Predictive Model for Flight Delay Data

Final Report of DSC 148 Winter '23

Chester Kai Ni, Sujay Talanki
A16019458, A16433981
Professor Jingbo Shang
Winter 2023, March 19, 2023

Introduction

As a cornerstone of long-distance travel, flying is relied upon as a vital mode of transportation by millions of travelers every year. Despite this, due to its logistical complexity, flights are prone to delays and cancellations. Travelers intending to use flights as part of a trip will likely have to take into account the possibility of delays or cancellations affecting their plans. In this project, we build a model to predict the extent to which any given flight will be delayed based on factors such as airline, location, time, and more.

1. Dataset

1.1 Choosing a Dataset

For this project we use a dataset "Airlines Delay" [1] uploaded to Kaggle by user Heemali Chaudhari under Creative Commons License CC0: Public Domain [2]. The data aggregated in this dataset was originally collected by the U.S. Department of Transportation's Bureau of Transportation Statistics to track flight delays, cancellations, and other details, and features over 1.9 million data points of flights spanning the year 2008.

To provide a background understanding of the state of the United States in the year 2008, we can reference Wikipedia's list of significant events in 2008 [3]. One of the most important events that occurred during 2008 was the Financial Recession.

In September 2008, multiple big banks collapsed and arguably marked the beginning of a collapse of the global economy.

Given that our data is exclusively taken from the year 2008, this and other major global events will likely result in our model performing with lower accuracy on flights from other years. This concern is outside of the scope of our project, but we note that the most straightforward method to mitigate the impact of current events on our model would be to use data collected from a span of multiple years.

1.2 Features

A complete list of features used is provided as follows with explanations and formatting as necessary:

Numeric Features

Month – (1-12)
DayofMonth – (1-31)
DayOfWeek – (1-7)
DepTime – Actual time of departure
CRSDepTime – Scheduled time of departure
ArrTime – Actual time of arrival
CRSArrTime – Scheduled time of arrival
ActualElapsedTime – Actual duration between arrival and departure (mins)
CRSElapsedTime – Scheduled duration between arrival and departure (mins)
AirTime – Flight duration (mins)

ArrDelay – Duration between planned and scheduled arrival time (mins)
Distance – Distance of flight (miles)
TaxiIn – Duration between landing and arriving at gate (mins)
TaxiOut – Duration between leaving gate and takeoff (mins)
Cancelled – Boolean for whether the flight was cancelled (0-1)
Diverted – Boolean for whether the flight was diverted (0-1)
CarrierDelay – Delay due to carrier (mins)
WeatherDelay – Delay due to weather (mins)
NASDelay – Delay due to NAS (National Air Space) logistics (mins)
SecurityDelay – Delay due to security (mins)
LateAircraftDelay – Delay due to aircraft lateness (mins)

Categorical Features
UniqueCarrier – Carrier code
Origin – Flight origin location
Dest – Flight destination location
CancellationCode – Type of cancellation if applicable: A for carrier, B for weather, C for NAS, and D for security

1.3 EDA
Before proceeding with building and tuning a model, we begin by performing Exploratory Data Analysis on the chosen dataset. Given that there are almost 2 million data points and almost 30 features, we make sure to observe each feature and determine which ones would be useful for our analysis. In addition, we develop various visualizations to probe for trends or patterns that may provide early insight on the dataset.

We first generate a correlation matrix and visualize it as a heatmap [Fig 1].

This provides an easily accessible means of identifying features that relate with each other. We focus on features that have high correlation with the feature “DepDelay”.

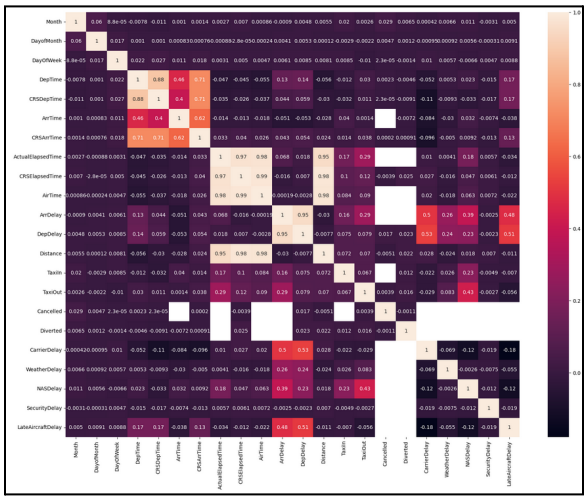

Fig 1: Correlation Matrix

Note that we later use a categorical feature to replace “DepDelay” for our classification model, but for now “DepDelay” serves sufficiently.

Here we find that unsurprisingly, the feature “ArrDelay” has a correlation of 0.95 with “DepDelay”. This implies that in a majority of cases, flights that depart behind schedule were already behind schedule when they arrived at the airport. Furthermore, the feature “DelayCategory” has a correlation of 0.88, implying that knowing the reason for delay is often enough to have an idea of the magnitude of delay.

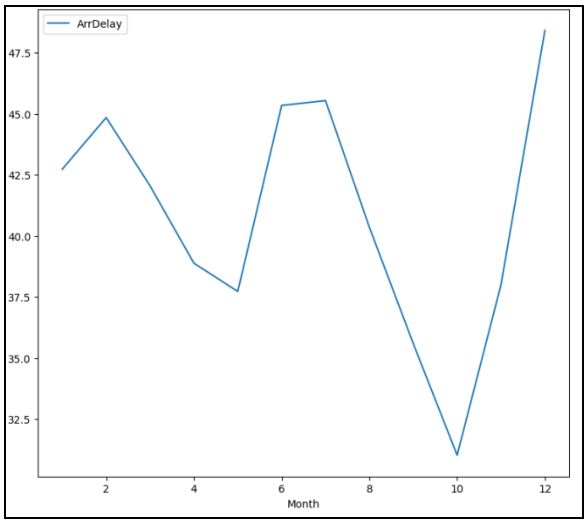

Fig 2.1: Average Delay by Month (mins)

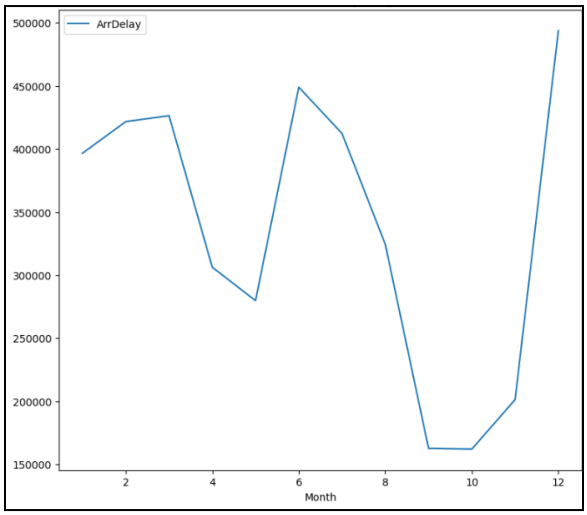

Fig 2.2: Sum of Delay by Month (mins)

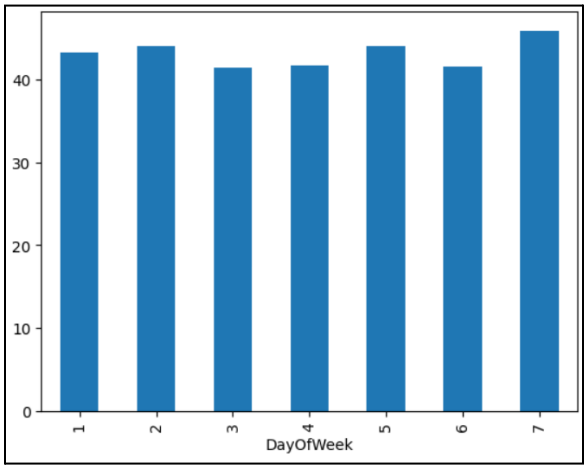

Fig 2.3: Average Delay by Day of Week (mins)

From there, we observe the mean and sum delay throughout the months of the year. For this, we plot two line graphs [Fig 2.1, Fig 2.2]. While these figures do not present a high level of granularity, they do appear to show that spring and fall months generally see less delay, whereas summer and winter months see more.

This may be a result of summer and winter months coinciding with peak vacation times during the year. On the other hand, the dip in delays during September and October may be related to the financial collapse that climaxed during that time.
Our data is insufficient to provide any conclusive explanations for these trends.

We also observe the mean delay per day of week, but find that there do not appear to be any notable trends that occur [Fig 2.3].

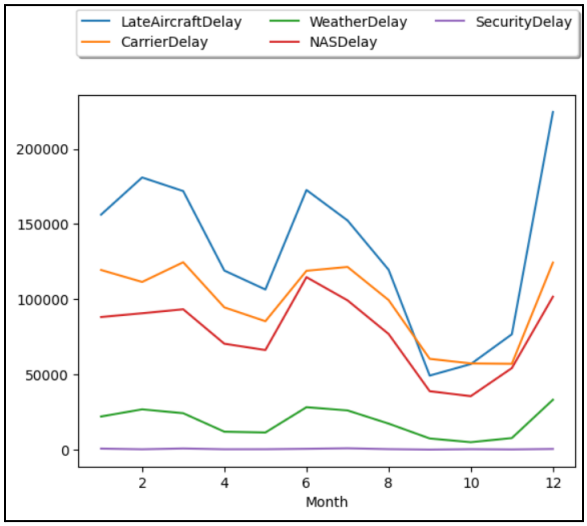

Fig 3: Instances of Delay Type by Month

To further explore the progression of flight delays throughout the months, we plot a multiple-line graph showing the count of each delay type by month [Fig 3]. This chart shows that through most of the year, the most common type of delay is Late Aircraft Delay, followed by Carrier Delay.

This appears to align with the earlier findings from the correlation matrix that show some amount of correlation between the total departure delay and the features "LateAircraftDelay" and "CarrierDelay".

Most travelers likely have some amount of preference for different airlines, and a popular reason for these preferences may be the frequency and amount of delays associated with each airline. To investigate this, we plot a bar graph showing the average duration of delay by airline [Fig 4].

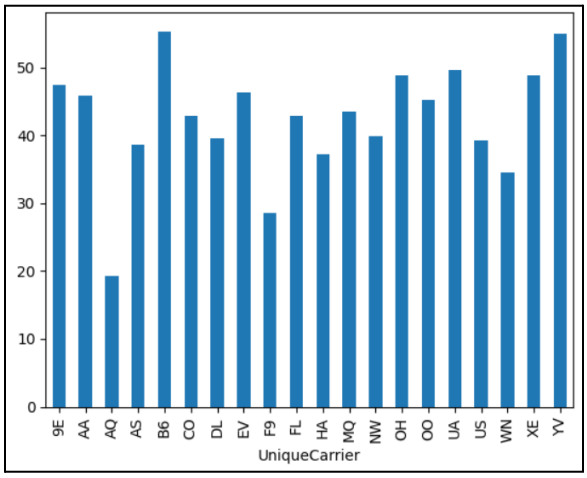

Fig 4: Average Delay by Unique Carrier (mins)

Note that the unique carrier codes for each airline follows the IATA format established by the International Air Transport Association [7]. For a legend containing the full name of each airline and more information, refer to the Federal Aviation Administration's web page titled ASQP: Carrier Codes and Names [8].

To further explore the distribution of delay duration, we generate a density plot of the feature "DepDelay" [Fig 5]. This plot shows that while almost all flights are delayed by less than 2 hours (120 minutes), there appear to be some major outliers in our dataset.

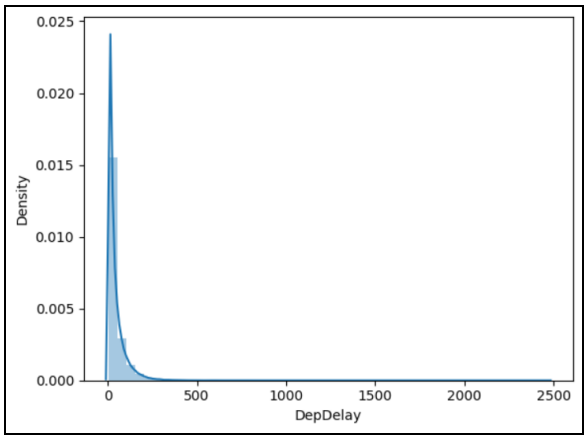

Fig 5: Delay Duration Density Plot

Given that the maximum values appear to be almost 2500 minutes (41.7 hours), it would appear either that some flights were truly delayed for almost two full days or a massively high number was inputted into this dataset to effectively serve as a placeholder for flights where departure delay time was not applicable.

1.4 Classification Task

For our classification task, we convert the given numeric "DepDelay" feature into a categorical feature. We choose to do so because intuitively, delay duration can be categorized into categories that roughly correspond to how significant a delay is.

We initially also considered creating a classification task to predict whether a flight would be entirely canceled or not. We decided not to move forward with that due to the very small proportion of flights that are canceled. Within the dataset of almost 2 million flights, only around 600 flights are canceled.

Proceeding with our chosen predictive task, we design 6 categories to serve as bins for "DepDelay", beginning with a delay duration of less than or equal to 30 minutes.

We intuitively deem this amount to be at most a mild inconvenience for most travelers. From here, we progress to 1 hour, 2 hours, 3 hours, 5 hours, and more than 5 hours.

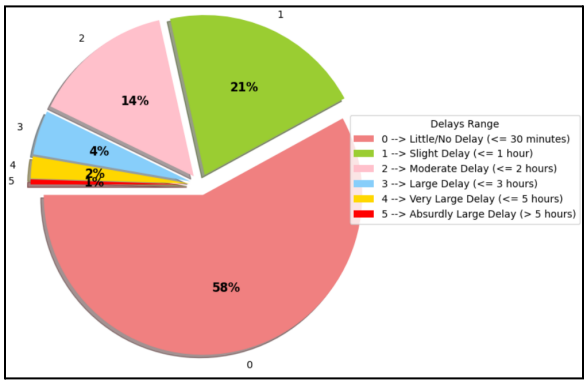

Fig 6.1: Initial Delay Categories Pie Chart

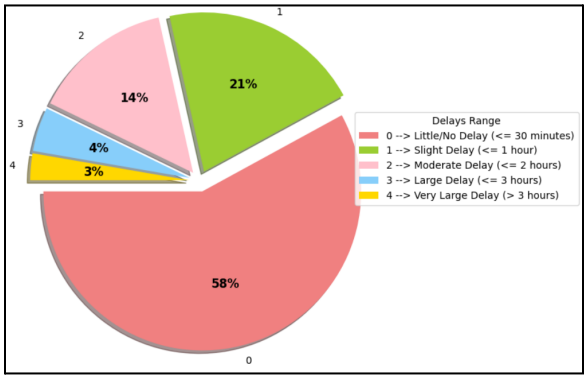

Fig 6.2: Updated Delay Categories Pie Chart

To visualize the proportion of data points that fall into each category, we visualize using a simple pie chart [Fig 6.1]. From this visualization, we clearly see that only around 2% of flights fall into category 4 (Very Large Delay; 3-5 hours) and only around 1% of flights fall into category 5 (Absurdly Large Delay; more than 5 hours). Therefore, we combine these two categories to create our finalized categorical feature "DelayCategory" as our y-value and generate an updated visualization correspondingly [Fig 6.2].

Finalized Categories:
Little/No Delay (0-30 minutes)
Slight Delay (30-60 minutes)
Moderate Delay (1-2 hours)
Large Delay (2-3 hours)
Very Large Delay (3 hours or more)

2. Predictive Model

2.1 Defining the Predictive Task

The ultimate goal of this project is to predict the amount of time that a flight will be delayed given a series of other factors. For this problem, we separate flights by delay duration using bins (refer to section "1.4 Classification Task"). Therefore, this is a classification task.

2.2 Evaluation

We start by exploring multiple classifier algorithms and evaluate them by comparing precision, recall, and F1 statistics. We make these preliminary comparisons between classifiers without tuning hyperparameters and per-model feature engineering to provide an equal platform and determine the one best suited for our dataset.

2.3 Baseline

After choosing a classifier, we proceed to further improve our model by tuning hyperparameters and feature engineering. We use the initial model built using the chosen classifier without tuning hyperparameters and feature engineering as a baseline to compare and monitor our model. To this end, we produce confusion matrices and use accuracy, precision, and recall statistics to monitor the performance and progression of our model.

2.4 Classifiers

We begin by exploring four classifier algorithms: Random Forest Classifier, XGBoost Classifier, LightGBM Classifier, and Support Vector Machine Classifier.

Random Forest Classifier (RFC) is a popular standard classifier algorithm that uses a "forest" of decision trees to perform classification. By implementing multiple decision trees, the RF algorithm is able to mitigate the tendency of models based on decision trees to overfit.

XGBoost Classifier (XGB) is a classifier algorithm that implements gradient boosting to improve efficiency, flexibility, and portability [4]. The GBDT parallel tree boosting algorithm is also contained within the XGBoost library, and can be used with XGBoost and other predictive models to optimize performance.

LightGBM (LGBM) is an algorithm built by Microsoft that also implements gradient boosting. It is capable of highly efficient performance with larger datasets, making it easier to work with, especially during hyperparameter tuning and feature engineering [5].

Support Vector Classification (SVC) provided by SciKit Learn uses support vectors to build memory efficient predictive models. SVM based algorithms take advantage of kernel functions to work with high-dimensional data, with different kernels providing additional flexibility across different types of datasets and predictive problems [6].

2.5 Preliminary Testing

We build pipelines using libraries provided by SciKit Learn to perform basic preliminary cleaning and feature extraction.

Given the size of our model and limitations in computing power, we take a sample of 1/40 of the dataset.

```
model_xgb = model_performance(xgb.XGB

Precision: 0.9390949248395014
Recall: 0.9185482447649183
F1 Score: 0.9285665412935806

model_lgbm = model_performance(lgbm.L

Precision: 0.9322327413313671
Recall: 0.9111452944945715
F1 Score: 0.9213935705949285

model_rfc = model_performance(rfc(n_j

Precision: 0.8586246951664254
Recall: 0.7516382241743849
F1 Score: 0.7931717058097839

model_svc = model_performance(SVC())

Precision: 0.9135153265326468
Recall: 0.8887742546835012
F1 Score: 0.9007802719133648
```

Fig 7.1: Recall, Precision, F1 Comparison

```
np.mean(cross_val_score(model_xgb, X, y, cv=5))
0.9473553274489035

np.mean(cross_val_score(model_lgbm, X, y, cv=5))
0.9451454712059183

np.mean(cross_val_score(model_rfc, X, y, cv=5))
0.8841777276466978

np.mean(cross_val_score(model_svc, X, y, cv=5))
0.9353765822248892
```

Fig 7.2: Cross Validation Comparison

From the results of these preliminary tests, we find that the XGB and LGBM models achieve the best overall performance. Both RFC and SVC are much slower, even with the smaller sampled dataset, and perform at a meaningfully worse level [Fig 7.1, Fig 7.2].

Given the similar performance level of XGB and LGBM and that LGBM is designed to better perform well with larger datasets, we elect to proceed with LGBM.

3. Model

3.1 Initial Setup

Having chosen a classifier, we can begin the optimization process. To set a baseline, we build a LGBM model with identical configurations as in the previous step. Due to our limitations in computation power, we proceed with 3-fold cross validation instead of 5-fold. Our baseline accuracy score using this model is 0.945, so the goal is to improve upon this score.

3.2 Tuning Hyperparameters

We implement SciKit Learn's Grid Search feature to tune our classifier's hyperparameters. During the Grid Search process. Grid Search provides an interface to test all permutations of any user-specified range of hyperparameters and determines the best configuration based on cross validation [9].

3.3 Feature Engineering

In optimizing our model, we use the list of features listed in section "1.2 Features". The original dataset contains several additional features; any features not listed in the aforementioned section are dropped.

Numeric Features

For each numeric feature, we standardize all values using the Standard Scaler from SciKit Learn's preprocessing library. To address missing values, we impute the feature's mean for missing values.

Categorical Features

For all categorical features, we apply One-Hot Encoding, and we impute the feature's mode for missing values.

3.4 Optimized Model

To compensate for our limits in computational power, we run multiple Grid Searches, each time adjusting the range of parameters as necessary. We complete five rounds of this process, assessing the cross validation score each time.

```
print(lgbm_grid.best_score_)
print(lgbm_grid.best_params_)

0.9526631812502285
{'model__bagging_fraction': 1, 'model__bagging_freq': 190, 'model__feature_fraction': 0.95, 'model__learning_rat
e': 0.18, 'model__n_estimators': 20, 'model__num_iterations': 200, 'model__num_leaves': 40}
```

Fig 8: Final Cross Validation Score

We are able to improve our cross validation score from 0.945 to 0.953 [Fig 8], which is a 0.8% improvement. With access to more time and/or computational resources, we believe there is still plenty of headroom for improvement. We find the best hyperparameters for the LightGBM classifier algorithm as below:

```
for k in lgbm_grid.best_params_:
    print(str(k) + ": " + str(lgbm_grid.best_params_[k]))

model__bagging_fraction: 1
model__bagging_freq: 190
model__feature_fraction: 0.95
model__learning_rate: 0.18
model__n_estimators: 20
model__num_iterations: 200
model__num_leaves: 40
```

Fig 9: Final Hyperparameters

4. Literature

4.1 Airline Flight Delay Prediction Using Machine Learning Models by Yuemin Tang

Stella Yuemin Tang from the University of South California published the paper Airline Flight Delay Prediction Using Machine Learning Models for the ICEBI 2021: 2021 5th International Conference on E-Business and Internet in Singapore [11].

In this paper, Tang explores several Machine Learning models for predicting flight delay on a dataset of flights leaving JFK Airport from November 2019 to December 2020.

Tang reports results on seven different predictive models: Logistic Regression, KNN, Gaussian Naive Bayes, Decision Tree, SVM, Random Forest, and Gradient Boosted Tree. Of these seven models, the Decision Tree model is found to have the best performance in all four metrics used, Accuracy (0.9778), Precision (0.9777), Recall (0.9778), and F1 (0.9778).

4.2 Predicting Flight Delays by Fabien Daniel
Kaggle user Fabien Daniel performs in-depth Exploratory Data Analysis and builds a predictive model using polynomial regression and explores regularization techniques to improve predictive performance. Daniel uses a similar dataset to ours, uploaded directly by the United States Department of Transportation onto Kaggle for flight delay data in the year 2015.

Daniel notably investigates connected flights, observing whether flights are delayed by the immediate travel history of the airplane. In the post, Daniel includes visualizations comparing delays across different airlines, maps containing airline travel paths, and discusses subjects such as temporal variability and the relationship between arrival and departure delay. Daniel's final model achieves a Pearson R Score of -0.037.

5. Results

5.1 Tuned Hyperparameter Results
Based on the results of our hyperparameter tuning, we test the updated model and compare it to the original model by evaluating the cross validation, precision, recall, and F1 scores. Score comparisons are listed below:

Score	Initial	Tuned
Cross Validation	0.945	0.953
Precision	0.932	0.940
Recall	0.911	0.924
F1	0.921	0.932

5.2 Reflection
In the process of completing this project, we work with various predictive modeling techniques to determine the one best suited to our dataset to find that models based on Gradient Boosting are able to achieve the best prediction results with predicting airline delays according to cross validation, precision, recall, and F1 scores and are able to do so relatively efficiently.

Working with a larger dataset of almost 2 million data points, the limits with our computational power were made very evident. Based on the results of our final model and the process of building it, we believe that with access to more time and computational resources, we would be able to further develop our model and significantly improve predictive performance.

In addition, during our Exploratory Data Analysis, we found that our dataset contains sufficient data to allow many other avenues of analysis. We could have chosen several other predictive tasks, such as comparing delay across airlines and mapping flight paths to compare delay across geographical regions.

References

1. Kaggle: Airlines Delay Dataset
https://www.kaggle.com/datasets/heemalichaudhari/airlines-delay
2. Creative Commons License C00: Public Domain
https://creativecommons.org/publicdomain/zero/1.0/
3. 2008 in the United States
https://en.wikipedia.org/wiki/2008_in_the_United_States
4. XGBoost
https://xgboost.readthedocs.io/en/stable/
5. LightGBM
https://lightgbm.readthedocs.io/en/v3.3.2/
6. Support Vector Machines
https://scikit-learn.org/stable/modules/svm.html
7. International Air Transport Association
https://www.iata.org/en/services/codes/
8. ASQP: Carrier Codes and Names
https://aspm.faa.gov/aspmhelp/index/ASQP__Carrier_Codes_and_Names.html
9. SciKit Learn: Grid Search CV
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
10. SciKit Learn: Standard Scaler
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
11. Airline Flight Delay Prediction Using Machine Learning Models
https://dl.acm.org/doi/10.1145/3497701.3497725
12. Predicting Flight Delays
https://www.kaggle.com/code/fabiendaniel/predicting-flight-delays-tutorial