MANIPULATE FLIGHT DELAY IN AN EFFICIENT WAY BY PREDICTIONS WITH MACHINE LEARNING

A PREPRINT

Minjun (Elena) Long
ml6vq
ml6vq@virginia.edu

Mengchen (Veronique) Wang
mw5ew
mw5ew@virginia.edu

Sara Feng
sjf8yqz
sjf8yqz@virginia.edu

April 27, 2020

ABSTRACT

As flight delay is a concerning problem nowadays, it would be beneficial for both travelers and airports if patterns and predictions of these delays could be revealed. Effective predictions would not only help travelers to better prepare for the delays, but also help airports to control and reduce chained delays which is caused by the inefficient arrangement of airport facilities. Thus, we decide to work on a dataset from U.S. Department of Transportation's Bureau of Transportation Statistics, using Regression, Classification, and Neutral Network in Machine Learning to fit models, compare models, and ultimately find a way to reasonably accurate predict future flight delays. This can benefit travelers and airports, including Virginia airports and its residents.

***K*eywords** Regression · Classification · Neutral Network · Transportation · Virginia · Machine Learning · Cross validation

1 Introduction

Nowadays, while air travel becomes a common transportation, constant flight delays seem to be a problem. Delays are especially annoying since people do not expect or well-prepare for it, which usually ends up with wasting hours in the airports. Delays usually are caused by several factors, including weather, previous flights, crew issues, and air traffic control, etc. Many of these factors could be utilized to find out a pattern of delayed fights and let airports predict the departure delay for future flights. For example, delay time could be reduced if we plan ahead and control air traffic in an efficient way. These predictions also benefit travelers, as they could prepare for it accordingly; for example, bring additional entertainment devices. We believe accurate predictions on future flights would benefit the community, including Virginia and its residents. With these information ahead, travelers could be better prepared before they come to the airport and adjust their schedule accordingly. Besides, airports in Virginia could also use these predictions to better prepare for chained delays. Virginia has several airports, including Dulles International Airport (IAD), Norfolk International Airport (ORF), Richmond International Airport (RIC), and Charlottesville Albemarle Airport (CHO), etc. With predictions, they could prepare snacks and drinks for customers and also arrange boarding gates, runways, shuttle bus, and jet bridges in a more efficient way. Lastly, with these information, airports could also reduce delay time since they could assign jet bridge, control air traffic, and even adjust flights efficiently. The result of this application would benefits not only residents but also airport staffs. In this report, we are going to discuss the experiments, including Regression, Classification, and Neutral Network, that intended to find a way for reasonably accurate prediction on flight delays.

2 Method

Flight delays, we believed, was the top concern of passengers in regarding of traveling, and the predictions of departure delays would carry the most important information. Since we saw this problem as a supervised learning which wanted to predict delays for future flights, we chose Departure Delay, a feature of our dataset, as our labels.

First of all, for any future flights, if we could give a prediction on the delay time as a side information for passengers and airports, we believed it would be beneficial. For example, if someone booked flight $UA9023$ from LAX to IAD, we could use the known information, including scheduled time and scheduled arrival, to predict how long this flight would delay in minute. This would be a Regression problem, because Regression consists a set of machine learning methods that allow us to predict a continuous outcome variable y based on the value of one or multiple predictor variables x. The goal of regression model is to build a mathematical equation that defines y as a function of the x variables. Then, this equation can be used to predict y on the basis of new variables of the predictor variables. In our project, the y would be departure delay time, and x would be measurable variables airline name, tail number, origin airport, destination airport, elapsed time, air time, distance, scheduled arrival time, year, month, day, day of week, scheduled departure time, and scheduled time. Our data set could use regression for analysis since our outcome y is a numerical variable which is delay time, and it is possible to predict a numerical number from regression method.[3] We evaluated results of Regression models with root mean square error (RMSE) and also with $5-$fold Cross Validation. More detailed information, including which specific Regression models we tried and why, is in the next section.

Predicting delays, in our opinion, could be grouped into two different categories. The first one was to predict the mean delayed time for future flights, which could be done with Regression as we stated above, and the second one was to predict level of delays. By grouping different delays times into classes, we could give passengers and airports information like "it is highly possible that this flight would have a minor delay." We thought that, delaying for 7 minutes did not have a big difference from delaying for 9 minutes, so grouping would make sense and provide good enough information as well. We made a new column of labels, used these labels as y, and tried classification models. The classification methodology fitted our propose well, since it is the task of approximating a mapping function f from input variables x to discrete output variable y. The mapping function predicts the class or category for a given observation.[6] This is a process of categorizing a given set of data into classes, and it can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The main goal for this task is to identify which class or category the new data will fall into.[9] In our new labels, we grouped delays into categories "no", "minor", "short", "medium", "long", and "severe". To evaluate the performance of Classification models, we used accuracy scores as well as classification report which contained precision, recall, and f1-score. More detailed information, including which specific Classification models we tried and why, is in the next section.

Lastly, we thought Neutral Network might be a good fit for our data set as well since Neural network is reducible to classification model, which means that a neural network can pretend to served as a classification model. Artificial neural networks are relatively crude electronic networks of neurons based on the neural structure of the brain. Its process records one at a time, and then it compares their classification of record with the known actual classification of the record to learn. Errors from the initial classification of the first record is fed back into the network, and used to modify the networks algorithm for future iterations. The neural network algorithm can be used to find one model that results in good classification of the new data since the ensemble methods could let us combine multiple weak neural network classification model. This method would reduces the variance in the strong model. [2] Since our data set was very large and included many features, we thought that Neutral Network would perform good with it and avoid over-fitting. To evaluate Neutral Network model, we used evaluate function to get the accuracy.

3 Experiments

Before we uploaded to Google Colab, we first did some basic data cleaning in Excel. We deleted some rows with clearly duplicated data or data contained special characters. We stepped back and looked at the big picture of our data. Our data was 564MB big, and it was to too large to upload to Google Colab through an Github URL link, so we decided to upload from local each time. After uploading the data, our next step was data visualization. This was the step we could get a closer look to the relationship between attributes. We created histogram to show the frequency of each attributes, and we also had heat map to check correlations and performed scatter plots for some selected attributes and analyzed their relationship to our predicted y value, which was departure delays.

Our second step was data cleaning. This was one of the most important parts since we had to make sure we got the clean data so that we could use them to get accurate results. The first problem we ran into was due to the huge data we had. To analyze which model we should use, we first use 90 percent of the whole data as training set. However, it was too large and caused Colab to crash for too many times, so we decided to use smaller percent of the data as training set, which might not be as accurate as 90 percent training set, but it could save us a huge a amount of time of re-running the code due to the crashing. We splited the data into the training set (65 percent) and test set (35 percent) since we wanted to save some time. In the data cleaning section, we first dropped the columns with little information,

including variables like "TAXI OUT", "WHEELS OFF", and also columns that provided redundant information, which would cause a multi-collinearity. We also removed some columns to match the real world situation, for example, variables like "ARRIVAL DELAY", as we would not know it in advance for flight delay prediction. Next, we checked to see if there was any missing value in the data with $.isNull()$. We detected these row numbers and dropped the entire row. We also wrote a Clean function to drop any N/A or Null data. We roughly dropped around 105,000 data entries. Since our data set was extreme large, dropping 105,000 data was not a big problem; we still left with 3,714,171 instances. After cleaning all the rows with missing value, we created pipelines to perform column transformations including scale standardization for numerical variables and OneHotEncoder for categorical variables.

Our third step was model selection. In this section, we tried regression models, classifiers and neutral network. We first looked into regression problems, which we tried to have a prediction value for future flights' delays. After data cleaning, we fitted Linear Regression, Decision Tree Regressor and Random Forest Regressor with our training label, which was the departure delay.

The first model we fit was the Linear Regression model. We checked whether there exists a relationship between our label and predictors by the graphs in the data visualization. It did not have to be a causation, but that there was some noticeable association between variables.[1] Based on the correlation results, we observed that, for our label "DEPARTURE_TIME", there were a couple of predictor variables has an association with it, so we decided to continue implementing linear regression model. Taking linear regression was the first step because it was the simplest regression, and we knew that if both a simple model and a complex model could provide the same level of performance and get the very close RMSE score, we would prefer the simple model. The first time we fitted a Linear Regression model, we got a resulting RMSE of $8.3007e^{-14}$. We doubted the correctness of it since it was almost impossible that a linear model could get that error on a real-world problem with nonlinear relationships. We looked back to our dataset and found out that we had columns "DEPARTURE_TIME" and "SCHEDULED_TIME", which caused a linear relationship between our label and the predictors. Considering that in real world, we had no idea of the actual departure time, we decided to drop this column to meet the real world condition.

Our second model was Decision Tree Regressor. Decision tree is in the form of a tree structure, which breaks down a dataset into smaller and smaller subsets while an associated decision tree is incrementally developed at the same time. The final result is a tree with decision nodes and leaf nodes. We chose to Decision Tree Regressor because it could give more weight for the better predictors by placing them near the root, and we believed it could give a better result than Linear Regressor.

Due to the weakness in Decision Tree, including lack of stability and computationally expensive to train, we considered Random Forest regression as our third model. Random forest is a bagging technique with trees run in parallel. It operates by constructing a multitude of decision trees at training time and outputting the mean prediction of the individual trees. We chose this model because it does not rely too heavily on any individual feature, and makes fair use of all potentially predictive features. Lastly, it usually provided higher accuracy, and had the power to handle a large data set with higher dimensionality. We thought it should be a good for our dataset and used n_estimators = 5 in the model selection part. [4]

For regression models, we tried to use Cross Validation for more reasonable RMSE results. We have tried $5-$fold Cross Validation and Leave-one-out Cross Validation, which a special case of cross-validation where the number of folds equals the number of instances in the data set. However, as we did some research about Leave-one-out Cross Validation and tried running the code, we found out it was infeasible because our data set was too large. As a result, we decided only to use $5-$fold Cross Validation in our project.

Then, we looked into classifiers. As we talked before, we could group different delays times into classes and predict which class it would belong to. In our project, we try three different classifications: Random Forest Classifier, K Neighbors Classifier, and Gaussian Classifier.

For the classification, we first made a new label called 'DELAY_LABELS.' We reassigned each observation to different types in 'DELAY_LABELS.' Since the 'DEPARTURE_DELAY' had unit "minutes", we assigned the observations in to label "NO" if the 'DEPARTURE_DELAY' was less than or equal to 0. Setting the labels to "Minor" if the 'DEPARTURE_DELAY' was between 0 to 10; to "Short" if the 'DEPARTURE_DELAY' was between 10 to 30; to "Medium" if the 'DEPARTURE_DELAY' was between 30 to 60; to "Long" if the 'DEPARTURE_DELAY' was between 60 to 120. If the 'DEPARTURE_DELAY' was over 120, we then considered the label be "Severe." Besides, we re-did some of the data pre-processing. As we looked into our categorical predictors, including "ORIGIN_AIRPORT" and "DESTINATION_AIRPORT", there were many coding data. For example, there were not only airport named as

"LAX" and "IAD", there were also airport coded as "000103". This meant there must be another file or document that matched these coding number to the actual airport names. However, after a long time search online, we were not able to find it. Thus, most categorical columns actually were not as useful in the prediction as it should be. Thus, we decided to drop these columns and continued with other predictors.

We first tried Random Forest Classifier, which is an ensemble tree-based learning algorithm. It is a set of decision trees from subset, which is randomly selected from training set, and aggregates the votes from different decision trees to decide the final class of the test object. Random forest classifier also has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing. We chose to use this classifier with our data set since it is one of the most accurate learning algorithms and have produced a high accurate classification in many data sets. Secondly, since our data set is large, it would be reasonable to use random forest classifier since it runs efficiently on large databases. We used all the default hyper-parameters in the model selection part. [5]

Then, we tried KNeighbors Classifier with $n\_neighbors = 5$, which principle is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. We chose it because KNeighbors Classifier has been successful in classification situations where the decision boundary is very irregular. We believed was a reasonable suit for our flight delays dataset since delays class usually did not have a regular boundary.[7]

The last classifier we chose was Gaussian Naive Bayes, which belongs to the family of simple "probabilistic classifiers" based on applying Bayes' Theorem with naive independence assumptions between features. We chose Naive Bayes because first of all, it was extremely fast compared to more sophisticated methods. Also, it performed well in the multiple class classification. Since we have 5 classes, we believed it is worthy to try Gaussian Naive Bayes. [7]

For all three classifiers, we used accuracy_score to get the test accuracy. However, we also looked at classification_report to check the precision, recall, and f1-score for each classifier in order to make a more accurate interpretation of the results.

Lastly, we considered Artificial Neural Network as one of our model building tools. In our built model, we use layers with "relu" and "linear" activation, and performed Batch Normalization to find coefficients that were better and fitted the data. We used Mean Absolute Error as the loss, 'adam' as optimizer, and ten epochs when fitting.

After trying all these models, we first compared within categories, i.e. found out the best Regressor and the best classifier. By comparing the RMSE, we chose Random Forest Regression as our best Regressor. By comparing the test accuracy, precision and recall score, we chose Gaussian Naive Bayes as the best classifier. Then, we compared among Random Forest Regression model, Gaussian Naive Bayes model, and ANN model, and finally decided that Random Forest Regression was the best model. Detailed explanation on the decision would be talked in the conclusion model. We then tuned the hyper-parameters and re-trained the model to find out the final model RMSE of the test set. However, a problem occurred as there were four levels in the 'TAIL_NUMBER', which is a categorical columns, were not included in the training set. We realized that our test set was too large that could include information that was not included in the training set. This would possibly cause a less accurate result than we expected in the test set. We decided to drop the data with these four 'TAIL_NUMBER' in the test set. We used GridSearchCV and $cv = 5$ for hyper-parameter tuning. Since the training set is too large, our program crashed too many time, so we used a slice of it for hyper-parameter tuning.

4 Results

With 65 percent of the data (train set) set, linear regression model had a RMSE for about 36.2481866. For the Decision Tree Regressor, it had a RMSE for about 0.094527. Random Forest Regressor with number of estimators equaled to 5 had a RMSE of about 18.585021. However, as we run the RMSE with $5-$fold Cross Validation, Decision Tree Regressor had the mean of RMSE for about 52.555319 with a Standard deviation of 0.174935. Random Forest Regressor got a result that the mean of RMSE was about 40.097131 with Standard deviation of 0.116543. Thus, according to the results in $5-$fold Cross Validation, Random Forest had the lowest validation RMSE.

For the Classifiers and Neutral Network, Random Forest Classifier had a test accuracy of about 0.45997; KNeignbors Classifier had a test accuracy of about 0.567764; Gaussian Naive Bayes had a test accuracy of about 0.5954164. Besides, we also looked at their classification report. The precision and recall were high only for "NO" class for about 0.7, which was the class for no delay, for all three classifiers. Other classes, which we actually paid more attention to that were related to delays, all had precision and recall below 0.3. Our Neutral Network model gave

us a test accuracy of 0.5984085.

Our final model was Random Forest Regressor, which gave a final RMSE of about 37.0112413.

5 Conclusion

First we compared all three Regression models. Linear Regressor had the largest RMSE as it was too simple to handle this kind of real-world problem. Due to the existing limitation for this simple model, we would not consider Linear Regressor as a good model. The results of Decision Tree Regressor and Random Forest Regressor were close. Thus, we looked at both the pros and cons for these two models. For Decision Tree Regressor, it was simpler and did not rely too much on data pre-processing. However, it was less stable. A small change in the data could cause a large change in the structure of the decision tree. Also, the calculation in decision tree might sometimes go far more complex beyond what it should be.[8] Random Forest Regressor was more complex, but except for that, it held most advantages of Decision Tree. Random Forest all had a higher accuracy (smaller RMSE). Thus, we believed Random Forest Regressor performed the best among Regression models.

For the classifiers, Gaussian Naive Bayes had the highest test accuracy. However, as we looked at the classification report, none of these three models performed good enough regarding the precision and recall score for any actual delayed class. In our opinion, what most passengers and airports care was the flight delays. Even though there could be flights departure earlier than scheduled time, we are usually concerned about how long we need to wait for delayed flights. Thus, due to low precision and recall score for delayed labels, we concluded that classifiers were not good fits for our data set. For the Neutral Network model, it only had a test accuracy of less than 60 percent. Since our data set was too large and contained too many information, this self-built model was too shadow to handle it.

Due to the inaccurate results in delays flights portion of Classifier and ANN, we chose Regression over Classifier and Neutral Network. Based on the observation and analysis we have so far, we concluded that there did exists a relationship between flights delay time and other measurable variables, such as airline name, tail number, origin airport, destination airport, elapsed time, air time, distance, scheduled arrival time, year, month, day, day of week, scheduled departure time, and scheduled time, which meant that flight delay time was predictable using these predictors. We found out Random Forest Regressor performed the best with a RMSE of 37.0112413. As RMSE can be interpreted as the standard deviation of the unexplained variance, 37.0112413 seems to be a large number. However, taking into the consideration variation in flight delays, we thought it was a reasonable result. As even for the same flight on different days, the departure delays could vary by a lot, it would be extreme hard to have a very small RMSE with only information we could know in advance.

Using our best trained model Random Forest Regressor, we could predict departure delay time and use this information to help the well being of Virginia and its residents. First, we could corporate with local airport station such as Dulles International Airport(IAD) and Charlottesville Albemarle Airport (CHO). By sharing our prediction model, they could arrange gates, departure lounge, and ground service more efficiently. In addition, they could add more airlines which are more arrived on time, or they could save the between time to arrange more flights. In this way, both local airport station and residents in Virginia could get benefit from it since airport makes profits with more flights, and residents have more opinion to choose the flight which fits their need.

Secondly, we could corporate with Virginia Department of Transportation. By sharing our prediction model, they could arrange dedicated airport lane or express bus. For example, if we find out that the flights which scheduled from 5pm to 9pm is more on time than the flights in the other time of the day, we could arrange one dedicated lane which near airport station or from a highway only used for residents who are on the way to airport station during 5pm to 9pm. During the time outside this range, this lane is same as other lane, and it is open to all. In this way, Department of transportation could reduce the traffic since it sometimes happens once people with all different direction are together. In residents' perspective, this dedicated lane might reduce the amount of time they spends to airports. Express bus could use the predicted time to scheduled the stop station schedule.

Thirdly, we could share our prediction model with the well being of Virginia and its residents. In this way, we could provides a well-rounded suggestions when residents are choosing their flights. For example, if they are in the situation of time-important such as attending to a next-day conference, they could use our model to find the estimated delay time for their several flights opinions, therefore they could calculate and compare the total travel time to find their best choice. In addition, this model is helpful especially for people who are having transfer flights. Residents could use our model to predict the possible delay time, so that they could save enough time to take the next flight. Our model

helps to reduce the cases that passengers are not able to catch the connected flights due to the previous flights' delays

Fourthly, we could corporate with airlines. By sharing our prediction model, they could use our delay time model to analyze and find the pattern about their operated flights. Therefore, they could use these information to figure out the exists or potential problems to reduce the number of similar delayed cases. This could benefit all people who taking these airlines, including Virginia residents.

However, there are still ways to improve our model. First of all, since we had a lot of missing data in the reason of delays columns, we decided to drop them from our dataset. However, we believe delay reasons also matter in the prediction of days, and hence, if we could collect more information for each instance, we might be able to give a more accurate prediction. Also, there are several columns that required matched documents or files. For example, the file that contains the airport coding and airport name. With this information, we could better use these categorical predictors and get a more accurate results. If we would have these documents, we might be able to increase the accuracy of our predictions. Secondly, our data set include airports all around the U.S., we might able to find a more accurate model with just Virginia data. Thus, for later improvement, we could separate the data set by states, fit models for each states, and check if that would give a better prediction on delays.

Besides, Google Colab crashed when we used more than 70 percent of the dataset to train models, and this was possibly the reason why our test set RMSE was generally higher than the training set. If we could run our model on a different platform that allows us to use 90 percent of the data as training set, we might be able to improve our model as well.

Additionally, we think better data cleaning and data pre-processing would help me improve the performance of the models. We did some research and found out most project due with flight delays usually had a huge part on data pre-processing. Even though we also put a lot of time and efforts on this process, it is always possible to do more with data cleaning. However, one of the requirements for more precise data pre-processing is to obtain the data matching documents to have the categorical columns information. We believed Neutral Network should give accurate predictions on this kind of large and complex data set. However, both classifiers and neutral network did not give accurate results. We think our inefficient data pre-processing might caused this problem. In our opinion, if the data could be pre-process better, the neutral network model should perform much better than the result we got this time.

Lastly, we also did some research on transfer learning for predictions on flights delays. Unfortunately, we did not find out something similar to what we want. However, if there is any pre-trained model that could be adopted as a base model in our project, we think it might be able to give a more accurate result as well.

References

[1] Linear regression. Yale Universiy.

[2] Neural network classification. Frontline Solvers.

[3] Regression analysis essentials for machine learning. Statistical tools for high-throughput data analysis.

[4] Afroz Chakure. Random forest and its implementation. Towards Data Science, 2020.

[5] Afroz Chakure. Random forest classification and its implementation. Towards Data Science, 2020.

[6] Kirill Fuchs. Machine learning: Classification models. Medium, 2017.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Saed Sayad. Decision tree regression. saedsayad.

[9] Mohammad Wasseem. How to implement classification in machine learning. edureka, 2019.