Predicting Flight Delays at U.S. Airports

ORIE 4740 Final Project

Authors: Sherrie Chen (sc2289), Gallen Gu (gx52), Andrew Zhou(jz956), Nina Xie(jx295)

Abstract

When traveling by air, everyone wants to avoid flight delays which can be caused by a variety of reasons, such as air carrier delay and aircraft leaving late. Using the 2015 Flight Delays and Cancellations dataset that tracks the on-time performance of domestic flights in the United States, we examined the impact of many factors on determining flight delay time. This paper describes our data pre-processing, feature selection, and model fitting processes. Among a mix of linear and nonlinear models that we experimented with, XgBoost model had the best performance with a prediction error under 3 minutes. However, if we were to develop a fast mobile business application for visualizing flight delays, we would recommend using linear models such as ridge regression, which has reasonably good performance of prediction error under 7 minutes.

1. Introduction

Waiting for delayed flights is frustrating for most passengers who have travelling plans or business schedules. Even though passengers are keen to learn about their flight delays, flight delay predictions are not available at all airports. This project will explore a dataset of 2015 flight delays and cancellations in the United States. The source of this data is the U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics, which tracks the on-time performance of domestic flights operated by large air carriers. There are 5,819,079 records and 31 variables in total in our dataset. Variables are indicative of flight date, airline, departure and destination airports, scheduled departure and arrival times, delay time, delay reasons, and more. These convey important information that can help people know how they should fly to avoid significant delays. In order to provide that guidance, our goal is to analyze what contributed to flight delays by finding the relationships between delay time and multiple factors, such as airlines, airports, scheduled time, and flight distance. We will first clean up the dataset to get useful values and select feature subset to be investigated. Once the data has been properly processed, we will apply multiple machine learning models to train and fit on the training and validation sets, in order to predict flight delays accurately.

2. Data Pre-Processing

2.1 Data Manipulation

We first looked into all provided features to decide which ones would be helpful to our regression. This dataset covers diverted and cancelled flights besides delayed flights, but those status are not needed as we only focus on delay performance in this project. Thus, we removed the DIVERTED, CANCELLED, and CANCELLATION_REASON columns. Delay types are

also not useful in making a prediction of how much a flight would delay, so columns such as AIRLINE_DELAY and WEATHER_DELAY were dropped. FLIGHT_NUMBER and TAIL_NUMBER seemed to be related information, but we realized they were factual nominal data which had too many distinct values. Considering that these information can be covered by other variables such as airline, departure and arrival airports, and time, we decided to leave out these two columns because transforming them into dummy variables would result in data that is too sparse. After dropping all unrelated features, we removed all records that had missing values from the dataset because keeping those NaN values would make our modeling inaccurate.

2.2 Exploratory Data Analysis and Data Sampling

To serve our goal of finding important factors to flight delays and predicting delay time, the ARRIVAL_DELAY column was set to be the target variable, or label, for our predictive models. To explore how our target variable was distributed, we plotted bar and box graphs to reveal ARRIVAL_DELAY's mean and standard deviation. The bar graph (Figure 1) shows the overall distribution of flight delay time, with the highest frequency at around 0 but right skewed to the positive. The mean of flight delay is 4.4 minutes, with a standard deviation of 3.9 minutes and a median of -5.0 minutes. This means most flights were approximately ontime or delayed only for a small amount of time.

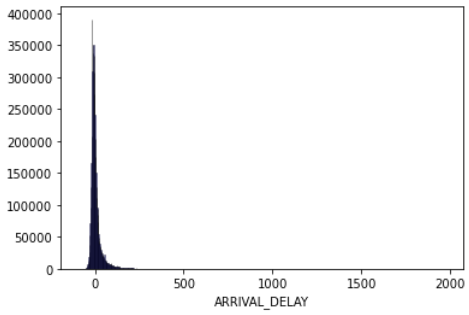

Figure 1: Bar graph of ARRIVAL_DELAY

The box plot (Figure 2) shows that the middle quantiles of ARRIVAL_DELAY centers around 0, but it has a really long tail in the positive direction, indicating that the data is skewed by outliers.

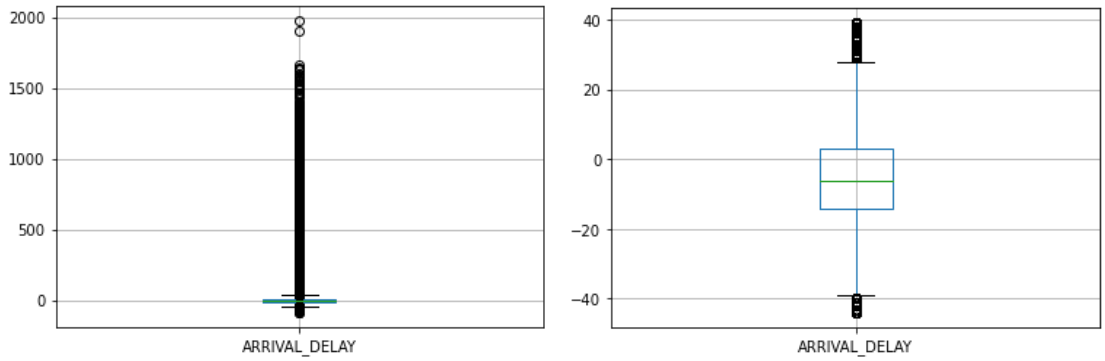

Figure 2: Original box plot of ARRIVAL_DELAY *Figure 3: After Removing Outliers*

2.3 Data Cleaning

Knowing that the target variable data is affected by outliers, we checked for it by using the IQR rule. The first and third quartiles of this data were found, and the interquartile range (IQR) was calculated as the difference between them. We then removed the data less than first quartile - 1.5*IQR and data greater than third quartile + 1.5*IQR. After removing the unreasonable values, or outliers, we re-ploted a box chart (Figure 3) of ARRIVAL_DELAY, which now is entirely centered around 0. This box plot shows that the range of ARRIVAL_DALAY is between -43 and 40 minutes, and the mean of it is about -4 minutes.

3. Feature Engineering

The dataset was split to a training set of 70% data and a test set of 30% data. We did feature transformation on ORIGIN_AIRPORT and DESTINATION_AIRPORT columns because they were nomical, non-numerical values and could not be used in modeling directly. Thinking about how airports factor into our delay analysis, we decided to represent them by their levels of delay rate. On the training set, we calculated the percentage of flight delays for each origin and

destination airport, as well as the overall percentage of flight delays. Then for each flight in both training and testing data, we added two additional predictors - ORIGIN_AIRPORT_ DELAY_LEVEL and DESTINATION_AIRPORT_DELAY_LEVEL - which equal to 1 if the airport has a chance of delay greater than the overall percentage of flight delays, and 0 otherwise. Categorical variables were converted into indicator, or dummy, variables. This particularly applied to the AIRLINE variable, which was turned into 13 new columns for each of the flight carriers. After needed feature transformation, we standardized all the data and separated them by features and the target variable that measures the duration of flight delays.

4. Dimension Reduction

Because our dataset is large and most of the features are in different scales, we chose PCA (Principal Component Analysis) to reduce the dimension of our training data, thereby avoiding the curse of dimensionality and understanding the linearly independent features in our model. By measuring how much variance each principal component carries, we found that the top 25 most important principal components explain over 99% of the original training data, as shown in the step graph in Figure 4. We decided to discard principal components that explain little information. Therefore, we reoriented our data by multiplying training and testing data's transpose with the transpose of vectors composed of the 25 selected principal components.

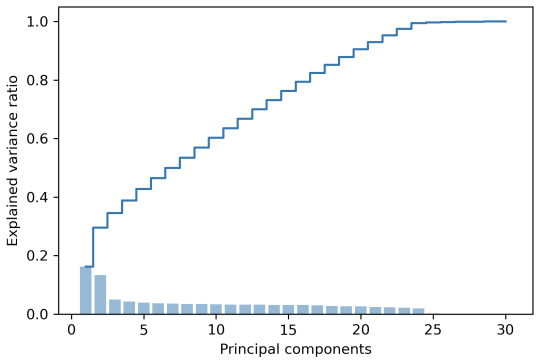

Figure 4: Percentage of Variance Explained By Principal Components

5. Model Fitting

5.1 Train/Test Split and Cross Validation

The dataset has been split into a training set and a testing set by 70% and 30% ratio respectively. There is no validation set since we used cross validation technique to tune the model parameters in the project.

K-fold cross validation has been applied to the training set to generate the CV score and guide the model tuning process. The number K refers to the number of groups that a given dataset is to be split into. Each observation in the dataset is assigned to one of the groups, and each group can be used as the hold out set 1 time and be used to train the model K-1 times. We chose 10-fold cross validation for our model to avoid excessively high bias or variance. By averaging all the 10 testing errors from each split, we obtained a final CV score that shows whether our model overfits and how it generalizes to an unseen dataset.

5.2 Linear Models

(1) Ridge Regression and Least Squares

We first fit our training data using linear models because they are fast to train and straightforward to interpret. We used generalized cross-validation to choose the tuning parameter alpha of ridge regression. We made sure the alpha values used in cross validation ranges from very small to large numbers. Least squares is simply ridge regression with alpha equals to one. We decided to use mean absolute error (MAE) instead of mean squared error (MSE) because many flight delays under 1 minute will scale our error to an excessively small number that does not accurately reflect our model performance. The MAE 6.45 minutes for both ridge regression and least

squares. And the distribution of their difference from the true arrival delay is similar (Figure 5). Because hyperparameter tuning does not improve our linear model, we speculated that ridge regression is not fit for our training data with many observations and reduced multicollinearity from the previous PCA transformation. The lowest CV (cross validation) score for both models is 6.54 minutes, which has a slightly higher test error than the training data. But we would say that these two linear models generalize reasonably well and they do not overfit.

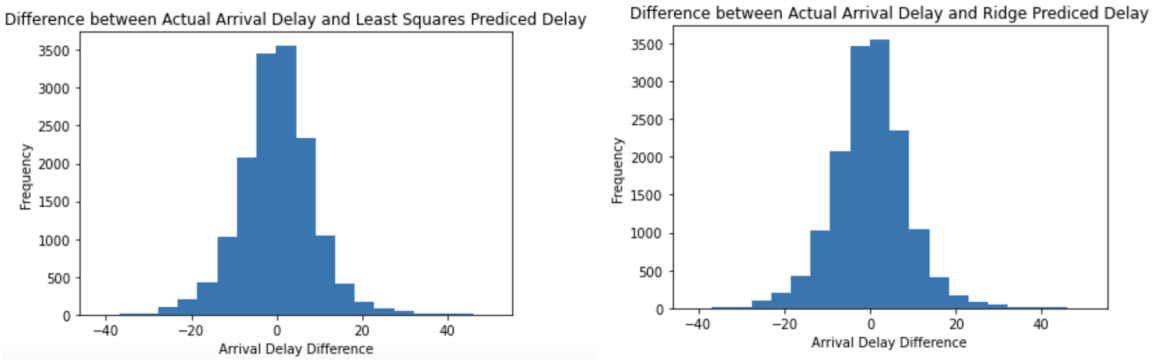

Figure 5: Linear Model Results for Least Squares (Left) and Ridge Regression (Right)

(2) Lasso Regression

We performed 10-fold cross validation to select the best alpha value for lasso, and then fit lasso regression to find out the MAE error. The difference between Ridge and Lasso is Lasso shrinks some predictors to zero, so it focuses more on the few most important predictors that might contribute to arrival delays. But the Lasso also did not improve the prediction accuracy or the cross validation score. The MAE is still 6.45 minutes and the residual distribution is similar to Figure 5.

(3) Linear Regression using Stochastic Gradient Descent

Stochastic gradient descent (SGD) is an iterative method for optimizing an objective function with suitable smoothness properties, and it replaces the actual gradient by an estimate thereof.

We experimented with SGD because it converges faster on large datasets with data points greater than 10,000. However, it even increased MAE to 6.48 minutes probably because our training iterations have not reached the minimum empirical error. All four of our linear models have similar performances of predicting flight delays with error within 7 minutes. They gave reasonably good results with fast computation, but we believe that there is more room for optimization.

5.3 Non-linear Models

(1) Random Forest Regressor

The Random Forest Model is based on the Decision Tree algorithm, which is an upside-down tree-like structure that makes decisions based on the conditions present in the data. It is very similar to the decision-making process in our human brains.

Decision trees sometimes produce over-fitting problems in application, and the random forest that we are going to introduce next solves this problem on the basis of decision trees. A random forest fits many large (deep) trees to bootstrap-resampled versions of the training data, or in another word, is a bagging of multiple decision trees. Each of the trees built over a random extraction of the observations from the dataset and a random extraction of the features. This guarantees that the trees are de-correlated and therefore less prone to over-fitting. Besides the ordinary classification tasks, the random forest model can also be able to do regression analysis.

When predicting, each tree will give a prediction value, and each prediction will be averaged to return the final result. By adjusting the specific conditions in each tree and the bagging framework, the random forest will be trained to predict the flight delay time as accurately as possible. Figure 6 is an example of a random forest regressor with 3 sub-trees.

In order to find the best hyperparameters of the model and framework, we have used the grid search method to try different combinations of the given values of parameters. Then, we used the 10-fold cross validation to determine the best instance among all the attempts. In our specific case, we have tried the maximum depth of each tree to be [6,8,10], the maximum number of features selected for each tree to be [“auto”,”sqrt”], and the number of trees in the forest to be [100,500,1000]. Among all the combinations of these parameters, we finally achieved a Mean MAE of 5.16 minutes, which is lower than using the linear models.

(2) XgBoost Regressor

Boosting is also a common and effective algorithm based on decision trees. Unlike bagging algorithms such as random forest, boosting algorithms do not assume the sub-trees are mutually independent. Instead, boosting treats each subtree as strongly correlated with each other. As a consequence, a boosting model will first train a base learner (subtree) from the initial training set, then adjust the sample according to the learning performance. Next, it is going to train the next base learner based on the adjusted sample until the number of base learners reaches the pre-specified number N, and finally performs a weighted combination among each base learner. Figure 7 is an example of the boosting regression model with 3 base learners (subtrees).

Here in our project, we have used the XgBoost in the “SK-Learn” package. We still applied the grid search method to try different parameter values. Under the learning rate (eta) parameter, we gave values [0.02,0.05,0.1]; For the maximum depth of each decision tree, we tried values [6,8,10]; And in order to find the best total number of decision trees, we used the same values as in the random forest regressor, [100,500,1000]. According to the 10-fold cross validation scores, the lowest error rate (MAE) among all the combinations is 2.73 minutes. We have found the best flight delay time prediction model to be the XgBoost, and by using it, we are confident to tell our

customers that the error of forecasts of aircraft delay time can be roughly controlled within 3 minutes.

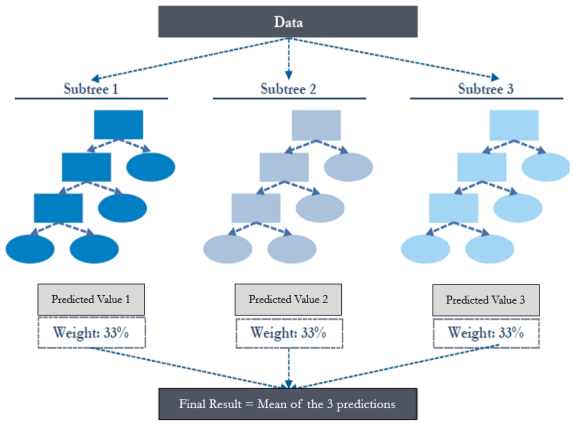

Figure 6: Random Forest Example

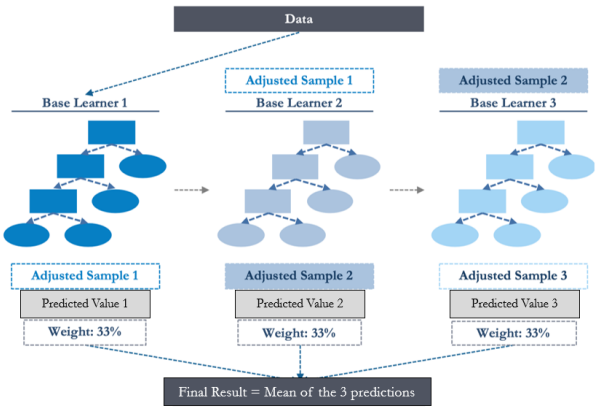

Figure 7: Boosting Example

6. Conclusions

In conclusion, our linear model was able to predict the duration of flight delays at U.S. airports with an absolute error controlled within 7 minutes, whereas the non-linear model was able to control the flight delay error under 3 minutes. Because linear models were more interpretable and computationally fast, given that a prediction deviation of 10 minutes is acceptable and passengers would want a quick estimation on their flight delays, it is possible to extend our ridge regression model to business applications that inform passengers about how early or late their flight would arrive. We could wrap our program with a mobile website or a mini-app that is easily accessible to busy travellers at airports.

7. Bibliography

"2015 Flight Delays and Cancellations." Kaggle, 9, Feb. 2017, www.kaggle.com/usdot/flight-delays?select=flights.csv.