**Capstone Project 1**

**Final Report**

# Title: Flight Delay Prediction with Machine Learning Models

**Cuthbert Lo**

**Feb 15, 2018**

**Table of Contents**


1. **Introduction……. Page 2**
2. **Dataset…….. Page 2**
3. **Data Cleaning and Wrangling……..Page 3**
4. **Data Exploration……. Page 4**
5. **Feature Engineering…….. Page 8**
6. **Modeling……... Page 9**
7. **Further Improvement……. Page 17**
8. **Conclusion……... Page 17**

## 1.Introduction

Every air traveler must have experienced a flight delay due to various reasons, like aircraft technical problem, weather, air traffic, late aircraft arrival, etc. However, no one can never accurately predict the chance of flight delay unless for some obvious reasons like adverse weather which may eventually cancel a flight. According to FAA definition, a delay is defined as the difference in minutes between scheduled and actual departure time more than 15 minutes. The aim of this project is to build a model with some publicly accessible dataset to build a model to predict the probability of a flight delay.

Air traveler, airlines, travel agents, ground handling agent, airport operators or other parties that needs to work according to flight schedule can use the model to predict the likelihood of a flight being delayed, so that they can plan the trip, aircraft, passenger handling, and other resources accordingly.

## 2. Dataset

We used two datasets which are available to public for free. The first dataset is the Airline On-Time Performance dataset from Bureau of Transportation Statistics. The second one is historical METAR (airport weather report) from Iowa Environmental Mesonet of Iowa State University (https://mesonet.agron.iastate.edu/ASOS/). Due to time and computer resources constraints, dataset we use only consists of 3 months of data April 2017 to June 2017 for the busiest 20 airports during the period which more than 784,000 rows.

### 3. Data Cleaning and Wrangling

First of all, we need to combine two datasets into one. Departure time, origin airports are the key columns to merge two dataset. But the time zone used in BTS dataset is local time while weather reports are in UTC. Before merging, we need to first convert local time to UTC. After merging two datasets, there are some duplicated rows during the merge operations which has been removed.

A flight status has more than just delayed or not delayed, a flight can be cancelled as well due to various reasons. Since this status is not of our interest in predicting a flight delay, data of cancelled flight has been removed.

In weather data, some of the columns may not have values. Like skyc1, skyc2, skyc3 and skyc4, as well as skyl1, skyl2, skyl3 and skyl4, are describing the cloud coverage of the sky, when the sky is clear or very few layers of cloud, they will either be empty or not utilizing all 4 columns. Those empty fields will be filled with -1 to indicate a null value.

In some records, wind direction and wind speed are missing which believe to be due to instrumental error because even when wind is calm there will be an entry “00000KT” in text report and zero in wind direct and wind speed columns. Those rows has been removed.

### 4. Data Exploration

There are more than 784,000 rows of data with 40 columns for 3 months of data.

### 4.1 Flight Delay Rate

The overall percent of flight delay is 22.3% of top 20 busiest airports across the country.

```
print('Percent of delayed flight in dataset: ', 100* df.IsDelay.sum()/len(df))
```

```
Percent of delayed flight in dataset:  22.313489322768664
```

When assessing the delay rate, we can look from angles like origin airports, airlines, time of departure, day of week. From Fig 1 and Fig 2, the most delayed airport is JFK (New York John Kennedy), EWR (Newark) and DFW (Dallas) while the most delayed VX (Virgin America), B6 (JetBlue), NK (Spirit).

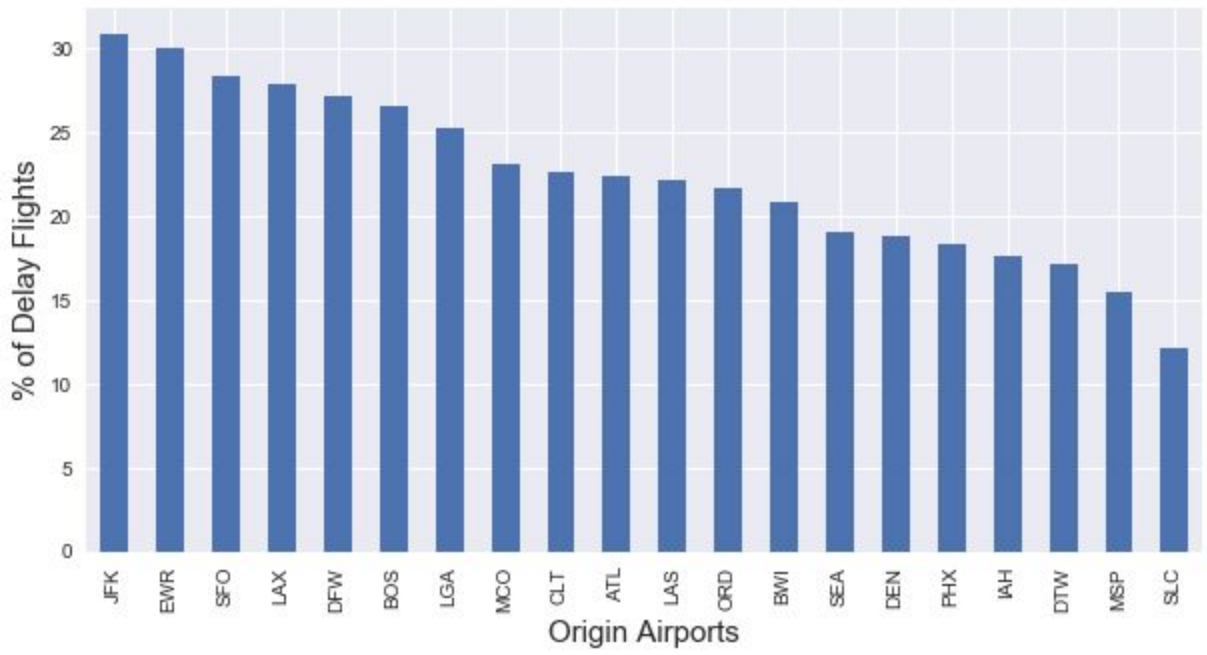


Fig. 1 the rate of delay flight of 20 busiest airports

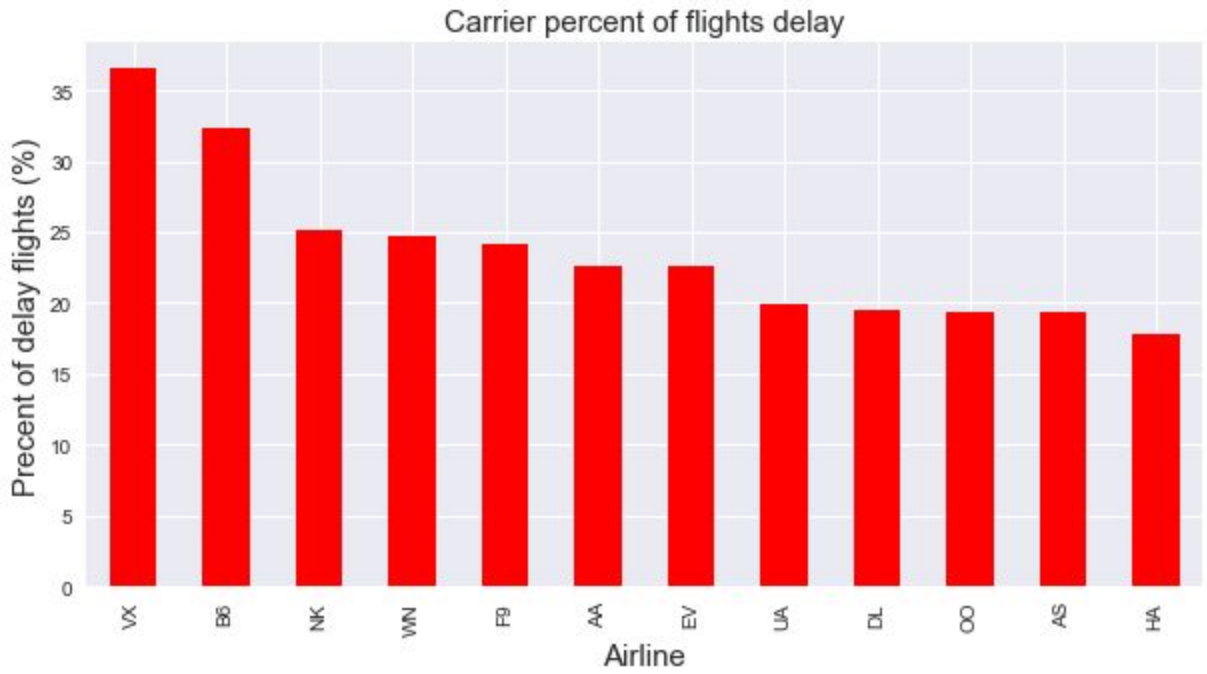


Fig 2. the rate of delay flight of different airlines

Looking at each airlines performance at each airports in Fig. 3 below, we can see not every airline operates at all 20 airports of concern or there is no delay recorded during those 3 months of interest. There are only 2 airlines, AA and DL, have operations in all 20 airports. HA (Hawaiian) has exposure in only 5 out of 20.

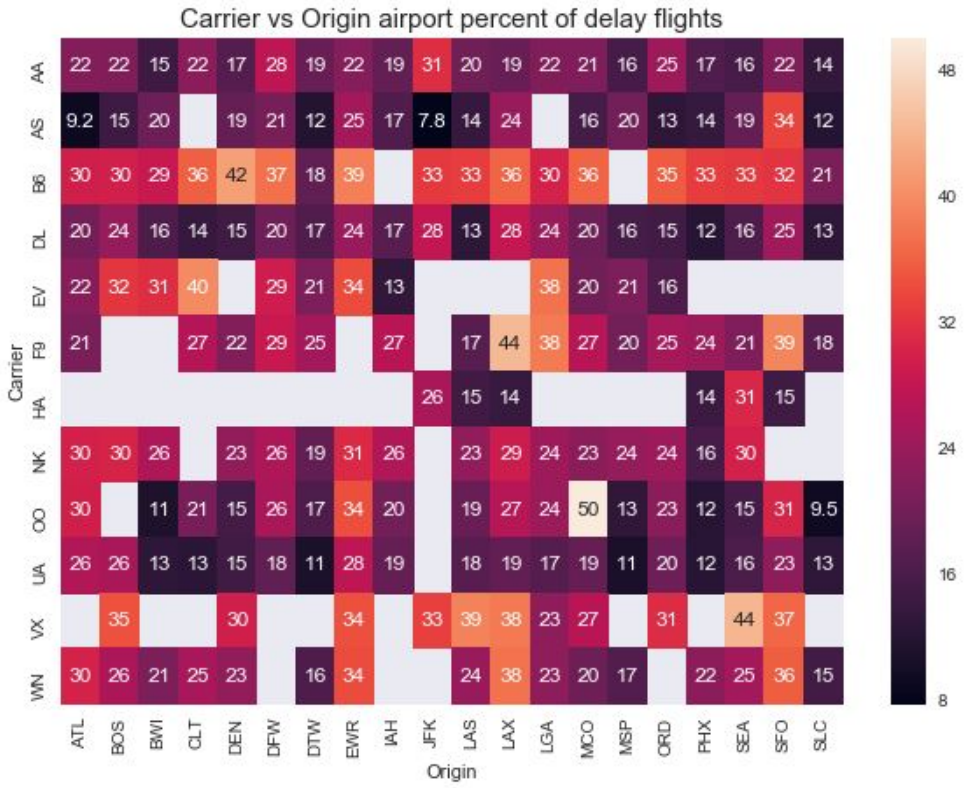


Fig 3. the rate of delay flight of airlines vs origin

In Fig 4., we can see the Wednesday, Thursday and Monday are the top 3 days of week in rate of delay.

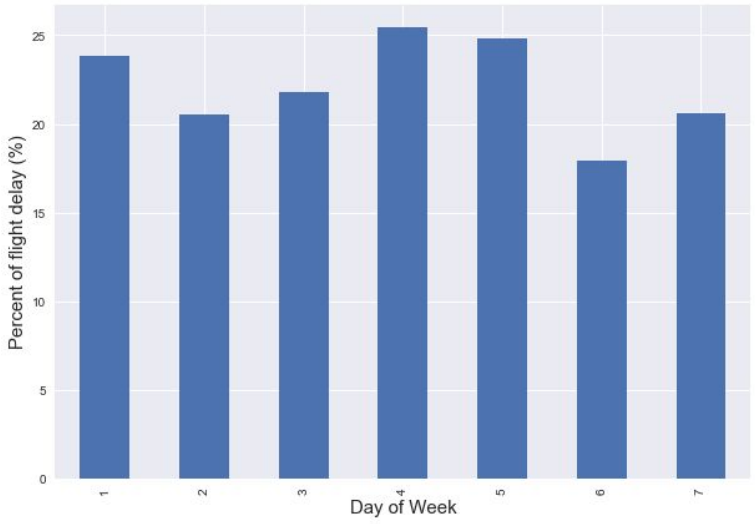


Fig 4. the rate of delay flight of day of week

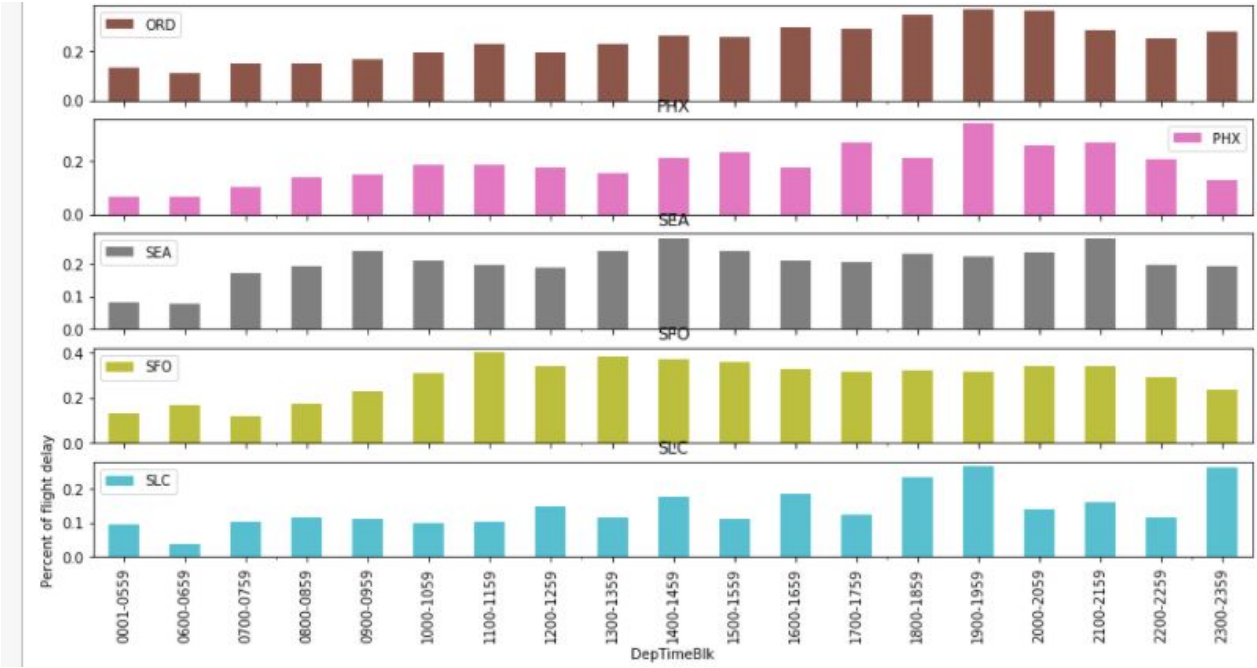


Fig 5. the rate of delay flight of day of week

In Fig 5. above shows five of the twenty airports the rate of delay flights throughout the day, we can see different airports have different delay pattern. For example, in the first graph showing Chicago O'Hare Int'l Airport, the peak delay period is between 1600-2100 hours while the second last one graph showing San Francisco Int'l Airport that the peak is around mid-day.

In Fig. 6 below, we can see when weather is TS BR which is thunderstorm and mist, the delay ratio is reaching 100% which is understandable. And for RA, rain only, the ratio is about 30%, a few percent higher than normal.

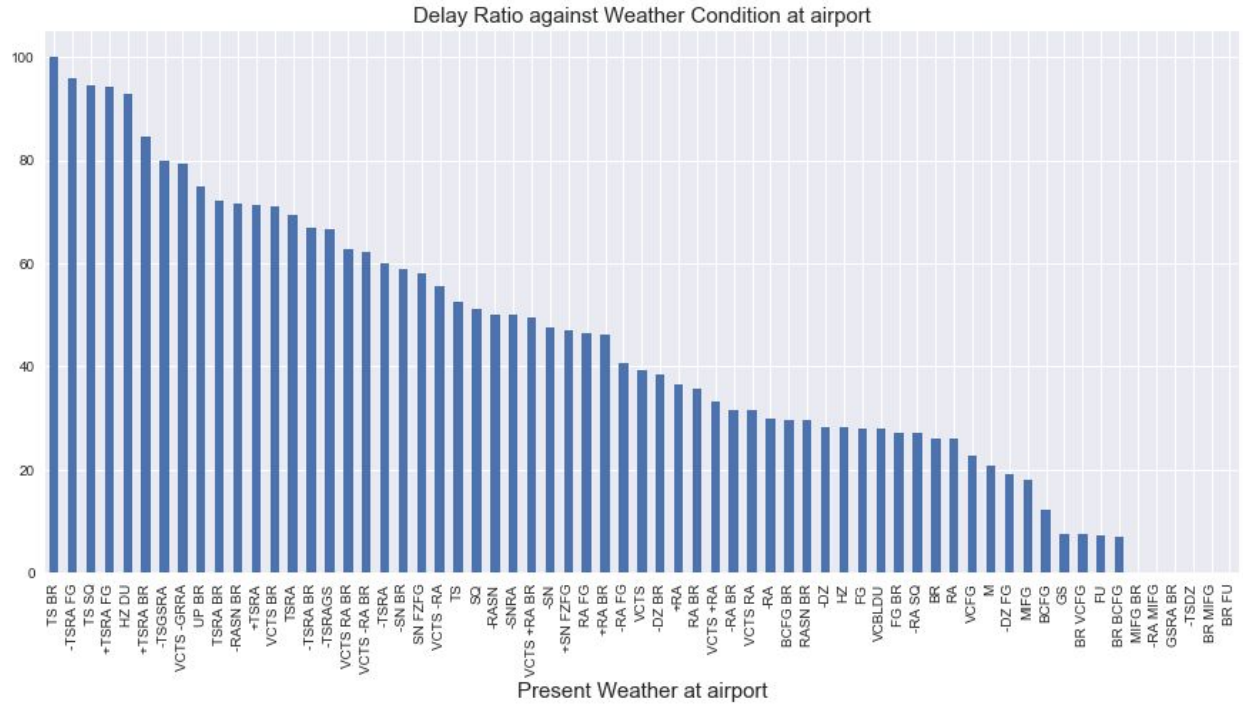


Fig. 6 Delay ratio and weather at airport

**5. Feature Engineering**

The combined dataset has 93 columns/features. Some of the columns are not related or not useful. This model is for prediction of departure delay, any post flight information are irrelevant, therefore WheelsOn, TaxiIn, ArrTime, ActualElapsedTime, AirTime, Cancelled, Diverted, actual minutes of delay, etc  can be dropped. Other columns like city name of origin and destination airports can be identified by airport codes instead.

Two rows are created “AptPreviousDateDelay” and “AptNoOfFlight”. The first one is the no. of flight delay on previous day at the same airport, the second one is total no. of flight on the day at the airport. Eventually, there are  40 columns/features for model training.

```
In [199]: %%time

df_merged.drop(['FlightDate','TailNum','FlightNum','Flights',
                'CarrierDelay','WeatherDelay','NASDelay','SecurityDelay','LateAircraftDelay',
                'CancellationCode','FirstDepTime','TotalAddGTime','UniqueCarrier','AirlineID',
                'OriginAirportID','OriginAirportSeqID','OriginCityMarketID','OriginStateFips',
                'OriginStateName','OriginCityName','OriginTimeZone',
                'DestAirportID','DestAirportSeqID','DestCityMarketID','DestStateFips',
                'DestStateName','DestCityName',
                'DepTime','DepDelay','DepDelayMinutes','DepDel15','DepartureDelayGroups',
                'ArrDelay','ArrDelayMinutes','ArrDel15','ArrivalDelayGroups',
                'ArrTime','ActualElapsedTime','AirTime',
                'WheelsOn','TaxiIn','TaxiOut','WheelsOff',
                'Cancelled','Diverted','UTCFlightDateTime',
                'valid','station','lat','lon',' mslp','DateHr',' metar'],
               axis=1, inplace=True)

CPU times: user 361 ms, sys: 646 ms, total: 1.01 s
Wall time: 1.98 s
```

## 6. Modeling

We would like to predict a certain flight will be delayed or not, therefore, there are two classes namely Positive or 1 means will be delayed, Negative or 0 means will not be delayed. The two classes are slightly imbalance which is around 78% is Class 0 (no delay) and 22% is Class 1 (delay) in three months of data. We use supervised learning binary classification algorithm the build the models. The models are trained using 70% data and the remaining 30% is used for prediction and evaluation of models' performance.

### 6.1 Data Pre-processing

Before we can feed any data to train the model, there are some preprocessing steps which must be done.

#### A. Label encoding

Categorical features like present weather like rain, fog, storm, etc. which is in standard weather code and origin, destination airports which are in text. We need to first convert those into labels by sklearn LabelEncoder function.

```
In [8]: from sklearn.preprocessing import LabelEncoder
        le = LabelEncoder()
        tmp = df[cat_list].apply(le.fit_transform)
```

### B. Data splitting

After encoding labels to categorical features, we split the dataset into training set and test set at the ratio of 70% to 30%.

### C. Hyperparameters tuning

Before we can identify the best suitable model for the problem, we have to tune the model by finding out the best hyperparameters for each model. We ran a grid search to find the best hyperparameters for each ML model, then we use the best hyperparameters found to initiate the classifiers, then plug in  the training set to training the model and the test set for prediction. Below are the graphs showing the ROC curve and Precision-Recall curve for each ML models performance.

### 6.2 Model Selection

After pre-processing the dataset and grid searching best hyperparameters, we can start training the model and use the test set to evaluate model performance. We compared 8 different classifiers for this problem, the best performance model is Gradient Boosting while the second is Random Forest.

### 6.3 Metrics selection and Benchmark

In this business case of this model, we would like to minimize false positive prediction which means minimizing the case that predicted to be delay but actually it's not. In layman terms, the airline tells you that your flight will be delayed and you arrives at airport late then find out your flight is on time and gate is closing soon. This is the scenario we would like to avoid.
Therefore, we need to optimize precision. In the following, we will evaluate each model and select the best suitable one. It's good to have a benchmark for evaluating any metrics, the most dominant is negative which account of 78% and positive class is 22%. We need to have a model that performance better than a blind guess.
For each model, there is a confusion matrix as well as the classification report showing the result of different metrics for evaluation.

### 6.4 Models

Gradient Boosting

This model is among the best performing one in precision that reaching 0.63 in positive class and 0.81 in negative class.

Confusion matrix:
[[17565 725]
[ 4024 1229]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.81 | 0.96 | 0.88 | 18290 |
| Delay | 0.63 | 0.23 | 0.34 | 5253 |
| avg / total | 0.77 | 0.80 | 0.76 | 23543 |

Fig. 7 Metrics for Gradient Boosting Model

Logistic Regression

In the first run, precision for negative class is 0.78 and 0.51 for positive class. Through optimize the class weight to “class_weight={0:0.3,1:0.7}”, we see a significant improvement in recall, however, precision in positive class has decreased by 0.13. Since we are optimizing for precision in this problem, we will keep the default class weight.

Confusion matrix:
[[18110 180]
[ 5069 184]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.78 | 0.99 | 0.87 | 18290 |
| Delay | 0.51 | 0.04 | 0.07 | 5253 |
| avg / total | 0.72 | 0.78 | 0.69 | 23543 |

Fig. 8 Metrics for Logistic Regression balanced class weight

Confusion matrix:
[[15319  2971]
 [ 3411  1842]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.82 | 0.84 | 0.83 | 18290 |
| Delay | 0.38 | 0.35 | 0.37 | 5253 |
| avg / total | 0.72 | 0.73 | 0.72 | 23543 |

Fig. 9 Metrics for Logistic Regression class_weight={0:0.3,1:0.7}

Decision Tree

Confusion matrix with balanced class weight:
[[18013   277]
 [ 4863   390]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.79 | 0.98 | 0.88 | 18290 |
| Delay | 0.58 | 0.07 | 0.13 | 5253 |
| avg / total | 0.74 | 0.78 | 0.71 | 23543 |

Fig. 10 Metrics for Decision Tree with balanced class weight

Confusion matrix with class_weight={0:0.3,1:0.7}:
[[15052  3238]
 [ 3052  2201]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.83 | 0.82 | 0.83 | 18290 |
| Delay | 0.40 | 0.42 | 0.41 | 5253 |
| avg / total | 0.74 | 0.73 | 0.73 | 23543 |

Fig. 11 Metrics for Decision Tree with class_weight={0:0.3,1:0.7}

Random Forest

In Fig. 3, the precision is improved slightly when changing the class weight.

Confusion matrix with balanced class weight:
[[17484  806]
 [ 4090  1163]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.81 | 0.96 | 0.88 | 18290 |
| Delay | 0.59 | 0.22 | 0.32 | 5253 |
| avg / total | 0.76 | 0.79 | 0.75 | 23543 |

Fig. 12 Metrics for Random Forest Model with balanced class weight

Confusion matrix with class_weight={0:0.3,1:0.7}:
[[17600  690]
 [ 4179  1074]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.81 | 0.96 | 0.88 | 18290 |
| Delay | 0.61 | 0.20 | 0.30 | 5253 |
| avg / total | 0.76 | 0.79 | 0.75 | 23543 |

Fig. 13 Metrics for Random Forest with class_weight={0:0.3,1:0.7}

Neural Network

Confusion matrix:
[[16713  1577]
 [ 4440   813]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.79 | 0.91 | 0.85 | 18290 |
| Delay | 0.34 | 0.15 | 0.21 | 5253 |
| avg / total | 0.69 | 0.74 | 0.71 | 23543 |

Fig. 14 Metrics for Neural Network

k-Nearest Neighbor

Confusion matrix:
[[18282     8]
 [ 5248     5]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.78 | 1.00 | 0.87 | 18290 |
| Delay | 0.38 | 0.00 | 0.00 | 5253 |
| avg / total | 0.69 | 0.78 | 0.68 | 23543 |

Fig. 15 Metrics for k-Nearest Neighbor

Naive Bayesian

Confusion matrix:
[[15506  2784]
 [ 3803  1450]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.80 | 0.85 | 0.82 | 18290 |
| Delay | 0.34 | 0.28 | 0.31 | 5253 |
| avg / total | 0.70 | 0.72 | 0.71 | 23543 |

Fig. 16 Metrics for Naive Bayesian

Support Vector Machine

Confusion matrix:
[[18290 0]
[ 5253 0]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Not Delay | 0.78 | 1.00 | 0.87 | 18290 |
| Delay | 0.00 | 0.00 | 0.00 | 5253 |
| avg / total | 0.60 | 0.78 | 0.68 | 23543 |

Fig. 17 Metrics for Support Vector Machine

In Fig. 18 below, shows the ROC curve and the PR curve of all models, Gradient Boosting is orange line and Random Forest is the green line, for both curves, the orange line is among the best and green line performs slightly worse than the orange.

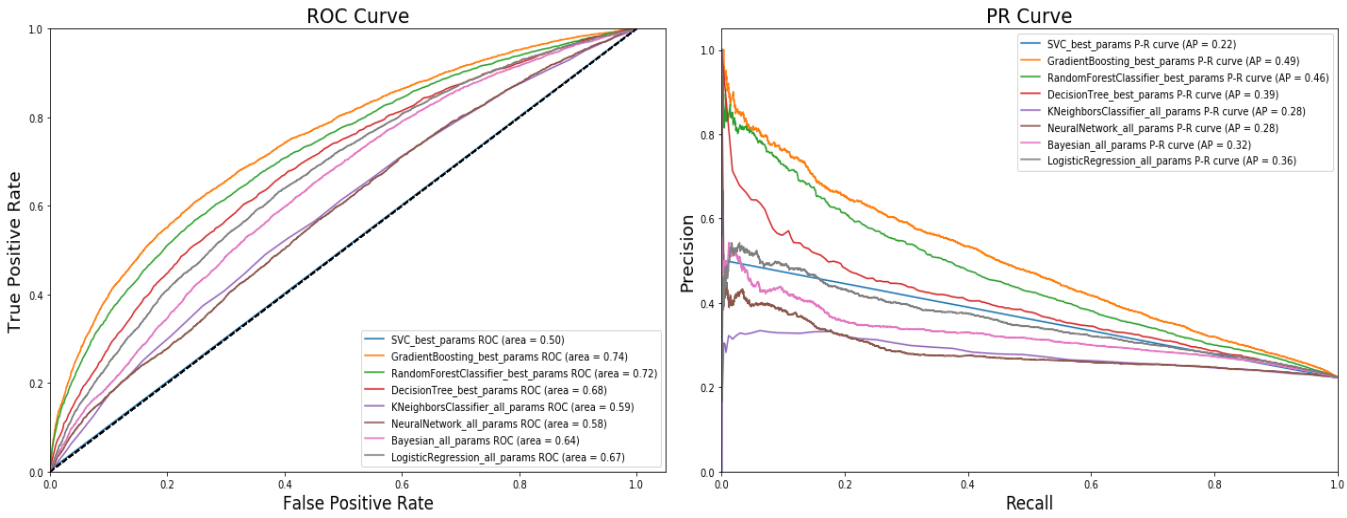


Fig. 18 ROC Curve and Precision-Recall Curve for all models

## 7. Further improvement

Currently this model is built based on dataset in the public domain which provide limited perspective to the problem like weather, basic flight information and airports information. If adding some perspective like airlines operations information e.g. passenger counts, aircraft maintenance history; and historical air traffic control information which would help the model to be more generalize in solving the flight delay prediction problem.

## 8. Conclusion

After exploring all datasets, we used five different supervised classification algorithms (Logistic Regression, Gaussian Naive Bayes, Random Forest, Decision Tree, Gradient Boosting, Neural Network, k-Nearest Neighbor and SVM) to train the predictive model by using 70% of the whole data. The remaining 30% was used to evaluate the model.

1. Using various performance evaluation metrics, we found that the Gradient Boosting classifier gives the the best model performance.
2. We achieved the ROC AUC to be about 0.74. The AUC for PR was not great (about 0.49) but the precision is reaching 0.63 in positive class and 0.81 in negative class.

This model is good enough to predict the likelihood or probability of a flight delay. As due to the computation resources we have for this project, only 20 airports and 3 months of data is studied, the lack of data will limit the training of the model. By overcoming the dataset size, computation resource problem and those improvement mentioned in Section 7, the model can be further improved.

Sources:

Bureau of Transportation Statistics, from
https://www.transtats.bts.gov/Tables.asp?DB_ID=120&DB_Name=Airline%20On-Time%20Performance%20Data&DB_Short_Name=On-Time

Iowa Environmental Mesonet, Iowa State University, from
https://mesonet.agron.iastate.edu/ASOS/