Airline Flight Prediction using Machine Learning Algorithms (CS 4269 Spring 2019)

Max Engel Samer Bendhary Alexander Reed

Abstract

We created a flight delay prediction model using historical flight and weather data for flights in and out of Nashville, TN. This application will allow future passengers and airlines to better estimate their travel times and risks. We incorporated flight data from 2015 and weather data from the day of the flight to train and test several models constructed with various supervised machine learning algorithms.

1. Introduction

Air travel has become one of the preferred methods of transportation, particularly for long distance travel. In 2017, US airlines generated $222 billion in revenue (fli, 2017). Nearly every US business relies on air travel and is willing to pay a premium to ensure timely travel. However, flight delays can impede such timely travel, leading to flier frustration and potential losses in revenue. An accurate flight delay prediction model could change the way fliers book flights and airlines schedule and price flights.

While this model could be useful for passengers in some ways, it could give airlines further reason to charge more for flights. Additionally, it could also lead to some flights becoming too crowded or empty. The airlines would need to price their flights appropriately to manage these issues. A flight prediction application could greatly facilitate air travel and allow fliers and airlines to better hedge their timing risks.

2. Data

The flight information comes from Kaggle, which is an online repository of public data sets. The data set we used contains information about domestic flights operating large carriers in 2015. The data is collected by the U.S. Department of Transportation (DOT) Bureau of Transportation Statistics and published in the DOTs monthly Air Travel Consumer Report (fli, 2015).

The important features in the dataset are listed here:

Important Features
year
month
day
departure time
origin airport
destination airport
departure delay (minutes)
distance (miles)
arrival time
arrival delay (minutes)
cancelled status (yes or no)
cancellation reason
weather delay

The original dataset contains around 6 million instances, so we reduced it to only contain flights leaving from or arriving to the Nashville International Airport. This brought the dataset to about 50,000 entries. Then, we randomly selected 100 entries from each month to have a representative sample. We did this because of limitations in the number of API calls we could make for the additional data we were looking to add. We then split the dataset into training, validation and test. The training dataset received 80% of the total dataset and both validation and test each received an equal share of the remaining 20%. We further altered each of the training, validation and test sets to have an equal number of instances with the 'delayed' and 'on-time' target label. Kaggle's dataset is a government-provided log, and thus, we assume that every flight is logged and there is no sampling bias. However, it is important to note that the data is from 2015. This makes the data susceptible to the trends that existed in 2015. These trends might include weather patterns, flight scheduling protocols, airline technology, etc. After some research, there are not major fluctuations in quantity of delayed flights between 2015 and now that should impact our model in a significant way (del).

Weather is a large influencer on flight delays, as pilots are less likely to fly in poor weather conditions. So, we added weather information for the data and location from which the airplane departs and lands. The weather data comes from

https://www.worldweatheronline.com/developer/ API. The features added to each flight by making POST calls to the API given the data and location of the flight are listed below.

Features from Weather API
maximum temperature (F)
minimum temperature (F)
amount of snow (cm)
average wind speed (mph)
average visibility (miles)

The dataset has a natural distribution of about 80% 'on-time' labels and 20% 'delayed' labels. After several iterations using the natural distribution, we found that some models were limited in their development and often prematurely jumped to an 'on-time' classification. Thus, for the purposes of this project, we took the liberty to create a dataset that contains 50% on-time labels and 50% delayed labels with an even selection from the twelve months. We understand this alters the natural distribution, but we also found this 50/50 split gleaned greater insights into the comparisons of the various models.

It is important to note that we did not use any normalizing factor for location. Thus, features are all treated equally, independent of geographical location.

We defined accuracy as the ratio between correct guesses of 'on-time' or 'delayed' labels to total number of tested entries.

3. Decision Tree

We created decision trees as a first approach to our problem. Our program builds a tree where each node consists of either a feature or classifier. If every instance of the data reviewed contains the same classifier, we create a node in our tree representing that classifier. Else, we use an entropy gain function to determine which feature best divides the training data so each section of the divided data contains the highest proportions of the same classifier. After dividing upon this feature, we continue down each generated branch with the corresponding subset of the original data. We continue this division and checking process until the nodes of the decision tree are all classifiers. To avoid overfitting to our data, we ran a pruning algorithm that prunes each branch of the tree and tests the pruned tree against the validation set. We achieved our best results (depicted below) using the following features:

Best Features
airine
day of week
month
minimum temperature at destination (F)
minimum temperature at origin (F)
snowfall at destination (cm)
distance (miles)
visibility at origin (miles)
average wind speed at origin (mph)
scheduled departure time

Using the original, 80/20 distribution of labels, we generated the following results using our custom decision tree model:

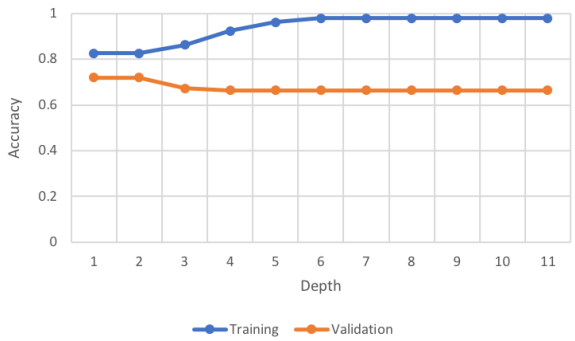

Figure 1. Accuracies of decision tree using 80/20 distribution over varying depths of decision tree

We see an expected increase in training accuracy and an expected decrease in validation accuracy as depth increases. This demonstrates the bias-variance trade-off. The random splitting of the validation set gave our validation set a 70/30 split, resulting in lower accuracies. Our tree appeared very lopsided, where it was eager to pick the 'on-time' label. We believe this is due to the uneven distribution of labels in the data. We ran the same tests on the 50/50 distribution and achieved the following results:

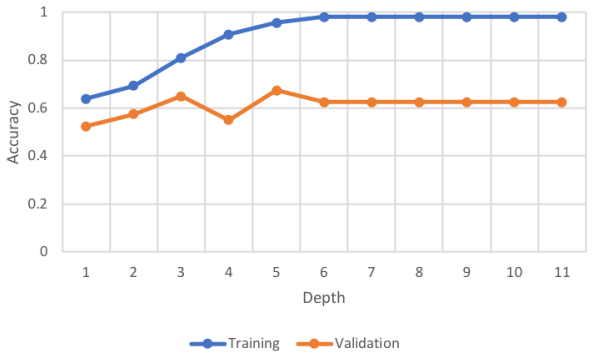

Figure 2. Accuracies of decision tree using 50/50 distribution over varying depths of decision tree

Here, we see an expected increase in training accuracy and an increase then decrease of validation error. This further demonstrates the bias-variance trade-off as training accuracy consistently increases, while validation accuracy increases, reaching its best representation of the validation data, then decreases as it demonstrates overfitting. Upon examining the structure of the tree, we found that the tree consistently first splits on the departure time, which we bucketed into 'morning', 'afternoon' and 'evening'.

4. Neural Network

We used the Keras python package to implement our neural network. First, we tried all subsequent tests on the true population with the 80/20 split. We found that regardless of the hyperparameters, our network performed with a success rate that hovered close to 96%. The following results are using the altered distribution with the 50/50 split.

Our program builds a neural network consisting of various numbers of hidden layers with varying numbers of nodes while also utilizing different numbers of epochs and batch sizes. After trial and error, we were able to generate an accuracy rate of 62.5% with 2 hidden layers of 16 and 8 neurons, respectively, a learning rate of .001, utilizing 150 epochs, and a batch size of 10 using the same features from the decision tree.

We first tried using two different error functions to test accuracy: mean squared error and binary cross-entropy. We found that, as we increased the number of neurons and layers in our network, binary cross-entropy began to outperform mean squared error. We found that this is likely due to nature of the classification problem (delayed or on-time), in which binary cross-entropy performs well.

Next, we tried different learning rates and found that smaller learning rates performed best. We have a sample of significant size, so the smaller learning rates were able to better approach the respective loss minimums.

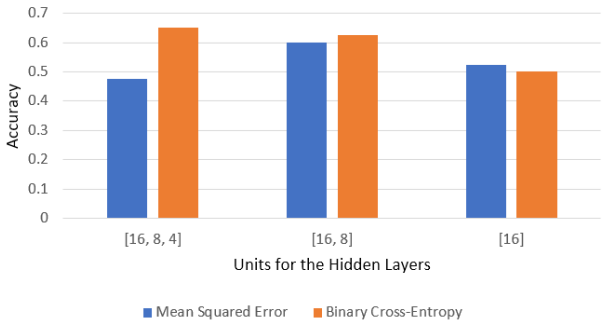

Figure 3. Accuracy of various error functions with differing neural networks

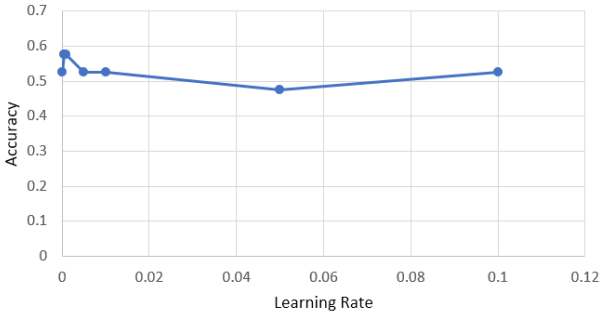

Figure 4. Accuracy of various learning rates

We then considered the number of layers in the neural network. We found that three layers, with one input layer, one output layer, and one hidden layer was ideal for the highest accuracy. The resulting decrease is likely due to overfitting. Figure 5 below shows how the best accuracy comes from using one hidden layer

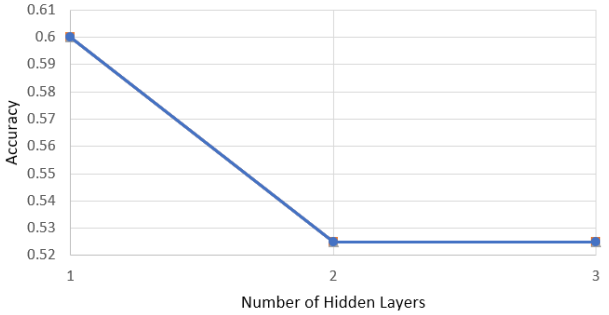

Figure 5. Accuracy of different numbers of hidden layers using 16 neurons

We also looked at the effects of changing the number of neurons in the hidden layer. Since using one hidden layer produced the best accuracy, we decided to test different numbers of neurons. Figure 6 below shows that as we increased the number of neurons, the accuracy also increases,

and then decreases after about 32 neurons.

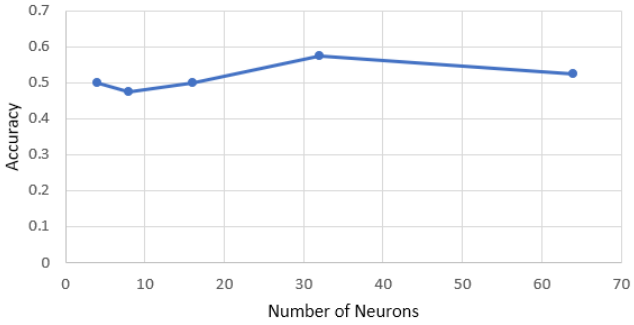

Figure 6. Accuracy of different quantities of neurons

However, combining the two does not produce the best results. We applied different combinations of hidden layers and number of neurons and found that using 3 hidden layers with 16, 8, and 4 neurons in each layer produced the best result. Figure 7 below demonstrates that this is comparatively the best combination of neurons and hidden layers.

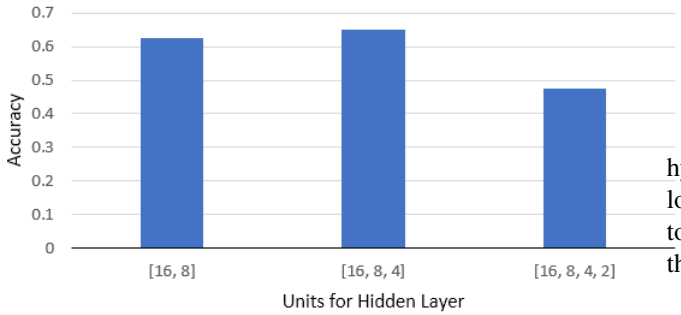

Figure 7. Accuracy of varying hidden layer sizes and designs

Next, we looked at how modifying the number of epochs changes the accuracy. Using a two hidden layer model with 16 and 8 neurons, a learning rate of .001, and a batch size of 30, we found that there is much fluctuation in accuracy. Figure 8 below shows that using 2 and 100 epochs produce the best results at .55 accuracy.

Lastly, we examined the impact of batch size on accuracy. Using the same hyper-parameters as before, but with 100 epochs, we see that using a batch size of 20 produces the optimal accuracy. Figure 9 below shows an upward trend and then a decrease after a batch size of 20. Using batch sizes of 20 produced the best generalization without overfitting the training set.

Ultimately, we found that with an even distribution of labels, our model was able to achieve over 50% accuracy. The network's hyperparameters played a significant role in achieving the 62% accuracy rate, as variations existed in all

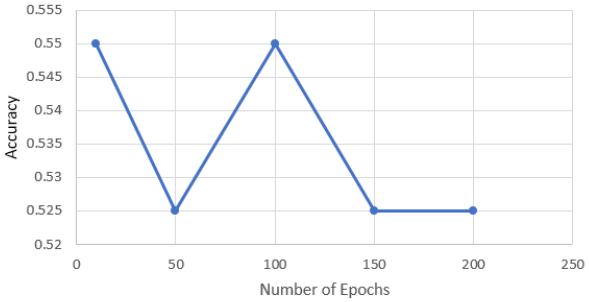

Figure 8. Accuracy of varying numbers of epochs

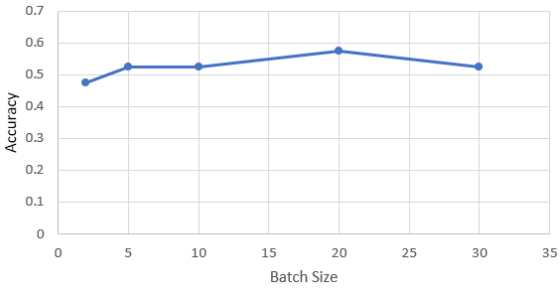

Figure 9. Accuracy of differing batch sizes

hyperparameter tests. Our relatively large sample size allowed for finer tuning of parameters but also susceptibility to overfitting. We had to engineer our network to mitigate the bias-variance tradeoff.

5. Naive Bayes

We next tested our algorithm with a Naive Bayes classifier. We used Scikit-Learn's Gaussian Naive Bayes classifier to construct our model. This model utilizes the following likelihood function:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

We first tested our model on the original dataset and the same features used previously which received 70.5% accuracy on the original dataset and 51.9% accuracy on the modified dataset.

Then, we wanted to see how changing list of features impacts the accuracy. Using only features in the first data set without weather information added, we achieved 81.4% accuracy on the original dataset and 65.5% accuracy on the modified dataset. The features are listed below.

Previously Used Features
airline
distance (miles)
day of the week
month
snowfall at origin (cm)
minimum temperature at origin (F)
minimum temperature at destination (F)
visibility at origin (miles)

Original Features
airline
distance (miles)
day of the week
month
scheduled departure

Features with Origin Airport Weather
airline
distance (miles)
day of the week
month
snowfall at origin (cm)
minimum temperature at origin (F)
minimum temperature at destination (F)
snowfall at origin (cm)
visibility at origin (miles)
average wind speed at origin (mph)

Features with Origin and Destination Airport Weather
airline
distance (miles)
day of the week
month
snowfall at origin (cm)
minimum temperature at origin (F)
minimum temperature at destination (F)
snowfall at origin (cm)
snowfall at destination (cm)
visibility at origin (miles)
visibility at destination (miles)
average wind speed at origin (mph)
average wind speed at destination (mph)

As previously mentioned, this dataset was highly predictive but did not demonstrate the effectiveness of the model as the proportion of labels was so unbalanced. Thus, we used our dataset containing an even distribution between delayed and on-time flights. Our results are detailed below. The validation set of the 80/20 dataset had a 70/30 distribution, leading to lower accuracies. It is important to note

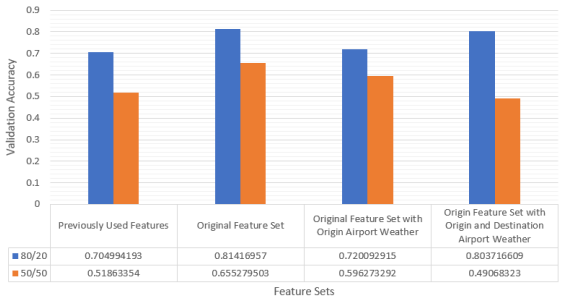

Figure 10. This figure depicts the accuracies of differing Naive Bayes models using 50/50 and 80/20 target label distributions and differing sets of features

Naive Bayes assumes all features are independent of each other. However, this assumption does not hold true with our dataset as there are high levels of correlation between features (i.e. temperature and snowfall). This could have resulted in lower levels of predictability than in previous models. Furthermore, the accuracy of the model in the 50/50 distribution goes down after adding weather features which are not independent of each other, which skews the probability distribution. Additionally, to use the Gaussian likelihood estimator, we took the liberty of assuming that our large sample would take on a Gaussian distribution.

6. K-Nearest Neighbors

We then decided to use the K-Nearest Neighbors algorithm to build a model using our training data. We used Scikit-Learn's K-Nearest Neighbors package to build our model. This package uses the Euclidean Distance Formula to compute the distance between two points. Since our target feature is binary we were sure to use an odd number for k in order to avoid a situation where the nearest neighbors are equally split between classes.

We tested several KNN models by varying the hyperparameter k (number of neighbors to compare against during classification) as well as by varying the dataset used. We used 1, 3, 7, 11, 15, 19, 23 and 51 as our values for k. The datasets represent delayed as a value of '1' and on-time as a value of '0'. All other feature values are continuous and are not grouped into buckets. Upon reviewing our results, we see differing behaviors between our models with the 80/20 split and the 50/50 split. Our 80/20 split achieves an 84 % accuracy rate with k = 15 - 51 and our 50/50 split achieves a 65% accuracy rate with k = 3, 7. These variations are relatively small and could be attributed to variations in the data we collected in the 50/50 split. It appears that the 50/50 split is better at finding the necessary indicators in the data with fewer neighbors. There will be greater indication of 'delayed or 'on_time' in

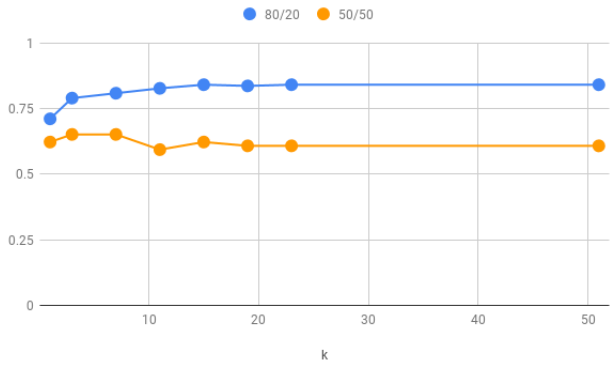

Figure 11. Accuracy of KNN model using varying k values and both datasets

a smaller proximity than in the 80/20 split where our model might have to search through a greater number of entries to find any indication of a delayed flight.

7. Conclusion

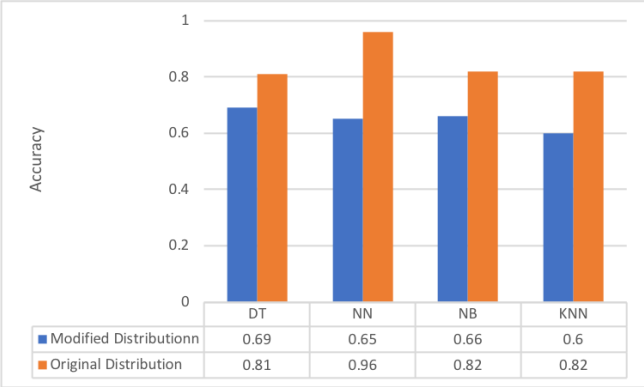

Figure 12. Final comparisons of models on both label distributions

We found that the neural network works best on the original distribution while the decision tree works best on the modified distribution. We believe the neural network was able to perform best on the 80/20 distribution because it was best able to detect and weight which features were most predictive of the 'delayed' or 'on_time' labels. Additionally, the iterative nature of the neural network was able to fine-tune its weights and biases, allowing it to be less affected by potentially faulty data that could be misleading. Thus, the neural network can more easily pick up nuances in the 80/20 distribution that are not apparent in the 50/50 distribution. It is important to note that 'black-box' nature of a neural network makes it difficult to define its concrete strengths and weaknesses in this selection process.

We believe the decision tree worked best on the modified data because of the predictiveness of departure time. We found that flights departing in the afternoon are more likely to be delayed than those in the morning. Our labels are binary so the baseline accuracy with our equal distribution is 0.5 and our model only had to add an accuracy of 0.19 to achieve 0.69 accuracy. The decision tree was able to isolate this feature when predicting, and potentially remove noise from other features, to make its decisions.

Ultimately, we believe flight delay data is hard to use in a predictive manner. In industry, there does not exist a reliable flight-delay prediction application. There are numerous other factors that we did not include, and are not open-sourced, that could affect a flight's timeliness. We hope that airlines and airports will open-source this data, as we believe the next step of development of this model would be to included a greater quantity and range of data. We would need to obtain more, likely confidential, information. This includes, but is not limited to: age and use of aircraft, airport traffic on departure day, organization and preparation of crew and mechanics, etc. We enjoyed working on this project and hope to continue working once more data is available.

References

On-time performance - flight delays at a glance. data retrieved from Bureau of Transportation Statistics, https://www.transtats.bts.gov/homedrillchart.asp.

2015 flight delays and cancellations, 2015. URL https://www.kaggle.com/usdot/flight-delays.

Total operating revenue streams of u.s. airlines from 2004 to 2017 (in billion u.s. dollars), 2017. URL https://www.statista.com/statistics/.