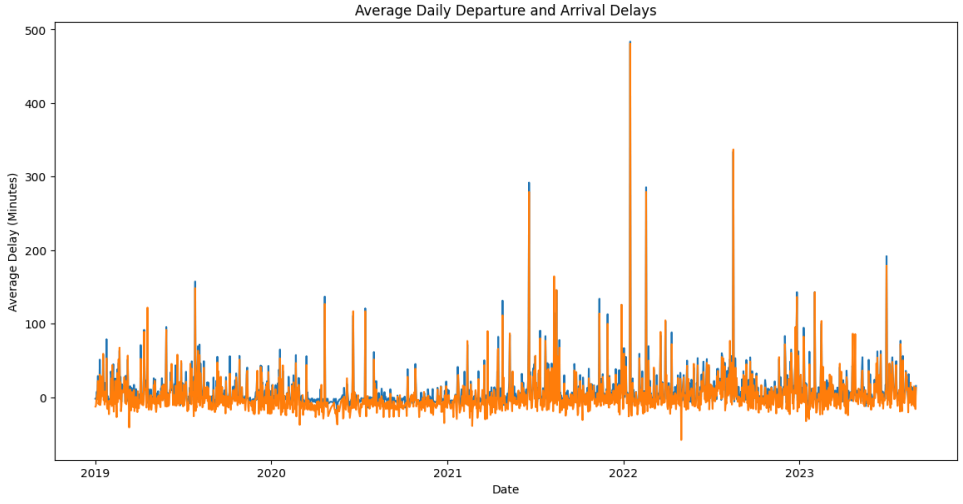
The current aviation industry faces a serious challenge as flight delays contribute to more greenhouse gas emissions, intensifying climate change and global warming. Data shows that the amount of CO2 emission generated by flight disruptions are equivalent to the annual emissions of around 2 million cars [1]. In turn, the growing frequency of extreme weather events then worsen flight delays, thus creating a vicious cycle. This issue is further compounded by a post-pandemic surge in air travel demand, leading to busier flight schedules and more frequent disruptions [2].

To address this, the project aims to improve the understanding of contributing factors to flight delays and enhance the precision of delay forecasts. By developing a predictive model for delay times in U.S. domestic flights, the team seeks to provide actionable insights for airlines and airports to optimize operations and mitigate disruptions, ultimately breaking the vicious cycle of global warming.

## 2. Data

For training and testing the model, the team decided to combine two databases: the US flight database and US weather database. The US flight database, spanning from 2018 to 2023, includes information such as flight date, airline, airport locations, actual and scheduled departure times, a cancellation code indicating the main cause of delay, a canceled flag (indicating if the flight was delayed), a diverted flag (indicating if the flight was diverted), actual and scheduled elapsed times, and delay times categorized by general delay types, such as NAS (National Air System), security, late aircraft, and weather [3]. The time column is represented in the form of HH:MM. The US weather database contains a comprehensive collection of weather events data across 49 states in the United States from 2016 to 2022 [4]. The team selected a few features from the weather database which are weather event, severity, and precipitation, and used a left join to merge these data to flight databases on matching dates and locations, excluding data for 2023 for consistency, as the weather database does not cover that year.

*Figure 1: Diagram of Average Daily Delays*



Exploratory Data Analysis (EDA) is conducted in order to understand the data and derive a good predictive model. Irrelevant features, such as delay times with various categories, scheduled and actual departure time, were identified. The various categorical data presented in the database, such as region and weather, are valuable for the prediction model as they can capture distinct behavior.

The findings in Figure 1 showed that the day with the highest average departure delay was Jan 15th, 2022, with an average delay of 483.67 minutes. This day also has the highest average arrival delay, which is 481.00 minutes. Research shows that on that day, a severe winter storm was hitting across the U.S., particularly in the Southeastern region, implying the significant impacts of extreme weather events in winter times on flight delays, providing the team valuable insights to be further studied in following steps [5].

## 3. Methods

This section outlines the detailed steps performed to first clean and preprocess the data, train and evaluate predictive models, and conduct scenario-based simulations. These methods were designed to ensure data consistency, select the best-performing model, and explore the potential impact of various delay factors on flight delay times.

### 3.1 Data cleaning

First of all, the team conducted several preprocessing steps to the database for data cleaning purposes. The team split the date into “DayOfWeek”, “DayOfMonth”, “Month”, and “Year”, applying one-hot encoding to “DayOfWeek” and “DayOfMonth” while dropping Monday and January to avoid multicollinearity. The aforementioned irrelevant features, such as the time column, were removed. Delay times with various categories were converted into boolean variables, indicating whether a specific type of delay occurred, with a threshold of 180 minutes(3 hours)---the time commonly recommended for travelers to arrive early at the airport and check-in their flight. Lastly, NaN were replaced with 0, and categorical variables were also transformed into numerical values by assigning unique numbered labels to each category.

### 3.2 Model training

The target variable for the predictive model was defined as the delay time, calculated as the difference between actual elapsed time and scheduled elapsed time. The dataset was split into training (80%) and testing (20%) subsets. The team implemented and evaluated three kinds of models for comparison: Linear Regression with Lasso regularization, Random Forest (RF), and Classification and Regression Tree (CART). Model performance was assessed using Root Mean Square Error (RMSE) due to its interpretability and its ability to emphasize the larger errors that are potentially more disruptive.

In order to improve model accuracy and prevent overfitting, the parameters of random forest and CART were hypertuned. Random forest parameters are tuned by manually implementing reasonable parameters. The final parameters are shown in Table 1:

*Table 1: The List of Parameters of Random Forest*

| Parameters | random_state | max_depth | n_estimators | max_samples | bootstrap |
|---|---|---|---|---|---|
| Value | 2 | 10 | 150 | 0.8 | True |

For CART, Grid Search tuning with 5 cross validation is used to find the best parameters. The tested parameters are below:

Param_grid = {'max_depth': [10, 20, 50, None],

'min_samples_split': [2, 5, 10],

'min_samples_leaf': [1, 5, 10],

'max_features': ['sqrt', 'log2', None],

'criterion': ['gini', 'entropy']}

As a result, the selected best parameters are shown in Table 2:

*Table 2: The List of Parameters of CART*

| Parameters | random_state | max_depth | min_samples_split | min_samples_leaf | max_features | criterion |
|---|---|---|---|---|---|---|
| Value | 1 | 50 | 2 | 1 | None | entropy |

After the three models were trained, their performance for flight delay prediction was evaluated based on RMSE and the results are as follows:

RMSE_LR_L1 = 20.1757

RMSE_RF = 14.6054

RMSE_CART = 11.6922

The lowest RMSE indicates that the model performs better in predicting flight delays compared to other models. Among the evaluated models, CART achieved the lowest RMSE of 11.6922 and was therefore selected as the final predictive model for flight delay times. To further assess the accuracy of the predictions, the Mean Absolute Error(MAE) of the CART model was calculated, yielding a value of 3.8747. This MAE indicates that, on average, the predicted flight delay times deviate from the actual values by 3.87 minutes, demonstrating the high accuracy of the CART model.

3.3 What-if Scenarios Simulation

The team employed What-if scenarios simulation to explore the impact of altering specific factors on predicted flight delay time. This analysis systematically changes the values of certain features and observes the resulting changes in the model's predictions. As shown in

Appendix A, the team focuses on two primary types of scenarios: delay factors and weather types.

The delay factor scenarios investigate the influence of five factors: Carrier, Security, NAS, Weather, and Late Aircraft. Each factor is represented by a binary feature, indicating its presence or absence. The simulation quantifies the difference in predicted delay time when the factor is present compared to when it is absent.

Similarly, the weather type scenarios examine the effects of various weather conditions, including “Cold”, “Fog”, “Hail”, “Precipitation”, “Rain”, “Snow”, and “Storm”. The simulation compares the predicted delay time for each weather type’s presence against its absence, providing insights into their respective impacts.

To better illustrate the process, consider the scenario for the “Carrier_delayed” factor. The following approach was applied across all five delay factors and seven weather types, generating a comprehensive overview of their individual influences on flight delays.

The analysis begins by filtering the test data into two subsets: one where “Carrier_delayed” is present (value of 1) and one where it is absent (value of 0). Next, two modified datasets are created using the create_test_data function. The first dataset is based on the subset where “Carrier_delayed” was originally absent. In this dataset, “Carrier_delayed” is changed to 1, simulating the scenario where a carrier delay is introduced. The second dataset is based on the subset where “Carrier_delayed” was originally present. Here, “Carrier_delayed” is changed to 0, simulating the scenario where a carrier delay is removed.

These modified datasets are then used to obtain predictions from the trained CART model. The simulate_factor function calculates the difference between the actual mean delay time from the original test data and the predicted mean delay time for each simulated scenario. This difference quantifies the impact of the carrier delay on the predicted delay time.

## 4. Results

The simulation results for each scenario were analyzed to gain insights into the impact of various factors on flight delay times. The analysis highlighted which factors significantly influence delay times, particularly when specific delay causes are present. The findings in Appendix A were summarized and presented in Tables 3 and 4, illustrate these trends and provide a detailed analysis of the observed patterns in the data.

*Table 3: The Difference in Flight Delay Depending on the Occurrence of Delay Causes*

| Delay existence [0,1] | Delay Cause | | | | |
|---|---|---|---|---|---|
| | Carrier | Security | Weather | NAS | Late Aircraft |
| 1 | 5.63 | 7.49 | 6.03 | 9.09 | 6.15 |
| 0 | 6.02 | 5.70 | 5.75 | 2.92 | 5.80 |

Table 3 provides insights into the influence of different delay factors, particularly their existence (1) or absence (0), on the overall delay. National Air System (NAS) delays emerge as the most impactful when introduced, suggesting their critical role in predicting flight delays. Security delays also exert a notable influence, especially when they occur. In comparison, carrier, weather, and late aircraft delays exhibit more moderate but consistent effects.

Interestingly, the asymmetry between the existence and nonexistence of some delay factors suggests that the model reacts differently to their presence versus absence. This asymmetry may reflect underlying nonlinear relationships within the data or the model's predictions.

*Table 4: The Difference in Flight Delay Depending on the Occurrence of Weather Events*

| Weather Event Occurrence [0,1] | Delay Cause | | | | | | |
|---|---|---|---|---|---|---|---|
| | Cold | Fog | Hail | Precipitation | Rain | Snow | Storm |
| 1 | 7.46 | 8.39 | 9.36 | 10.29 | 11.29 | 12.19 | 13.20 |
| 0 | 9.26 | 6.44 | 17.91 | 4.27 | 6.01 | 0.20 | 12.26 |

Table 4 also highlights the role of weather events in causing delays. Events like snow, storm, rain, and hail stand out as the strongest contributors to delays, with the largest differences between their occurrence (1) and nonoccurrence (0). Snow and hail, in particular, demonstrate asymmetric effects, suggesting potential interactions with other variables in the model. Moderate contributors, including cold, fog, and precipitation, show smaller differences, indicating a lesser but still meaningful impact on delays.

The asymmetric impacts observed for some weather events, such as hail, reveal that their absence often has a greater influence on the model than their presence. This could be attributed to the rarity of these events, which might magnify their impact on delays when they do occur. These findings align with real-world data and predictive models, where delays are shaped by multiple interconnected factors, and systematic prediction errors reflect the complexity of these influences.

## 5. Discussion

The results of the simulation demonstrate the complex interactions between various factors contributing to flight delays. Delays categorized as part of the National Airspace System (NAS) often include operational issues that arise after a flight has left the gate but before it has taken off or during its journey. These delays could be caused by factors such as heavy air traffic, changes in air traffic control instructions, and waiting for clearance for the runway to take off [6]. Notably, NAS delays appeared as the most impactful among all delay types indicating the importance of systematic challenges in air traffic management and its significant effect on total delays. Moreover, the cost for airlines is negatively affected by these delays caused by the NAS, which also leads to increased fuel consumption and greater

greenhouse gas emissions. Detecting key elements that lead to NAS will improve airline efficiency and reduce the environmental footprint of aviation.
According to the FAA (Federal Aviation Administration), more than 75 percent of all air traffic delays of 15 minutes or more are caused by weather events, such as low visibility, hail, high winds at takeoff, and thunderstorms[7]. Our simulation findings demonstrated that severe weather events, such as snow, storms, rain, and hail, showed the largest differences in delay times when comparing scenarios where the weather event occurred versus when it did not occur. This indicates they were the most impactful weather event types for delays. With this analysis, airports can better allocate resources. For example, comprehensive de-icing equipment for ground crews can be used to take precautions for extreme weather conditions to minimize the delay time.

To recap on the weaknesses of the prediction models that potentially impacted the findings and hindered the accuracy, the team has identified some limitations, primarily due to the limited feature engineering such as encoding and transformation. There might be more complex interactions between features or external data sources that could improve accuracy. For example, airport-specific characteristics or air traffic control data could be valuable.

Similarly, there is room for improvement for the What-if analysis, as the team applied simplified assumptions and a limited scope. It primarily considers single-variable changes, potentially overlooking complex interactions between factors. The analysis also focuses on the presence or absence of delays rather than varying severity levels. These limitations could lead to an incomplete understanding of the model's behavior in diverse situations and hinder comprehensive scenario planning.

## 6. Conclusions and Future Directions

This study demonstrates the interconnectedness of operational and environmental factors contributing to flight delays, with NAS delays and severe weather events, such as snow and storms, emerging as significant contributors. By combining advanced predictive modeling with scenario-based simulations, actionable insights were developed to optimize airport operations and mitigate disruptions. For instance, the findings highlight the importance of prioritizing de-icing equipment for snow-prone airports and optimizing air traffic control during peak congestion hours.

Looking forward, several opportunities exist to expand the study. Future work could incorporate real-time air traffic and weather data to enhance model accuracy and practical applicability. Extending the scope to include international flights and assessing the influence of varying regulatory frameworks could provide a more global perspective, making this project more relevant and contemporary. Additionally, integrating environmental impact analysis would offer a holistic view of how delay mitigation strategies can reduce aviation's carbon footprint. By pursuing these directions, the aviation industry can enhance its resilience against delays, improve passenger satisfaction, and contribute to global sustainability efforts.

**References**

[1] “What exactly is the environmental impact of flight disruptions?,” AirHelp, https://www.airhelp.com/en/blog/what-exactly-is-the-environmental-impact-of-flight-disruptions/ Accessed: Dec. 4, 2024.
[2] “Air Travel Demand Is Breaking Records. Airline Profits Are Not.” *NBCNews.Com*, NBCUniversal News Group, 8 July 2024, www.nbcnews.com/business/business-news/air-travel-demand-breaking-records-airline-profits-are-not-rcna160663.
[3] Suhagiya, Het. “Flight_delays_cleaned.” *Kaggle*, 8 Mar. 2024, www.kaggle.com/datasets/hetsuhagiya/flight-delays-cleanded.
[4] Moosavi, Sobhan. “US Weather Events (2016 - 2022).” *Kaggle*, 23 May 2023, www.kaggle.com/datasets/sobhanmoosavi/us-weather-events.
[5] J. W. Childs, “Winter storm izzy: Hundreds of flights canceled, several states declare emergencies,” The Weather Channel, https://weather.com/news/news/2022-01-15-winter-storm-izzy-hundreds-of-flights-canceled Accessed: Dec. 4, 2024.
[6]“Types of delay - ASPMHelp.” https://aspm.faa.gov/aspmhelp/index/Types_of_Delay.html#:~:text=Delay%20that%20is%20within%20the,are%20also%20reported%20through%20OPSNET
[7]B. Berg, “How weather delays flights, the key factors explained - AFAR,” *AFAR Media*, Aug. 20, 2024. https://www.afar.com/magazine/weather-conditions-affecting-flights-delays#:~:text=According%20to%20the%20FAA%2C%20more,equally%20hindering%20effect%20on%20travel.

# Appendix A

Delay Factor

```
[ ]  import pandas as pd

     def create_test_data(df, is_delay, features, delay_type):
         # Create a copy of the DataFrame to avoid modifying the original
         df_copy = df.copy()
         df_copy[delay_type] = is_delay
         x = df_copy[features]
         y = df_copy[delay_type]
         return x, y

     def simulate_factor(df, features, factor, model):
         result = []

         # Filter rows based on the factor's condition
         delay_df = df.loc[df[factor] == 1]
         non_delay_df = df.loc[df[factor] == 0]

         # Create test data for delayed and non-delayed cases
         test_delay, y_not_delay = create_test_data(non_delay_df, 1, features, factor)
         test_not_delay, y_delay = create_test_data(delay_df, 0, features, factor)

         # Predict using the model
         simulate_delay = model.predict(test_delay)
         simulate_not_delay = model.predict(test_not_delay)

         # Calculate differences between actual and simulated means
         result.append(y_not_delay.mean() - simulate_delay.mean())
         result.append(y_delay.mean() - simulate_not_delay.mean())

         return result
```

```
# Simulating factors
carrier_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, "Carrier_delayed", cart_model)
print(f"Carrier delay exists: {carrier_result[0]:.2f}, Carrier delay does not exist: {carrier_result[1]:.2f}")

security_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, "Security_delayed", cart_model)
print(f"Security delay exists: {security_result[0]:.2f}, Security delay does not exist: {security_result[1]:.2f}")

NAS_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, "NAS_delayed", cart_model)
print(f"NAS delay exists: {NAS_result[0]:.2f}, NAS delay does not exist: {NAS_result[1]:.2f}")

weather_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, "Weather_delayed", cart_model)
print(f"Weather delay exists: {weather_result[0]:.2f}, weather delay does not exist: {weather_result[1]:.2f}")

late_aircraft_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, "Late_Aircraft_delayed", cart_model)
print(f"Late aircraft delay exists: {late_aircraft_result[0]:.2f}, late aircraft delay does not exist: {late_aircraft_result[1]:.2f}")
```

```
Carrier delay exists: 5.63, Carrier delay does not exist: 6.02
Security delay exists: 7.49, Security delay does not exist: 5.70
NAS delay exists: 9.09, NAS delay does not exist: 2.92
Weather delay exists: 6.03, weather delay does not exist: 5.75
Late aircraft delay exists: 6.15, late aircraft delay does not exist: 5.80
```

## Weather Type

```
[ ]  def create_test_data(df, is_delay, features, delay_type):
         # Create a copy of the DataFrame to avoid modifying the original
         df_copy = df.copy()
         df_copy[delay_type] = is_delay
         x = df_copy[features]
         y = df_copy[delay_type]
         return x, y

     def simulate_factor(df, features, value, factor, model):
         result = []

         # Filter rows based on the factor's condition
         delay_df = df.loc[df[factor] == value]
         non_delay_df = df.loc[df[factor] == 0]

         # Create test data for delayed and non-delayed cases
         test_delay, y_not_delay = create_test_data(non_delay_df, value, features, factor)
         test_not_delay, y_delay = create_test_data(delay_df, 0, features, factor)

         # Predict using the model
         simulate_delay = model.predict(test_delay)
         simulate_not_delay = model.predict(test_not_delay)

         # Calculate differences between actual and simulated means
         result.append(y_not_delay.mean() - simulate_delay.mean())
         result.append(y_delay.mean() - simulate_not_delay.mean())

         return result
```

```
# Simulating factors
cold_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, 1, "Type", cart_model)
print(f"Cold weather event exists: {cold_result[0]:.2f}, Cold weather event does not exist: {cold_result[1]:.2f}")

fog_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, 2, "Type", cart_model)
print(f"Fog weather event exists: {fog_result[0]:.2f}, Fog weather event does not exist: {fog_result[1]:.2f}")

hail_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, 3, "Type", cart_model)
print(f"Hail weather event exists: {hail_result[0]:.2f}, Hail weather event does not exist: {hail_result[1]:.2f}")

precipitation_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, 4, "Type", cart_model)
print(f"Precipitation weather event exists: {precipitation_result[0]:.2f}, Precipitation weather event does not exist: {precipitation_result[1]:.2f}")

rain_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, 5, "Type", cart_model)
print(f"Rain weather event exists: {rain_result[0]:.2f}, Rain weather event does not exist: {rain_result[1]:.2f}")

snow_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, 6, "Type", cart_model)
print(f"Snow weather event exists: {snow_result[0]:.2f}, Snow weather event does not exist: {snow_result[1]:.2f}")

storm_result = simulate_factor(pd.concat([X_test, y_test], axis=1), features, 7, "Type", cart_model)
print(f"Storm weather event exists: {storm_result[0]:.2f}, Storm weather event does not exist: {storm_result[1]:.2f}")
```

```
Cold weather event exists: 7.46, Cold weather event does not exist: 9.26
Fog weather event exists: 8.39, Fog weather event does not exist: 6.44
Hail weather event exists: 9.36, Hail weather event does not exist: 17.91
Precipitation weather event exists: 10.29, Precipitation weather event does not exist: 4.27
Rain weather event exists: 11.29, Rain weather event does not exist: 6.01
Snow weather event exists: 12.19, Snow weather event does not exist: 0.20
Storm weather event exists: 13.20, Storm weather event does not exist: 12.26
```