Predicting flight on-time performance

Arjun Mathur, Aaron Nagao, Kenny Ng

I. INTRODUCTION

Time is money, and delayed flights are a frequent cause of frustration for both travellers and airline companies. After just a few minutes delay, the consequences range from economic (missed connections and cancellations) to environmental (wasted fuel) and social (loss of productivity and airport congestion). Thus, we used flight arrival and departure data to classify whether a flight would be on-time, so that airlines can learn which features predict delays and work to mitigate them.

II. DATA COLLECTION AND FEATURIZATION

Our dataset consists of flight information sent by major airlines to the U.S. Department of Transportation, and collated for the American Statistical Association's 2009 Data Expo [1]. For delayed flights, airlines report what caused their delays, summarized in the table below [2]:

Aircraft arriving late	36.55%
Air carrier delay	27.76%
Weather	20.15%
Volume	6.22%

Our dataset contains features that are related to these causes, including previous flight information, air carrier (airline), and origin/destination airport, so we expect it to be predictive. We supplemented our dataset with airplane age information and weather information obtained from scraping Weather Underground.

To obtain the airplane age, we looked up the plane tail number for each flight in the Federal Aviation Administration (FAA) database using a simple GET request. To mine the weather data, we first filtered the training and test examples for all unique dates and airport locations. Then, we used the requests and BeautifulSoup Python packages to download the weather data for each unique date at each airport and parse the data to CSV format. And finally, leveraging Python's shelve library and the pytz package, we matched each flight example to weather data recorded within half an hour of the flight's scheduled departure time and created the following new features: temperature (Fahrenheit), visibility (miles), wind speed (MPH), precipitation level (binary indicating whether any precipitation presently), and weather

conditions (categorical). Weather conditions (such as freezing rain , thunderstorms, and patches of fog) were ranked on a 0-9 scale depending on its impact on flight delays. An example of the inutition is as follows: thunderstorms have a lower rank than freezing rain since freezing of any type has direct impact on the runways (which leads to delays) whereas thunderstorms have no effect on takeoff and do not affect the plane upon its ascent above the trophoshere.

The carrier and origin/destination airport were categorical, so we generated binary features for all 20 airlines, the 30 highest-traffic origin airports, and 30 highest-traffic destination airports. 36% of flights did not depart from these popular airports, so these examples had 0 values for their binary features. Some algorithms (e.g. SVM with Gaussian kernel) required the data to first be standardized with mean 0 and variance 1, while others (e.g. random forest) did not.

The dataset is relatively large, so we randomly sampled 1000 training examples and 100 testing examples to use from each month in 2008 (for a training set of 12000 examples and a test set of 1200 examples). Plotting bias/variance learning curves confirmed that adding more training examples would not improve accuracy.

For our project milestone, we had defined a flight to be delayed if it had any positive delay. However, because our prediction accuracy was low, we instead use a different definition: a flight is delayed only if the flight is >15 minutes delayed, which is the official cutoff defined by the FAA. Notably, by the >0 minute criteria, 50% of flights were delayed; by the >15 minute criteria, only 25% are delayed.

III. METHODS AND RESULTS

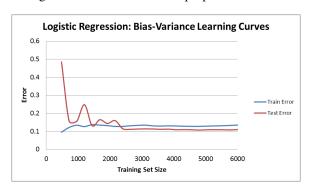
A. Logistic Regression

For initial testing, the matlab glmfit library was leveraged to create a logistic regression model. A Bayesian regularized logistic regression model with Newton's method optimization was also created. Cross validation was used to optimize parameters for the regularized MAP estimate. The following are our results:

1

Logistic	Accuracy	Precision	Recall	F_1
Regression				score
Maximum	.8933	.9694	.4318	.5975
Likelihood				
Estimation				
MAP	.8942	.9794	.4318	.5994
Regularized,				
$\lambda = 2.2$				

The following bias-variance curve was plot to ensure convergence and check for error properties.



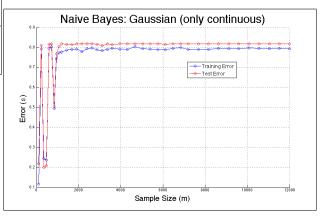
The plot indicates convergence after ~ 2500 training examples. Furthermore there is low variance. Logistic regression achieved the best F_1 score of 0.5994.

B. Naive Bayes

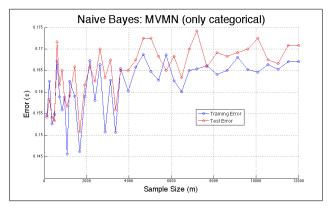
For classification with Naive Bayes, we used the supervised learning methods from MATLAB 's Statistics Toolbox.

Initially, in our Naive Bayes classifier, we attempted to fit a Gaussian distribution to all the features. However, for some training samples (mainly small sample sizes), the variance of certain features was zero (which resulted in a degenerate normal fit). This signaled that some of our features may not necessarily follow a normal distribution. Specifically, we realized that 14 of the features are continuous whereas the other 89 features (created from the non-numerical data for weather conditions, airline carrier, airport origin, airport destination, and previous flight delay) are categorical. Then, in order to create a Naive Bayes model that takes into account both continuous and categorical data, we considered the following methods: (1) independently fit a Gaussian Naive Bayes model on only the continuous part of the data and fit a multivariate multinomial Naive Bayes model on only the categorical part of the data; transform the entire dataset by taking the class assignment probabilities as new features; and fit a new Gaussian Naive Bayes model on these new feature, or (2) transform all continuous features into a categorical representation by binning. At first, we attempted the former method by using different

distributions to independently fit the data. However, we discovered vastly different results from the two different distributions.



From the learning curve of a Naive Bayes model using a Gaussian distribution, we see that the training error is unacceptably high and there is a small gap between training and testing error, which is indicative of high bias.

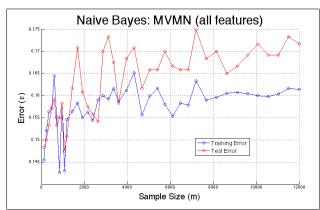


However, from the learning curve of a Naive Bayes model using a multivariate multinomial distribution, we see that there is high variance (or overfitting).

Since high bias can be remedied by introducing a larger set of more predictive features, this further tells us that either the continuous features are relatively insignificant or that these features do not follow a normal distribution. Thus, we opted to follow the latter method and transform all the continuous features to categorical representations.

To transform from continuous to categorical, we computed percentiles for each continuous feature. Then, we binned the continuous features using the percentiles as bin boundaries and domain knowledge (and intution) about flight delays. For example, scheduled departure and arrival time was binned into early (1) and late (0) flights and visibility was binned into no visibility (2), low visibility (1), and okay visibility(0) (since anything

above "low visibility" is deemed okay to fly by the National Weather Advisory). After transforming all the features into categorical representations, we plot the following bias-variance learning curve:



Our results from the various distributions and features experimented with for Naive Bayes classification is summarized below:

Naive Bayes	Accuracy	Precision	Recall	F_1
				score
Gaussian	0.183	0.001	0.949	0.002
Distribution (only				
continuous				
features)				
Multivariate	0.829	0.908	0.879	0.893
Multinomial				
(only categorical				
features)				
Multivariate	0.828	0.918	0.872	0.894
Multinomial (all				
features)				

From our results, we have that the multivariate multinomial Naive Bayes model (both subset and entire set of features) performed the best (and about equal), achieving an 83% accuracy and a F_1 score of ~0.89, indicating both high precision and high recall.

C. SVM

For the SVM, we used the Python library scikit-learn, which wraps liblinear and libsvm.

Initially we used a Gaussian (rbf) kernel: $K(x,z) = \exp\{-||x-z||^2\}$. To optimize the Gaussian kernel parameter γ and the SVM regularization constant C, we used exhaustive grid search with exponential grid spacing. Each (γ,C) pair was evaluated using stratified 10-fold cross-validation, where "stratified" means that each fold contained the same proportion of late and non-late examples as the complete set. We found that higher values of C worked best, which means that the SVM aimed to classify all training examples correctly

(especially delayed examples), at the expense of having a more complex decision boundary.

SVM	Accuracy	Precision	Recall	F_1
				score
Gaussian:	0.8883	0.97	0.40	0.566
$\gamma = 0.0001, C =$				
1000				
Gaussian:	0.8092	0.48	0.58	0.53
$\gamma = 0.0001$, for				
each class				
$C_i \propto freq(i)^{-1}$				

Plotting bias-variance learning curves showed that the Gaussian kernel had > 99% accuracy on the training set but was less accurate on the testing set, indicating high variance. Thus we also used a linear kernel. To optimize the regularization parameter C, we again used grid search and stratified 10-fold cross-validation.

SVM	Accuracy	Precision	Recall	F_1
				score
Linear: $C = 0.01$	0.8892	0.97	0.41	0.576
Linear:	0.8375	0.55	0.63	0.587
$C_i \propto freq(i)^{-1}$				

The linear kernel generally performed better than the Gaussian kernel.

D. Multiclass Classification

We also ran multiclass classification because only one previous project tried it. To do so, we split our late class into two classes: class 1 between 15 and 45 minutes late, and class 2 for > 45 minutes late. An SVM with linear kernel and one vs. all strategy almost never predicted class 1, so we used an SVM with a Gaussian kernel and one-against-one strategy (create one SVM per pair of classes, and to predict, the class which receives the most votes is selected).

Confusion Matrix:

		Predicted		
		0	1	2
	0	906	51	23
Actual	1	64	19	45
	2	29	8	55

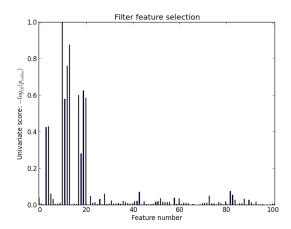
Overall accuracy was 0.8167; recall for class 1 was only 0.15, but recall for class 2 was higher at 0.60. This is because our classifier predicted class 2 57% more often than class 1, despite the fact that more flights are actually class 1. This suggests that class 1 flights (between 15 and 45 minutes late) do not have strong distinguishing characteristics in the dataset, compared to class 2 flights.

E. Random Forests

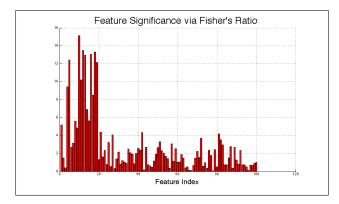
After performing parameter optimization on the number of trees in the forest, and the size of the random subsets of features considered when splitting a node, we found that a random forest classifier with 100 trees considering 100 features had 0.8917 accuracy, 0.93 precision, and 0.44 recall. We confirmed that more trees is better only up to a certain critical value.

IV. ADDITIONAL METHODS

A. Feature Selection



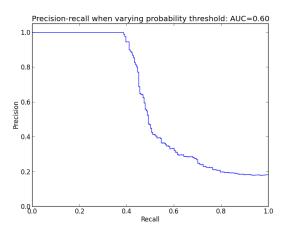
First we performed filter feature selection to measure how informative each feature was. The score metric we used was the p-value from a univariate statistical test. By far, the most important feature was whether the previous flight was delayed. Next were the weather features we added to the dataset (see high scores for features #12-20): specifically the precipitation and the condition-level (e.g. "partly cloudy") at both the departure and arrival airports. The other important features were the scheduled departure time and arrival time (see features #3 and 4).



Alternatively, we also used Fisher's ratio: $\frac{(m_1-m_2)^2}{(v_1+v_2)}$, where m_1 and m_2 are the means of our two classes (delayed or not delayed) and v_1 and v_2 are the variances, to measure the discriminating power of each feature. Once again, after previous flight delay, the most significant features are weather conditions (feature indices 10-13 and (16-19) and arrival and departure times (feature indices 4-5).

Finally, forward search on logistic regression was performed. The algorithm chose the following features (in order): (1) is_previous_flight_delayed, (2) departure conditon-level, (3) departure time, (4) arrival precipitation, (5) departure wind speed, (6) departure visibility, (7) arrival condition level, (8) origin airport, (9) destination airport. Overall both feature selection algorithms showed that weather information, departure time, and previous flight delay were the most important features.

B. Precision vs. Recall



Because only 25% of flights were (> 15 minutes) delayed, our dataset was skewed, so it was important to study precision and recall rather than just accuracy. This graph shows a precision-recall curve when varying the decision threshold of the hypothesis function, for a linear SVM. We also varied SVM parameters to optimize either precision or recall (refer to SVM results above), achieving an F1 score of 0.587.

V. CONCLUSION

Overall, accuracy from our algorithms were relatively good: all algorithms were about 90% accurate. However, there is still room for improvement in recall. Our multiclass classification (>15 min & >45 min) showed that flights between 15 and 45 minutes late do not have strong distinguishing characteristics in the dataset, so better features might improve recall.

The results from both our feature selection methods revealed the following as the most predictive features: the previous aircraft arriving late, weather, and departure time. These features matched air carrier's reported causes of delay.

In addition, more complex network-based algorithms may show significant improvement. Often, when a few flights are delayed early in the day, this causes a domino effect that delays later flights. Our model started to capture this effect with our previous_flight_delay feature, but network algorithms that fully represent this structure may improve performance.

VI. REFERENCES

- [1] The data, American Statistical Association, Data Expo 2009 http://stat-computing.org/dataexpo/2009/the-data.html
- [2] Flight Delays by Cause, Bureau of Transportation Statistics http://www.transtats.bts.gov/ot_delay/ot_delaycause1.asp
- [3] Flight Standards Service-Civil Aviation Registry http://www.faa.gov/about/office_org/headquarters_offices/avs/offices/afs/afs700/
- [4] Historical Weather, Weather Underground http://www.wunderground.com/history
- [5] How FlightCaster Squeezes Predictions from Flight Data http://www.datawrangling.com/how-flightcaster-squeezes-predictions-from-flight-data