Predicting Flight Delays

Dieterich Lawson - jdlawson@stanford.edu
William Castillo - will.castillo@stanford.edu

Introduction

Every year approximately 20% of airline flights are delayed or cancelled, costing travellers over 20 billion dollars in lost time and money. Many factors affect flight delays including air traffic control backups, equipment delays, and weather. Our goal was to leverage the massive amount of data available on flight punctuality and weather to forecast whether or not a flight will be delayed.

Data

We used two main sources of data for the this project: The Bureau of Transportation Statistics' (BTS) Airline On-Time Performance dataset[1] and National Oceanic and Atmospheric Administration's (NOAA) ISD-Lite dataset[2]. The BTS dataset contains an exhaustive listing of flights to and from in the US since 1987 and includes features such as departure date, airline carrier, origin airport, number of minutes delayed, and more. The NOAA ISD-Lite dataset is made up of weather observations collected via NOAA's network of weather stations going back to 1900. The ISD-Lite contains features such as millimeters of precipitation in the last hour, wind speed, sky coverage, and more.

To combine these two datasets we performed multiple transformations on the data. The BTS dataset uses only local timestamps while the NOAA dataset uses UTC timestamps. Additionally, the NOAA dataset uses an identifier called a 'WBAN' to identify where the weather observation was recorded, but the BTS data uses its own airport identifier called 'Airport Sequence ID'. These inconsistencies had to be resolved before the datasets could be joined.

The final dataset was 17.7 GB and contained 135 million flights. It included weather data at both the origin and destination airport as well as the flight information, 33 features in all. We set up a 20-machine cluster on Amazon's EC2 to perform these transformations and run our machine learning algorithms.

Methods

Given a single flight, we attempted to predict whether or not it would be delayed, i.e. we performed binary classification on the flight. We used several different classifiers including SVMs, Naive Bayes, and Random Forests. In general, we tried to choose algorithms that

[1] http://www.transtats.bts.gov/TableInfo.asp?Table_ID=236&DB_Short_Name=On-Time&Info_Only=0
[2] http://www1.ncdc.noaa.gov/pub/data/noaa/isd-lite/isd-lite-technical-document.pdf

parallelize well, so that we could run them on top of Apache Hadoop and take full advantage of the large size of our dataset.

Support Vector Machines

We started by trying different approaches on a smaller dataset to find the best performing classifier. Some of our initial attempts included SVMs. When testing with SVMs, we selected one airport, San Francisco International, and limited the scope to one year of flight data so that we could experiment locally with a reasonably sized dataset instead of on the cluster. We used the SVM library in MATLAB to see how SVMs performed on the data and to determine if we should scale this approach to the cluster. When exploring SVMs, we tried various kernels including Gaussian, linear, and quadratic, but ultimately did not see a reason to continue training SVMs and instead focused on other methods.

Naive Bayes

To classify flights using Naive Bayes, we implemented our own distributed version of Naive Bayes on top of Apache Hadoop. Before training on the data, we ran discretization scripts that we created to turn real-valued features like 'estimated flight time' into categorical variables. We then split our dataset into 10 pieces and performed 10-fold cross validation. We also tried training on only the flights from specific airports, and performed 10-fold cross validation on that as well. Because Naive Bayes classifiers train relatively quickly, this was the only classifier that we were able to train on the entire 135 million row dataset.

Random Forests

Given the size of our dataset, we wanted to use classifiers that could be implemented and run on our cluster. One easily parallelized classifier is the random forest, which is made up of an ensemble of decision trees. We used MATLAB's random forest library when testing locally, and Apache Mahout's random forest implementation when training on the cluster.

Each decision tree was trained on ⅔ of the data, called the 'bag', and tested on the the other ⅓, called the out-of-bag data. The number of testing errors from each tree is then summed up and divided by the number of trees we specified, giving a metric known as the 'Out-of-bag error' (OOB). We used 100 trees for the majority of our analysis, but we also explored the effect of increasing the size of the forests. Specifically we tested 100, 250, 500, and 1000 trees in an attempt to maximize precision and recall.

As is standard for random forests, we used a random subset of features to train each decision tree, and varied the subset with each tree. After initial local testing to ensure the Random Forest approach was working, we investigated the features we had chosen in an attempt to simplify the trees and improve precision and recall. Using the OOB error, we measured the effectiveness of our classifier and determined the most influential features for successful prediction.

SVM Results

The results we achieved with SVMs were not promising. Our SVM achieved 14.4% error rate, but its precision and recall were extremely low (19.4 and 12.6 percent, respectively) because it predicted that almost all flights were on time. Because only 20% if flights are delayed, this appears to be a working classifier if you only consider the error rate. On some smaller data sets we saw close to 90% accuracy but this was not consistent. Because of this, we ended up abandoning SVMs for more promising methods.

Naive Bayes Results

Our Naive Bayes classifier performed surprisingly well - achieving error rates between 15 and 18 percent. However, closer investigation of the classifiers showed that they were also mostly predicting that flights would not be delayed. The precision was also fairly high - around 75% - but the recall varied widely, from 37.32% on Newark International flights to 7.6% on the entire dataset.

Dataset	Error Rate (%)	Precision (%)	Recall (%)
Newark Liberty International	15.00	76.4	37.32
O'Hare International	17.97	72.63	22.88
All Flights	15.62	70.46	7.6

Figure 1 - Naive Bayes results

Random Forests Results

Random forests were the best performing classifiers that we tried. Their error rates hovered around 84%, but their precision was as high as 90% and their recall was as high as 44%.

Dataset	Error Rate (%)	Precision (%)	Recall (%)
Newark Liberty International	16.44	86.44	22.37
O'Hare International	17.60	84.72	19.80

Miami International	17.00	90.94	5.11
Random Subset	15.41	76.40	37.32

Figure 2 - Random Forest Results

We experimented with different numbers of trees on data from Newark International Airport in an attempt to optimize precision and recall, as shown in the precision-recall curve plotted in figure 3.

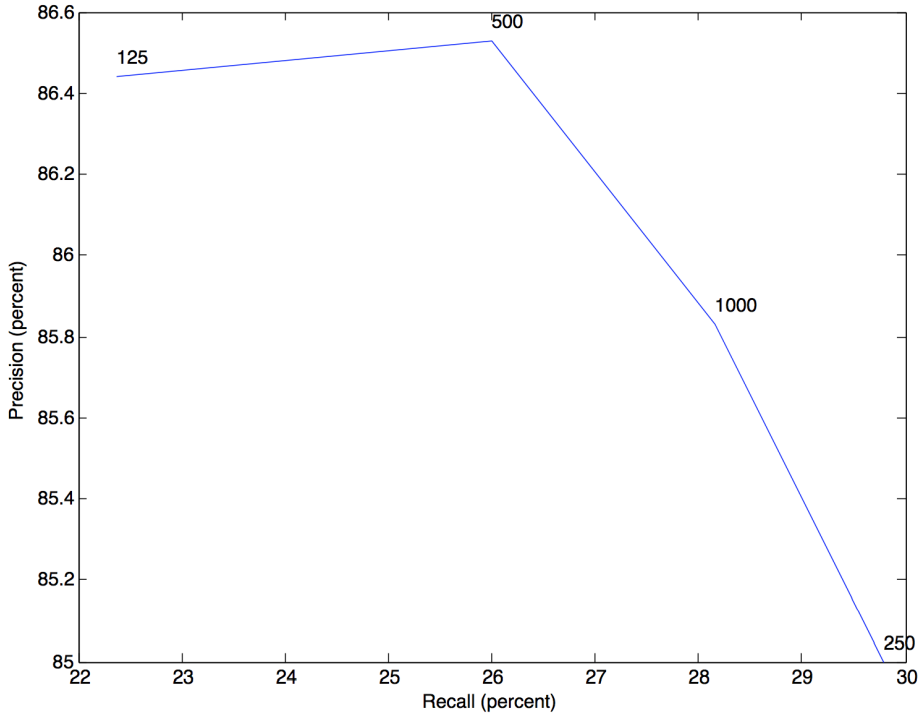

Figure 3 - Precision vs Recall for Number of Trees

The graph shows a seemingly steep tradeoff between precision and recall, but gains of a few percent in recall can be had for fractions of a percent loss in precision. It also shows that more trees isn't always better - 250 trees performed the best of all the forests that we trained.

Feature Selection

Using our Random Forests results from the data we were able to gain insight into which of our features was most useful for predicting the output. The main method we used to determine feature importance is most commonly known as permutation importance [3]. In this method we consider the examples that were not used to create a particular decision tree in the forest and permute one feature at a time. We then classify all the permuted examples, yielding an error rate,

and subtract the original error of the data before the permutation. Combining the results of this for every tree in the forest and dividing by the number of tree yields a metric for feature importance.

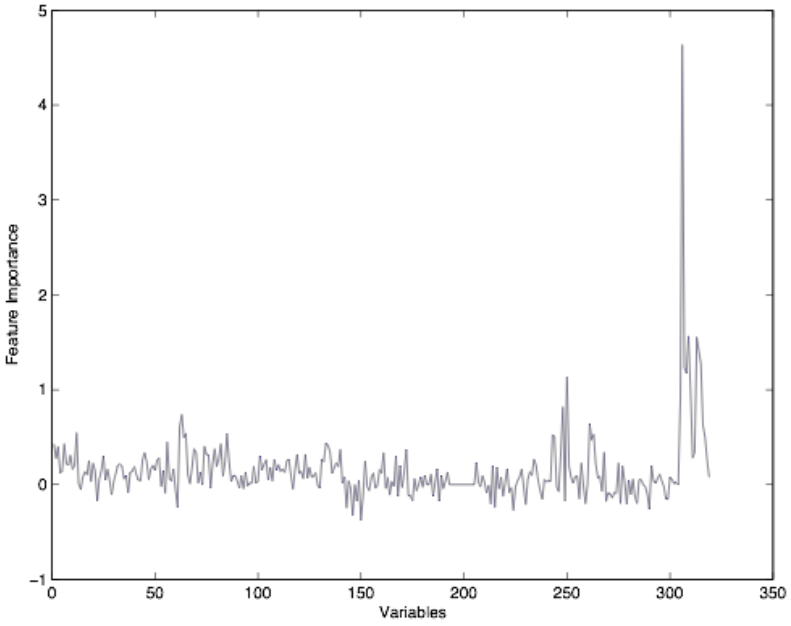

Figure 4 - Random Forest Variable Importance

The results of this process showed us that the most important variables were the previous flight arrival delay, which is represented by the huge spike to the right of the graph, followed by origin airport precipitation (small spike to the right) and departure hour (small spike to the left).

Conclusions: Precision vs. Recall

At face value, a 12-15% error rate on the test set is good, but is potentially deceiving. As relatively few flights are delayed, a classifier that simply predicts that no flights will be delayed could achieve similar error rates. Thus, our main challenge was to instead improve recall, the percentage of delayed flights that we correctly classified as delayed. Companies in industry that work in flight delay prediction report similar numbers for testing accuracy and precision, but their recall is higher - roughly around 60% [4]. These companies also cite improving recall as their biggest concern.

Because most of our approaches yielded similar precisions and recalls, we believe that improvements in these areas would only come with different data, i.e. more features, further data processing, or better domain knowledge, instead of better machine learning algorithms.

References

[1] Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.
[2] Breiman, L. (1996). *Out-of-bag estimation* (pp. 1-13). Technical report, Statistics Department, University of California Berkeley, Berkeley CA 94708, 1996b. 33, 34.
[3] Genuer, R., Poggi, J. M., & Tuleau-Malot, C. (2010). Variable selection using random forests. *Pattern Recognition Letters*, *31*(14), 2225-2236.

[4] http://www.datawrangling.com/how-flightcaster-squeezes-predictions-from-flight-data