DS-GA-1001
Fall 2017 Introduction to Data Science
Project Report

ANALYSIS OF ON-TIME PERFORMANCE IN NEW YORK AIRPORTS

Group Name: Teletubbies

Yu Xiong(yx1201), Jin Han(jh5695), Sijia Liu(sl6496), Xin Guan(xg702)

December 8, 2017

Abstract

Due to the increase in air traffic, flight takeoff and landing schedules have been a crucial task in the recent years. Flight delay in the United States has resulted in the huge cost for airlines, airports, and passengers. Traditionally, flight schedule was a problem in the operation research field where scholars employ advanced analytical methods to help key-players make better choices. However, these approaches were unable to capture the dynamic nature and uncertainty of aircraft scheduling. In this paper, we propose an empirical model using machine learning methods. Machine learning is the research that explores algorithms which learn from data and provide prediction base on it. Specifically, we examined methods including Logistic Regression, Decision Tree, Random Forest, Naïve Bayes, and XGBoost. We tested our model against an authoritative dataset with our evaluation metrics. We were able to achieve a significant lift against the baseline model.

1.Introduction

The demand for air transportation has been drastically increasing for its short travel time. The increasing number of aircraft takeoffs and landing within a given time period has resulted in an overload issue in many airports. Air traffic delays have become more and more common. Flight delays have caused huge losses for all players of commercial aviation. Due to the uncertainty nature of flight delays, passengers have to plan hours ahead in order to attend their appointments at the destination on time. The occurrence of flight delay has increased the risk and trip cost for passengers. Airlines also suffer from additional cost resulted by their aircraft's delay. For example, airlines have to bear with the possible penalties and fines from the airport regulations. They also have to compensate their customers and crews for the extra travel time. Flight delays also jeopardize airlines' customer relations efforts because customers' loyalty always comes from the airlines' reliability.

1.1 The business problem and its motivation

A survey done by United States Joint Economic Committee shows that the annual cost of domestic flight delays to the US economy was estimated to be $31–40 billion in 2007. Such cost has never stopped increasing over the past ten years, motivating the development of better aircraft scheduling mechanism and a more precise delay prediction method.

The result of the flight delay data mining problem should be a platform which allows airport administrators to enter information on a specific flight and get an estimation of a flight delay status. A mature product can help people who have no programming background to get an idea of whether a specific flight will delay or not. Such product will benefits both airports in scheduling the flight's takeoffs and landing and the passengers in planning their travel time. Airlines can also get values from such product because they can improve the flight on-time performance before flight delays happen and incur the loss to the company.

1.2 Previous Works

The flight delay problem has been approached using many different modeling methods depending on the different research purpose. One common approach is statistical analysis. It employs regression models, correlation analysis, econometric models to evaluate the efficiency of flight systems and to detect correlations to understand principal causes of the flight delay. [1] Another approach is using probability models which employ analysis tools that estimate the probability of an event based on historical data.[2] developed a probabilistic model using genetic algorithms to estimate departure delay distribution in Denver International Airport. The recent approaches use machine learning methods. Khanmohammadi et al. predicted rood delay using fuzzy inference system.[3] Balakrishna et al. built taxi-out delays prediction model using reinforcement learning algorithm and tested their model on dataset at JFK Airport and Tampa Bay Airport.[4]

1.3 Problem Formulation

In this paper, we would like to explore the delay prediction problem using a machine learning approach. Our data mining problem is a supervised classification problem. We take the dataset from Bureau of Transportation Statistics and produce a model which predict the flight on-time performance at NYC airports.
More specifically, given a training dataset, we would like to derive a model that could maximize the test accuracy on the test dataset. Details on our training dataset, which contains the features X, input information related to the flight and the target variable Y, flight status, are explained in Section 2.1.

2. Data Analytics

The analysis in this paper is based on data from the Airline On-Time Performance Data from the database of Bureau of Transportation Statistics (BTS)[5]. The raw data consists of all historical records of flight information from January 1995 through September 2017. The database itself contains on-time arrival data for non-stop domestic flights by major air carriers and provides additional information such as departure and arrival delays, origin and destination airports, flight numbers, scheduled and actual departure and arrival times, canceled or diverted flights, taxi-out and taxi-in times, air time, distance and etc.

2.1 Data Collection

2.1.1 Descriptive Statistics

We decide to investigate the flight on-time performance by formulating models with flight delay parameters of flights that arrived at or departure from New York in 2013 to 2016 interval. Our dataset contains 2036850 rows and 28 columns. The target variable, which is the departure delay, counts for the difference between the scheduled departure time and the actual departure time from the original airport gate in minutes. Each of the flight information is described using 28 columns after selection. We categorized all 28 columns in into 9 buckets:

- Time Indicator
- Flight number
- Delayed Information
- Carrier
- Arrival time and Departure Time
- Air time
- Distance
- Origin and Destination Airports
- Delay causes: Carrier delay, Weather delay, National Air System Delay, Security Delay, Late Aircraft Delay

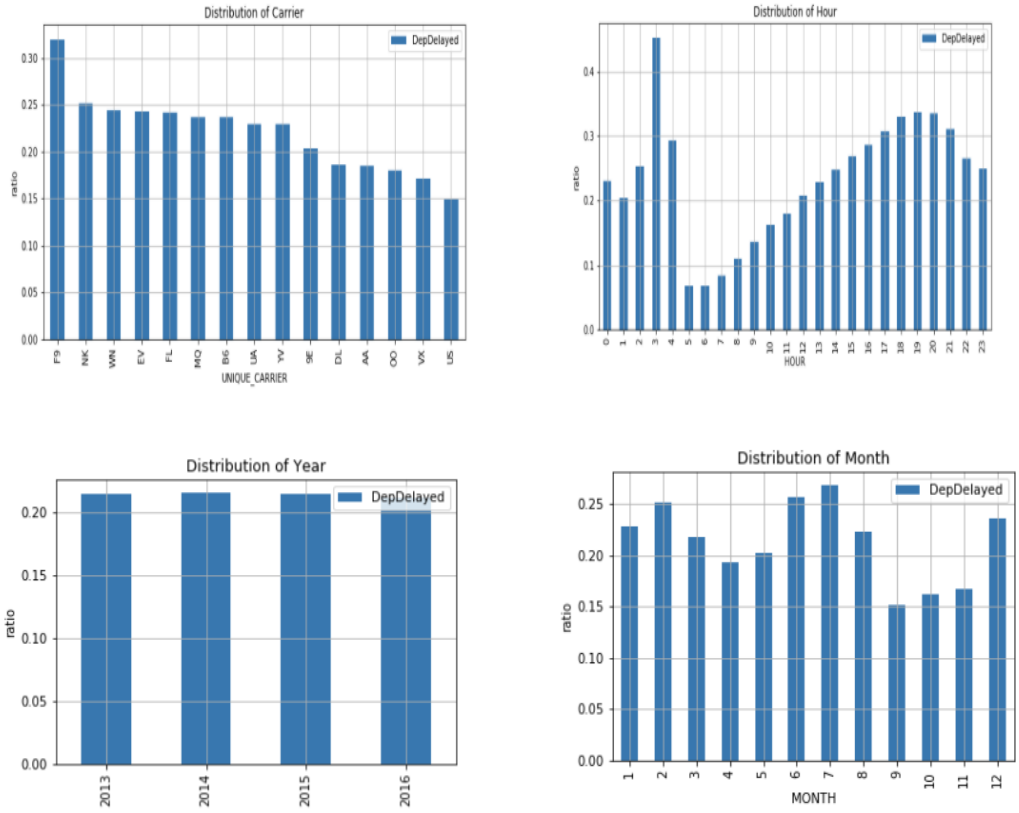


Figure 1: Frequency distribution of month, hour, year and airline company.

We looked at some descriptive statistics and visualization of each variable in our data. In Figure.1, we plotted four frequency distribution for four variables: *month, hour, year and airline company*. The total number of delayed flights does not change much over the three year period we selected, indicating that the delay is not getting improved over years. From the 'month' plot, we see that more delay happens during summer and winter. A possible explanation could be that some severe weather conditions such as storm and snow that cause flight delay happen more in summer and winter. The top 5 airlines that have the highest numbers of flight delay are Frontier Airlines(F9), Spirit Airlines(NK), Southwest Airlines(WN), ExpressJet Airlines(EV) and AirTran Airways(FL). An interesting pattern we observe is that delay number increases from morning to past-midnight in an everyday cycle. This situation happens because there are reduced flight crews and staffs during the midnight. Therefore it suggests that if one wants to avoid delay, he probably should get up early and get the early 4 am flight. Moreover, we plotted out the covariance graph in Figure 2 to investigate the degrees of correlation amongst themselves so that we can figure out what algorithm best describe our dataset.

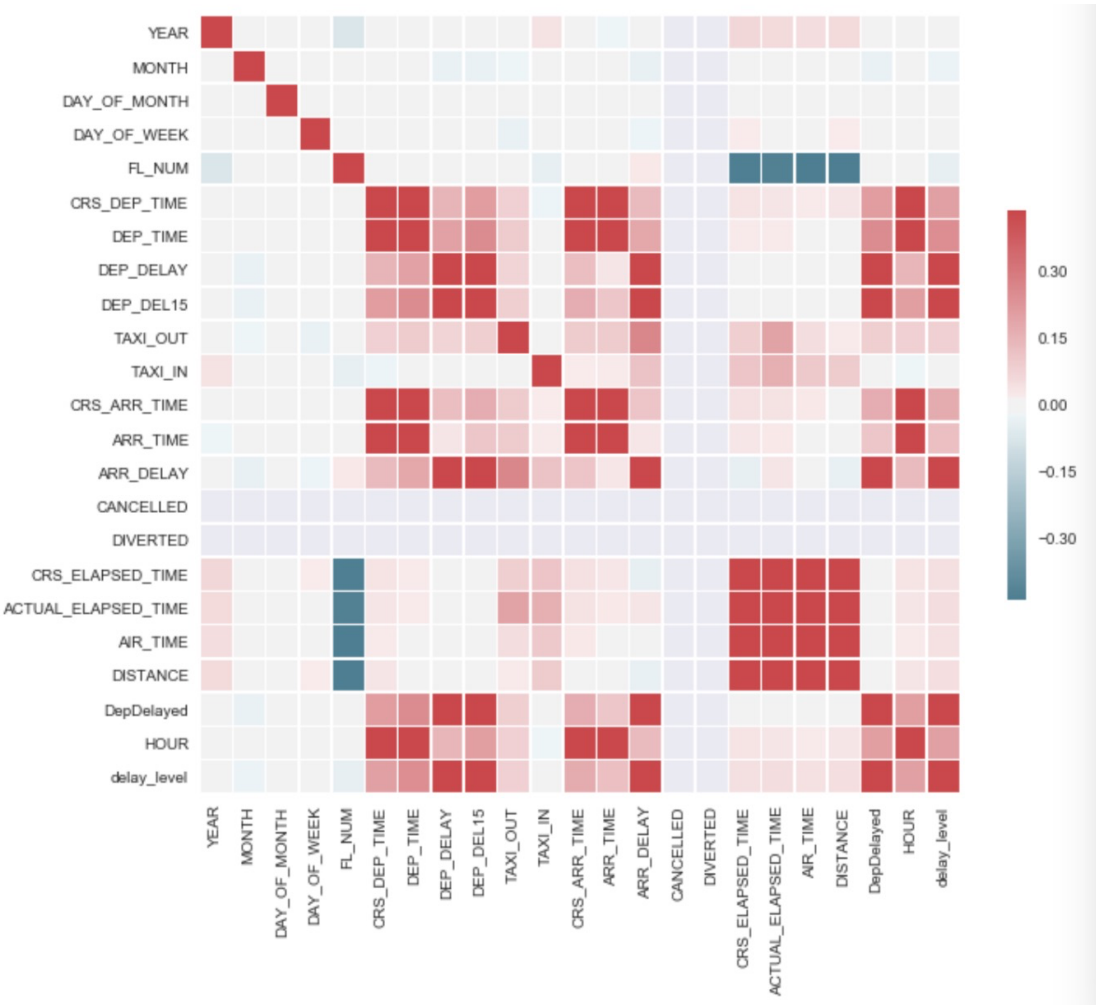


Figure 2: Correlation map of the features

2.2 Data Cleaning and Feature Engineering

After a rough data exploratory analysis, we found the following problems, which were also addressed in this section.

1. Among 28 columns, 12 of them contain null values. 6 of them, including Carrier delay, Weather delay, National Air System Delay, Security Delay, Late Aircraft Delay and cancellation code, have more than 80% null.

Column Selection: We dropped out the columns with more than 80% nulls. Because the large absence of the data points does not contribute to be imputed as filling in a fixed value or computing the column-wise mean.

Missing Value Handling: Except for the 6 features I mentioned above that has significant missing values, all remainings have missing value less than 0.5%. Therefore, we directly

dropped out any rows contain null values in their columns. Also, due to the large volume of the dataset, dropping missing value would be more efficient than computing the mean value without losing the generosity.

2. Some of the column variables provide duplicated information. For example, both flight number and tail number are labels of a particular flight. Moreover, some of the column variables are unnecessary for our on-time performance analysis. For example, CRS Departure Time and Actual Departure Time are unnecessary since we can directly investigate the variable Delay Time, which is the difference between them.

Feature Dropping: We would like to improve our dataset by dropping some duplicated or irrelevant column variables. 1) Since we are not analyzing data by year, we dropped the variable Year. 2) We are not investigating the flight on-time performance by the particular company. Therefore, the flight number and airline code are irrelevant. 3) Delay time is the target variable we are trying to predict and it is the difference between the scheduled time and actual time.Thus, any other indicator representing the timetable or delay timing of the flight is unnecessary.
Feature Creating: We also create a new feature called *Hour* for statistical examine of the relationship between the on-time performance and hours. We believe that hour, which is not included in the raw data, should be an important and meaningful feature in our analysis and modeling.

3. Some of the columns are categorical such as Origin and Destination airports and etc which is hardly to be incorporated into our models without transformation.

Categorical Columns Conversion: We transformed all categorical features, which are Unique Carrier Types, Original Airports, Destination Airports, to dummy variable with yes=1 and no=0.

4. Our target variable itself, delay time, is a numeric value that accurate to minutes. It is obviously very hard to predict the on-time performance in minutes. Therefore, it is better to perform a classification analysis on on-time performance instead.

Redefine the Target Variable: In order to make the outcome more convenient and explicit for passengers, companies, and airports, we decided to classify the departure delay into 3 basic levels: Early Departure, On Time and Delayed. Since the delay time is specified in minutes, we think that it is okay if the flight departures no later than fifteen minutes of the scheduled departure time. We classified our target variable as followed:

i. The flights that arrive the gate more than fifteen minutes after the scheduled time are considered as “delayed”;
ii. The flights that arrive at the gate more than five minutes before the scheduled time are considered as “early departure”
iii. The flights that departures less than five minutes early or less than fifteen minutes late are considered as “on time”.

Figure 3 shows the distribution of the 3 levels of on-time performance in our choosing dataset. As we can see, there are approximately 28.88% of flights arrived in advance, about 49.75% of flights arrived on time and around 21.36% of flights delayed, which suggests that no class dominates our class set.

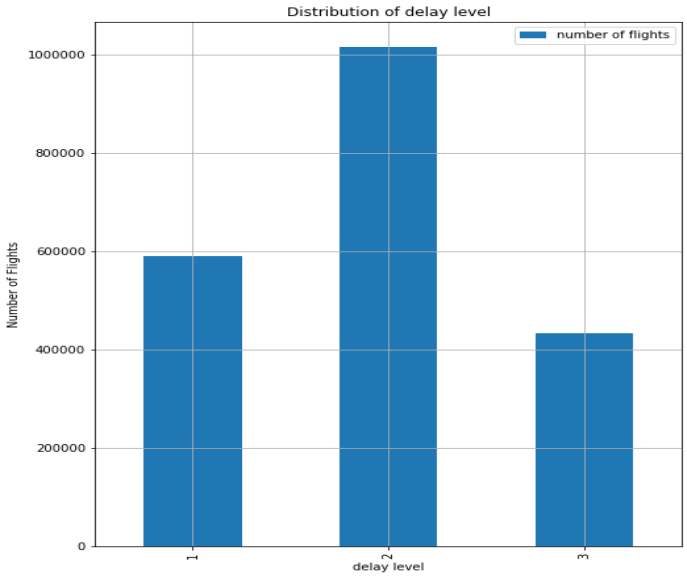


Figure 3. Distribution of delay level

After the data cleaning and feature engineering, we have our final dataset with 2036850 rows and 260 columns.

2.3 Feature Selection

Selection of good features is an important phase in pattern recognition. We applied the feature selection from Decision Tree to identify the best features from our feature-engineered dataset for the purpose of classification. More specifically, we fitted Tree Classifier by using entropy with our feature-engineered dataset. After getting the feature importance, we only kept the features whose weights were more than 10^{-4}. And our data after feature selection has 2036850 rows and 127 columns.

Figure 4 demonstrates that the feature importance decays at almost 0 after the first 80 features. The second graph shows the top 20 most important features of the dataset. They are air times, hours, month, the day of weeks, flight distance, and some dummies from unique carrier type and original airports.

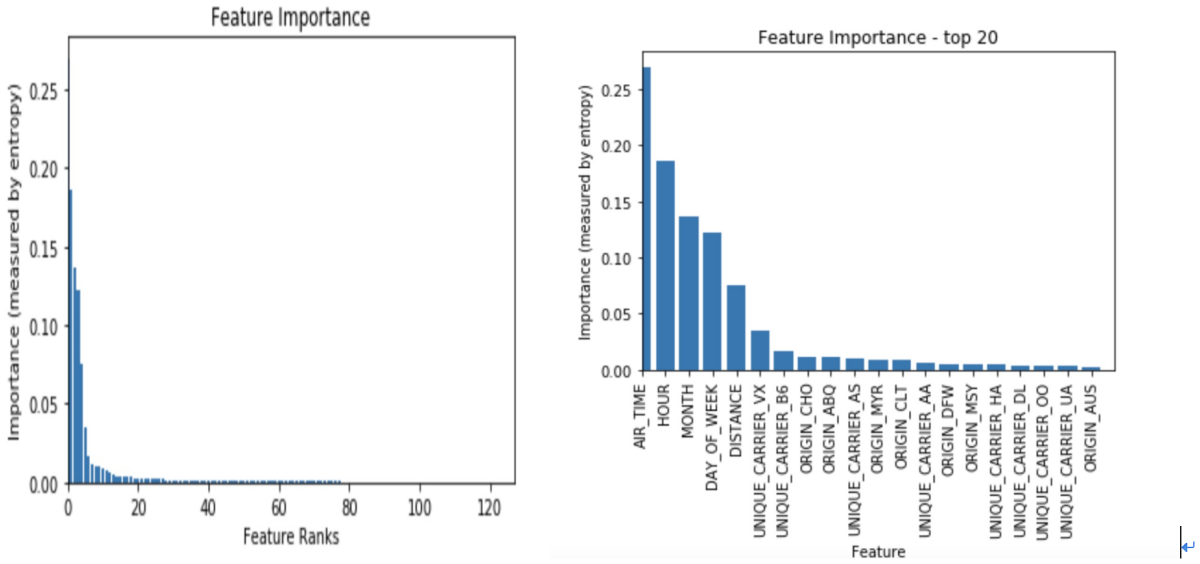


Figure 4: Left Plot: The distribution of feature importance in the Extremely Random Tree measured by entropy. Right Plot: The top 20 features with largest feature importance.

3 Model Selection

3.1 Baseline Models

Random Guess: The simplest way to approach this classification problem is by guessing at random. In our problem formulation, we classified our target variable, *departure delay*, up to 3 basic levels: Early Departure, On Time and Delayed. Assuming that each flight has equal probability of arriving early, arriving on-time and delayed, we get an expected accuracy of 33.33%.

Logistic Regression : We used the data before feature engineering in Section 2.2 as our training data and developed a Logistic Regression model Training data was randomly selected from the complete raw dataset. The rest of the raw dataset is used as testing data. The ratio between train data and test data is 8:2. For our target variable, *departure delay,* we get the accuracy of 35.43%.

3.2 K Nearest Neighbors

K Nearest Neighbors(KNN) is a simple machine learning algorithm that categorizes an input by the majority votes of its nearest neighbors. The model is non-parametric, simple to implement and robust. It naturally deals with multi-class cases and can do well with enough representative data. These features of KNN make it a good model for our dataset. KNN has two important parameters, n neighbors, and Euclidean Distance. In our model, we use n_neighbors = 5 and

Euclidean Distance=2 and get a cross-validation accuracy of 48.60% and a test accuracy of 51.54%.

On the downside, KNN is computationally expensive. Its classification depends on distance calculation making it very sensitive to outliers and irrelevant attributes. To obtain a good prediction, KNN needs a well-selected distance function.

3.3 Naive Bayes

We also implement the Naive Bayes method, which is a supervised-learning algorithm based on the Bayes' theorem with the "naive" assumption of conditional independence. The assumption of conditional independence makes it less robust than KNN, but it is computationally fast and it works well with high dimensions. It is also less influenced by outliers and irrelevant attributes. Despite the fact that it can work very well in some real-world settings, famously document classification and spam filtering, its conditional independence assumption does not hold for most real-world problems.

The two most classic Naive Bayes variants are Multinomial Naive Bayes model and Bernoulli Naive Bayes model.[7][8]MultinomialNB implements the naive Bayes algorithm for multi normally distributed data while Bernoulli Naive Bayes is for multivariate Bernoulli distributed data. We used the BernoulliNB classifier because our target variable only has three levels and does not need smoothing. We get a cross-validation accuracy of 51.12% and a test accuracy of 51.06%.

3.3 Logistics Regression (L1 Norm)

Logistic Regression(LR) is the most convenient implementation among all tools. It is very simple to implement and takes little to train. One major drawback of LR is the conditional independence assumption for all its features. The LR model relies heavily on transformation for nonlinear variable and we hope that the feature engineering in Section 2.2 can effectively help us to capture the nonlinearity.

We trained an LR model using L1 regularization. For the penalty factor C for the L1 regularization, we check the cross-validation accuracy for C = 0.001 , 0.01 , 0.1 , 1 , 10. We get the peak performance at C = 0.01. The LR cross-validation accuracy and LR test accuracy we get are 52.33% and 52.37%.

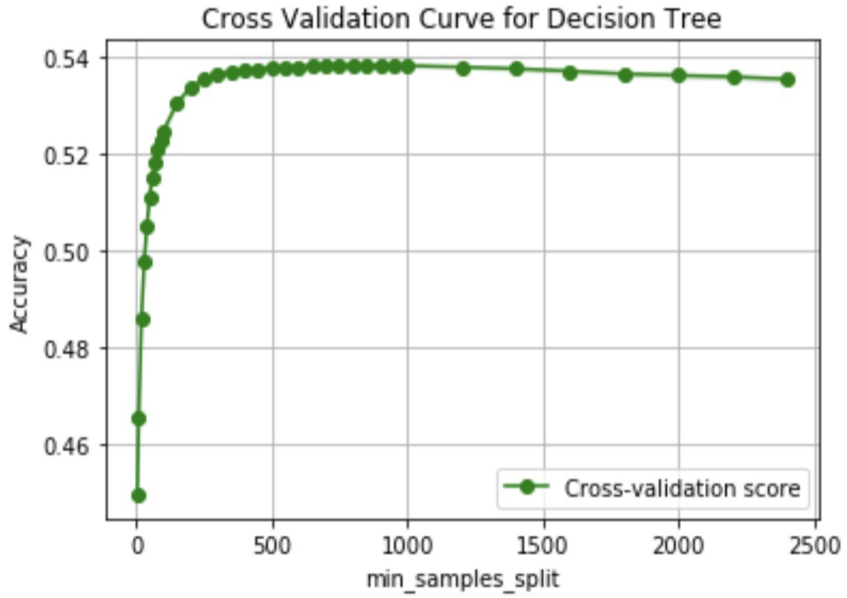
3.4 Tree approach

3.4.1 Decision Tree

A decision tree is a non-parametric supervised learning method. It is simple to implement, easy to understand and requires little data preparation. The deeper the tree, the fitter the model and the

more complex the decision rules[7]. A decision tree can fit all most all the test dataset within its own region which can lead to the overfitting problem. Such situation happens when decision tree model develops hypothesis to reduce training set error at the cost of testing set error. The bias is low, but the variance tends to be high.

As shown in figure 5, we obtained the highest cross-validation accuracy with min_sample split = 200. We use n_estimators = 30 and the min_sample split = 200 as our best Decision tree model



and achieved a test accuracy of 44.57% and cross-validation accuracy of 53.83%.

Figure 5: The cross-validation curve of Decision Tree

3.4.2 Random Forest

To correct the overfitting problem in the Decision tree model, we implemented the Random forest model. Random forest help to mitigate overfitting by fitting a number of decision tree classifiers on sub-samples of the dataset. Averaging the results from each decision tree to help to improve the predictive accuracy and to control over-fitting.

We turned our random forest model in a similar way as we tuned the Decision tree model. As shown in figure 6, for our random forest classifier, we used n_estimators = 30 and the min_sample split = 70 and the accuracy we obtained from this model is 54.52% and the cross-validation accuracy is 54.74%.

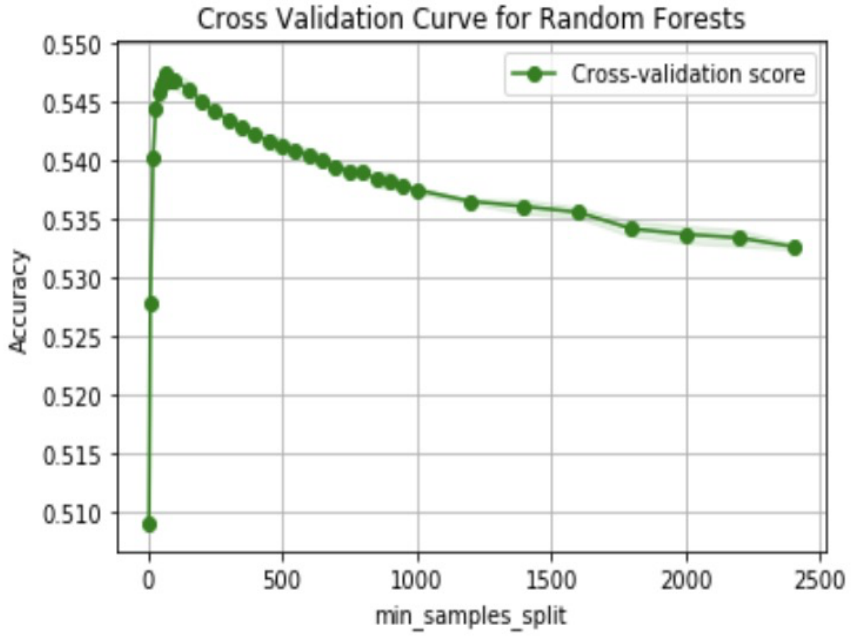


Figure 6: The cross-validation curve of Random Forests

3.5 XGBoost

XGboost[8], which is also known as Extreme Gradient Boosting, is an ensemble tree method based on gradient boosting tree. It could solve the problem of overfitting efficiently by considering regularization term. Besides, it has a more efficient and accurate performance by using ensemble trees model than other decision trees model. We applied XGboost with the max_depth=20, num_boost_round=10, and nfold=5 and we can get the test accuracy of 54.74% and the cross-validation accuracy of 58.69%.

4 Evaluation

We have a relatively large dataset and balanced class distribution. In this section, we designed our evaluation metrics by comparing the test accuracy of each machine learning models. We evaluated out best model giving the highest test accuracy of 54.74% and having a 1.54 lift against our baseline Logistic model. We also probe into the error analysis of our model with the best performance.

4.1 Performance Analysis

The performance of our models is shown in Table 1. By comparing the test accuracy and cross-validation accuracy of various models we applied, the best model is XGBoost with the test accuracy of 54.74% and the cross-validation accuracy of 58.69% . It has a 1.54 lift against our

baseline Logistic model. Besides, we could conclude from the table that for the case we discussed, the ensemble tree methods outperform other linear models and the models based on probability.

Model	**CV Acc**	**Test Acc**
Random Guess (Baseline)	n/a	33.34%
Logistics Regression(Baseline)	37.67%	35.43%
K Nearest Neighbors	51.54%	48.71%
Naive Bayes (MultinomialNB model)	45.80%	45.83%
Naive Bayes (BernoulliNB model)	51.12%	51.06%
Logistics Regression(L1 Norm)	52.37%	52.33%
Decision Tree	53.83%	44.57%
Random Forest	54.74%	54.52%
XGBoost	58.69%	54.74%

Table 1:Performance results of trained models. CV Acc is the cross-validation accuracy of the tuned model. Test Acc is the accuracy of the test set.

4.2 Error Analysis

Even Though the accuracy is relatively high compared with the baseline, there are still some reasonable errors, resulting from data mining process and the constraints of models. By error analysis, we could explore the non-trivial irreducible errors and improve the process of our prediction. Here we analyzed the prediction of XGBoost model with the best performance among all models.

We first analyzed the comparing the test set and the predicted label by visualizing the label distribution. As shown in Figure 7, the distributions are likely to each other but there are still some differences. Even though the number of label 1 in both distributions are the largest, the number of label 2 in the test set is a little bit larger than that in prediction. It shows that there is some misclassification during the prediction process. Besides, the number of label 1 and label 3 in prediction are both larger than that in the test set.

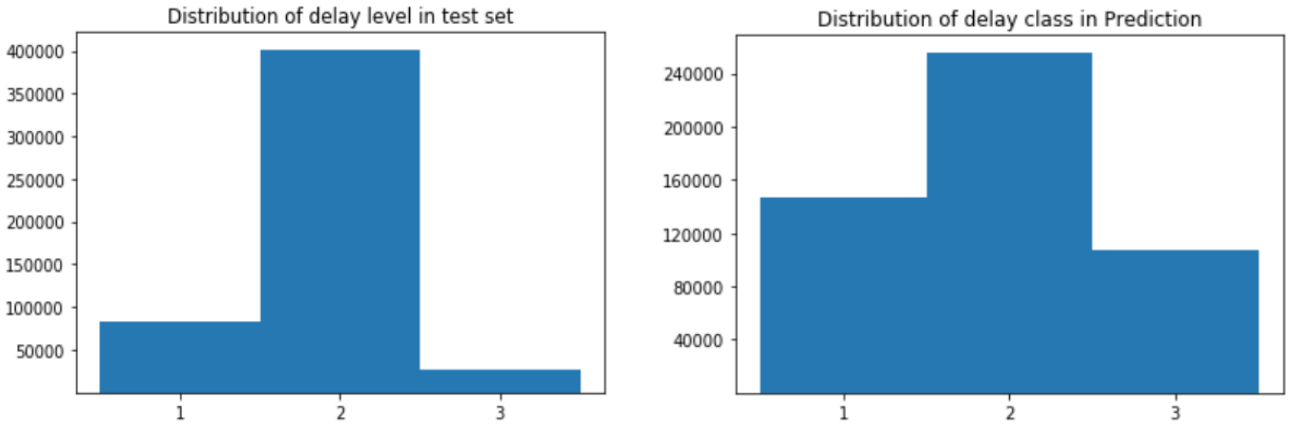
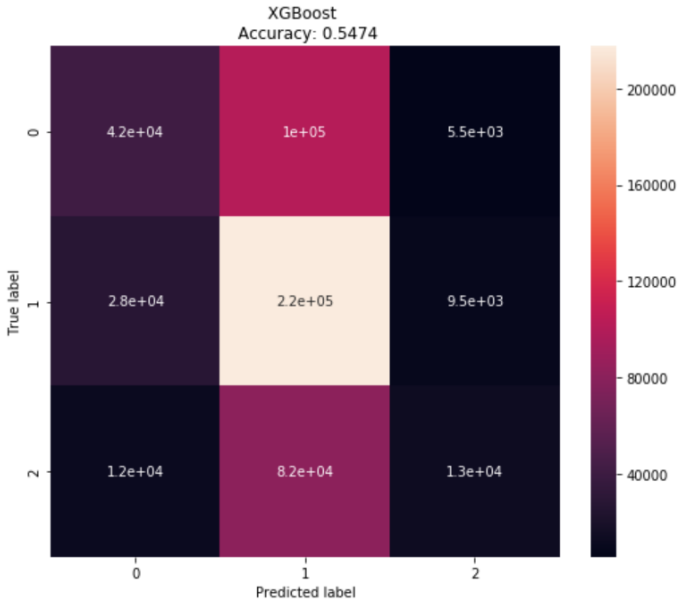


Figure 7: Left: The distribution of test set. Right: The distributions of predictions generated by XGBoost model on the test set.

We then applied Confusion Matrix to obtain further details about model errors in multi-class classification problems. According to Figure 8, there are 272785 correctly classified examples in test set and . And the detailed condition of misclassified examples of each categories could also be shown in Figure 8. The prediction of label 1 is the most accurate compared with others while



the condition of label 0 being misclassified as label 1 is the worst.

Figure 8: The Confusion Matrix of XGBoost

We attribute our misclassification into four different error sources: ignorant features, classification confidence, Data Deficiency and Absence of Generality which would be discussed in section 5.2.

5 Productization

As we have analyzed in section 1, the prediction of flight on-time performance has a significant value for both airports and passengers and we believe our classification models would provide a relatively accurate view of it. As it was shown in Section 4, our classification models outperform the baseline model by 20% in accuracy. According to the error analysis in Section 4.2 even though there would be some wrong labels, the prediction is still meaningful in reality and could be improved to a system to predict the on-time performance in the real world. Therefore, in this section, we are going to discuss the deployment of our models, the possible technical risks as well as the ethical considerations we may be faced with.

5.1 Deployment

The ultimate purpose of our analysis is to solve real-world problems and create business value. As discussed in Section 1, the business value of our prediction could be realized by transforming the whole analysis process into a prediction system, an industrialized product which could be used by both airports and airlines. The workflow of this system is shown in Figure 9.

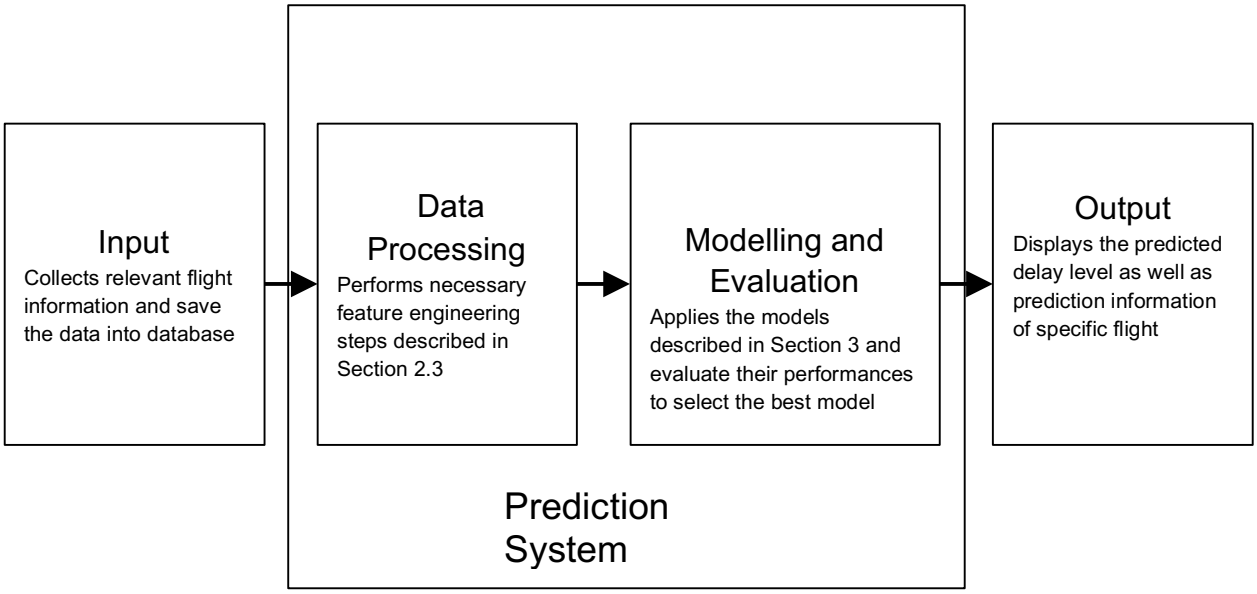


Figure 9. The workflow of Flight Delay Prediction System

The system could be dynamically tuned by inputting the latest features of flights to adapt to the real conditions and a feedback system could also be applied to adjust the parameters of the built-in models of the system.

5.2 Limitations and Risks

Users should be aware of the following limitations in our model:

Ignorant features: Our model suffered from the fact that it lacks some features which may have a profound effect on predictions such as weather. Since we collected the data from the database of Bureau of Transportation Statistics (BTS) with the absence of weather information, we may fail to take the weather condition into consideration When there is inclement weather condition, flights are unable to departure for safety reason and airport capacity is reduced due to the increased aircraft separations. We should consider the weather as a feature for modeling in our further study.

Classification Confidence: During the data processing, we divide the delay condition into three levels based on the actual delay time in minutes. However, the threshold may be lack of persuasion and thus leads to a great proportion of misclassification of group 1. This reason may contribute to the inaccuracy of the prediction.

Data Deficiency : Our data may be biased because it only consists of a small proportion of data from the Bureau of Transportation Statistics database. The daily flight flow is tremendous in each New York airports so that the database which records the on-time performance from 1995 to now should be very large. However, we are unable to study the whole dataset because it may take the huge amount of time for the model to generate a prediction.

Absence of Generality : We are investigating the data from all New York airports while each airport condition is unique. In this case, if the users deploy the model in other airports, it should consider the characteristics of local airports such as the capacity of the airports, the quantity of flight flow and other possible influential local features in the airports/locations. Our model may be reevaluated in other airports/locations.

5.3 Ethical Considerations

Our models aim to predict the delay status for a specific flight. The modeling results can benefit the airline companies in that they can prevent the delay before it actually happens. It can help the passengers to better plan their time. Yet there is a probability that the prediction is actually inaccurate. We would strive for a higher accuracy, but the error always occurs. Prediction errors may hurt the airlines and passengers in the following scenario: a passenger used the mature product to evaluate the delay status for a coming trip and saw that the flight with the airline he often chose had a high delay possibility thus giving up buying that airline ticket for this trip.The inaccurate prediction may cause the change of passenger preference, and therefore increase the possibility of churn problem and hurt airlines' profit.

6 Conclusion

In this report, we implemented data-driven methods and quantitative analysis to predict the on-time performance of flights. By applying several models and comparing their performances, we

achieve the highest accuracy 54.74% of XGBoost model. Besides, by taking advantage of the predicted delay class, our work could effectively improve the allocation of airport resources and produce economic values directly.

However, there are still some constraints of our model and we believe that our model could be enhanced by applicating deep learning and mining more relative features such as weather. We could also predict flight arrival delay situations.

Finally, our work would be industrialized into a prediction system which cold be applied by airports and airlines. We have enough reasons to believe that our product could make a difference to the future civil aviation industry.

7 Collaboration Statement

Starting with a clear plan, we distributed the workload equally among our group members and also helped each other a lot. Every member drafted on their data processing methods and models, and completed the final report together.

References

(1) L. Hao, M. Hansen, Y. Zhang, and J. Post. New York, New York: Two ways of estimating the delay impact of New York airports. Transportation Research Part E: Logistics and Transportation Review, 70(Supplement C):245–260, Oct. 2014.

(2) Y. Tu, M. O. Ball, and W. S. Jank. Estimating flight departure delay distributions—a statistical approach with long-term trend and short-term pattern. Journal of the American Statistical Association, 103(481):112–125, 2008

(3) S. Khanmohammadi, C. A. Chou, H. W. Lewis, and D. Elias. A systems approach for scheduling aircraft landings in JFK airport. In 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pages 1578–1585, July 2014.

(4) P. Balakrishna, R. Ganesan, and L. Sherry. Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of Tampa Bay departures. Transportation Research Part C: Emerging Technologies, 18(6):950–962, Dec. 2010.

(5) "Bureau of Transportation Statistics." Bureau of Transportation Statistics, www.bts.gov/.

(6) P. Balakrishna, R. Ganesan, L. Sherry, and B. S. Levy. Estimating Taxi-out times with a reinforcement learning algorithm. In 2008 IEEE/AIAA 27th Digital Avionics Systems Conference, pages 3.D.3–1–3.D.3–12, Oct. 2008.

(7) Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

(8) API design for machine learning software: experiences from the scikit-learn project, Buitinck et al., 2013.

(9) "Introduction to Boosted Trees." Introduction to Boosted Trees — xgboost 0.6 documentation. Accessed December 08, 2017.