Prediction of airline departure delays

Tony Zhang and Menghua Wu

December 16, 2016

1 Introduction

Often, people purchase plane tickets to minimize monetary cost. However, price is not the only consideration when consumers purchase tickets—not all flights are created equal. In particular, flyers often care about and take into account the risk that a flight will be delayed. This paper hopes to quantify this risk on the basis of past flight delay data.

The United States Bureau of Transportation Statistics (BTS) compiles comprehensive datasets annually regarding the nation's transportation infrastructure, including aviation, maritime, highway, and rail. [1] In this paper, we focus on the aviation dataset, which reports on a wide range of variables concerning individual flights, including carrier, origin and destination, and flight delays. We will primarily concern ourselves with classification: whether a flight is delayed or not.

We downloaded data from the month of June 2015, with a total of approximately 500000 individual flights. Notably, some of the delay data is missing. We used these data to build a classifier that predicts whether or not a flight's departure is expected to be delayed. We applied several methods of classification and compare their results here.

While we will briefly remind the reader of particular aspects of each model, we assume familiarity with them. As a reference for implementation details and model specifics, we direct the reader to [2].

2 Data preprocessing

For simplicity, we only considered the following data features. These features are also likely to be available to customers who are planning their flights.

- Departure date and time
- Airline
- Origin and destination airports

Since airline and airports are categorical features, we encoded them as one-hot vectors. In the data subset we considered, there were 14 airlines and roughly 250 airports.

For date and time, we began by encoding them as days into a year and minutes into the day. This representation leaves much to be desired, however.

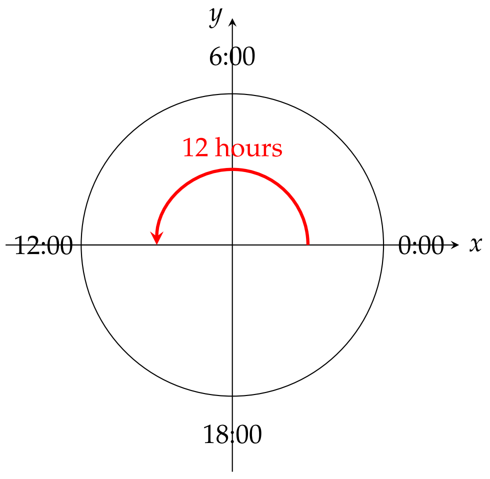
2.1 Cyclic features

Features such as time of day and day of year are naturally cyclic. Representing them on a linear scale therefore imposes an unnatural metric on them. For instance, if we naively represent times as "minutes into day", the time 23:59 will be considered "more similar" by any machine learning model to noon than to 00:01, which disagrees with our intuitive notions of closeness.

A better representation of cyclic data would respects our notions of closeness. An obvious choice is to map a linear data point $x \in [0, T)$ onto a unit circle by

$$x \mapsto \left(\cos\frac{2\pi x}{T}, \sin\frac{2\pi x}{T}\right). \tag{1}$$

For instance, if we were to encode the time of day, noon would have "angle" $\frac{2\pi x}{T} = \pi$ and would thus get mapped to the point $(-1, 0)$:



We encoded dates (with T being days in a year) and time (with T being minutes in a single day) in this fashion. In preliminary testing, we found that this encoding noticeably improved classification accuracies for a variety of models, including logistic regression and random forests.

2.2 Dataset partitioning

For the purposes of the following experiments, we limited ourselves to a random sample of 40000 flights from June 2015; we found this size a good compromise between computational tractability and sample representativeness. We partitioned the data points into training, validation, and test sets, with 80/10/10% of the data, respectively.

3 Logistic regression

To establish a baseline accuracy and to benchmark our future efforts, we began with a standard implementation of a logistic regression classifier with L_2 regularization. Recall that such a classifier attempts to minimize a loss of the form

$$J(w,b) = \frac{1}{2}w^T w + C\sum_i \log\left(1 + e^{-y^{(i)}(w^T x^{(i)}+b)}\right) \tag{2}$$

where the sum is taken over all training data points. The parameter C controls the amount of regularization; a smaller value of C produces greater regularization.

We performed a sweep over log-uniformly spaced C from 10^{-5} to 10^5 (with factors of 10 between each C). Optimizing for validation accuracy, we selected $C^* = 100$, which yielded a test accuracy of 0.661.

Unsurprisingly, replacing the first term in Equation 2 with an L_1 norm on w gives us the loss function for a classifier with L_1 regularization. Performing a sweep over C as we did previously and optimizing for validation accuracy again, we found an almost identical test accuracy of 0.660.

We present the validation accuracies under both regularization schemes for the values of C we tested in Figure 1.

4 Multilayer perceptrons

We next attempted to improve upon our previous results with multilayer perceptrons: vanilla neural networks. With an existing implementation, we minimized mean cross-entropy loss with the Adam optimizer, computing gradients by backpropagation. Recall that cross-entropy loss for a single data point $x^{(i)}, y^{(i)}$ is given by

$$J^{(i)}(w) = -y^{(i)} \log \hat{y}^{(i)} - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \tag{3}$$

where $y^{(i)}$ is a binary training label (either 0 or 1) and where $\hat{y}^{(i)}$ is the probability estimate from the network.

We also imposed L_2 regularization on the network weights, governed by regularization parameter α, such that the training loss function is given by

$$J(w) = \frac{1}{2}\alpha w^T w + \sum_i J^{(i)}(w). \tag{4}$$

We experimented with various architectures, activations, and α. Given the large space of possible hyperparameter combinations, it was infeasible to find a best set of hyperparameters exhaustively. As such, we explored the effects of each hyperparameter by holding values of the others fixed.

4.1 Architecture

In experimenting with network architecture, we fixed $\alpha = 10^{-5}$. We began by exploring different numbers of hidden layers, finding that deeper networks provided no significant improvement over single-hidden-layer networks. Thus, we restricted our attention to networks with just a single hidden layer.

We show our findings in Figure 2 for three different activation functions on the hidden nodes; it's apparent that varying the number of hidden nodes did not provide any substantive improvement in our results over our baseline.

4.2 Regularization

To see the effects of regularization, we considered networks with logistic, tanh, and ReLU activations on a single hidden layer of 25 nodes. As we did for the parameter C in section 3, we performed a sweep over log-uniformly spaced α from 10^{-5} to 10^5. We present our results in Figure 2. We see marginally better performance by the ReLU model with $\alpha = 10^{-3}$. With a test accuracy of 0.647, however, this model did not improve upon our logistic regression baseline.

5 Random forest classification

We next turned to ensemble methods to improve upon our previous results. We began by experimenting with existing implementations of random forest classifiers. In particular, we explored the effects of varying hyperparameters such as

- bootstrapping
- number of decision trees
- impurity threshold
- maximum tree depth
- feature masking
- warm starting

In the following sections, we explore the effects of varying hyperparameters, by tweaking the value of one at a time, holding all else equal to the following defaults: bootstrapped samples, 10 estimators, a maximum leaf-node impurity threshold of 10^{-7}, and no maximum

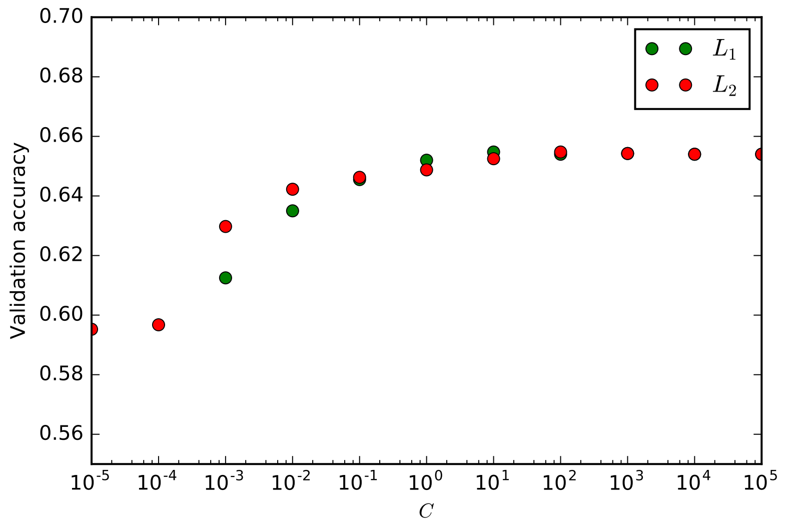


Figure 1: Logistic regression validation accuracies under L_1 and L_2 regularization for varying values of regularization parameter C.

tree depth, feature masking, or warm start. Trees were trained under Gini impurity. The results of these experiments are presented graphically in Figure 3.

5.1 Bootstrapping

Recall that bootstrapping involves training estimators on subsets of the training data sampled *with* replacement from the dataset, emulating sampling from the general population. When we sampled without replacement instead (i.e. no bootstrapping), training accuracy increased, but validation accuracy decreased, potentially due to overfitting on more sample points during training.

method	bootstrap	no bootstrap
training acc	0.976	0.999
validation acc	0.653	0.619

5.2 Number of estimators

A key hyperparameter in random forests, as in all ensemble methods, is the number of estimators, which impacts overfitting and prediction accuracy. Increasing the number of trees improved both training and validation accuracy. However, since training more estimators is computationally intensive, we did not train more than 32 trees. Our results are as follows and in Figure 3.

n trees	2	8	16	32
training acc	0.974	0.967	0.989	0.997
validation acc	0.635	0.643	0.651	0.658

We see that there is slight, but nontrivial, improvement in validation accuracies as our ensemble grows.

5.3 Maximum tree depth

By default, the depths of the individual decision trees are unconstrained, so that the model training runs until all nodes are expanded into valid leaves. We experimented with imposing a maximum depth as a form of early stopping. Unsurprisingly, a maximum depth generally decreased training accuracy, but increased validation accuracy in some cases.

max depth	4	8	16	32	256
training acc.	0.657	0.679	0.750	0.899	0.976
validation acc.	0.644	0.656	0.667	0.654	0.647

Indeed, a maximum tree depth effectively reduces overfitting to training data by regularizing the complexity of individual decision trees.

5.4 Masking features

When building each tree, we can ignore a subset of features when looking for the best split. Feature masking did not significantly affect training or validation accuracy. This may be due to the sparsity of our data, a consequence of our use of one-hot encodings.

5.5 Early stopping

If we do not constrain the depth of the tree, an alternative regularization appraoch is to increase the minimum tolerable Gini impurity at each leaf. However, changing this value had little effect, and was not as effective as limiting tree depth.

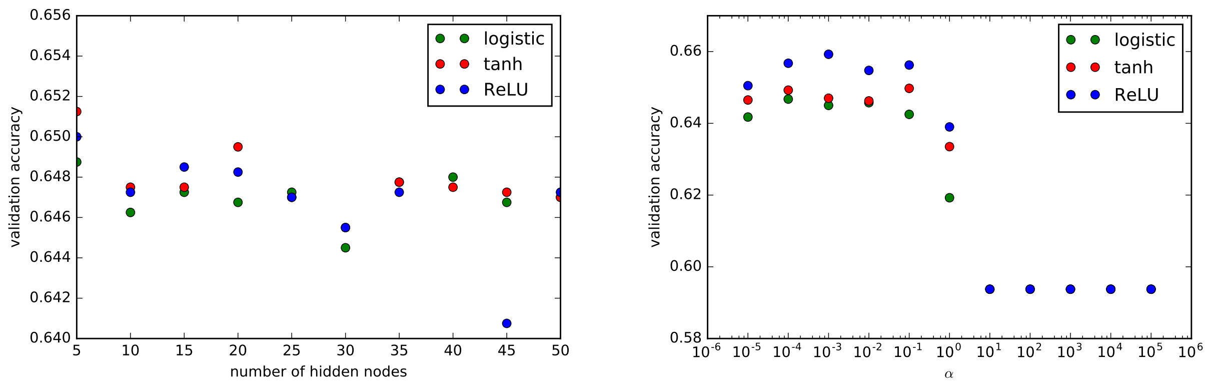


Figure 2: Single-layer multilayer perceptron validation accuracies. The plot on the left shows the results of varying the number of hidden nodes while fixing $\alpha = 10^{-5}$. On the right, we fixed the size of the hidden layer at 25 nodes and varied α.

5.6 Warm start

For each estimator, we can either train a new tree from scratch (default), or use the results of the previous run as a starting point (warm start). Preliminary experiments showed that using a warm start had little effect. We did not explore usage of warm starts further.

5.7 Optimal model

Based on validation accuracy, we selected the best model of 32 estimators, a max depth of 16, and default values for all other parameters. We obtained a testing accuracy of 0.687, a slight, but significant, improvement on our logistic regression baseline.

6 Boosting

We tried various boosting methods using existing libraries and tested different weak learners as the base estimator, which we discuss below.

6.1 Adaboost-SAMME

We also applied a variant of the Adaboost algorithm, Adaboost-SAMME (stagewise additive modeling with a multi-class exponential loss function).

With an ensemble of 50 decision tree classifiers, the Adaboost algorithm produced a training accuracy of 0.669 and a testing accuracy of 0.668. Unlike with our previous models, training accuracy is not drastically greater than testing accuracy.

When we increase the number of estimators, we notice that both training and testing accuracy improve. However, Adaboost does not overfit as much as other methods we have explored. In fact, it performed comparably as well on testing data as on training data.

n estimators	4	8	16	32	128
training acc.	0.649	0.655	0.662	0.668	0.673
validation acc.	0.654	0.661	0.667	0.672	0.670

We also manipulated the learning rate ν, a regularization parameter which exponentially weights the predictions from earlier stages of boosting. Whereas a standard forward-stagewise additive model would take the form

$$f = \sum_{i=1}^{m} \gamma_i f_i \tag{5}$$

for coefficients γ_i on ensemble members f_i, regularizing with ν gives

$$f = \sum_{i=1}^{m} \nu^{m-i} \gamma_i f_i. \tag{6}$$

Note that setting $\nu = 1$ recovers the unregularized model.

Fixing an ensemble size of 32, we found that decreasing learning rate worsened both training and validation accuracies. The results are graphically displayed in Figure 4.

learning rate	0.2	0.4	0.6	0.8	1.0
training acc.	0.654	0.663	0.665	0.667	0.668
validation acc.	0.661	0.668	0.671	0.672	0.672

We do not further consider regularizing by learning rate.

6.2 Boosted logistic regression

We tried an ensemble of 50 logistic regression models with Adaboost. It attained a training accuracy of 0.657 and a testing accuracy of 0.661. When we increased to 100 estimators, the training accuracy only reached 0.660.

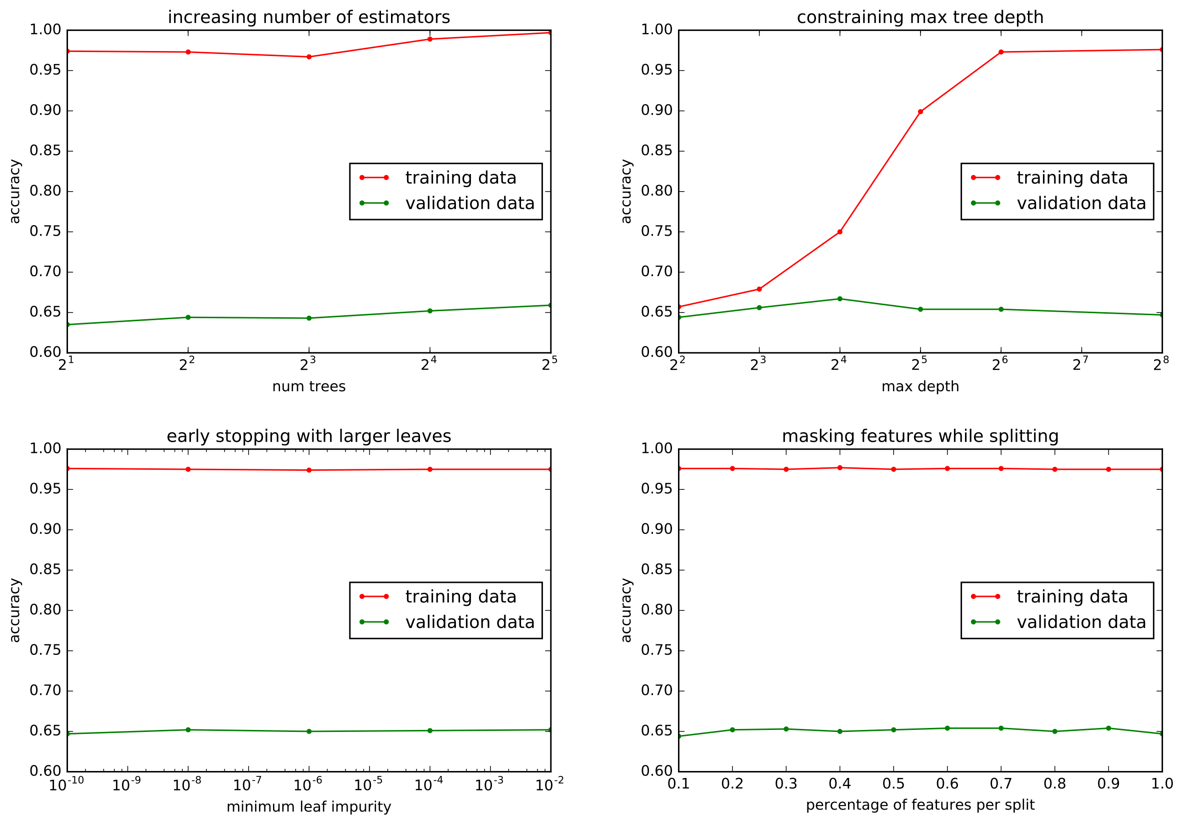


Figure 3: Varying parameters for random forest classifier. Setting a minimum Gini impurity on leaves and masking features do not significantly affect validation or testing accuracy.

The resulting accuracy of 0.662 was lower than that of boosted decision trees, so it seems that logistic regression is not as good a weak learner for boosting.

7 Extensions and variations

In addition to the cyclic data mapping, we also explored other feature mappings. To see their effects, we applied the best models from the previous sections on these new features. We also explored the effect of using larger training datasets on model accuracies.

7.1 Geographic relations within data

So far, we have treated the airport feature as a categorical variable, discarding any geospatial information it carried. In particular, by using a one-hot encoding of airport information, we imposed a discrete topology on the airports. That is, the "distance" between any two airports (as one-hot vectors) is the same.

However, geographical information is very likely to be helpful in making predictions. For example, it is conceivable that airline companies would work harder to ensure punctual service on profitable long-distance flights. Therefore, we added 4 additional features to the data representing the latitude and longitude of both origin and destination airports.

With these adjustments, our best random forest model, training accuracy decreased to 0.811, but testing accuracy remained about the same, at 0.688.

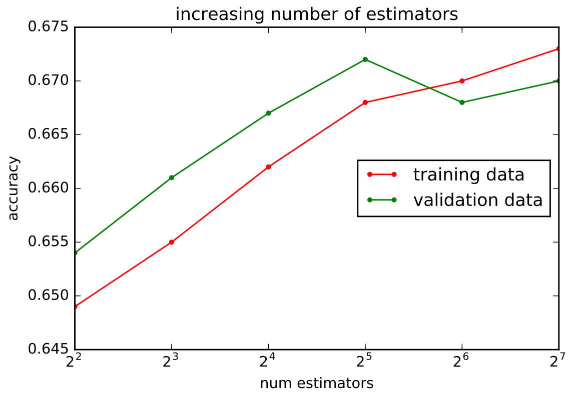
In addition, we tried representing origin and destination only with latitude longitude coordinates; that is, we did not represent airports with one-hot vectors, greatly reducing the number of features, from 469 to 7.

Note that doing so emphasizes the implicit assumption made by geographic encoding of airports that close airports are similar, which may not at all be the case. For example, there is no reason to believe that the three major airports in the New York metropolitan area, John F. Kennedy, LaGuardia, and Newark, have similar contributions to the probabilities of their flights being late.

Still, removing one-hot encodings reduced the computational load of training our models. Below, we present accuracies of the best models in each model class (as determined by validation accuracy)

model	LR	RF	AdaB
training acc.	0.604	0.840	0.650
testing acc.	0.602	0.673	0.635

The fewer features significantly reduced the accura-



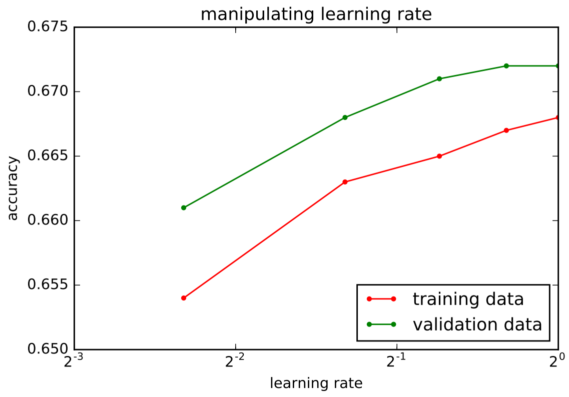


Figure 4: Varying parameters for the Adaboost-SAMME classifier, with a decision tree weak learner. Increasing number of estimators accuracy, but decreasing learning rate did not improve results.

cies of both the logistic regression model and the Adaboost ensemble model. Interestingly, the optimal random forest model did not suffer as much of a loss in accuracy, which makes sense intuitively.

Indeed, decision trees effectively discretize feature space by considering splits. Therefore, removing our discrete features (the one-hot vectors) did not actually remove much information for the random forest models.

7.2 Geographic displacement

In addition to absolute geographic locations, we also considered the displacement for each flight. In particular, we included features for the *change* in latitude and longitude over the course of the flight.

When we consider all the previous features, including discrete airports as one-hot vectors, the best models attain the following accuracies.

model	LR	RF	AdaB
training acc.	0.676	0.816	0.669
testing acc.	0.667	0.684	0.674

Interestingly, for the random forest model, our features were weighted approximately according to what we would intuitively expect. Each "feature" as part of some one-hot vector was individually weighted less than the date or time features; latitude, longitude, and displacement were similarly heavily weighted.

feature	weight
date	0.260
time	0.131
airline	0.095
airport one-hots	0.286
origin (lat long)	0.112
destination (lat long)	0.105
displacement	0.046

In particular, date contributes a lot to the final delay estimation.

With only latitude, longitude, and displacement data for airports (no one-hot vectors), our best models attained the following accuracies.

model	LR	RF	AdaB
training acc.	0.656	0.846	0.655
testing acc.	0.650	0.684	0.661

While these accuracies are an improvement on using solely position data of origin and destination, While these accuracies still do not reach that of representing airports as one-hot vectors, they are a significant improvement from using only position data of origin and destination.

feature	weight
date	0.280
time	0.270
airline	0.055
origin (lat long)	0.161
destination (lat long)	0.156
displacement	0.078

If we analyze relative feature weights, we see that displacement still does not contribute very much, but time of day's weight significantly increased. Thus, flight delay may be somewhat heavily related to what time of day a flight occurs at.

7.3 Additional temporal features

We also expect that flight delays are correlated with the day of the week; indeed, we imagine airports are busier on weekends, and thus weekend flights are more susceptible to delays. Therefore, we encode the day of the week as a cyclic feature (see section 2.1). We also encode the day of the *month* in the same way, since it is conceivable that delays might depend on time of month. With these additional cyclic features, our testing accuracies did not improve much from our previous best model, as shown below.

model	LR	RF	AdaB
training acc.	0.676	0.817	0.660
testing acc.	0.670	0.684	0.662

7.4 Larger datasets

While it was computationally infeasible to perform all our previous experiments on a larger training dataset, we explored the effects of having more training data by evaluating some of the previous models on a larger sample of 10^5 flights, still from June 2015.

The larger dataset produced general accuracy improvements. The logistic regression baseline improved from 65% to 67%; all other models we tested also showed a similar accuracy improvement on the order of $1-2\%$. Below we provide the testing accuracies using the best models (determined by validation).

	LR	RF	AdaB
n=40000.	0.649	0.659	0.648
n=100000	0.672	0.684	0.662

This improvement is expected, since large datasets are more tolerant of overfitting and more representative of the population data.

8 Conclusions

In the end, we found that our optimal random forest model trained on a larger dataset gave an accuracy of about 70%, only a few percentage points higher than the baseline accuracy produced by logistic regression.

Indeed, as we found, much of the work in applying machine learning techniques to a new problem "in the wild" involves cleaning data extracting meaningful features, and applying and encoding domain knowledge into our representation of the data (for example, our cyclic feature mapping).

Notably, more expressive models did not produce meaningful improvements in accuracy over our most basic logistic regression baseline. The best "fancy" model (the random forest in section 5.7) performed only slightly better. The biggest accuracy gain throughout all our experiments came from the use of the cyclic feature mapping from section 2.1. Further improvements might be attained from increasing the size of our dataset, both in sample size and number of features. However, we have found that it is difficult to predict with high accuracy, whether or not a flight will be delayed, from basic information available to customers before the flight.

9 Divison of labor

Both group members divided the work equally. However, we each focused on different sections of the paper. M.W. focused on ensemble methods, including random forests and boosting, while T.Z. concentrated on logistic regression, multilayer perceptrons, and feature mapping.

References

[1] United States Bureau of Transportation Statistics, *Airline On-Time Statistics*, 2016, https://apps.bts.gov/xml/ontimesummarystatistics/src/index.xml.

[2] scikit-learn developers, *scikit-learn User Guide*, 2016, http://scikit-learn.org/stable/user_guide.html