# Analysis of Flight Delays: Capston Proj

Proj by Col Rakesh Pedram

## Problem Statement

According to air travel consumer reports, a large proportion of consumer complaints are about frequent flight delays. Out of all the complaints received from consumers about airline services, 32% were related to cancellations, delays, or other deviations from the airlines’ schedules.

There are unavoidable delays that can be caused by air traffic, no passengers at the airport, weather conditions, mechanical issues, passengers coming from delayed connecting flights, security clearance, and aircraft preparation.

## Objective

The objective of this project is to identify the factors that contribute to avoidable flight delays. You are also required to build a model to predict if the flight will be delayed.

### Project Setup

For the project I used the following libraries:-

Pandas and Numpy for data manipulation and structuring.

Mathplot and seborn for visualization

Beautiful Soup for web scrapping

Sklearn for Machine learning tasks, including pipelines and ensemble

I have done the project in **Google colabs** with files loaded from google drive.

### Task 1: Import, Aggregate, Web Scrape

#### a. Data Import and Aggregation

The three files from the Data set were imported and their shapes and info was checked.

##### Join datasets based on appropriate keys (e.g., airport code)

Studying the structure. I have decided to make three joins to keep airfield data like elevation, no\_of\_runways etc in a single flat df.

This process of merging took max time, The shape of the df was checked to confirm that the shape is not larger than the Airlines table. I made a temp df to finally link all this data.

My joins are listed:-

* Airports left join on runways on common field
* Airlines joined to first df ArirlineTo<>iata\_code
* Airlines joined to first df AirlinesFrom<> iata\_code

### b. Web Scraping and feature engineering

I used Beautiful Soup library for web scrapping. There was limited data in the wiki tables. The complete structure of the wiki page was studied and the code written to extract the relevant information from the columns.

##### Airline Experience

Some data remained blank and had to be filled up by manually looking up the web as dropping this data was not feasible due to large numbers. Eg foundation dates of major airlines from

##### Passenger traffic

The passenger traffic was extracted from the web page and was only available for 64 major airports. The data where not available was imputed. A feature called hub\_type was created classifying airport traffic as small, med, large.

##### # Missing value analysis

print(merged\_df.isnull().sum())

The df was cleaned and missing values imputed or data dropped where it was insignificant.

## Data Visualization

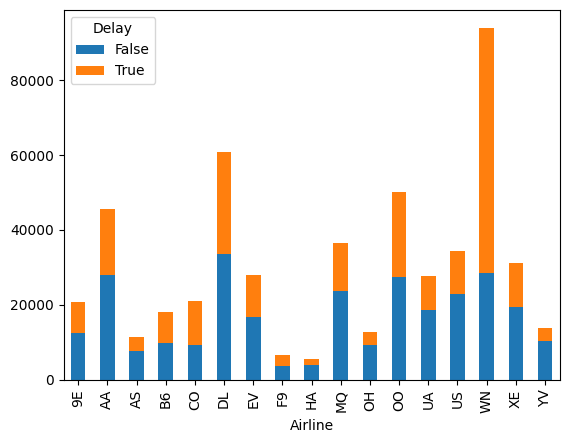
Various graphs were plotted to understand the data interdependence and impact on the target parameter: Delay

Figure 1Flt Operator Vs Punctuality

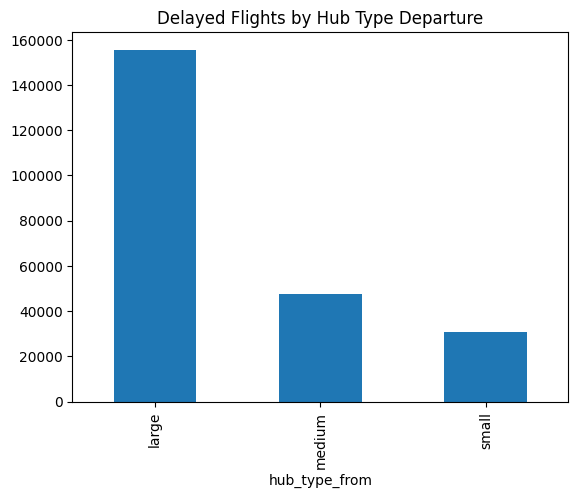


Figure 2 Delayed Flt Vs departure hub type

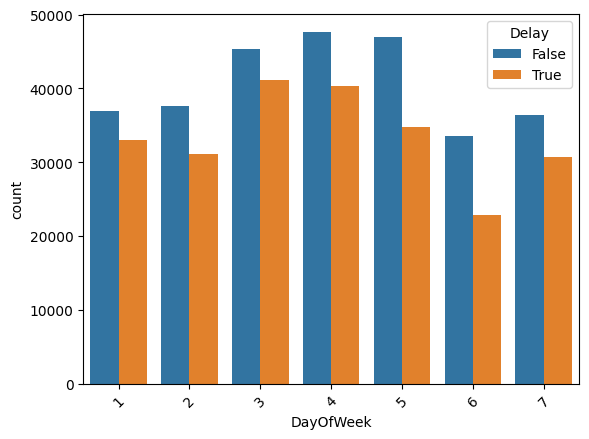


Figure 3 Delay Vs Day of Week

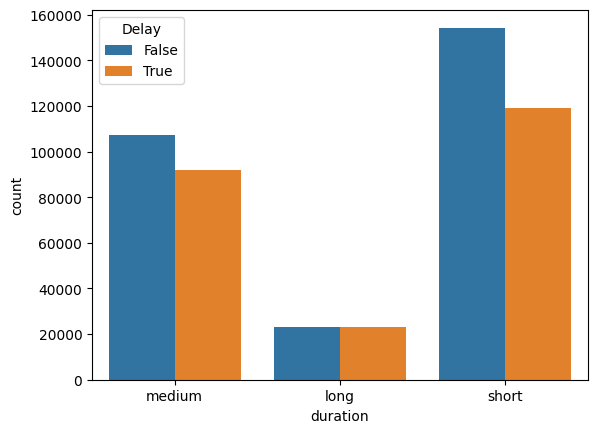


Figure 4 Delay Vs Duration

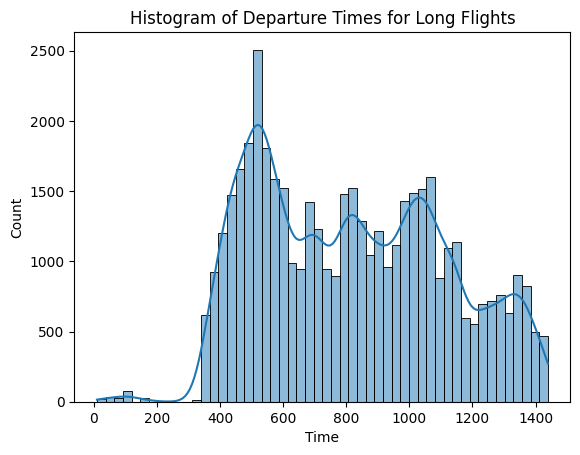


Figure 5 Delay of long flt Vs time of dep

### Task 4: Hub Comparison and Hypothesis Testing

###### My Hypothesis for Elevation

Two-Sided Hypothesis: I hypothesized there's a difference in delays without specifying the direction, the results also indicate that my hypothesis is likely correct.

Used a t test for this with skitlearn.

Results Breakdown

t-statistic (9.186): This measures the difference between the mean delays of high-elevation and low-elevation airports, scaled by the variability within each group. A large t-statistic suggests the difference is likely significant.

p-value (4.10e-20): This is extremely small (close to zero). It indicates the probability of observing a t-statistic as extreme.

Interpretation: Reject Null Hypothesis

The results strongly suggest that there is a statistically significant difference in mean delays between airports above and below 5000 ft elevation. Since the p-value is far below common significance thresholds (like 0.05 or 0.01), we reject the null hypothesis (which would assume no difference).

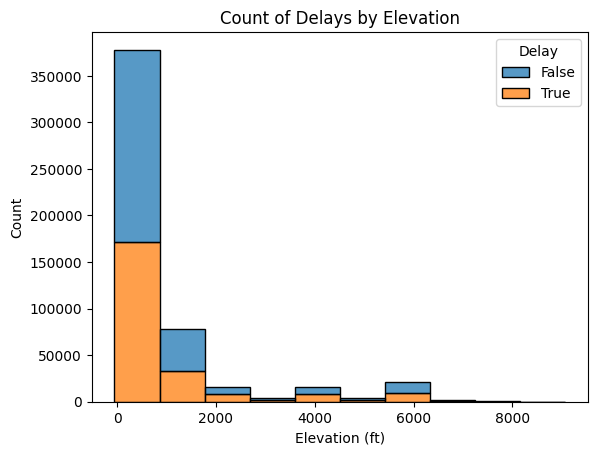


Figure 6 Delay Vs Elevation of dep airfd

### Hypothesis: number of runways at an airport affects flight delays

**Null Hypothesis (H0):** There is no significant relationship between the number of runways at an airport and average flight delays.

**Alternative Hypothesis (H1):** Airports with more runways tend to have lower average flight delays.

**Approach to testing:** As my Delay parameter is binary, Chi Square test is ideal. Contingency Table and Chi-Squared Test:

* Group the data based on the number of runways and whether the flight is delayed or not.
* Construct a contingency table.
* Perform a Chi-Squared test to check for a significant association between the number of runways and flight delay status (delayed vs. not delayed).

#### Results

Chi-squared Statistic: 2641.2504306544506

p-value of Chi test: 0.0

#### Interpretation of results

Since the p-value is extremely small (close to zero), we reject the null hypothesis.

This implies that there is a significant relationship between the number of runways at an airport and average flight delays.

The alternative hypothesis suggests that airports with more runways experience lower average flight delays.

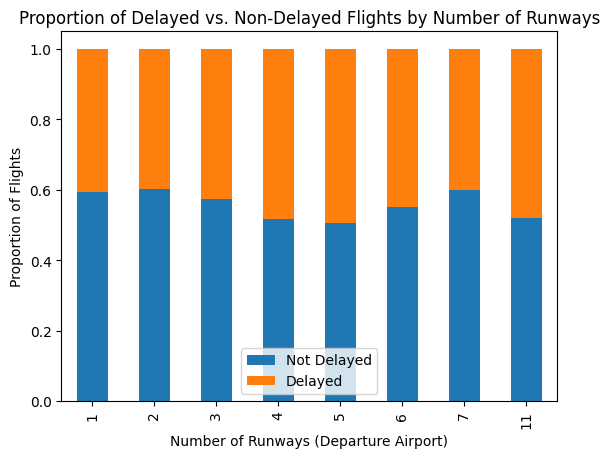


Figure 7 Runway Vs Delay Chi test visual

### Hypotheses: flight length and delays

Null Hypothesis (H0): There is no significant relationship between flight duration (length) and the likelihood of flight delays.

Alternative Hypothesis (H1): There is a relationship between flight duration and the likelihood of flight delays (this could be either that longer flights are more likely to be delayed, or less likely to be delayed)

#### Testing Approach : Chi-Squared Test

1. Categorize length (e.g., short, medium, long flights) based on my earlier classification 0, 120,240, max
2. Create a contingency table.
3. Perform the test to measure association.

#### Results

Chi-squared statistic: 682.9995414218747

p-value: 4.881283244205973e-149

#### Analysis of Results

* Since the p-value is practically zero, we reject the null hypothesis.
* This implies that there is a significant relationship between flight duration and the likelihood of flight delays.
* However, the direction of this relationship (whether longer flights are more or less likely to be delayed) cannot be determined solely from the chi-square test. We can get this from the coeff of the parameter in out models. From Gradient boosting Algo we see that this is a positive correlation with factor 0,28.

#### Visualisation of Chi test

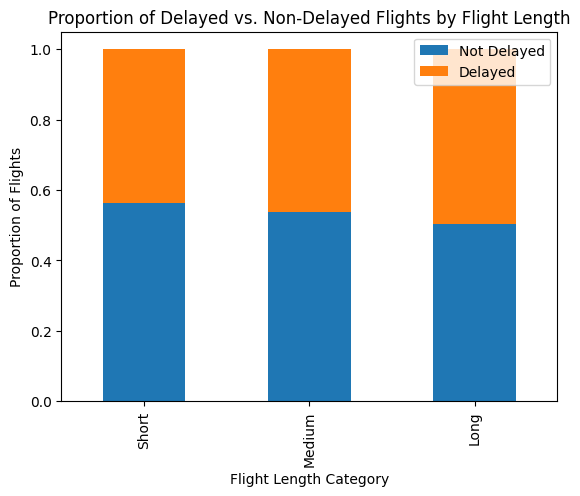


Figure 8 Visualisation Chi Test- Duration Vs Delay

### Correlation Matrix

The correlation between predictors and target variable were plotted with seaborn.

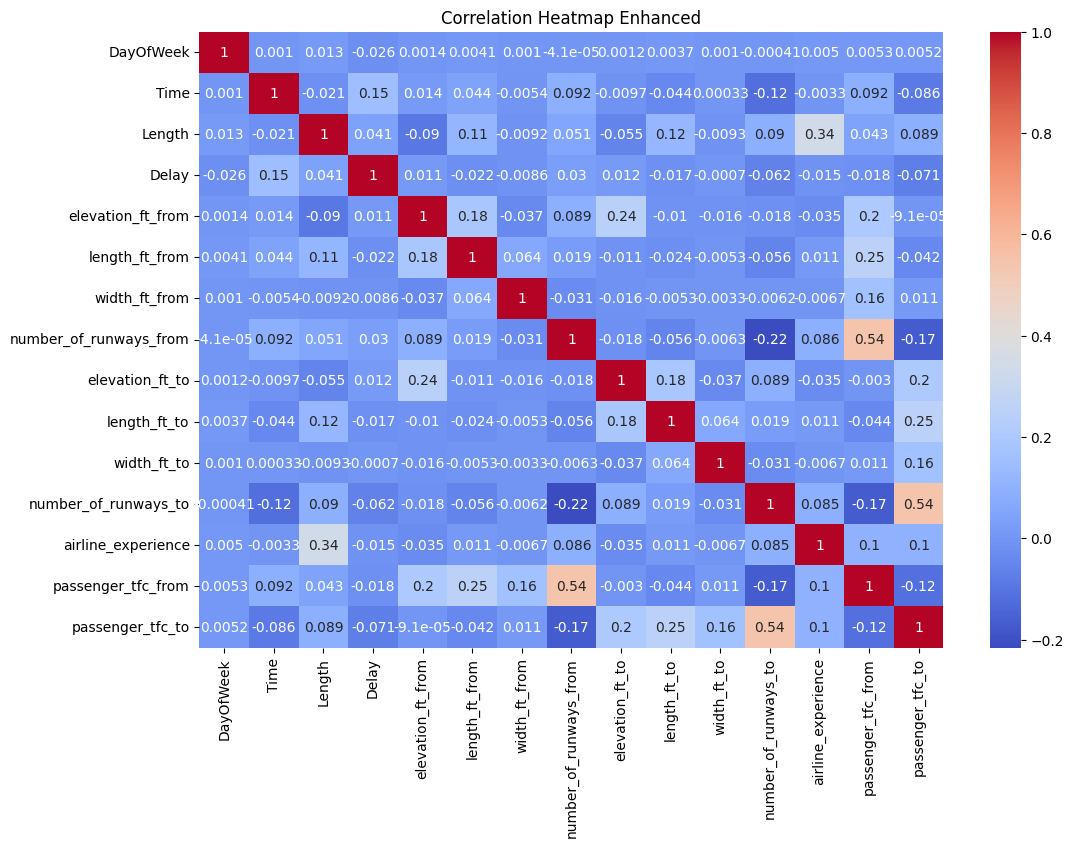


Figure 9 Heatmap Predictors Vs Target(Delay)

#### Analysis

##### Positive Correlations

**number\_of\_runways\_from & passenger\_tfc\_from (0.62):** Airports with more runways tend to have higher departing passenger traffic, which is intuitive.

**number\_of\_runways\_to & passenger\_tfc\_to (0.62):** Similarly, arriving passenger traffic is associated with airports having more runways.

**airline\_experience & passenger\_tfc\_from/to (0.20):** This suggests more experienced airlines might operate from or fly to airports with higher passenger volumes.

**length\_ft\_from & passenger\_tfc\_from (0.29):** Longer runways at departure airports might be correlated with larger aircraft accommodating more passengers.

##### Negative Correlations

**Time & passenger\_tfc\_to (-0.13):** A weak negative correlation could hint at potential off-peak arrival patterns. Further analysis of time-of-day trends is warranted.

**number\_of\_runways\_from/to & passenger\_tfc\_to/from (-0.26, -0.26):** Somewhat counterintuitively, airports with more runways seem to be weakly negatively correlated with passenger traffic. This requires further investigation.

## Machine Learning

### Encoding Categorical Variables

I have used pipeline for encoding the data. However encoding all types of data numeric, ordinal and categoric was causing repeated problem that I could not debug after a lot of effort. Hence, I used label encoder outside the pipeline code for encoding my nominal variables.

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df['Airline'] = le.fit\_transform(df['Airline'])

df['AirportFrom'] = le.fit\_transform(df['AirportFrom'])

df['AirportTo'] = le.fit\_transform(df['AirportTo'])

df.head()

### Pipleline

Column transformer was used to convert all data. The binary and labeled data was passed through.

# Build the Pipeline

preprocessor = ColumnTransformer(

transformers=[

('oe\_type', OrdinalEncoder(categories='auto'), ordinal\_features),

('scaler', StandardScaler(), numerical\_features)

],

remainder='passthrough'

)

### Multiple Models

I used a list of model to loop through with my test and train data. And stratified K fold was used for voting. I used the following model:-

* Logistic Regression
* Decision Tree
* Gradient Boosting
* Random Forest

The key code snippets are below:-

# Split into train/test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

X\_train\_transformed = preprocessor.fit\_transform(X\_train)

X\_test\_transformed = preprocessor.transform(X\_test)

# Model Building

models = {

'Logistic Regression': LogisticRegression(solver='liblinear'),

'Decision Tree': DecisionTreeClassifier(max\_depth=5), # Example depth control

# Hyper tuning parameter set

'Gradient Boosting': GradientBoostingClassifier(max\_depth=8, n\_estimators=100),

# tuning hyper parameters based on grid search

'Random Forest Classifier' : RandomForestClassifier(max\_depth= 8, min\_samples\_split= 10, n\_estimators=200)

}

# Stratified K-Fold for voting

skf = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

for name, model in models.items():

accuracy\_scores = []

for train\_index, test\_index in skf.split(X\_train\_transformed, y\_train):

X\_train\_fold, y\_train\_fold = X\_train\_transformed[train\_index], y\_train.iloc[train\_index]

model.fit(X\_train\_fold, y\_train\_fold)

y\_pred = model.predict(X\_train\_transformed[test\_index])

accuracy = accuracy\_score(y\_train.iloc[test\_index], y\_pred)

accuracy\_scores.append(accuracy)

print(f"{name} Average Accuracy: {np.mean(accuracy\_scores):.3f}")

#### Results

The results from my models are as listed below:-

* Logistic Regression Average Accuracy: 0.585
* Decision Tree Average Accuracy: 0.634
* **Gradient Boosting Average Accuracy: 0.666**
* Random Forest Classifier Average Accuracy: 0.645

The Gradient boosting was the most accurate and the hyperparameters were tuned.

### Hyper Parameter Tuning

I used GridSearch CV for tuning the Hyper parameter for both Random Forest and Gradient boosting algorithms.

The key code snippets for the GridSearchCV is below:-

from xgboost import XGBClassifier

from sklearn.model\_selection import GridSearchCV

##### # Define the parameter grid

param\_grid = {

'n\_estimators': [50, 100, 200],

'max\_depth': [3, 5, 8],

'min\_samples\_split': [2, 5, 10]

}

rf\_model = RandomForestClassifier(n\_estimators=100, max\_depth=5, random\_state=42)

grid\_search = GridSearchCV(estimator=rf\_model, param\_grid=param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train\_transformed, y\_train)

best\_params = grid\_search.best\_params\_

###### print("Best Hyperparameters:", best\_params)

The parameters were then copied into the model code and model refitted.

#### Result Visualisation

The successful models were then plotted and important top features extracted.

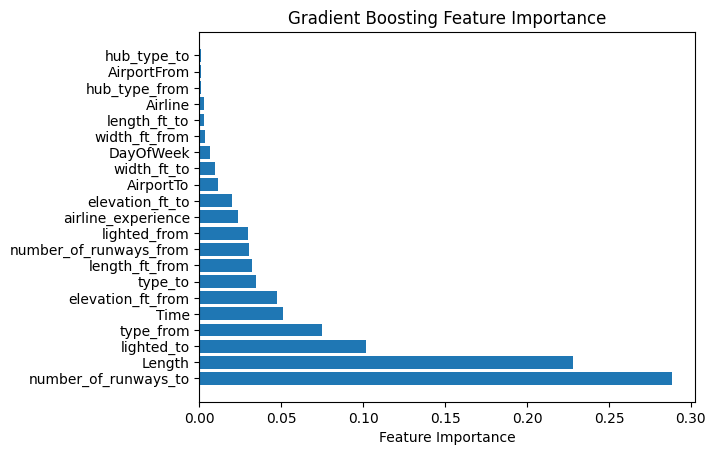


Figure 10 Gradient boosting. Features by importance

##### Top 10 Features (Gradient Boosting):

1. number\_of\_runways\_to: 0.288
2. Length: 0.228
3. lighted\_to: 0.102
4. type\_from: 0.075
5. Time: 0.051
6. elevation\_ft\_from: 0.048
7. type\_to: 0.035
8. length\_ft\_from: 0.032
9. number\_of\_runways\_from: 0.030
10. lighted\_from: 0.030

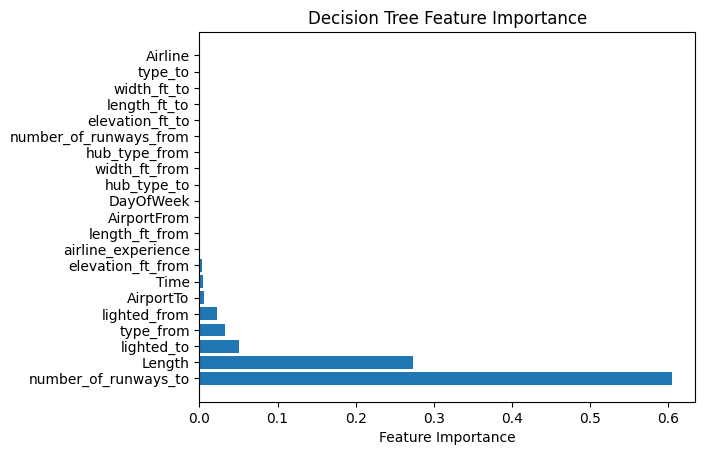


Figure 11 Decision Tree: Top Features

##### Top 10 Features (Decision Tree):

1. number\_of\_runways\_to: 0.605
2. Length: 0.273
3. lighted\_to: 0.051
4. type\_from: 0.033
5. lighted\_from: 0.023
6. AirportTo: 0.006
7. Time: 0.004
8. elevation\_ft\_from: 0.004
9. airline\_experience: 0.001
10. length\_ft\_from: 0.000

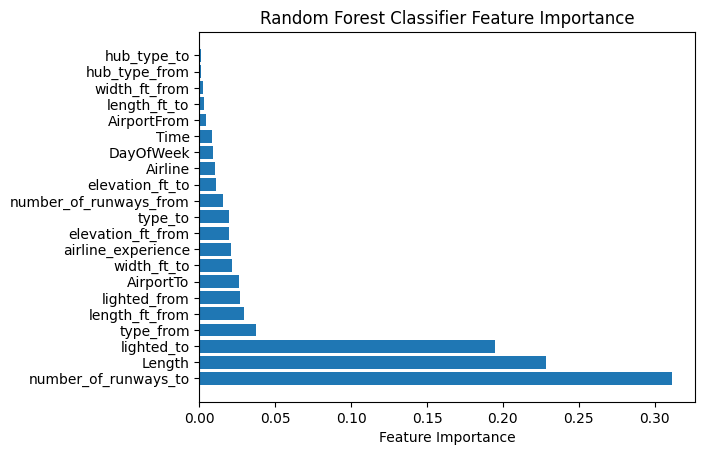


Figure 12 Top Features Random Forest Classifier

##### Top 10 Features (Random Forest Classifier):

1. number\_of\_runways\_to: 0.311
2. Length: 0.229
3. lighted\_to: 0.195
4. type\_from: 0.037
5. length\_ft\_from: 0.029
6. lighted\_from: 0.027
7. AirportTo: 0.026
8. width\_ft\_to: 0.021
9. airline\_experience: 0.021
10. elevation\_ft\_from: 0.020

# SQL Analysis

### Approach

The SQL code for the project was deceptively simple. The major challenge was in the data size and error handling techniques would have been suitable.

All three tables from the dataset were imported into sql after making tables and new schema.

### Challenges

With the data set size of 5 lakh, altering the table were major changes were required was difficult and my laptop frequently timed out despite increasing the global setting of wait time. I couldn’t run this multiple join query

SELECT r.airport\_ref, COUNT(\*) AS num\_delayed\_flights

FROM airline a

JOIN airports ap ON a.AirportTo = ap.iata\_code

JOIN runways r ON ap.id = r.airport\_ref

WHERE Delay = 1 AND r.airport\_ref IN (

SELECT airport\_ident

FROM runways r

GROUP BY r.airport\_ident

HAVING COUNT(DISTINCT r.id) >= 10

)

GROUP BY r.airport\_ref;

### Workaround

I exported a flat joined df from pythin into sql and ran my queries. As joins were not involved, the execution was fast.

### Results

The results of my sql are below:-

A screenshot of a computer

Description automatically generated

Figure 13 Dealyed Flt: Runway >10

A screenshot of a computer

Description automatically generated

Figure 14 Delayed Fly by Airlines

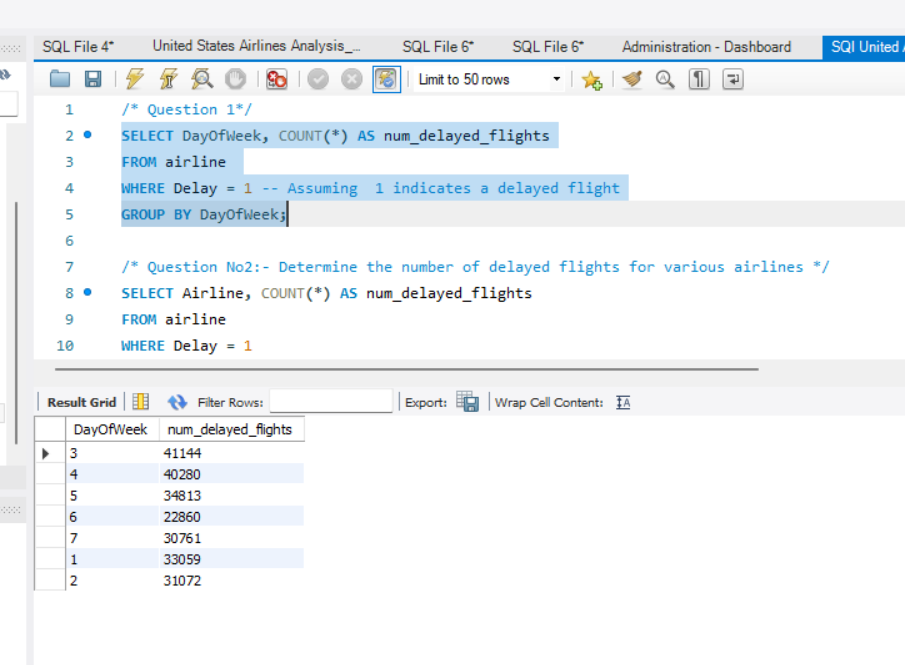


Figure 15 Delayed Fly by Day of week

## Tableau Visualisation

The tableau visualization were made to explain the dependance of various factors on the flight punctuality. It also gives a broad over view of the scale of the operations.

The entire visualization was made in a story. The picturization is below:-

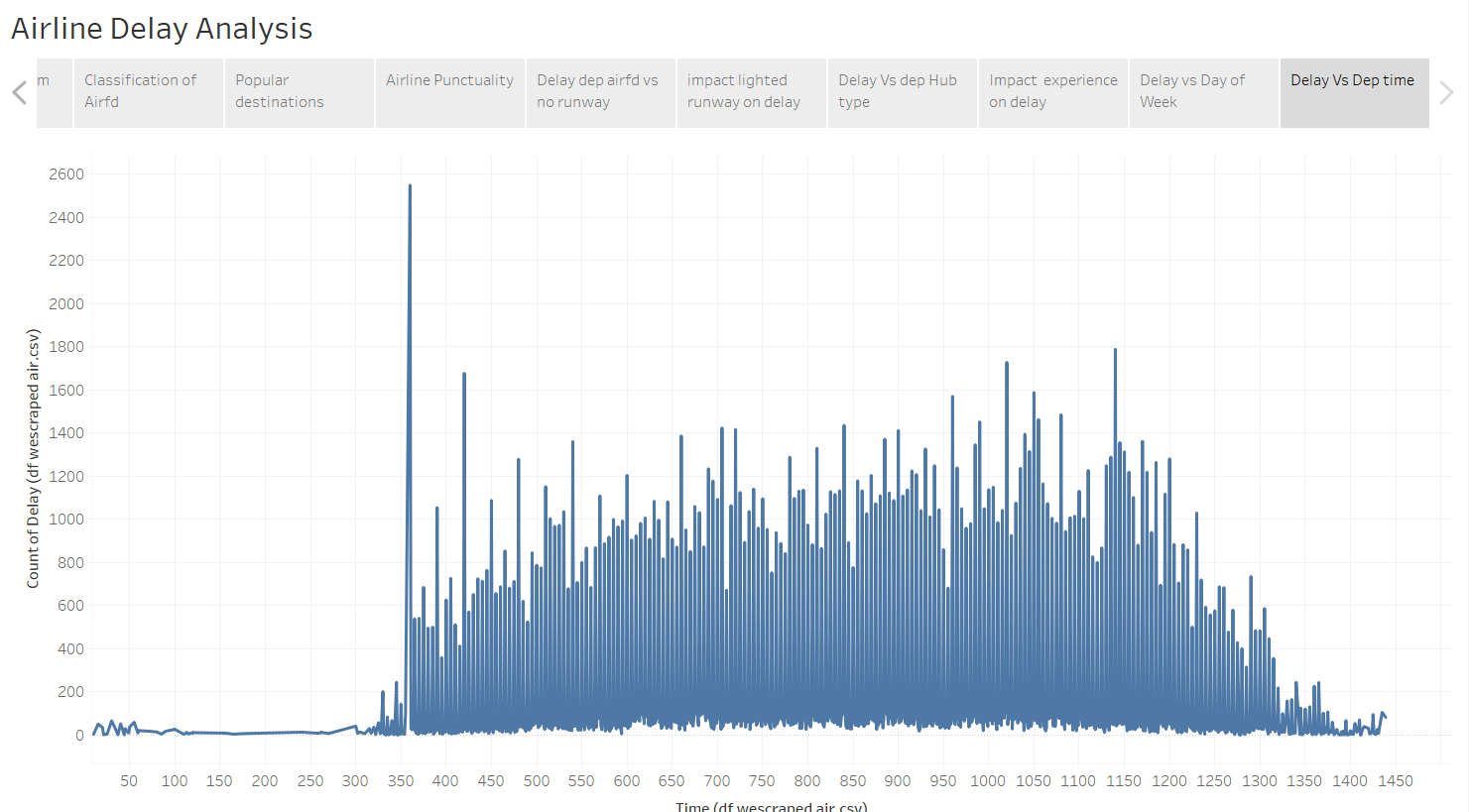


Figure 16 Visual Delay Vs Time dep

A graph on a white background

Description automatically generated

Figure 17 Visual Delay Vs Day of Week

A graph of a bar chart

Description automatically generated with medium confidence

Figure 18 Op Experience Vs Delay Analysis

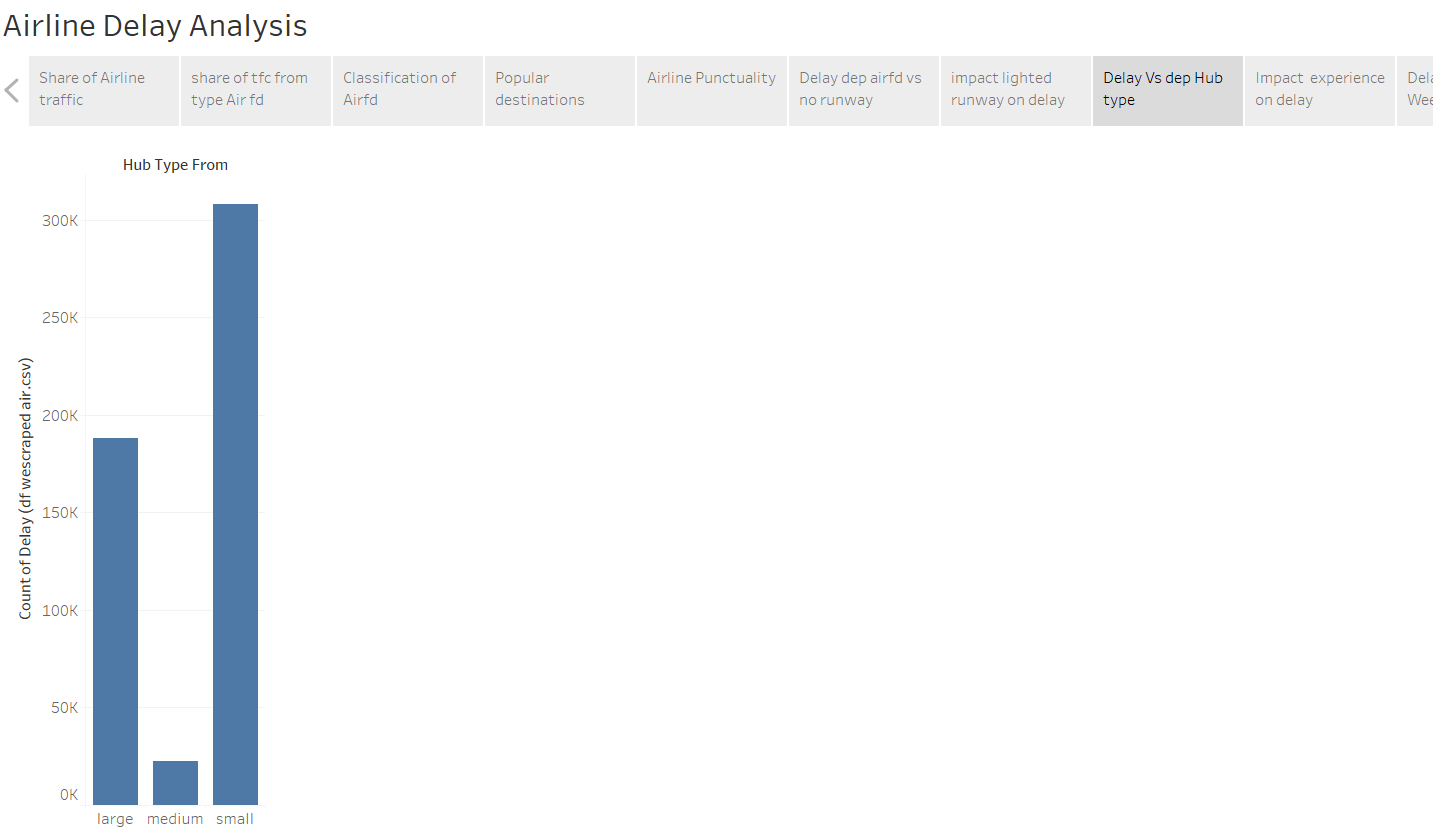


Figure 19 Delay based on Passenger capacity AF

A screenshot of a graph

Description automatically generated

Figure 20 Air Fd delay plot with number of runways

A graph of a bar chart

Description automatically generated with medium confidence

Figure 21 Airline Punctuality

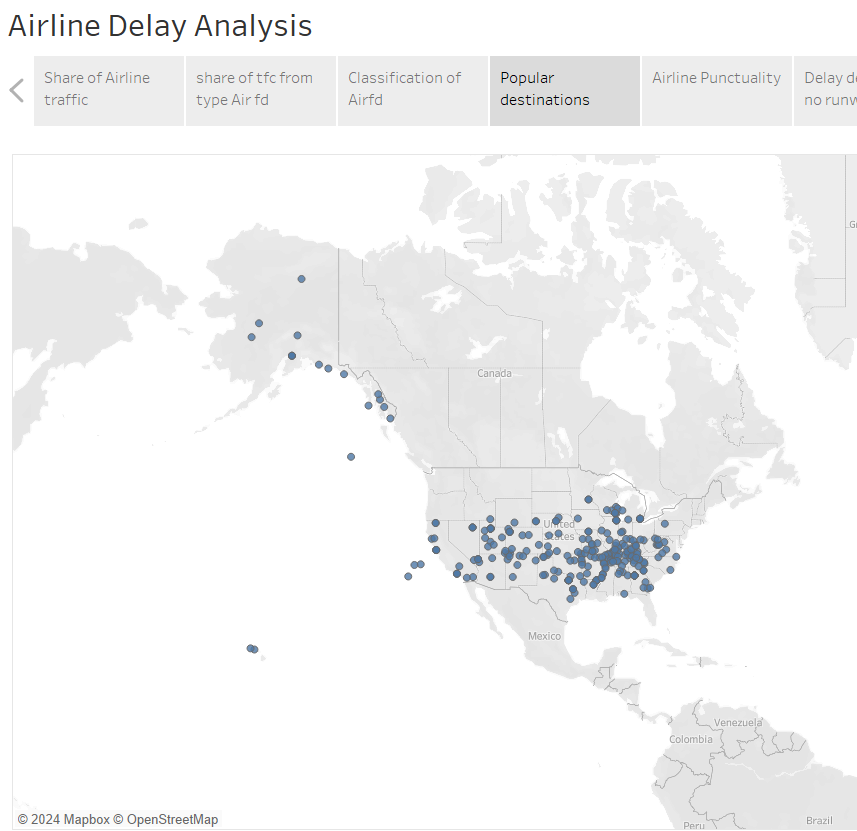


Figure 22 Most popular destinations

A graph of a bar chart

Description automatically generated with medium confidence

Figure 23 Largest operators by no of airports

Link on Public Tableau Server:

<https://public.tableau.com/views/UnitedAirlinesCapston/AirlineDelayAnalysis?:language=en-US&publish=yes&:sid=&:display_count=n&:origin=viz_share_link>

## Conclusion

This project was a challenge and the models made can be tuned better with more understanding. The size of the data made this realistic. Testing data through multiple models and hyperparameter tuning were new in this project.

Overall this was good culmination to my course.