Flight Delay Prediction - Project Report

by Dorian Buijse and Konrad Krawczyk

1. Introduction

1.1. Purpose statement

Throughout the year 2015, there has been over 5,4 million domestic flights within the US. All of their metadata are recorded and saved in the Department of Transportation's (DOT) Bureau of Transportation Statistics.

Flight delays cause significant financial and other losses to airlines, airports, and passengers. Their prediction is crucial during the decision-making process for all players of American aviation industry. Therefore, predicting the likelihood of delay based on flights' features bridges an important information asymmetry between airlines and passengers.

The primary use case of the algorithm will be: **predicting a potential delay, on a given day, for a given airport and airline.**

1.2. Data Overview

The dataset to be analysed consists of data about all domestic flights in the United States for the whole year 2015, for all airports and airlines. The data was collected and published by the DOT's Bureau of Transportation Statistics. The raw data file, with appropriate documentation, can be found on: https://www.kaggle.com/usdot/flight-delays

The whole set has over 5,400,000 examples. By default, every example has 31 features, although not all information is complete for all examples. It is nevertheless an extremely large and comprehensive dataset, which could allows for accurate statistical inferences.

2. Data Analysis

In order to investigate what factor could have a significant influence on delays (potential correlations or a lack thereof) we needed to analyse the data. Although the set had 31 pieces of data per sample, not all of them have been relevant to delay time. For example, flight number hardly could influence any delay. Also, since features such as taxi time or wheels-off time are directly related to departure time (it's a standard set of procedures with more or less fixed durations) these features should not be of any influence to our data set. For the data analysis, we picked several features that could potentially have some correlations with our target and could enhance our predictions.

Feature (Key)	Data type	Significance
Day and month (DAY, MONTH)	Numeric	Depending on the time of the year, different factors (weather, holidays, other events) could affect the traffic across all airports
Day of the week (DAY_OF_WEEK)	Numeric (counted from Monday onwards)	Trends can also unfold for particular days of the week (over the weekend e.g.)
Airline identifier (AIRLINE)	String (two-letter identifier)	Using the prediction algorithm, a user can compare airlines' punctuality which fosters informed consumer decision-making
Departure airport (ORIGIN_AIRPORT)	String (IATA code)	Creating a prediction model like this would potentially help in identifying airports that are poorly organized; also in estimating real travel time and minimizing the risk of missed connections
Destination time (SCHEDULED_DEPARTURE)	Numeric (format: HHMM)	At different times of the day traffic blocks are more likely to happen (for example during the night)

Figure 1. Table of features (independent variables) considered for data analysis

The value of dependent variable used for data analysis was the ratio of delayed to non-delayed flights. The ratio was calculated by using Counter() feature in Python. The ratio had to be calculated since using a number of flights per se would skew our patterns on favor of features with the greatest number of flights total. For every example within a sample in which a 'DEPARTURE_DELAY' was greater than a threshold we set (10 minutes), the number of delayed flights per day was incremented in the counter. Then within the same loop we divided the number of delayed by number of non-delayed flights, then passed it to a MatPlotLib bar chart generator.

The figures below show some of the patterns within the data.

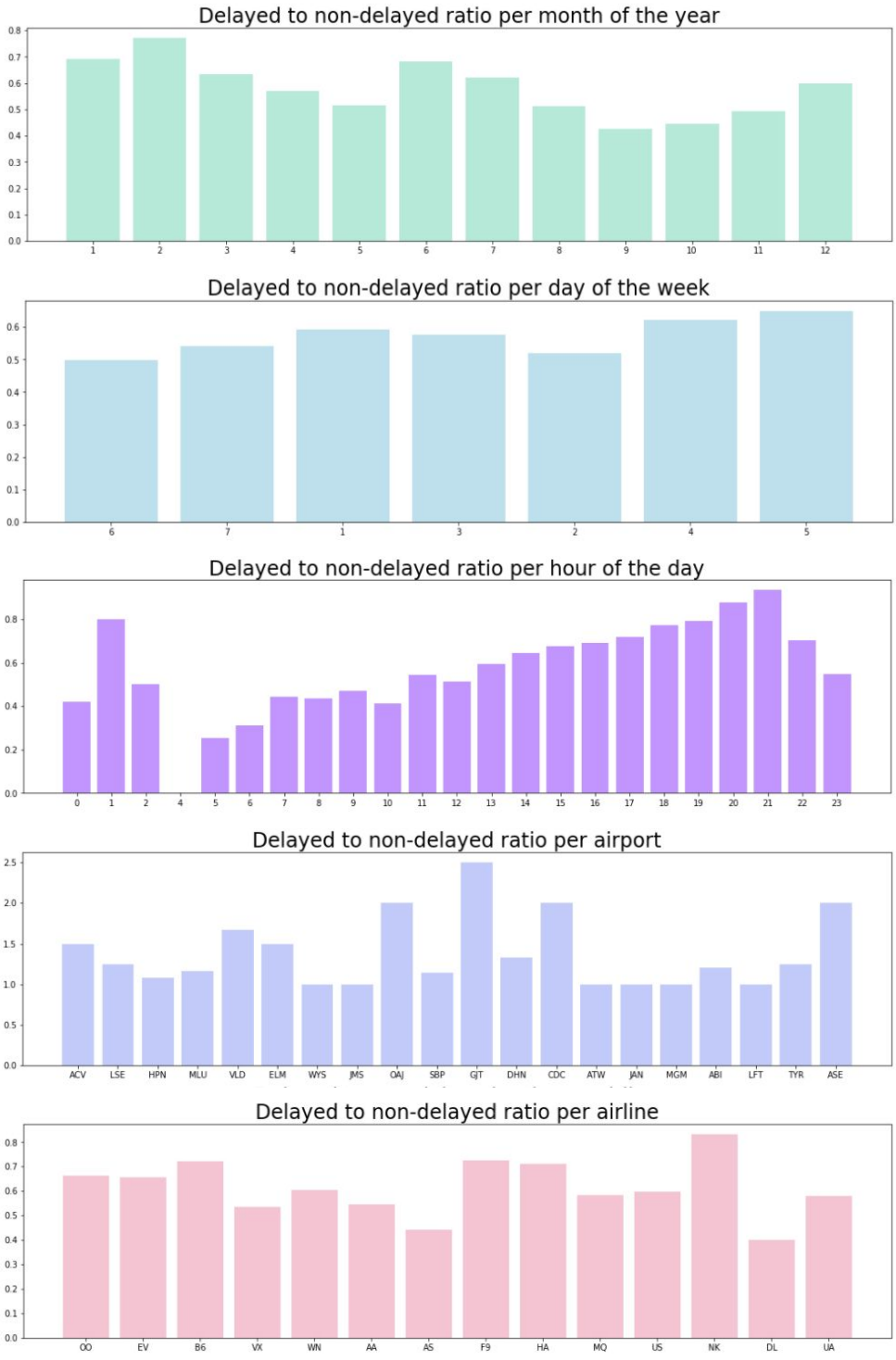


Figure 2. Ratios of delayed and non-delayed flights per month, day of the week, hour, departure airport and airline

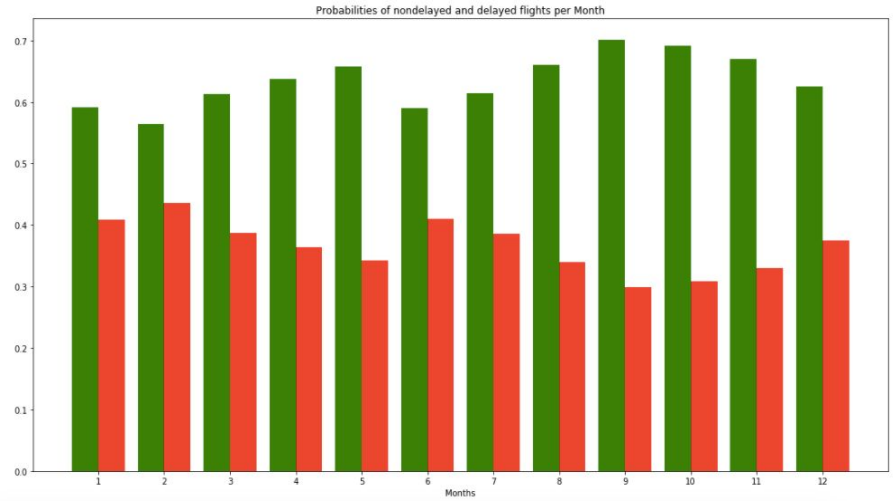


Figure 3.1.
Probabilities of delayed and non-delayed flights per month

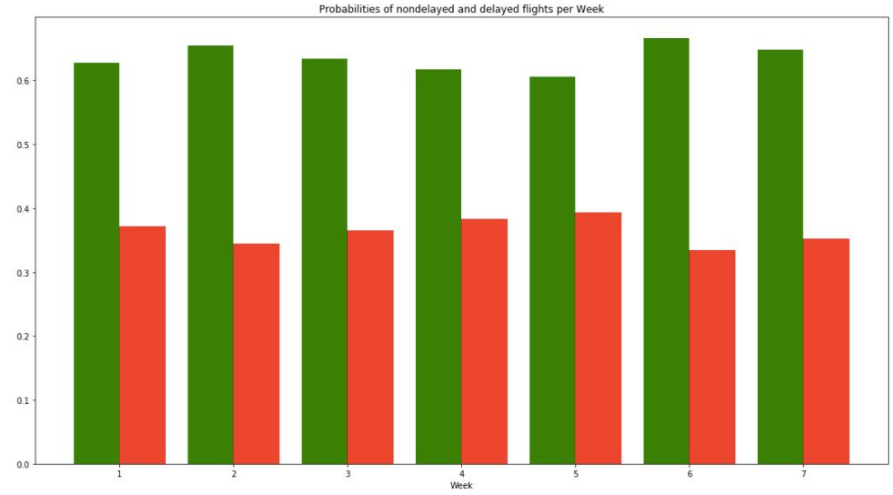


Figure 3.2.
Probabilities of delayed and non-delayed flights per week

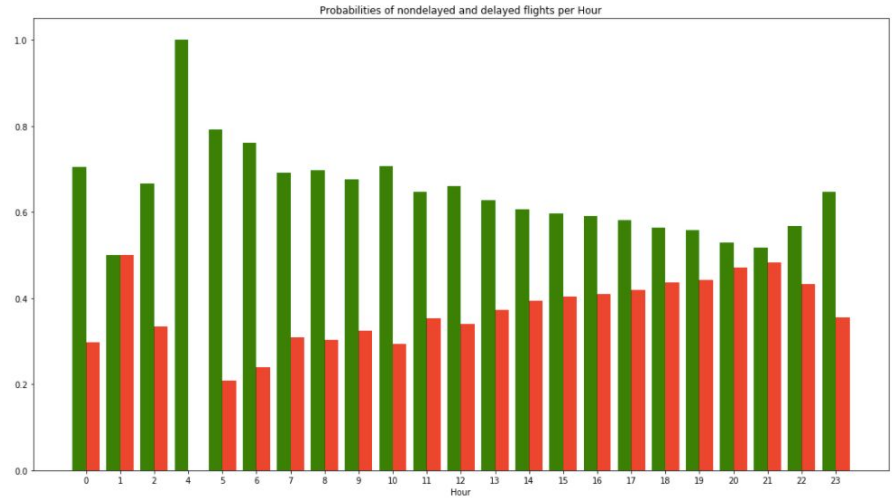


Figure 3.3.
Probabilities of delayed and non-delayed flights per hour

There are several conclusions that can be drawn from the data:

1. We have two seasonal spikes in delayed flights. The first one is during the winter season - possibly because of the weather conditions and winter holidays. The second one is during the summer, especially in June - when many people travel for summer vacation.
2. For days of the week, Friday and Monday have the highest ratio of delayed to non-delayed flights, probably because the traffic is significantly higher right before and after the weekend.
3. Evening and late night flights are usually most likely to be delayed, either because of the increased traffic or other factors, such as low visibility, difficult weather conditions or workforce fatigue.
4. A majority of airports with a high number of delayed flights are regional, low-scale airports. It might be useful to check for inverse correlation between airport traffic and ratio of delayed to non-delayed flights.
5. Low-fare airlines tend to have a significantly greater amount of delays. The top three: Spirit Airlines (NK), Frontier (F9) and JetBlue (B6) are all low-cost operators that are more likely to compromise on customer experience to cut down maintenance costs. The biggest 3 airlines in the US - American, United and Delta are all less likely to delay, with Delta having the best score in this category.

3. Methods & Approaches

3.1. Algorithms

For the prediction model building, we decided to use built-in functions within the sklearn kit.
The following table shows the types of algorithms we used, with their advantages and challenges with regard to our dataset:

Algorithm	Pros	Cons
Neural network (Multi-Layer Perceptron)	With a large amount of data and a limited number of features NN is likely to give good predictions	Very slow, especially given that we're dealing with very large amounts of data
Random forest	Reduced variance (relative to decision trees), trees are decorrelated	Not as easy to visually interpret as decision trees
Logistic Regression	Can provide probabilities, can be used with kernel methods, faster than neural network	Some risk of underfit

3.2. Data processing

Since we worked on a large data file, we had to prepare several datasets for different purposes. We had to fix some technical issues with the set, primarily in the airport data - airport codes for October consisted of 5 numbers instead of two capital letter. After fixing the dataset, we created the first subset - with only delayed flights that had a known reason for the delay.

For the sake of simplicity, we also limited the number of airlines and airports to only include the major ones in the analysis. However, we decided not to do that, as it would require us to arbitrarily determine limits of our dataset.

For the data analysis part, we created a set of 12,000 flights (trimmed down from 5.4m) by looping through each row and only taking each multiple of 480 since 5.4m/480≈12k 5.4m/480≈12k.

Later on, we created several other files, one with 72,000 flights and one with 148,000. These files let us compare the influence of data size on the accuracy of our prediction algorithm. It was especially important since our data has an asymmetry of delayed versus non-delayed flights, which causes bias against delays in our predictions. It is for this reason that, for our final models, we used a total dataset of 406,559 flights, of which 204,760 delayed and 201.799 non-delayed.

The features we used for final implementation were:

- Month (numeric)
- Day of the week (numeric)
- Departure time (numeric, converted to full hours)
- Airline (converted to dummy variables)
- Departure Airport (converted to dummy variables)
- Arrival Airport (converted to dummy variables)

The target we ended up using was: **arrival time (discrete).**

Where class 0 is defined as “Any delay smaller than or equal to 10 minutes”, and class 1 is defined as “Any delay larger than 10 minutes”

Some other features we used were:

- Airtime (numeric)
- Distance (numeric)
- Scheduled arrival time (numeric)

Some of the features, such as airport code or airline, were saved as categorical variables (strings), which cannot be an input to the sklearn prediction functions. Therefore they had to be converted into so-called *dummy variables*. Every unique string in the feature had to be converted into a separate feature, with a value of 1 for every data sample where that specific string shows up in the feature. We used the *get_dummies* function from Pandas to do this.

Aside from that, due to relatively large differences between numbers (hours vs day of the week for example) we had to scale our features, using the *StandardScaler* from sklearn.

3.3. Training and Cross-Validation

At the beginning, the objective was to get the algorithms to make sufficiently accurate predictions on the training set. Our assumption was that if predictions are not accurate enough on the train set, they will not work on the test set either. Poor prediction on the training set means that there is a bug in the prediction function, or our data is not suitable for making predictions.

The following features were compared during the CV phase:

Algorithm	Parameter for adjustment	Value set
Multi-Layer Perceptron	Hidden layer sizes	(30, 30, 30)
Random forest	Minimum samples split	14
Logistic Regression	C-value	0.0003

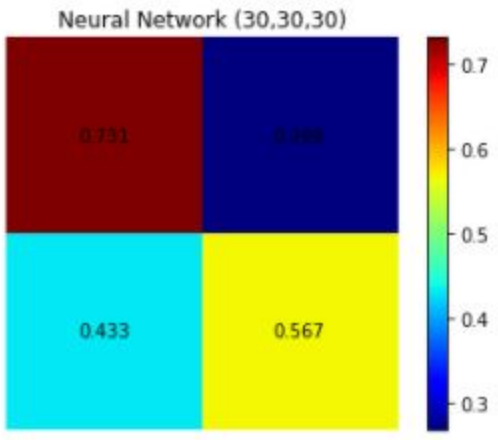
For the cross-validation part, the GridSearchCV function has been used, with 10-fold validation.

4. Results

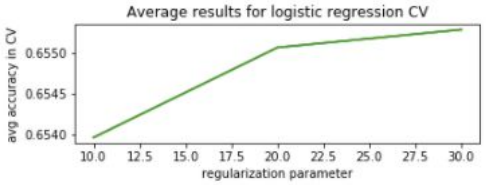
4.1. Multilayer Perceptron:

As seen on the below, our mlp algorithm was 6.7% better than chance (0.5) at predicting a delay. Where as it was over 23% more accurate on the non delayed flights.

The multilayer perceptron algorithm performed the best of all three algorithms, and was remained consistent when tried on the test set.

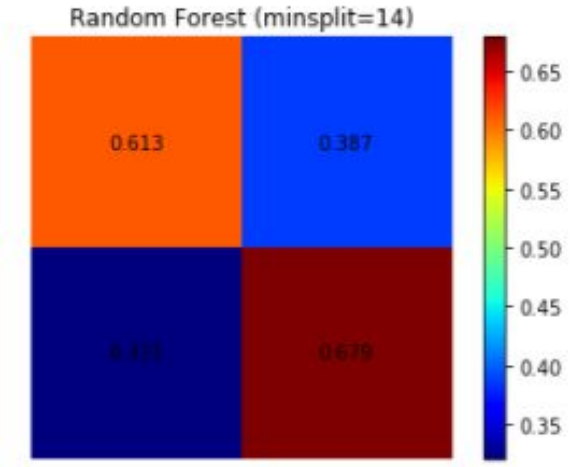



```
-----------------------NEURAL NETWORK:-----------------------

Cross-Validation: means & stds for given

Best score: (30, 30, 30)

Classification report for the best c-parameter (train set):
             precision    recall  f1-score   support

          0       0.70      0.75      0.73    183368
          1       0.65      0.59      0.62    141878

avg / total       0.68      0.68      0.68    325246


Confusion matrix for the best c-parameter (train set):
[[137490  45878]
 [ 58045  83833]]
Classification report for the best c-parameter (test set):
             precision    recall  f1-score   support

          0       0.69      0.73      0.71     45946
          1       0.62      0.57      0.59     35366

avg / total       0.66      0.66      0.66     81312


Confusion matrix for the best c-parameter (test set):
[[33593 12353]
 [15311 20055]]
```

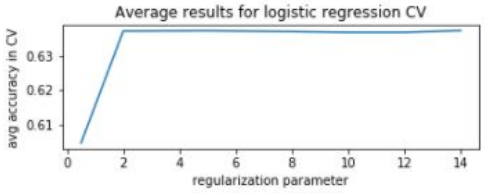


4.2. Random Forest

Our random forest algorithm was showed no favor to either of the two classifications and classified both delayed and non-delayed with a similar success rate (68% and 61% resp.)

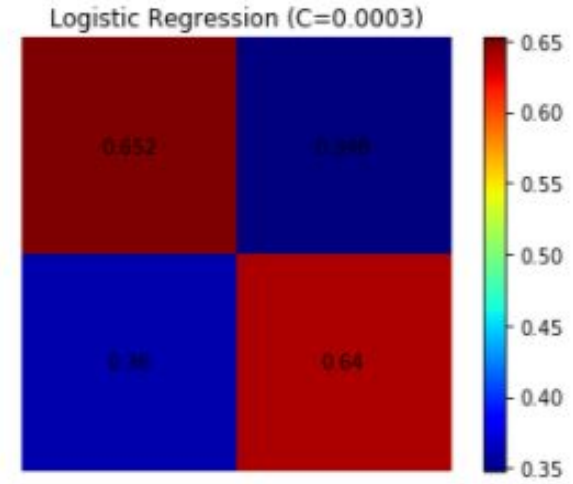



```
-----------------------RANDOM FOREST:-----------------------

Cross-Validation: means & stds for given

Best score: 14

Classification report for the best c-parameter (test set):
             precision    recall  f1-score   support

          0       0.71      0.61      0.66     45946
          1       0.57      0.68      0.62     35366

avg / total       0.65      0.64      0.64     81312


Confusion matrix for the best c-parameter (test set):
[[28143 17803]
 [11359 24007]]
```

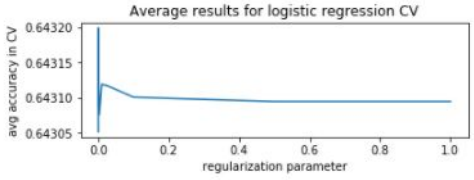


4.3. Logistic Regression

Logistic Regression was even more well balanced than our random forest model, with both classifiers hovering around the 65%.




```
-----------------------LOGISTIC REGRESSION:-----------------------

Cross-Validation: means & stds for given c-values

Best score: 0.0003

Classification report for the best c-parameter (train set):
             precision    recall  f1-score   support

          0       0.70      0.65      0.67    183368
          1       0.58      0.64      0.61    141878

avg / total       0.65      0.64      0.65    325246


Confusion matrix for the best c-parameter (test set):
[[118765  64603]
 [ 51264  90614]]

Classification report for the best c-parameter (test set):
             precision    recall  f1-score   support

          0       0.70      0.65      0.68     45946
          1       0.59      0.64      0.61     35366

avg / total       0.65      0.65      0.65     81312


Confusion matrix for the best c-parameter (test set):
[[29940 16006]
 [12736 22630]]
```



5. Evaluation

5.1. Approach

During our research we were very happy with the preprocessing and reduction of our data. Our 12k file was used to create the three models, the 72 and 148k files were used to test the models on larger datasets, and when we made sure the algorithms were sound, we had a balanced dataset of over 400k items to train and test our algorithms on. Once we were confident in handling the data the algorithms were easily applicable too. Although we would have liked to have more features, we managed to set up a structure that still worked somewhat for our features. The one thing we would have liked to research, was the influence of fluctuating the number of classifiers on our dataset. For now all we could predict is more or less than 10 minutes of delay.

5.2 Results

At first, it seemed like our models were not going to perform very well, however after introducing 400k data points, we managed to get all three of them to get above chance. That in and off itself feels like an accomplishment. Features were sparse, and many were not very efficient at predicting delay. Still this sparse data managed to find three methods that could predict delay vs non-delay with a 65% accuracy.

Since all three were able to classify with a similar f1-score, comparing the models' performance seems out of place. One thing to note, is that our test set was built of 56% non-delayed and 44% delayed, which may be the reason our 0 classifier's precision is higher.

In order to continue this research and get a better model, one would need to increase the number of features, and most likely improve the quality of the features. If this is done well, we hope to see a marketable app, that predicts the delay for us or even calculates whether a selected layover is manageable or not.