SOO SI MIN NOELLE

Harvard Airline Dataset Analysis | 2023

Table of Contents

Executive Summary

The report aims to clarify the steps for gaining insights into Airline on-time performance from the Harvard Airline Dataset using statistical, graphical, and technical analysis to answer the given questions.

The Harvard Dataset is a large dataset consisting of nearly 120 million rows. This report endeavors to address real-life queries by comprehending and modeling the dataset. Moreover, the report outlines efficient techniques for handling large datasets, employing tools like SQL, tables, and byte-size modification to facilitate computation of up to 35,000,000 data rows extracted from the total dataset spanning years 2003 to 2007. Other than the variables in the 'YYYY.csv' files, there are 4 accompanying sheets that carry airport's, carrier's and plane's data that we can join to our main tables if necessary.

Alongside explanation of the final answer to the question, the 📎 <paperclip> logo indicates portions for non-programming audience to understand how the data is processed to query the final results.

Firstly, exploratory data analysis and data cleaning is essential to understand the dataset provided. After which, a table is created for the years that have been selected for the analysis.

Q1. When is the best time of day, day of the week, and time of year to fly to minimise delays?
The best time of the year to fly to minimize delays is September, the best day of the week to fly is Saturday and the best time of the day is from 0400 to 0559.

2. Do older planes suffer more delays?
Yes, older planes suffer more delays, as shown via a linear relationship between plane age and percentage delays.

3. How does the number of people flying between different locations change over time?
The volume of people traveling between distinct locations varies depending on the routes taken and seasonal fluctuations. Notably, certain routes exhibit an annual pattern, with a decline in the number of travelers at the start and end of each year. It is possible to conduct separate analyses of specific routes to determine the precise trend of passenger traffic throughout the year.

4. Can you detect cascading failures as delays in one airport create delays in others?
Cascading failures can be detected via analysis of plain tail numbers from one airport to another. By tracking the routes, it was discovered that flights with delays under 100 minutes experienced cascading delays. In contrast, flights with delays exceeding 200 minutes were canceled, preventing further cascading delays from occurring.

5. Use the available variables to construct a model that predicts delays.
After conducting Pearson's Correlation Analysis, it was determined that the Arrival Delay and Departure Delay variables are highly indicative of the overall late aircraft delay. To streamline the model, solely numerical variables pertaining to delay values were included. Three models, namely Linear Regression, Decision Tree, and Random Forest, were employed to forecast late aircrafts. Metrics were subsequently utilized to evaluate the most appropriate model, Linear Regression.

Introduction

Overview of the dataset:

Dealing with a large dataset of up to 12GB uncompressed, the project will be selective in choosing particular years for modelling and leverage the use of SQLite in Python and RSQL in R.

After understanding the dataset and types using the `d.type` function, exploratory data analysis was performed on the data spanning from 2003 to 2007 because it contains a complete years' worth of data. Additionally, a wider range of data over an extended period yields a more representative sample, increased variability, more robust statistical analysis, and better insights into any rare occurrences. The selection was also influenced by the data's recency, which enhances the accuracy of predicting future trends.

Data pre-processing

Loading the data: To load the large dataset, we utilized the `bz2() function` to unzip the .bz2 files. After collecting the data spanning the 5-year period, the `append() function` was employed to consolidate all the data into a list, which was then transformed into a data frame named df. The `df.head() function` was then used to view the initial few columns of the data frame, while the `df.groupby("Year") function` was utilized to gain insight into the mean values deviating annually.

Data cleaning: given that our most important values are in the HH MM format, we clean the data by getting rid of values that are more than 24 hours and missing data using `.dropna function`. `Print df.dtypes` to check that the data has been converted into the appropriate integer types.

Data transformation: Reducing byte size by changing the integer types from 64 to 16 to increase processing power. Changing HH MM values from float to integer using the `astype() function` rounds decimal values to integers, making it faster to read and easier for the computer to process.

Once the dataframe is established, creating a database may not be required. However, for efficiency reasons, we employed SQL, a specialized language utilized to manage and manipulate relational databases. SQL is particularly suitable for this database as it enables us to process complex queries on large datasets in an efficient manner. The packages utilized in this project were optimized for algorithms and memory-efficient data structures, resulting in faster performance for common data manipulation tasks.

Creating tables and writing into our database: at this stage we commit our unzipped csv files into our database and create a table called `“ontime”` using SQL. As a precautionary measure, `if_exsist=“append”` will remove any CSV file of the same name that was in that data table previously

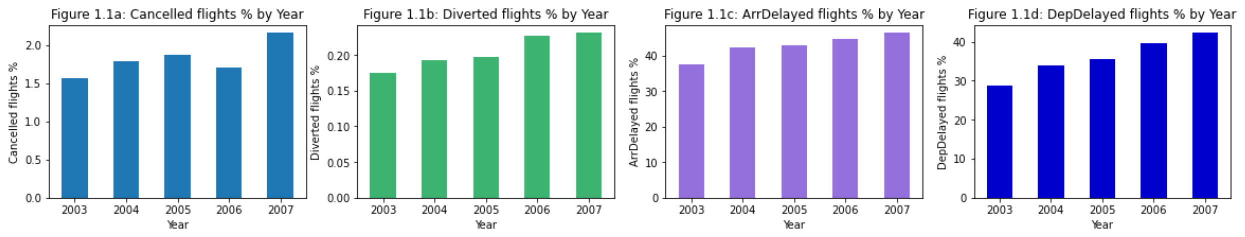
Q1: Best time to fly

Exploratory Data Analysis

Table 1: Output of Query1ByYear

	Year	% of Cancelled flights	% of Diverted flights	% of ArrDelayed flights	% of DepDelayed flights
0	2003	1.563819	0.175402	37.625429	28.653672
1	2004	1.792007	0.193344	42.338725	33.841936
2	2005	1.872813	0.196454	42.898688	35.541305
3	2006	1.707300	0.226634	44.768985	39.564350
4	2007	2.156761	0.230491	46.335105	42.194060

Table 2: Figure 1.1a, 1.1b, 1.1c, 1.1d



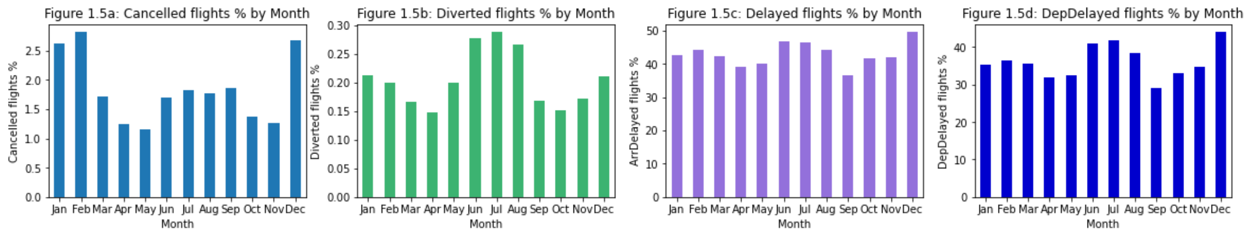
Defining variables: To analyse the best time periods to fly to minimize delays, flight delay data from years 2003 – 2007 was extracted to understand the variability of the data by years.

Visualizing the distribution of variables by year: `Query1ByYear` enables the extraction of variable values as percentages. `SELECT` is employed to query by columns, `FROM` is used to retrieve the data from the data table created, `GROUP BY` groups the results by year, and `ORDER BY` orders the results by years in rows. The output of `Query1ByYear` is presented in Table 1, which displays the percentage of flights by variables over the years. Based on the results presented in Table 1, it can be concluded that there is a notable increase in the percentage of delayed flights.

Q1.1: Best time of the year to fly to minimise delays

Analysing 2006 and 2007 data: The trends in flight delay are similar between the years 2006 and 2007, with only differences in the magnitude of the lowest and highest percentages. As a result, it is reasonable to analyze the best time of year to fly to minimize delays across all five years.

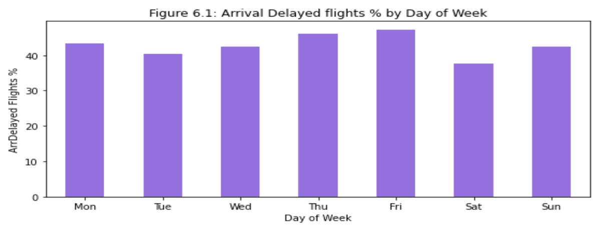
Table 3: Percentage of 2003-2007 cancelled, diverted, and delayed flights by Month



Answer 1.1: According to our analysis, September appears to be the optimal month of the year to fly, as evidenced by the comparatively low percentage of arrival and departure delays during this month. It is important to acknowledge, however, that May could also be a favorable option for those seeking to minimize the risk of cancellations or diversions, as the delay percentages during this period are relatively low, while the percentage of potential cancellations and diversions is at its lowest.

Q1.2: Best day of the week to fly to minimise delays

Table 4: Percentage of delay by Day of the week



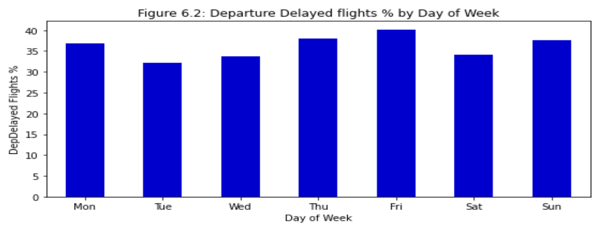
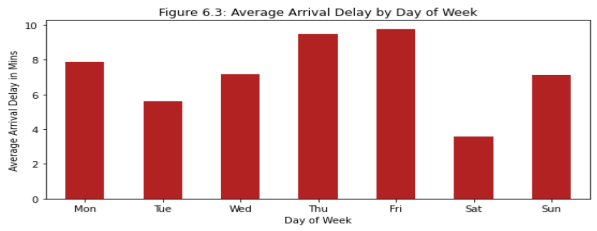
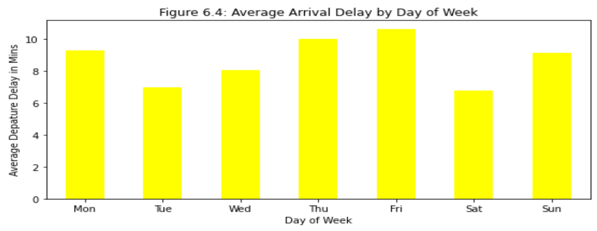


Table 5: Average delays (in mins) by Day of the week





Answer 1.2: Table 4 enables us to deduce that the day with the lowest percentage of delay is Saturday. Table 5 shows that the average delay in minutes for arrivals (3.56 mins) and departures (6.74 mins) are both the lowest on Saturday. Hence Saturday would be the best day of the week to fly such as to minimize delays

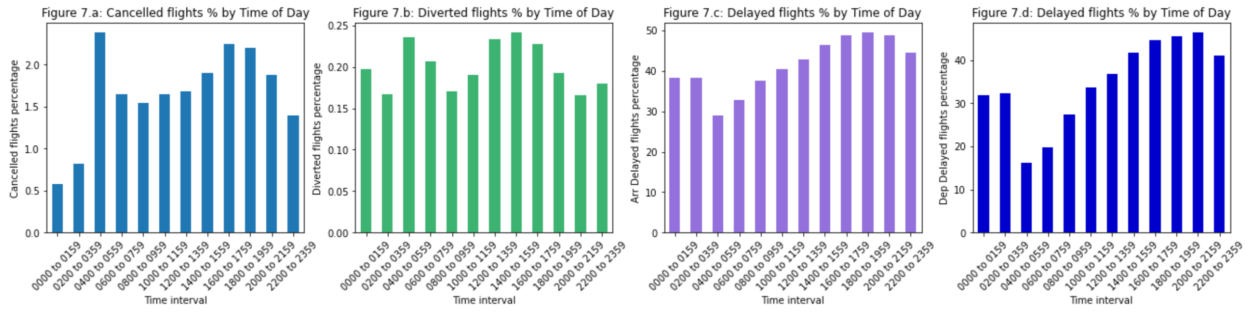
Q1.3: Best time of the day to fly to minimize delays

<u>Checking that the it is fair to query the entire data set for the average elapse time and average airtime:</u> in `Query7a`, we observe that the average airtime and average elapse time remains relatively stable across all days the week, hence making it possible to derive an hourly analysis without correcting skewness in the dataset.

<u>Query the sum of arrival delay and departure delay per every 2 hours</u>: Similar to above, we query the database for delay timings. In this portion, we used `CASE` to create a new column called '`TimeFrame`' which categorizes the flights based on the scheduled departure time '`CRSDepTime`' by referencing the scheduled departure time column and assigning each output into one of the assigned timeframes. The resulting table will have one row for each time frame and columns to show the cancelled, diverted, arrival and departure delayed flights respectively.

The process of indexing columns and assigning new column names in our data table involved the use of `'iloc'` function to index the columns by their positions rather than their labels, thereby ensuring successful indexing. This was done because the use of `'[]'` indexing failed to yield the desired outcome. Given that the timeframe column was selected in the 4th step, it was positioned in the 4th position of the table. To reorder this column to the first position, `'iloc'` function was utilized to successfully index the column.

Table 6: Flights cancelled, diverted, delayed by time intervals of 2 hours across 5 years



Answer 1.3: Table 6, indicates the best time of the day to fly to minimize delays is from 0400 to 0559 as it exhibits the lowest percentage of arrival and departure delay.

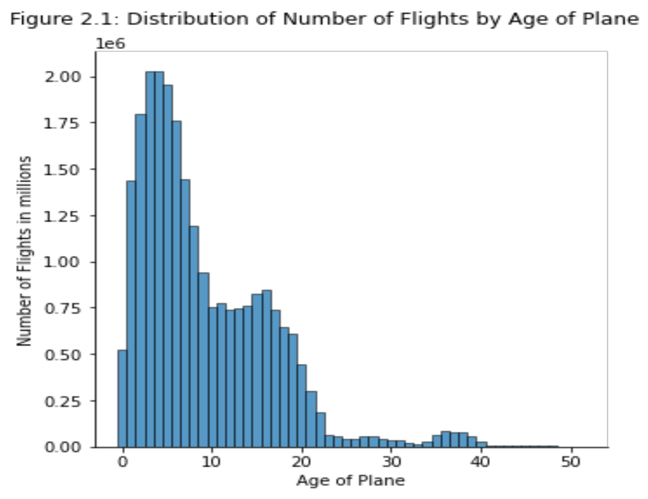
Q2: Do older planes suffer more delays?

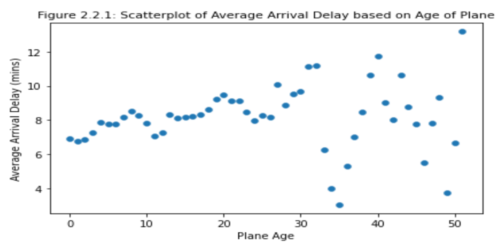
Understanding the data: After creating the `planedata.db` from the plane-date.csv, we observed that the tail number serves as the unique identifier for each aircraft and will be the `primary key` [i]for connecting the `planedata.db` to the `ontime.db.` To gain insights into the aircraft's production year, we used the `Query function` to determine the minimum and maximum years of production for the planes.

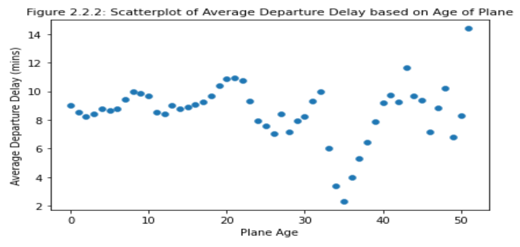
Querying the database to identify erroneous data: To identify and address erroneous data, we queried the database to find all instances where the plane's age is negative. We retained the rows with erroneous data in the 'planedata' table to prevent data loss.

Join 'planedata' table to 'ontime'table: To perform our analysis on the age of the plane relative to delay timings, we joined the planedata table to the ontime table by adding the age of the plane column to the original table.

Table 7: (F2.1)Distribution of Flights made by Plane Age, (F2.2.1) Average Arrival Delay by Plane Age, (F2.2.2) Average Departure Delay by Plane Age



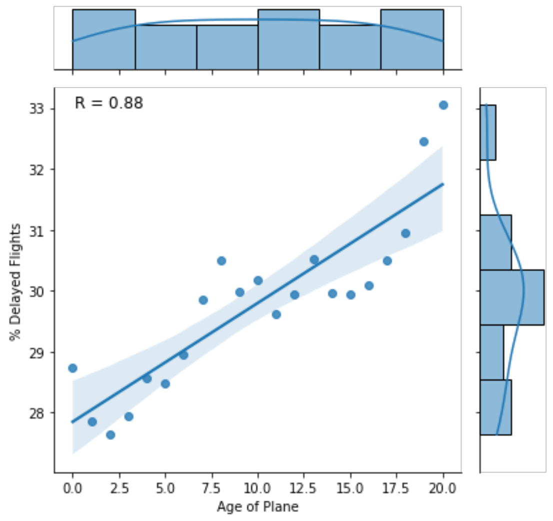


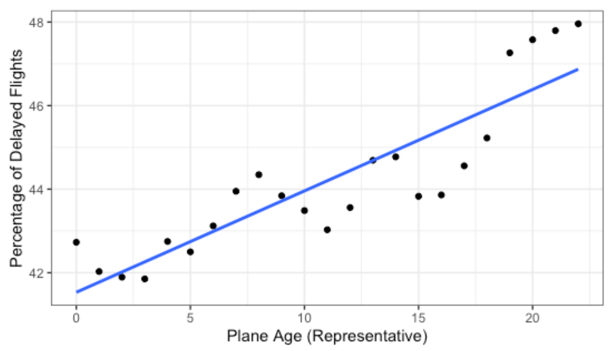


Analysis of Table 7: Based on Figure 2.1, there are fewer planes in the age range of 22 to 61. Therefore, we analyzed the delay variables against the age of the planes using averages and ratios instead of total percentages. Figures 2.2.1 and 2.2.2 do not reveal a clear pattern between average arrival delay and delay against the age of planes, so further analysis is necessary to determine any relationship between plane age and delay in minutes.

Table 8: Joint plot of the Percentage of delayed flights to the Age of Plane and Scatterplot (R)

Figure 2.7: Scatterplot on Percentage of Delayed Flights based on Age of Plane





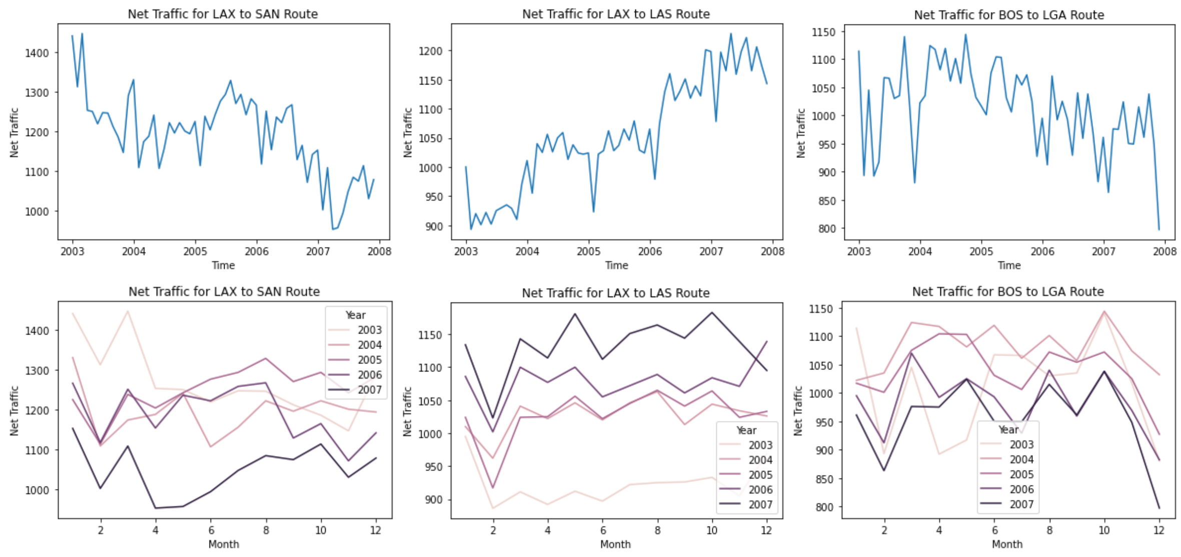
Answer 2: Table 8 displays the average delay time in minutes with respect to the age of the plane, revealing a positive correlation between plane age and delay time in minutes. The results are further supported by the high positive correlation (R value of 0.88) between the percentage of flights with delays and the age of the plane.

Q3: How does the number of people flying between different locations change over time?

In order to connect data tables, we utilized the `'destination'` column from ontime table and matched it to the `'airports.iata'` column from airports table. By utilizing the `concat function 'ontime.Origin || ' to ' || ontime.Dest AS Route'` in SQL, we were able to merge the columns of origin and destination in the query. Using the `matplotlib` library, we generated a line graph to visualize the net traffic volume of top flights over the years. To ensure that the time axis was properly represented in years, we indexed the year and month into a datetime function and labeled the newly created row as `'Date'`.

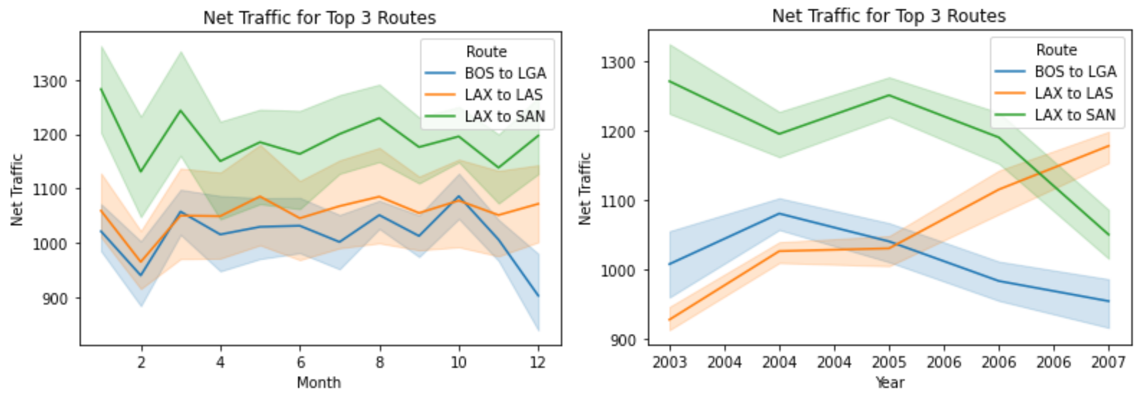
Extracting the top 5 travelled routes: By understanding the top 3 airport routes and city routes, we sample the data for the largest representative sample size to identify and spot trends.

Table 9: Tables of the analysis of net traffic against routes taken over time



Analysing the top 3 routes taken: The analysis of Table 9 indicates that there are variations in the net traffic volume over the five-year period. Specifically, the number of people leaving from LAX to SAN decreased over time, while the number of people leaving from LAX to LAS increased. Furthermore, for people leaving from BOS to LGA, a seasonal pattern is evident, with decreases occurring at the start of each year. Monthly, the trend of net traffic for 2003 is visibly different than the remaining years for all 3 routes.

Table 10: Tables analyzing the top 3 routes taken in in terms of yearly trend and overall trend



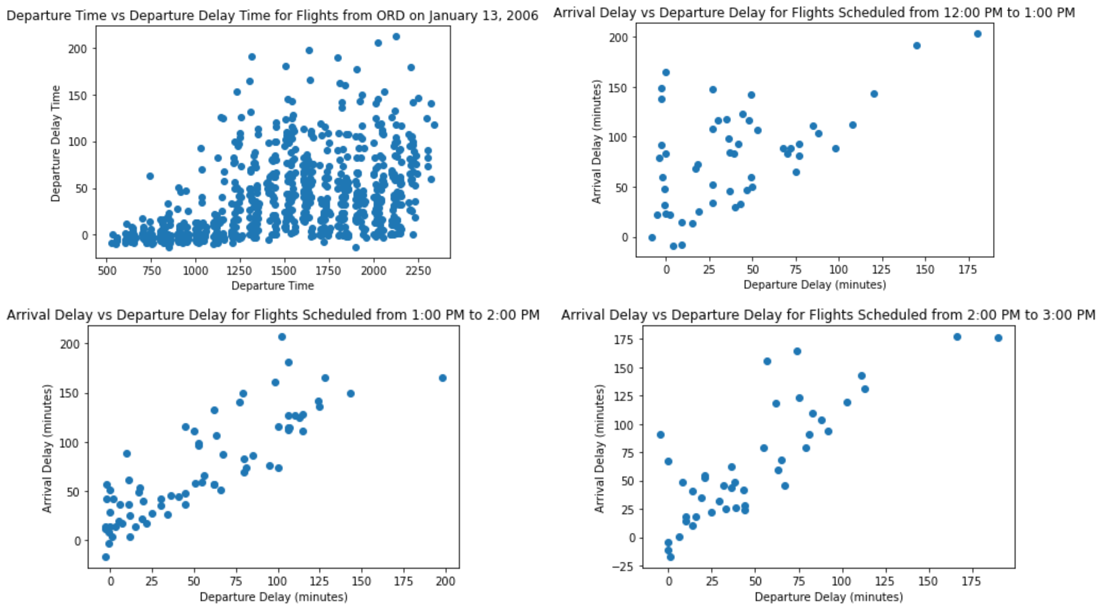
Comparing top 3 routes on the same axis: Upon querying for the top 3 routes taken, we use `seaborn` to plot the net traffic of the top 3 routes for 5 years against month to identify any seasonal trend in the net traffic volume. The net traffic over the years allows us to observe a change in the net traffic across the years and from 2003 to 2007.

Answer 3: The fluctuation in passenger traffic over time is route-dependent, with varying patterns observed for each route. There is no clear seasonality trend across all routes. However, by analyzing individual routes, the number of people flying over time can be determined. Notably, Table 10 reveals that the traffic peaks and dips for the LAX-LAS and LAX-SAN routes are inversely correlated, indicating a shift in the passenger base from LAS to SAN.

Q4: Can you detect cascading failures as delays in one airport create delays in others?

<u>Exploratory data analysis of individual flight numbers with delay of 1000, 250, and 100 minutes:</u> Querying the dataset with unique tail numbers as primary keys revealed that the most delayed planes were rescheduled for use on the following day, making it inappropriate to analyze maximum delay time for cascading delays. No cascading delays were observed for delays exceeding 250 minutes in the subsequent flight the same plane made on that day, possibly due to the plane being reassigned another flight to prevent further delays. For a particular 100-minute delay, all three flights queried resulted in further delays in their subsequent flights, indicating correlation between arrival delay and future departure delays for that Tail Number.

Table 11: Delay (mins) in ORD airport and the spread of Delay (mins) from 12 pm to 3pm



<u>Understanding delays within an airport:</u> From Table 11, we can observe that the departure delay time out of ORD begins to increase at 12pm on a particular day. Upon further analysis of the impacts of delayed departure time on the arrival time of planes entering the airport, we can see a linearity trend in the cascading delays. For example, between 12pm and 1pm, there are more arrival delays than departure delays for the planes. However, from 1pm to 2pm, planes with higher arrival delays have increasingly higher departure delays. This trend of cascading delays can be observed within airports, where a late arrival of a plane leads to late departure, resulting in a compounding effect on delays within that airport.

Table 12: Understanding cascading delays extended to other airports from ORD

```
Correlation between DepDelay in ORD to ArrDelay in BOS 0.8570193978749215
Correlation between DepDelay in ORD to ArrDelay in MSN 0.8982825080211401
Correlation between DepDelay in ORD to ArrDelay in LAS 0.9446255513325486
```

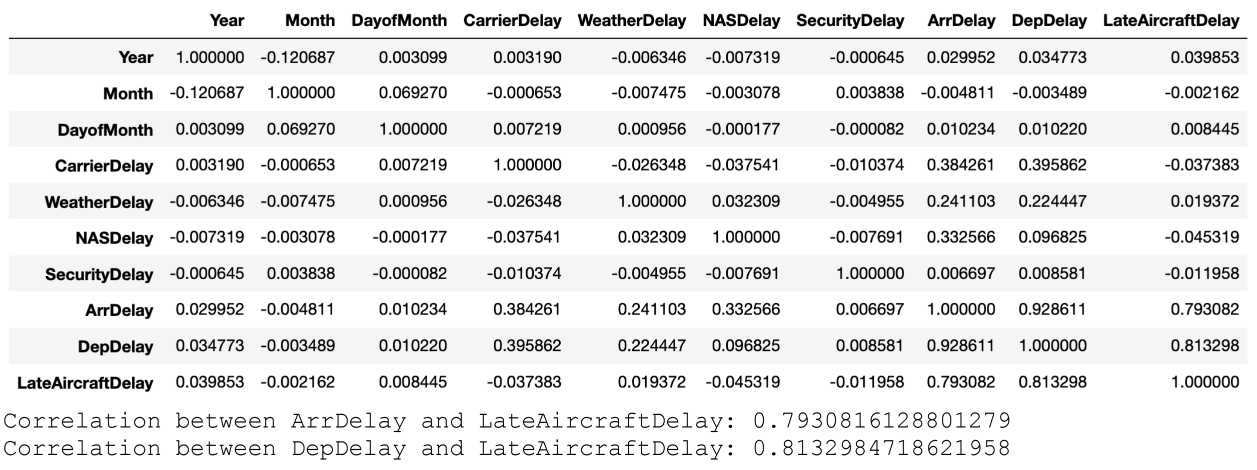
<u>Analyzing cascading delays extended to other airports:</u> Table 12 presents evidence to suggest that delays in the departure of aircraft from ORD airport have a cascading effect on subsequent airports the aircraft visits, with delays in departure leading to delays in arrival at subsequent airports. This trend indicates a linearity in the relationship between departure and arrival delays and suggests that late departures are highly likely to result in late arrivals at subsequent airports.

Answer 4: We can detect cascading delays in one airport creates delays in another by analysing arrival and departure delay

Q5: Use the available variables to construct a model that predicts delays.

The objective is to develop a model that can generate an output (LateAirCraftDelay) based on the input data (features) provided. Model calibration entails applying it to a dataset and creating a loss function (ie RMSE) that evaluates the accuracy of the predictions compared to the actual values. The primary objective of machine learning is to determine the optimal parameters and models that reduce the cost function's value to a minimum.

Table 13: Correlation between variables



	Year	Month	DayofMonth	CarrierDelay	WeatherDelay	NASDelay	SecurityDelay	ArrDelay	DepDelay	LateAircraftDelay
Year	1.000000	-0.120687	0.003099	0.003190	-0.006346	-0.007319	-0.000645	0.029952	0.034773	0.039853
Month	-0.120687	1.000000	0.069270	-0.000653	-0.007475	-0.003078	0.003838	-0.004811	-0.003489	-0.002162
DayofMonth	0.003099	0.069270	1.000000	0.007219	0.000956	-0.000177	-0.000082	0.010234	0.010220	0.008445
CarrierDelay	0.003190	-0.000653	0.007219	1.000000	-0.026348	-0.037541	-0.010374	0.384261	0.395862	-0.037383
WeatherDelay	-0.006346	-0.007475	0.000956	-0.026348	1.000000	0.032309	-0.004955	0.241103	0.224447	0.019372
NASDelay	-0.007319	-0.003078	-0.000177	-0.037541	0.032309	1.000000	-0.007691	0.332566	0.096825	-0.045319
SecurityDelay	-0.000645	0.003838	-0.000082	-0.010374	-0.004955	-0.007691	1.000000	0.006697	0.008581	-0.011958
ArrDelay	0.029952	-0.004811	0.010234	0.384261	0.241103	0.332566	0.006697	1.000000	0.928611	0.793082
DepDelay	0.034773	-0.003489	0.010220	0.395862	0.224447	0.096825	0.008581	0.928611	1.000000	0.813298
LateAircraftDelay	0.039853	-0.002162	0.008445	-0.037383	0.019372	-0.045319	-0.011958	0.793082	0.813298	1.000000

```
Correlation between ArrDelay and LateAircraftDelay: 0.7930816128801279
Correlation between DepDelay and LateAircraftDelay: 0.8132984718621958
```

Feature engineering: Selecting the most relevant and informative features is critical to optimize model performance. In Table 13, Pearson's correlation coefficient was utilized in this study to measure the linear relationship between variables, and the results revealed a strong positive correlation between arrival and departure delays, indicating their potential as predictors of late aircraft delays.

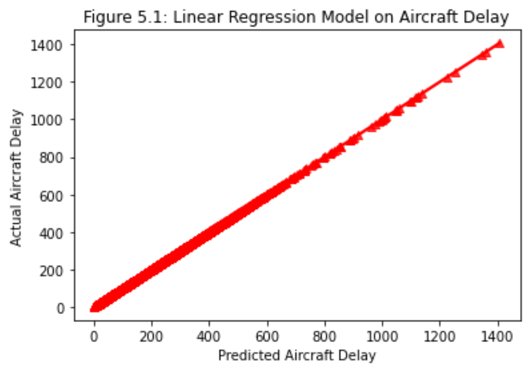
Creating the models

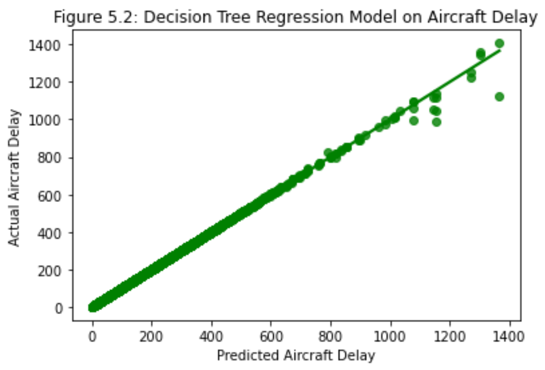
Data pre-processing: We passed the selected features through a pipeline[ii] that included `SimpleImputer()` and a numerical transformer [iii]to preprocess the numerical variables. Then, we created an independent variable X by selecting the features in `features list.`

Creating train/split datasets: For the model to accurately predict the arrival delay ('y') based on the input features of ('X'), splitting the `test_size=0.3` suggests that 30% of the total rows for the model is being used for testing the model, whilst the other 70% is for forming the prediction model.

Hyper parameter tuning: Defines the hyperparameters[iv] to be tuned for the overall pipeline, including the use of `'scaler__with_mean'` mean scaling and `'regressor__fit_intercept'` intercept fitting in the regressor. The `GridSearchCV function` is then used to perform a grid search [v]over the specified hyperparameters to find the optimal combination that produces the best model performance.

Table 14: Prediction Modeling graphs Actual by Predicted





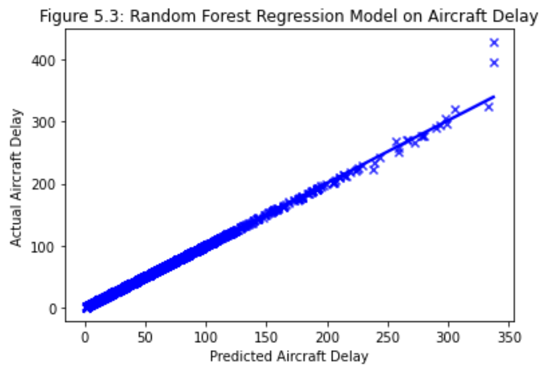


Table 15: Machine learning model accuracy metrics

	Linear Regression Values	Decision Tree Values	Random Forest Values
RMSE	2.889062e-14	2.170939	2.247487
MSE	8.346678e-28	4.712978	5.051197
MAE	1.846416e-14	0.176000	0.140269
R2	1.000000e+00	0.997660	0.997492

Figure 5.1 : The Linear Regression Model is the model of choice as it is perfectly suited for our prediction model as it is used for predicting numerical values (late aircraft delay). As seen from Table 15 it has the smallest error values and R2 value of close to 1 which is a perfect model.

Figure 5.2: The Decision Tree Model is a supervised machine learning algorithm used for predicting target variables such as airline delays. It recursively partitions data into subsets based on the most important features for predicting the target variable. The algorithm builds a tree-like model by choosing the feature that provides the most information gain and calculating impurity measures such as entropy, Gini index, and classification error. The RMSE value (2.17039) suggests some possible errors to be improved on, some possible further steps are to reduce overfitting and possibly inputting numerical data. The R^2 value (0.99766) suggests that it is still a strong and suitable model for predicting airline delays

Figure 5.3: Random Forest Model: The Random Forest is an ensemble learning method that combines multiple decision trees to enhance prediction accuracy and decrease overfitting. It builds diverse decision trees independently and randomly by using subsets of the training data and features. To make a prediction, it aggregates the predictions of the individual trees through majority vote or averaging. Based on the higher error values (RMSE 2.247487) and lower R^2 value (0.99749), we can conclude that the Decision Tree Model is not critically overfitted.

Answer 5: Overall, all 3 models have good performance with a high level of accuracy and low error rate. The high R^2 value suggests that the airline delay can be completely explained by model's input variables.

Appendix

[i] A primary key: is a unique identifier that is used to distinguish one record from another in a database table. It serves as the main reference point for linking data from multiple tables and for performing operations such as updating or deleting specific records.

[ii] Pipeline: A pipeline will apply imputation of the missing values with the function `SimpleImputer()` imputing either the median or mean observation for each variable.

[iii] Numerical Transformer: used to preprocess numerical data. The amin purpose is to transform raw numerical data into a format that can be more easily understood by the model. The one applied in my python code replaces any missing values in the data with mean value of the respective features and scales the features so that they have a mean of 0 and a standard deviation of 1.

iv Hyperparameters: is a parameter that is set before the training process of a machine learning model and is not learned during the training process itself. Instead, these parameters are set manually by the user or are tuned using a trial-and-error approach to improve the performance of the model.

`param_grid` is a dictionary that contains a set of hyperparameters to be optimized through hyperparameter tuning. The first example in the code is optimizing the `data_transformer__numerical__imputer__strategy` hyperparameter, which defines the strategy used to impute missing values in numerical data. The possible values for this hyperparameter are "mean", "median", and "most_frequent".

The `scaler__with_mean` hyperparameter determines whether to center the data before scaling it, while the `regressor__fit_intercept` hyperparameter determines whether to fit an intercept for the linear regression model or not.

v Grid search: Grid search is a hyperparameter tuning technique that involves searching through a predefined set of hyperparameters to find the combination that results in the best performance of a machine learning model.

This function takes in a pipeline object `pipe_lm`, which is the machine learning model that we want to optimize, along with a `param_grid` dictionary that specifies the hyperparameters we want to optimize and their possible values.

The `n_jobs` parameter in the `GridSearchCV` function specifies the number of CPU cores to use in parallel for the computation. In this case, -1 is used to indicate that all available cores should be used.

After specifying the hyperparameters and the pipeline`, GridSearchCV` fits the model on the training data (X_train and y_train) using cross-validation to evaluate the performance of each combination of hyperparameters. It then returns the combination of hyperparameters that results in the best performance.