Predicting Flight Delays

Wendy Gao, Shon Inouye, Alex Wu, Andrew Zhang
PSTAT 194 Final Project
March 19, 2018

1. Abstract

The objectives of this binary classification project is to identify factors that are predictive of flight departure delays and compare the performance of different machine learning models that predict whether a flight will be delayed. This project showed that month the flight took place, the airport where the flight departed from, and scheduled departure time were the most influential predictors of departure delay. The predictive models created were logistic regression, decision tree, random forest, and gradient-boosted tree. The model accuracies were improved with a 5-fold cross-validation. Gradient boosted trees performed the best (in terms of accuracy) out all the predictive models, with logistic regression, random forests, and decision tree following.

2. Introduction

Oftentimes when travelling, one can face many inconvenient challenges from packing only TSA approved items in carry-on luggage to getting through busy security lines. After getting through security, the travelling experience can be dramatically worsened when a flight delay occurs. While the process leading up to boarding can be repetitive and foreseeable, flight delays oftentimes seem quite the opposite of that and out of our control. However, research conducted on flights have outlined factors that affect the timeliness of a flight departure. These factors include—but are not limited to—the airline of the flight, airport the plane departs from, and the time and date of the flight.

This project uses the 2015 Flight Delays and Cancellations dataset from Kaggle and aims to identify variables that can be used to effectively predict a departure delay. In addition to the identifying key predictors, the project aims to compare and contrast predictive models and their performances. The original dataset consists of 5,819,079 observations and 31 variables. The variables used to create the predictive models are: Month, Day, Day of Week, Origin Airport, Destination Airport, Airline, Scheduled Time, Scheduled Departure, and Scheduled Arrival. These variables are selected because they offered information given prior to a plane taking off. Refer to the table below for short descriptions of each variable.

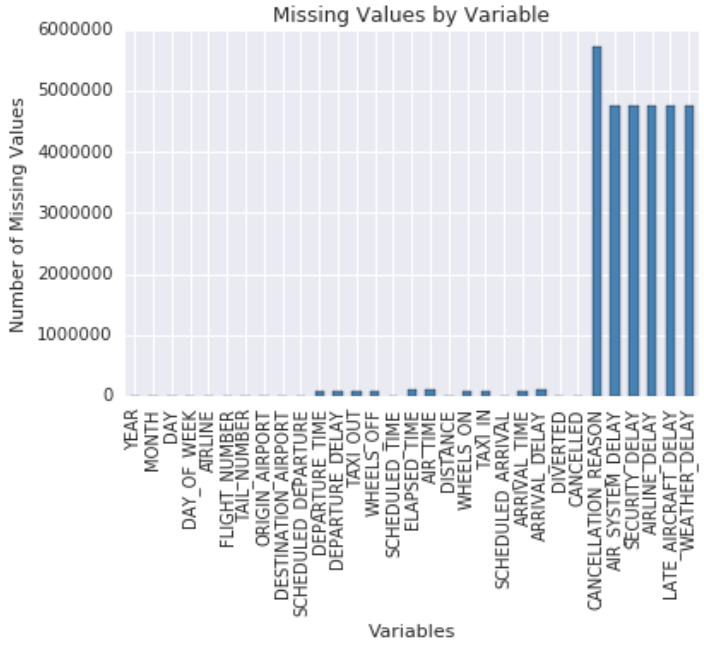
Variable	Variable Description
Month	Month the flight took place.

Day	Day of the month the flight took place.
Day of Week	Day of the week the flight took place.
Origin Airport	Airport the flight departed from.
Destination Airport	Airport the flight arrives at.
Airline	Airline the flight belongs to.
Scheduled Time	Formal flight duration given by airline.
Scheduled Departure	Formal flight departure time given by airline.
Scheduled Arrival	Formal flight arrival time given by departure.

The models created in this project are logistic regression, decision tree, random forest, and gradient-boosted tree.

3. Methods

Preprocessing of the dataset was done in Python 2.7 with both the Pandas and PySpark libraries. The dataset is originally in CSV format. It is read into Python using Pandas. At first, there are 5,819,079. Missing and incomplete observations are removed from the dataset. Since this project focuses on departure delays, a record is considered missing or incomplete when filtering through the DEPARTURE_DELAY variable. The remaining observations that are kept are records that have a value (in minutes) of the departure delay.



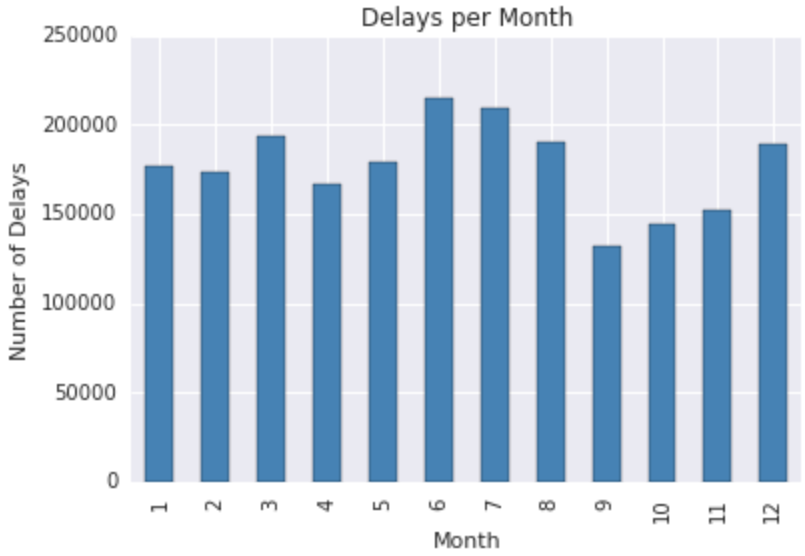
Afterwards, the data balance is evaluated. To do this, the binary dependent variable DELAY is derived from the DEPARTURE_DELAY variable. The DEPARTURE_DELAY variable was how many minutes the plane actually took off from the scheduled departure. If the plane takes off earlier than the scheduled departure, then DEPARTURE_DELAY is negative. DEPARTURE_DELAY is 0 if the plane departs on the exact scheduled departure time, and greater than 0 if the plane departs any later than the scheduled departure time. For observation i, the DELAY variable is 0 if the flight departs early or on time, and 1 if the flight departs any later. Afterwards, the data balance is assessed by the DELAY variable. The dataset is unbalanced, 63% of the data is 'No Delay' and 37% is 'Delayed.' Undersampling is done (because of the large number of observations) on the 'No Delay' observations until the number of 'No Delay' observations equals the number of 'Delay' observations. After a 1:1 ratio is achieved, 20% simple random sample is drawn. The sample is compared with the population dataset to confirm that the random sample is representative of the population sample. The sample is used to form the training and test dataset.

Other preprocessing steps include inputting the airport codes correctly. Observations from the month of October did not have proper IATA airport codes (e.g. LAX, SFO). To address this

issue, an airport IATA code lookup table was created to replace the airport IDs with the airport IATA codes.

Following adjusting the raw data into analyzable data, we began preprocessing in Pandas and Pyspark. The data was adjusted accordingly for the individual predictive models for analysis. This included changing data into the proper types, creating the correct vectors for input, and adding pipelines for our models to access the data.

After initially running our cleaned data through the models, the models yielded low accuracy rates. We attempted to change these accuracy rates through adjustments in pieces of our data in hopes of improving our model accuracy. One of the methods we utilized was OneHotEncoder, which mapped our categorical variables to vectors of binary variables. This helped organize our data into categorical vectors that were read more easily by our models.



Most Machine Learning models require integer types within the feature categories in order to run their algorithms smoothly. We spent a portion of our preprocessing creating dummy variables for categorical variables such as AIRLINE and ORIGIN_AIRPORT. Each of the airlines and origin airports were given unique numbers corresponding to their categorical inputs.

After the preprocessing step was finished, we created a parquet file to hold our updated dataset. This saved time in uploading our data to pandas during every session we worked on the project.

4. Models

After preprocessing the data, the predictive models are built. Four different machine learning models are created with PySpark: Elastic-net logistic regression, Decision tree, Random forest, and Gradient-boosted tree.

All these models are trained in a similar fashion. First, 20% of the dataset is randomly subsetted to create the holdout set. On the rest of the 80% of data, we perform 5-fold cross-validation for parameter tuning. The parameter chosen has the lowest cross-validation average error.

Because the process of training a series of models and then selecting the "best" based on the validation score can yield an overly optimistic estimate of a model's performance, we use the holdout set as an extra check against this selection bias.

The hyperparameters tried with the four different machine learning models:

- Elastic-net logistic regression
    - elasticNetParam (α) = {0, 0.3, 0.5, 0.7, 1}
    - regParam (λ) = {0.1, 0.01}
    - maxIter {0, 1, 5, 10, 25}
    - fitIntercept = {False, True}
- Random forest
    - numTrees = {5, 10, 25}
    - maxDepth = {3, 5,10,15}
    - maxBins = {5, 10, 20, 30}
- Decision tree
    - maxDepth = {3, 5, 10, 15}
    - maxBins = {5, 10, 20, 30, 35}
- Gradient-boosted tree
    - maxIter = {5, 10, 20}
    - maxDepth = {3, 5, 10, 15}
    - maxBins = {5, 10, 20}

5. Results

Model Accuracy

	Decision Trees	**Gradient Boosted Trees**	**Logistic Regression**	**Random Forest**
Training Accuracy	0.608178	0.611186	0.620115	0.528186
Test Accuracy	0.603879	0.608958	0.617391	0.536012

K-fold Cross-Validation. (k = 5 folds)

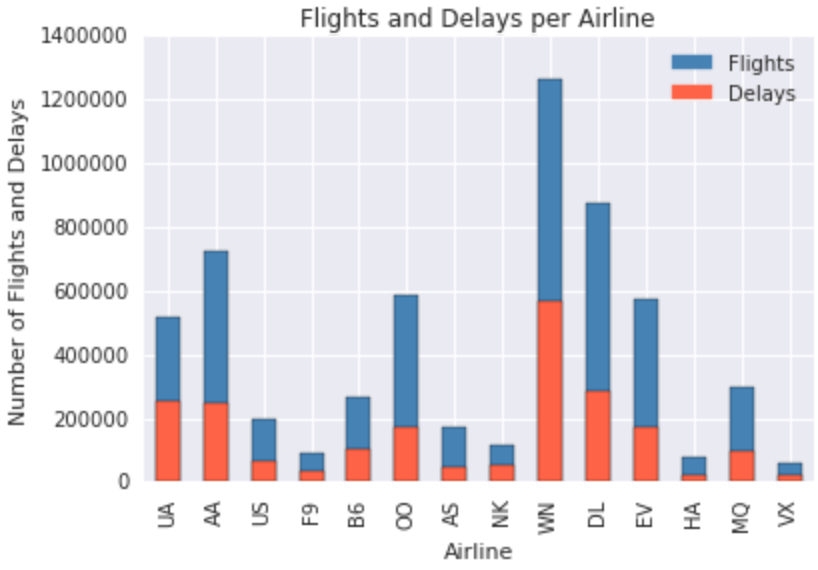
	Decision Trees	Gradient Boosted Trees	Logistic Regression	Random Forests
Training Accuracy	0.626541	0.670167	0.665911	0.644123
Test Accuracy	0.614331	0.630412	0.668337	0.626047

Gradient Boosted Trees performs the best with an accuracy of 66.8% after k-fold cross-validation.

6. Conclusion

We conclude that gradient boosted trees performed the best out of all the models in terms of model accuracy. K-fold cross-validation was essential for selecting hyperparameters to optimize model accuracy.

From evaluating the coefficients from the logistic regression model the top three predictors of departure delay are: month the flight took place, the airport where the flight departed from, and the scheduled departure time. An additional finding was that larger airlines (airlines that had more flights) tend to have more delays in respect to the total number of flights.



7. Acknowledgements

We would like to thank our professor Adam Tashman for introducing us to PySpark and giving us insightful advice to improve our model performances. Additionally, we would like to thank our teaching assistant Javier Zapata for guiding us throughout the quarter.