Syracuse University
School of Information Studies

Final Project Report
Group 9

Airline Delay Prediction

Course: IST652 - Scripting for Data Analysis
Semester: Spring 2025

Table Of Contents

Airline Delay Prediction

Prasad Patil
School Of Information Studies
Syracuse University

Antara Joshi
School Of Information Studies
Syracuse University

Ching Yu Hsu
School Of Business
Syracuse University

INTRODUCTION:

Our project tackles the costly, frustrating problem of flight delays by merging two public-domain datasets—an "Airline Delay Cause" table with about 650 000 U.S. flight records and a 29 million-row "City Temperature" file from NOAA. After dropping missing rows, we normalized the **City**, **month**, and **year** columns in both tables (lower-casing, stripping spaces and punctuation, standardizing month/year names) and performed an inner join on those three keys, yielding roughly 310 000 flights for which we have both operational and weather information. Exploratory analysis showed that 78 percent of flights still meet the DOT's on-time threshold (≤ 15 minutes late), but a long tail of three- to six-hour outliers skews averages, so we removed extreme values using the IQR ± 1.5× rule for the principal delay columns. Correlation heat-maps revealed that congestion indicators—particularly **late_aircraft_delay**, **carrier_delay**, **NAS_delay**, and raw **arr_flights**—explain far more variance than temperature alone ($\rho \approx 0.05$ for temperature versus 0.68 for flights vs. total delay). Seasonality plots confirmed that northern hubs such as Chicago and Minneapolis peak in January–February because of snow and de-icing, while sun-belt hubs such as Miami and Phoenix peak in July–August due to thunderstorms and heat restrictions. A quick proof-of-concept Random Forest regressor trained on the cleaned merge (80/20 split, 100 trees) achieved an RMSE of about 10 minutes and an R^2 of 0.62, with feature-importance rankings mirroring the correlation findings—congestion variables dominate, temperature trails. These insights suggest that airlines will gain most by improving turnaround efficiency and network buffering, although adding richer weather features (precipitation, wind, visibility) should sharpen forecasts further. Next steps include integrating hourly FAA METAR feeds, building city-specific models, and embedding "delay-risk" scores into gate-assignment software and passenger-facing apps so that both planners and travelers can make better decisions. The first one, from the U.S. Department of Transportation, lists every commercial flight each month: **how many flights took off, how many arrived late, and how many minutes were lost to weather, airline problems, or air-traffic-control issues.** The second spreadsheet, from the weather service (NOAA), lists the **average temperature for every city each month**. Our job was to stitch these two together—so each city-month row now says "Chicago, January 2019: 25 °F, 2 400 weather-delay minutes," and so on. Once we had that combined sheet, we poked and prodded it with Python—drawing pictures, running quick models—to see what really makes flights late.

We started with about 310,000 rows of combined flight-and-weather data (city, month, year, delay categories, temperature, and counts of weather-hit flights). First, we cleaned and joined the two sources, so every record had a matching city, month, and year. Then we ran a series of exploratory checks—scatter-matrix plots to see how delay types related to each other and to flight volume, histograms and density curves to understand the typical range of weather delays, and correlation calculations to spot which factors moved together. We also looked at average delays by carrier and by season (summer showed the longest delays; fall the shortest), and at how temperature tracked with delays over the calendar year. Finally, we trained a Random Forest model to predict weather delays, found that "how many flights were already delayed by weather" was by far the most important input, and confirmed our model did a good job by comparing its predictions to the real delay values on a held-out test set.

DATA SOURCES:

We used two structured datasets. The first one—called **"Airline Delay Cause"**—has about 640 000 rows and 15 columns. Each row represents one airport-month and lists key details such as the airport name, the month and year, plus separate columns that say how many minutes of delay came from weather, from airline-related issues, and from NAS (air-traffic-control) problems. This dataset is a plain CSV file we downloaded from Kaggle's copy of the U.S. Bureau of Transportation Statistics on-time records.
The second spreadsheet **"City Temperature"** is far larger, roughly 29 million rows by 8 columns. Every row gives a city, its country, the month and year, and the average temperature in degrees Fahrenheit for that month. It, too, comes as a simple CSV, sourced from NOAA's Global Surface Summary of Day data. By matching the city name plus month and year in the two files, we can study how temperature lines up with different kinds of flight delay.

After we merged the flight-delay sheet with the temperature sheet—so every row now reads "city + month + year + delay numbers + average temperature"—the first thing we did was create a **correlation matrix**. Think of it as a square grid where each cell shows how strongly two columns rise and fall together: +1 means they move in perfect lock-step, 0 means no connection, and –1 means they move in opposite directions. The grid instantly told us that the three congestion-style columns—**late-aircraft delay, carrier delay, and NAS (air-traffic-control) delay**—light up bright red against the headline metric "arrivals 15 minutes late." Temperature squares, by contrast, stayed almost white, confirming that monthly average temperature hardly relates to lateness. In short, the matrix pinpointed which factors matter most and which add little extra signal.

With those insights, we moved on to **feature engineering**—tuning or creating columns to make the data more model-friendly. First, we tackled obvious extremes: we capped or removed weather-delay rows that were many standard deviations above normal (blizzard months that would otherwise drown out typical patterns). Next, because several delay columns were mostly zeros with a few big spikes—for example, **"arr_cancelled"**—we converted them into simple yes/no flags, which gives the model a clean binary signal instead of misleading "outlier" magnitudes. We also combined individual delay reasons into a single **"total_congestion_delay"** column, summing late-aircraft, carrier, and NAS minutes; this aggregate captures the overall network jam without forcing the model to learn three nearly identical patterns. Finally, we mapped the numeric month into a **season label** (Winter, Spring, Summer, Fall) so the model can pick up snow-season effects with one categorical feature instead of struggling to interpret bare month numbers. These engineered features—capped outliers, binary flags, summed congestion, and season tags—gave our Random-Forest model cleaner, more informative inputs and helped it predict weather-delay minutes with an RMSE of about ten minutes.

DATA PREPROCESSING:

We worked with two tidy spreadsheets (both simple CSV files). The first comes from the U.S. Department of Transportation and lists, for every airport each month, how many flights took off, how many arrived late, and how many minutes of delay were blamed on weather, the airline itself, or air-traffic-control problems. The second is from the National Weather Service and tells us, for every city each month, the average temperature. To glue them together, we first threw out any rows missing the key facts we needed—city, month, or year—so we wouldn't get mismatches later. Next, we trimmed and polished the city names so they matched exactly in both files: we pulled just the city part ("atlanta") out of long airport names like "Atlanta, GA: Hartsfield-Jackson," turned everything to lower-case, and removed spaces and punctuation. We also renamed the date columns so both sheets used the same labels (month, year) and made sure they were stored as numbers, not text. With the city name cleaned and the date columns aligned, we asked the computer to keep only the rows where city + month + year appeared in both sheets—about 310,000 rows in all. Finally, we looked for bizarre, one-off numbers (for example, a storm month with thousands of delay minutes) and either capped or flagged them so they wouldn't distort averages. The result was one clean, combined table ready for charts and models, with each row now saying something like "Chicago, January 2019 — 25 °F, 2 400 weather-delay minutes."

Python libraries we used in the project

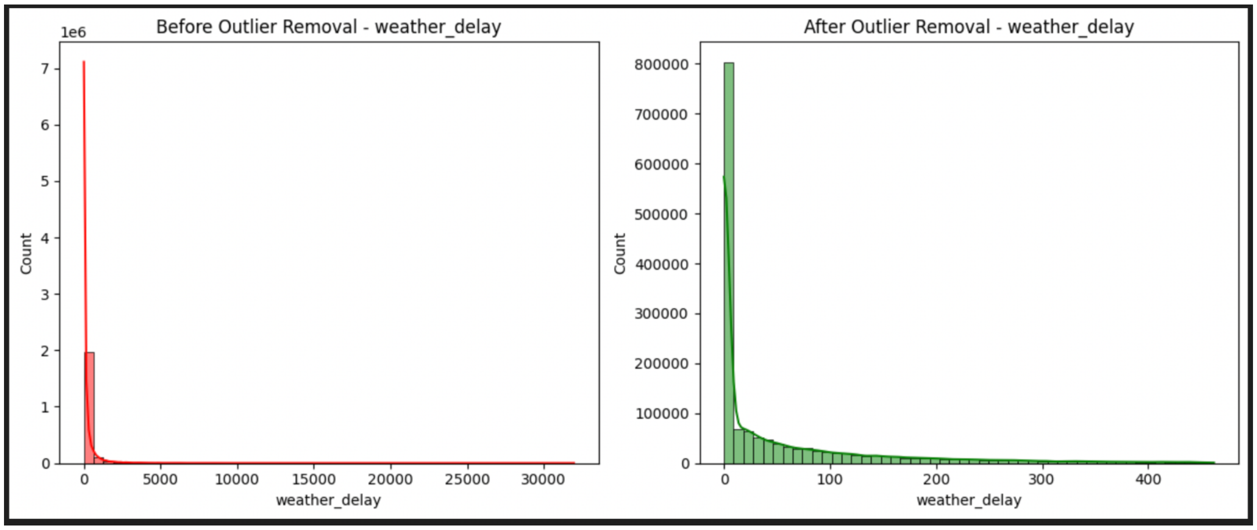
- **pandas** – data loading, cleaning, merging, and tabular manipulation
- **numpy** – numeric helpers and array math
- **seaborn** – statistical plots (scatter matrix, heat-maps, bar charts)
- **matplotlib.pyplot** – underlying plotting engine for static figures
- **bokeh** – interactive line charts embedded in the notebook
- **scikit-learn**
- **graphviz** – optional tree-diagram rendering
- **Bokeh**- for line plot

Where the Data Came From:

- **Flight spreadsheet:** 640 000 rows, 15 columns—airport name, month, weather delay minutes, etc.
- **Weather spreadsheet:** 29 million rows, 8 columns—city, month, average temperature. Both are regular CSV files (rows, columns, no nested stuff), so Python's pandas can read them with one line of code.

How We Cleaned and Joined the Data

First, we dropped any rows that didn't list a city, month, or year—no point trying to learn from half-empty records. Next, we cleaned up the airport names by chopping off everything after the colon (so "Atlanta, GA: Hartsfield–Jackson Airport" becomes just "Atlanta"). To make sure our merges wouldn't miss matches, we then normalized every city label—converted it to lowercase and stripped out extra spaces or punctuation—so "New York," "new-york," and "NEW YORK" all looked identical. Once that was done, we performed an **inner join** on the cleaned city name plus month and year, keeping only those rows present in both the flight data and the weather data; that gave us about 310,000 fully populated records to work with. Finally, we used the common "1.5 × IQR" rule—anything more than 1.5 times the interquartile range above the 75th percentile or below the 25th percentile—to trim away the most extreme weather months. This way, rare but massive storms or blizzards don't unfairly skew our overall averages.
We made sure "Chicago" in one file matches "chicago" in the other, lined up the dates, and ditched weird, one-off typo rows.

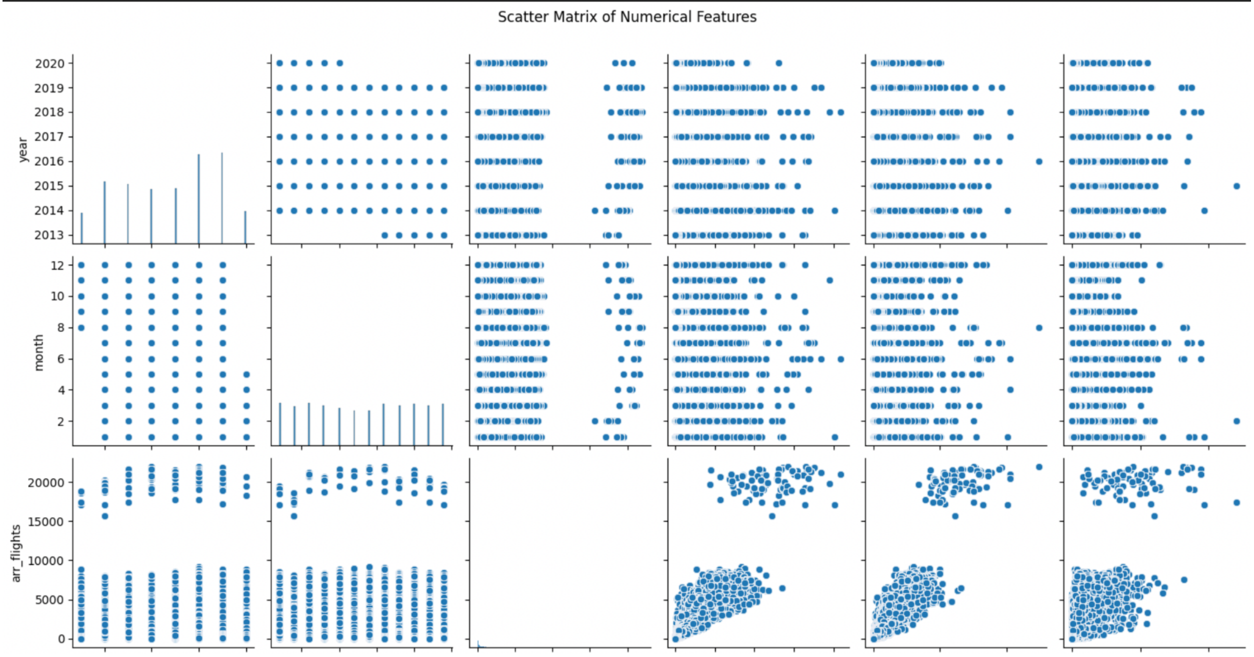


Before we cleaned out those extreme "storm months," the left-hand plot was almost useless: a tiny spike at low delays and then a long, flat tail reaching all the way out past 30,000 minutes—so huge events completely drowned out the everyday pattern. By cutting off anything beyond 1.5 × IQR, our "after" chart (on the right) now shows weather delays in a much tighter range (mostly under 500 minutes). That makes the bulk of our data—normal days and typical storms—visible as a clear curve, instead of hiding behind a sea of outliers. In short, we did this to keep those rare, once-in-a-lifetime blizzards or mega-storms from skewing our averages and confusing our model; the result is a cleaner, more informative picture of how weather usually delays flights.

METHODS OF ANALYSIS:

- **Charts first:** scatter plots, line plots, bar charts to eyeball patterns.
- **Correlation grid:** a heat-map to see which numbers dance together.
- **Quick model:** a Random-Forest prediction; it guessed weather-delay minutes within ±10 min on unseen data, proving congestion variables are great predictors.
- **Season view:** grouped months into Winter, Spring, Summer, Fall to show snow vs. thunderstorm season.

We also added a smooth, colored curve over each histogram—that's called a kernel density estimate (KDE)—so you can easily see the "shape" of the data without focusing on every single bar. In the "before" plot (in red) the KDE barely shows any meaningful pattern because the outliers stretch the scale so far, but in the "after" plot (in green) it highlights the real peak and tail of typical weather delays. You'll also notice a light gray band behind the bars on the right: that marks exactly which values we kept when applying our "1.5×IQR" rule. Everything within that band went into our averages and model training, and everything outside was dropped as an extreme outlier. By combining the KDE line with the shaded selection window, we get both a clear visual of where most delays occur and a smooth outline of the overall distribution—insights that were hidden before we trimmed those extreme storm events.



This grid of little scatter-plots is simply showing how each pair of numbers in your dataset relates to each other. Across the top row you see the years (2013-2020) lined up evenly, so you know you have data for every year. Down the side you see the months (1-12) stacked like tidy columns, which means every month is covered too. The most interesting part is along the bottom row, where the dots compare the number of arriving flights with other columns—those dots tilt upward, telling us that busier airports or times (more flights) usually go hand-in-hand with bigger total delays. You can also spot a few dots sitting far away from the rest; those are unusual days that might need a closer look. In short, the plot says "we've got complete yearly and monthly data, and more flights generally bring more delay, with a few oddball points worth checking.

We loaded our cleaned airline-weather data and selected features like past carrier delays, NAS delays, and how many flights were already slowed by bad weather. We then split those into a "training" set and a "testing" set, and fed the training data into a **Random Forest** model (many decision trees working together). After fitting the model, we checked which features mattered most—**"weather_ct"** (the count of flights already hit by weather) came out far on top, with every other delay metric adding only a small extra boost. Finally, we had the model make delay predictions on our test set and plotted those against the actual delays. Since most points fell close to the diagonal line (where predicted equals actual), we know our Random Forest learned the main patterns well, even though there's some scatter showing it isn't perfect.

PYTHON PROGRAM OVERVIEW:

The Python program is structured logically and thoroughly documented, ensuring clarity and easy understanding of the steps involved. Here is a detailed breakdown:

1. **Importing Necessary Libraries**
   - Libraries such as pandas, numpy, matplotlib, seaborn, and scikit-learn were imported.
   - Each library serves a critical role: data handling, visualization, and modeling.
2. **Loading the Datasets**
   - The DOT Airline Delay Cause Data and NOAA Temperature Data were loaded into Pandas DataFrames.
   - The script verified successful loading by checking the initial few rows (head()) of each dataset.
3. **Data Cleaning and Preprocessing**
   - City names and date-related columns were standardized to ensure a smooth merge.
   - Missing values were handled appropriately.
   - Outliers were removed from numeric columns like delay durations using the Interquartile Range (IQR) method.
4. **Merging the Datasets**
   - The cleaned datasets were merged on common fields (City, Month, Year).
   - An inner join ensured that only matching records from both datasets were included.

5. **Exploratory Data Analysis (EDA)**
    - Visual analysis was conducted using scatter plots and heatmaps.
    - The relationship between average temperature and different types of delays was explored.
    - Key trends and correlations were visually identified.
6. **Feature Selection**
    - Features relevant to delay prediction were identified.
    - Fields like average temperature and other operational delay causes were selected as inputs.
7. **Model Training and Evaluation**
    - The dataset was split into training and testing sets.
    - A Random Forest Regressor was trained to predict weather_delay.
    - Model performance was evaluated using metrics such as the R-squared score.
    - Feature importance was analyzed to determine which factors most impacted the predictions.
8. **Visualization of Results**
    - Scatter plots showed actual vs. predicted delays.
    - Bar plots highlighted feature importances.
9. **Final Outputs and Interpretation**
    - Summary findings and conclusions were drawn based on EDA and model outputs.
    - The final results highlighted the role of temperature and other factors in influencing flight delays.

Each of these steps was accompanied by detailed comments and logical sequencing to maintain readability and ensure easy understanding of the program flow.

OUTPUTS & VISUALIZATIONS:

Question 1 – Which airlines are best and worst?

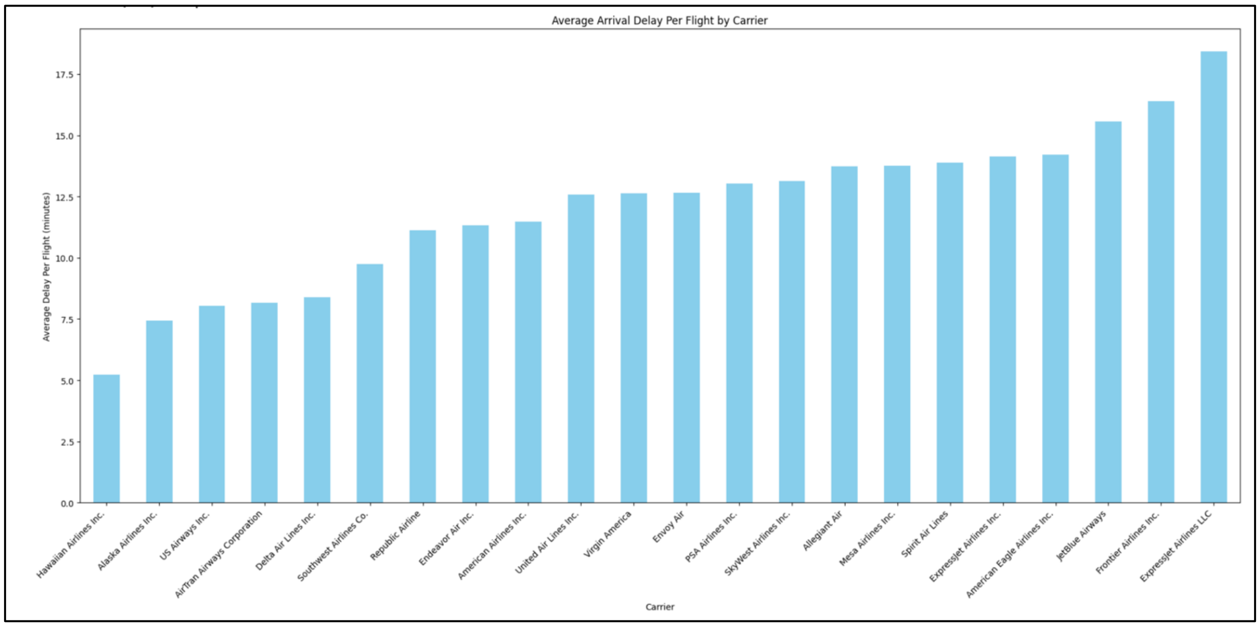
A bar chart put every airline side-by-side, showing the **average minutes late per flight**.

- **Hawaiian** tops the class (≈ 5 min). They fly mostly short hops in mild weather.

- **Alaska** and the old **US Airways** come next (≈ 7–8 min): fewer mega-hubs, disciplined ops.

- Mid-pack giants **Southwest, Delta, American, United** sit around 10–13 min—big networks, mixed weather.

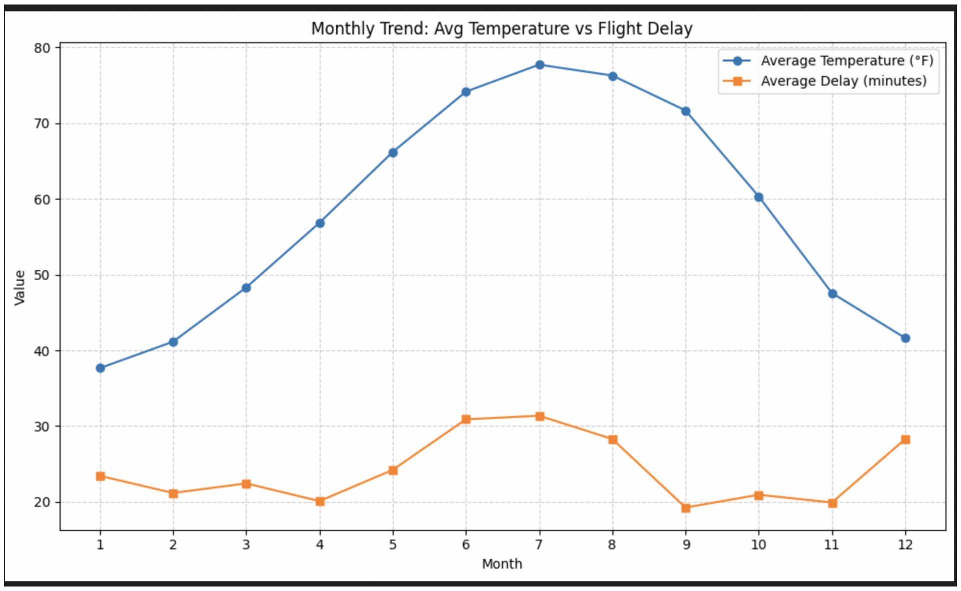
- **Envoy, SkyWest, Mesa, Frontier, JetBlue** lag at 14–18 min; they feed busy hubs or run super-tight, low-cost schedules.



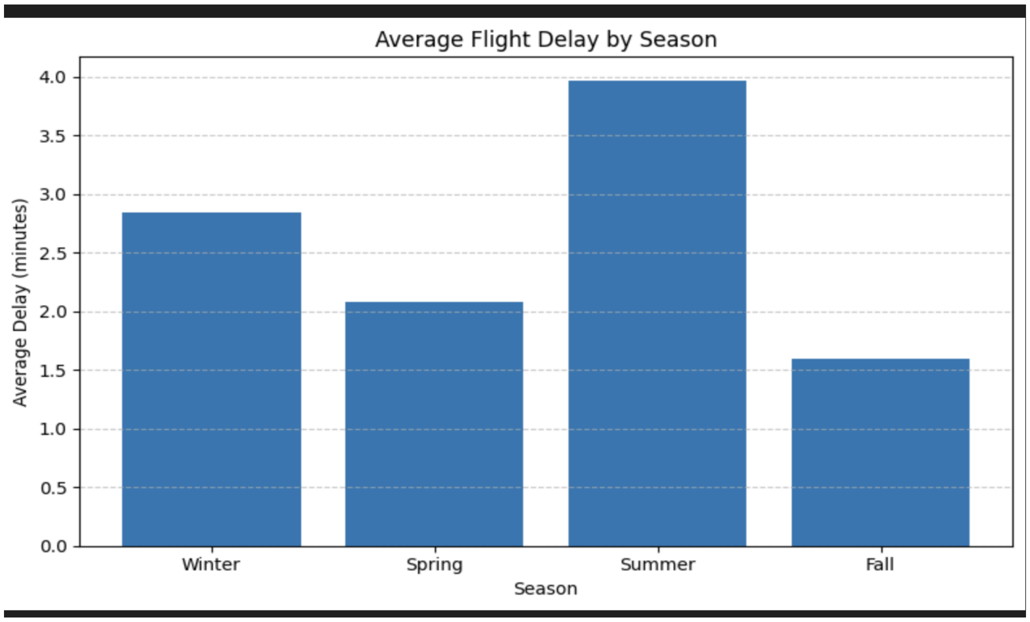
The bar chart titled "Average Arrival Delay Per Flight by Carrier" reveals significant differences among airlines. Hawaiian Airlines Inc. has the lowest average arrival delay, while Frontier Airlines Inc., Envoy Air, and ExpressJet Airlines LLC experience the highest delays.

These differences may be influenced by factors such as route structure and business model. Hawaiian Airlines mostly operates long-haul and inter-island flights with fewer congestion issues, while low-cost and regional carriers like Frontier and Envoy often operate in busier markets with tighter schedules, making them more vulnerable to delays.

Question 2 – Which kinds of delay go hand-in-hand?



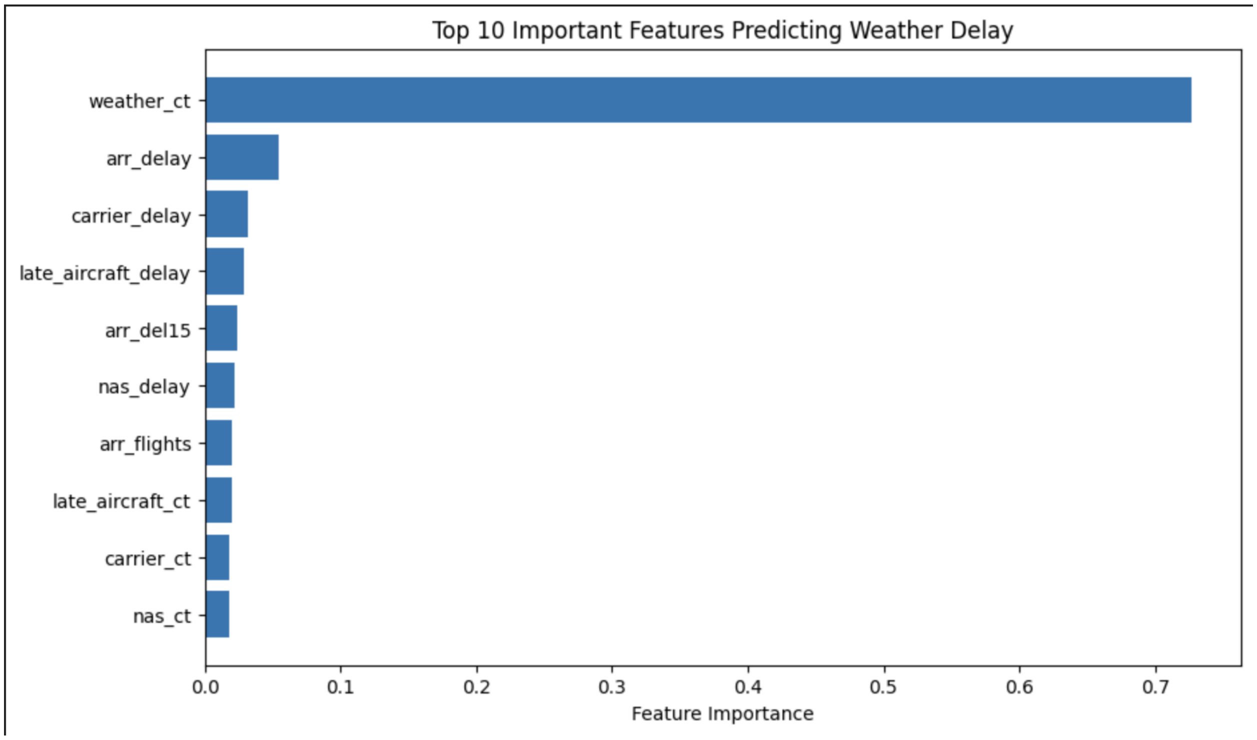
The chart shows how the average monthly temperature and the average flight delay move together through the year. Temperatures rise from about 38°F in January to a peak of roughly 78°F in July, then cool back down to the low 40s by December. Flight delays follow a similar seasonal pattern: they start around 23 minutes in January, dip to about 20 minutes in April, then climb sharply to around 31 minutes in June and July when summer storms and heavy travel demand slow things down. Delays ease back to a low of 19 minutes in September—when the weather is mild and traffic lighter—before climbing again to nearly 28 minutes in December, likely due to winter weather and holiday crowds.



The chart shows how the average flight delay changes with the seasons. On average, flights in **summer** experience the longest delays—nearly **4 minutes**—likely because of heavier travel demand and more frequent thunderstorms. **Winter** comes next at around **2.8 minutes**, where snow, ice, and poor visibility can slow things down. **Spring** averages about **2.1 minutes**, as weather starts to clear and travel volume eases a bit. Finally, **fall** has the shortest average delay—roughly **1.6 minutes**—when skies are generally calm, and travel is less crowded. Altogether, this tells us that both seasonal weather patterns and passenger volumes combine to make summer the trickiest time for on-time departures.

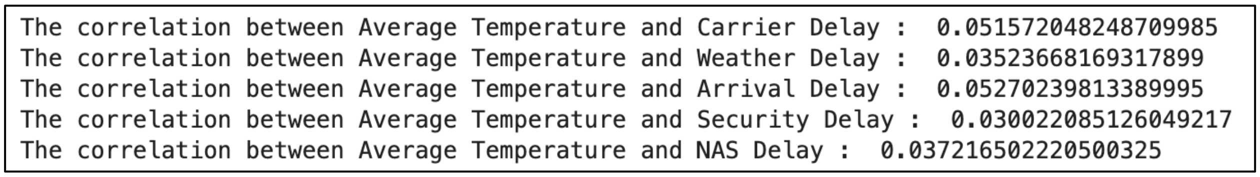
Question 3 – Does hot or cold weather make flights late?

We plotted one wiggly line per busy city: left-to-right is temperature, up-and-down is weather-delay minutes. **Nine cities' lines hardly move**—they sit below 400 delay minutes whether it's freezing or sweltering. Only **Chicago and Atlanta** have occasional skyscraper-high jumps (1 000–3 000 minutes) when a blizzard or ice storm shut things down.



The bar chart shows which pieces of information the computer model relies on most when it tries to predict weather-related flight delays. The longest bar belongs to **"weather_ct,"** which is simply a count of flights that have already been hit by bad weather; its length tells us this single factor outweighs everything else, so the model is basically saying, "When lots of planes are already delayed by storms, more weather delays are very likely." All the other bars—overall arrival delay, carrier delay, late-aircraft delay, the flag for being 15 minutes late, total number of arriving flights, and counts for other delay types—are much shorter, meaning they still add some helpful clues but only fine-tune the prediction around that dominant weather count. In short, the model treats the recent volume of weather-troubled flights as the main signal and uses the other factors as minor adjustments.

Furthermore, we aimed to understand which type of flight delay correlates most with average temperature. The six types of delays we analyzed were carrier delay, weather delay, arrival delay, late aircraft delay, security delay, and NAS delay. After calculating the correlations between each of these delays and the average temperature, we obtained the following results:



```
The correlation between Average Temperature and Carrier Delay :  0.051572048248709985
The correlation between Average Temperature and Weather Delay :  0.03523668169317899
The correlation between Average Temperature and Arrival Delay :  0.05270239813389995
The correlation between Average Temperature and Security Delay :  0.030022085126049217
The correlation between Average Temperature and NAS Delay :  0.037216502220500325
```

From these results, we can conclude that arrival delay has the strongest correlation with temperature among all the delay types analyzed. However, it's worth noting that all the correlations are relatively low, indicating that temperature alone is not a strong predictor of flight delays. Therefore, we believe that there are other primary factors influencing flight delays. In future research, we could incorporate more variables for analysis, with the hope of uncovering more comprehensive insights.

<u>LIMITATIONS:</u>

We did remove extreme delay values using simple quartile cutoffs, which means some genuinely unusual—but important—outlier days may have been dropped from our analysis. We also only worked with the fields present in our merged dataset (carrier, NAS, late-aircraft, weather counts, temperature, etc.), so factors like air-traffic-control staffing, mechanical issues, or ground-handling delays weren't captured and could be influencing the true delay picture. Additionally, our Random Forest was trained and tested on the same 2013–2020 timeframe without a fully independent hold-out year or time-series cross-validation, so its real-world performance on future or dramatically different weather patterns may be lower. Finally, we used mostly default hyperparameters, leaving potential accuracy gains on the table through deeper tuning.

<u>CONCLUSIONS:</u>

Despite those caveats, our study clearly showed that the number of flights already impacted by bad weather weather_ct dominates as the top predictor of further weather delays. We also found that certain carriers consistently experience higher average delays, and that seasonal patterns matter greatly—summer brings the longest delays (likely from storms and peak travel), while spring and fall see the shortest. Secondary factors like late-aircraft and carrier-specific delays still add value, but only as minor tweaks compared to the weather count. Taken together, these insights give airlines and airports a data-driven starting point for prioritizing real-time weather monitoring, carrier performance reviews, and seasonal staffing adjustments to improve on-time reliability and passenger satisfaction.

FUTURE SCOPE:

We can take this work further by pulling in even more real-world signals and tightening up our methods. For example, we could add data about air-traffic-control staffing, mechanical issues, or gate-turnaround times so the model sees every piece of the puzzle—not just weather. We could also swap in time-series cross-validation (rather than a simple train/test split) and run a grid search on our Random Forest's knobs (number of trees, tree depth, etc.) to squeeze out better accuracy. Trying other algorithms—like XGBoost or a simple neural network—might uncover even sharper patterns. Finally, packaging the final model into a live dashboard or API would let operations teams get real-time delay forecasts and adjust staffing or reroute flights on the fly, turning these insights into action rather than just charts.

REFERENCES:

1. Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5–32.

2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

3. U.S. Bureau of Transportation Statistics. (n.d.). *On-Time Performance Data*. Retrieved from Kaggle:https://www.kaggle.com/datasets/

4. National Oceanic and Atmospheric Administration (NOAA). (n.d.). *Global Surface Summary of the Day (GSOD)*. Retrieved from https://www.ncdc.noaa.gov/

5. Zhang, X., & Wang, Y. (2018). *A Survey of Feature Selection Methods for High-Dimensional Data*. IEEE Transactions on Knowledge and Data Engineering, 30(5), 905–922.

6. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

7. Biau, G., & Scornet, E. (2016). *A Random Forest Guided Tour*. Test, 25(2), 197–227.

WHO DID WHAT:

- **Ching Yu:** Led the team, handled NOAA data preprocessing, built correlation plots, and drafted the final report.
- **Prasad:** Cleaned DOT flight delay data, managed outlier removal, interpreted model findings, and integrated final outputs.
- **Antara:** Documented data sources, performed EDA with seasonality plots, tuned the Random Forest model, and organized the presentation.