UNDERSTANDING FLIGHT DELAYS

Abstract:

Flight delays have been the most fiendish and inevitable albatross to the world's aviation industry. An accurate estimation of flight delay is important in order to avoid consequences, thereby we are designing an application to detect the flight delays, which uses data collected from numerous websites and data sources and is currently being deployed to produce the results to users based on the effective factors of delay.

Introduction:

Delay is a key point in air transportation activity as it externalizes the negative impacts, mainly economic, for passengers, airlines, and airports. Given the uncertainty of the occurrences in delays, which could be due to the circumstances as assumed- Weather Conditions, delay due to wheels off, taxi-in, etc. This appears to be a complicated and multidimensional issue for all the aviation industry.

To better understand the entire flight ecosystem, huge volumes of data from commercial aviation are collected and stored in databases. Submerged in this massive amount of data, this project contributes by presenting an analysis of the available dataset on flight arrival and flight delay perspective.

Dataset

The datasets gathered for the analysis part includes data source collected from:

1. Bureau of Transportation Statistics:

This data source contains information on approximately 200 million domestic US flights reported to the United States Bureau of Transportation Statistics. It provides a complete outright statistic about each flight (such as date, time, departure airport, arrival airport) and the amount of time the flight was delayed. We collected data from 08-2020 to 07-2021 from this website to build our flight delays dataset.

Source: https://www.transtats.bts.gov/

2. NOAA's- National Centers for Environmental Information (NCEI):

1. This data source allows access to numerous chronicles on environmental weather data on earth, which are comprehensive to atmospheric, coastal, oceanic, and geographical data. This is the dataset we are using to build the weather data.

2. The original weather dataset did not match with any columns in our flight dataset, so we had to use another dataset called the station dataset that has the longitude and latitude values of each airport station and the DOT code of each airport. We merged the station dataset with the weather dataset based on the longitude and latitude of the airport to form a new dataset that is suitable for merging with our flight dataset. Source: https://www.ncei.noaa.gov/access/search/index

3. The weather dataset includes the hourly statistics of all the weather elements like wind, precipitation, temperature, etc from 08-2020 to 07-2021.

Features

We used the following features to form our dataset and train our model:
1. 'YEAR': Year of the flight. (yyyy)
2. 'MONTH': Month of the flight. (mm)
3. 'DAY_OF_MONTH': Day of the month. (dd)

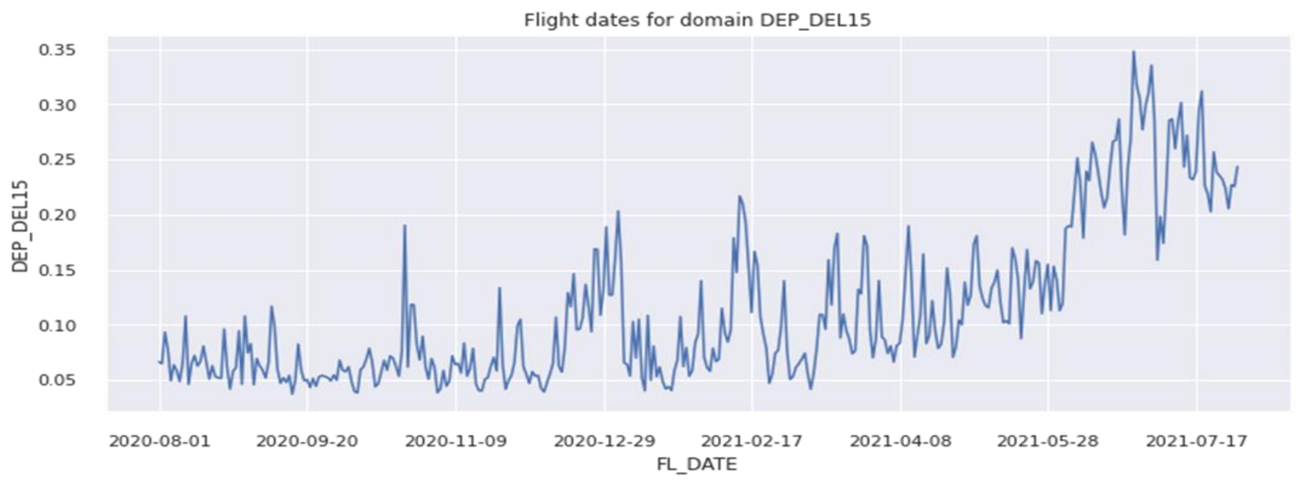


Figure 1: Plot between the number of delays in 15 minutes and the flight date.

4. ‘DAY_OF_WEEK’: Day of the week. (1 [Monday] to 7 [Sunday])
5. ‘OP_CARRIER_AIRLINE_ID’: Unique Integer ID associated with a specific Airline.
6. ‘ORIGIN_AIRPORT_SEQ_ID’: An identification number assigned by US DOT to identify a unique airport at a given point in time.
7. ‘DEST_AIRPORT_SEQ_ID’: An identification number assigned by US DOT to identify a unique airport at a given point in time.
8. ‘Departure_Hour’: Scheduled Departure Time. (hh - From 00 to 24)
9. ‘Real_Departure_Hour’’: Actual Departure Time (hh - From 00 to 24)
10. ‘TAXI_OUT’: Taxi Out Time. (Minutes)
11. ‘WHEELS_OFF’: Wheels off time. (Minutes)
12. ‘CRS_ELAPSED_TIME’: CRS Elapsed Time of Flight. (Minutes)
13. ‘AIR_TIME’: Flight Time. (Minutes)
15. ‘DISTANCE’: Distance between source and destination. (Miles)
16. ‘HourlyWindSpeed’: Wind speed
17. ‘HourlyVisibility’: Numerical quantity that represents the sky conditions.
18. ‘HourlyPrecipitation’: Hourly Precipitation values at every station.
19. 'HourlyStationPressure': Hourly Station Pressure values at every station.
20. 'HourlyRelativeHumidity': Hourly humidity values at every station.
21. 'HourlyDryBulbTemperature': Hourly atmospheric temperature at every station.

Analysis:

In Figure 1, the X-axis represents the flight dates. The Y-axis represents the departure delays in intervals of 15 minutes. As we observe, the departure delay for the year 2020, is steady starting from Aug, Sep, and Oct, while it gradually increases in the months Nov and Dec, remaining the same until Feb 2021, and fluctuating for about 3 months from March 2021 to May 2021 and drastic rise in peak during the months June and July which could be resulting from weather delays, as in Monsoon/Summer.

Figure 2 shows the delay of flights based on 5 different types of delays: Carrier Delay, Weather Delay, National Aviation System(NAS) Delay, Security delay, and Late Aircraft delay. We can see that carrier delay is significant among all other delays and security delay is the least significant type of delay. Moreover, all the delays are related to one another, as in all the delays are either increasing or decreasing at the same time.

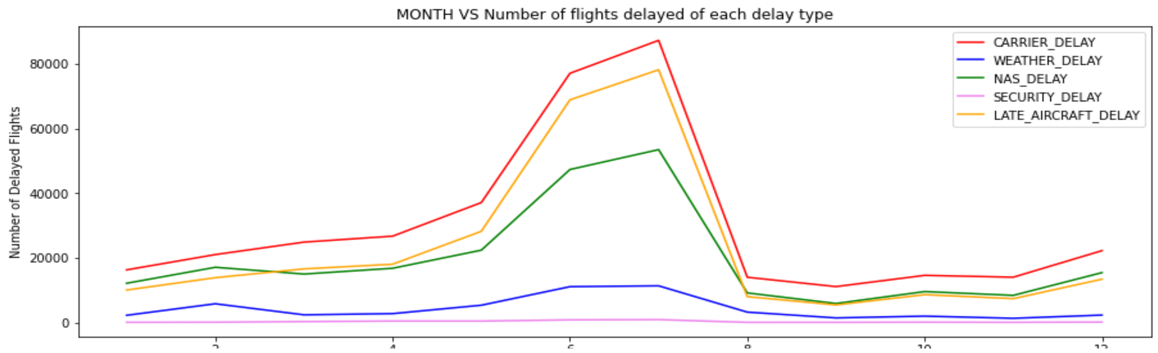


Figure 2: Plot between Month and number of delayed flights for every delay type.

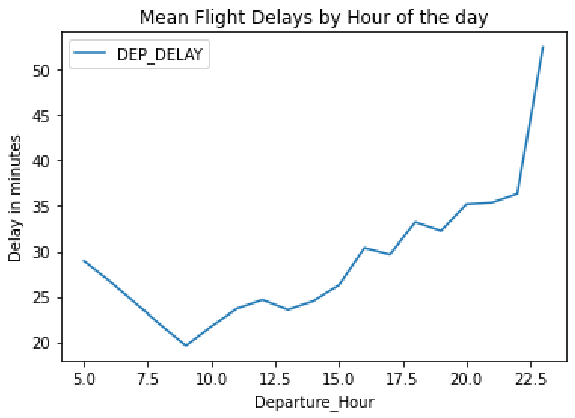


Figure 3: Plot between departure hour of the flight and departure delay.

In Figure 3, we can see that mean flight delays increase as we move along the day. Flight delays are serious at night which indicates that the airports are busy at night.

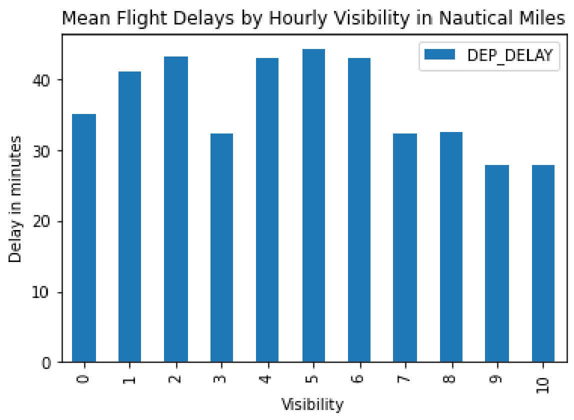


Figure 4: Plot between sky visibility and mean flight delays.

In Figure 4, we observe that delays are high when the visibility is low. Visibility is the measure of the distance at which an object or light can be discerned and is measured in nautical miles (NM). Sky visibility is low when the sky is filled with clouds or when there is rain, snow. So, it is reasonable to expect that delays are more when the visibility is low and less when the sky is clear (high visibility).

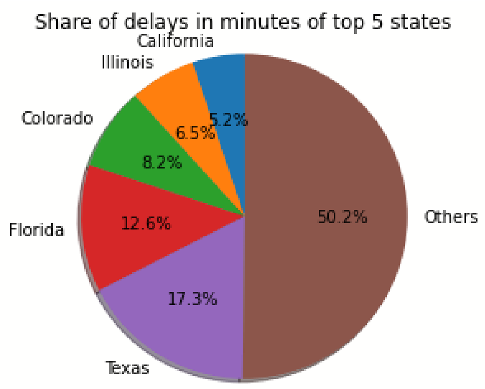


Figure 5: Pie chart showing the shares of delays of states.

We also tried to explore whether the geography of where an airport is located matters much or not. As we can see in Figure 5, Texas has the highest share of delayed flights followed by Florida, Colorado, Illinois, and California. Therefore, warmer states tend to see much more of a delay in flights. We can see that 4 out of the top 10 most populated states in the US contribute significantly to flight delays.

Implementation:

Data Preprocessing

1. Firstly, we took care of all columns with Null values in them. In most of the cases, we had to drop rows containing Null values as the rows consisted of data from various cities/states and replacing NaN values with imputed values from other regions will not be accurate and may skew our regression model.

2. Secondly, we had to take care of a few categorical features such as Airport Name, Origin State, Destination State, etc., and convert them to numerical data types to make it easy for our model to train on these features. Since most of the categorical columns consisted of too many unique values, One Hot Encoding was not preferred. Instead, we assigned each category with a unique incremental integer code.

3. After this, we had to take care to scale our features as there were large amounts of disparity in the range of various columns. We did this by subtracting the minimum value in a column from each value of the column and then dividing this result by the range of the column. We used the normalize() function from sklearn package for the same. The formula for the same is given below as follows:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

4. In our weather dataset, there were Null values for various relevant columns such as “HourlyPrecipitation”, “HourlyWindSpeed”, and more. We handled the weather-related Null values by replacing them with the average value of that feature for the same day. We did this to make the imputed values relevant to a particular day. For instance, on a rainy day, the hourly precipitation would be much higher than the average of the whole year. Therefore, taking the daily average increases the relevance of our features.

5. Finally, we split our main Airport dataset into sub-datasets as per the origin airport and merged the hourly weather dataset of that particular airport to form an origin-specific dataset. We did this for all origin airports. This was necessary because, in

our Django application, the user would be querying for a specific airport's delay. Training our model on data of all other airports would skew our predictions in the wrong direction as compared to training the model only on the relevant sub-dataset. The detailed implementation process is provided in the section below.

Final Implementation

Our project consists of a Django web application. This application provides an HTML form that allows the user to query for a specific flight. The inputs taken from the user are Origin and Destination Airports, Date, Airline Name, and the Scheduled Departure Time.

In our backend, the flow of our application is as follows:

1. Use the input values to get the airport code and read the appropriate CSV file for the same. (For eg, if the user inputs 'New York - JFK', we would read JFK.csv which has airport-related data just for JFK.)

2. Merge the airport-specific dataset with our preprocessed weather dataset to allow the model to gain a complete picture of the features for training.

3. Train our Decision Tree Model on this dataset and calculate performance metrics of the trained model such as Variance score, R2 score, Mean Absolute Error, etc.

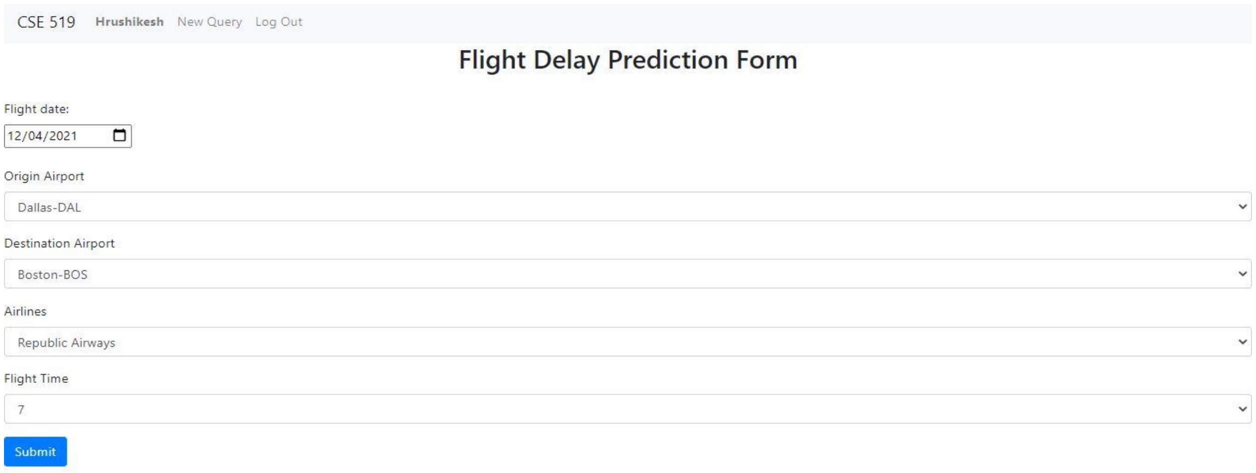
4. Use the date and time values to call our Weather API and get hourly predictions of weather for the period of the flight's scheduled departure. In case our API does not return the weather data required (if there is an error code in our response), then we impute the weather values for our prediction tuple.

5. Form our test dataset containing one tuple (User's query) and feed it to our trained model for prediction. Some features of this test tuple are given by the user itself (such as date, time, airline, origin airport, etc.) while others are retrieved from our dataset using logical inferences on the user input (For instance, to get the real departure time, we need to add our averaged "Wheels Off" and "Taxi Out" values to our scheduled departure time).

6. Return the prediction along with the performance metrics of the model to display on the Results page for the user.

Some of the snapshots for our Tool are shown below:

1) Flight Delay Prediction Form:

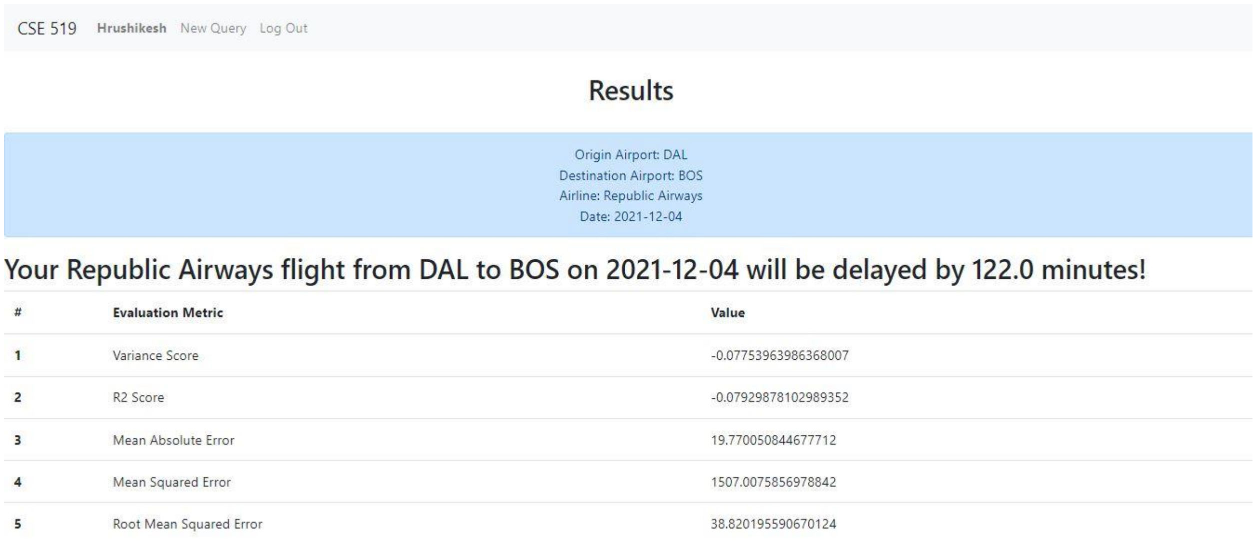


Flight Delay Prediction Form

Flight date:	12/04/2021
Origin Airport	Dallas-DAL
Destination Airport	Boston-BOS
Airlines	Republic Airways
Flight Time	7

Figure 6: Our UI form allows users to make a query about the delays at a certain flight.

2) Results Page:



Results

Origin Airport: DAL
Destination Airport: BOS
Airline: Republic Airways
Date: 2021-12-04

Your Republic Airways flight from DAL to BOS on 2021-12-04 will be delayed by 122.0 minutes!

#	Evaluation Metric	Value
1	Variance Score	-0.07753963986368007
2	R2 Score	-0.07929878102989352
3	Mean Absolute Error	19.770050844677712
4	Mean Squared Error	1507.0075856978842
5	Root Mean Squared Error	38.820195590670124

Figure 8: The results page confirms the details entered by the user on top. It then tells the prediction results (Delay time in minutes). Finally, it provides the evaluation metrics of the trained model.

Results

Since we dynamically train our model on a particular sub-dataset based on the user input of origin and destination airports, the performance of the model also varies accordingly. For instance, a major airport would have a greater number of flights traveling and thus its dataset would be much richer. An example is given above where the origin airport was Dallas while the destination was Boston.

In this case, the model's evaluation metrics are as follows:

Metric	**Value**
Variance Score	- 0.077539639863680
R2 Score	-0.07929878102989352
Mean Absolute Error	19.770050844677712
Mean Squared Error	1507.0075856978842
Root Mean Squared Error	38.820195590670124

Our tool provides these metrics for every query that a user does since every query would have a unique model trained specifically for that airport.

Future Steps:

There is a lot of potential for our tool to be optimized further in the future. Some of the possibilities worth exploring are as such:

- We have provided a User registration and Login interface in our tool as well so that each user may access his/her account. The relevance of this is because once a user queries a prediction, the backend could store the query in the database along with the predicted value of the model. The user could then return later to the tool and input the real delay of their flight so that our model may learn from the same and append the value to the airport dataset. This way, our model would become much better with time as it would learn on real-time data. Moreover, our dataset would get richer with each query as a new tuple would be added to it thus, increasing the efficiency of the future models.

- Another approach to solving the same issue would be to store all predicted tuples in our database. Once the flight date has passed for a tuple, we could query the Bureau of Transportation Statistics for the real flight delay and update our predicted value with the real one and then append the tuple to our dataset as before. This would make the tool seamless by reducing user interaction and help make our model better as API fetched values would be more accurate than user-inputted ones.

- Since we are fetching weather data predictions also from an API, they may not necessarily be accurate. Once the flight date has passed for a tuple, we could fetch the past weather data of that date and learn about the offset caused by the weather prediction API. This would result in higher accuracy of the weather data used which would once again, benefit our model.

- As of now, we have used a Decision Tree as our prediction model. However, in the future, we could use multiple models at the backend, compare their performances and respond with the best performing model. This would be crucial as different models perform differently for each dataset. Thus, we would be dynamically picking the best fit model for our dataset accordingly.

- Instead of training our model each time for an airport, we could train it the first time and then store the model as a pickle file in our dataset. Next time a user queries for the same airport, we could simply pick up the pickle file and use that for prediction. This would save a lot of time in retraining the model from scratch in every iteration. Rather we just train it for the added tuple in every iteration. This would not compromise the incremental efficiency of the model, nor would it cause delays that generally occur while training a model.

REFERENCES

[1]. Study of Flight Departure Delay and Causal Factor Using Spatial Analysis - Shaowu Cheng, [1] Ruiwei Liu,[2] Xiao Luo,[3] and Qian Luo[3]

[2]. Flight delay prediction based on deep learning and Levenberg-Marquart algorithm -Maryam Farshchian Yazdi, Seyed Reza Kamel.

[3]. Understanding Flight Delays at U.S. Airports in 2010, Using Chicago O'Hare International Airport as a Case Study -Paul Blackwood.

[4]. Statistical characterization of airplane delays Evangelos Mitsokapas, Benjamin Schäfer, Rosemary J. Harris &Christian Beck.

[5]. Complexity Analysis on the Influence Factors of the Flight Delay Risk Based on SNA Yueyao Wang[1], Yun Li[2]*