Analyzing and Predicting School Bus Delays and Breakdowns in New York City

Omar Afifi (*afifio@umich.edu*) Neil Mankodi (*nmankodi@umich.edu*)
Stephen Blough (*bloughst@umich.edu*) Aryan Ringshia (*aryanrr@umich.edu)*

Abstract

We analyze data containing information on breakdowns and delays for school buses in New York City. Our analysis is in three parts: First, we try to predict whether incidents are likely due to breakdowns or delays, based on several (largely categorical) features. Subsequently, we build models that are capable of incorporating these results for predicting delay times. Lastly, we offer an alternative time-series-based approach to delay prediction which relies more heavily on locational and temporal data. In a congested metropolis area like New York City, reliable transportation for students can be a problem and incidents are frequent. Understanding the immediate impact of incidents when they occur is therefore a crucial tool for any efficient transportation system.

Related Work

The most striking result preceding our work is Borseti and Berger's paper on an older version of the data set. [1] They focused on statistical analysis (e.g., how bus breakdowns are distributed across various manufacturers, accompanied by some basic OLS models. However, their work focused on analysis rather than learning.). Older versions of this data set available on Kaggle were also explored for PCA, but we avoided PCA because a lot of our data was binary or categorical, and its predictive power suffocated under dimension reduction.

Data Exploration, Preparation, and Preprocessing

The first issue we faced in this project was dealing with data preparation. Much of our ½ million rows of data were entered manually, so there were a lot of inconsistencies in formatting. [e.g. '1hr', 'One Hour', 45-60, '60 minutes', '60+'] all might be different values for one hour. Delays were also often entered as time-frames. Consequently, we were forced to abandon a regressive approach to this problem and we binned time frames to a granularity of ½ hour, with delays longer than 1.5 hours being classified as 'max', and we focused on correctly predicting these windows.

Most of our features were either categorical (e.g. district, delay reason), or binary (e.g. "has notified school", so we one-hot encoded all the categorical features we were able to use. (some features, like route numbers, seemed like good predictors but involved too many categories to be useful). We used timestamps for incidents, applying trigonometric transformations to the cyclic features, and using the day, hour, minute, and month as separate features as well.

Lastly, it's worth addressing that our data set was very imbalanced. The vast majority of incidents were less than one hour, and only about 20% of the data fell into the larger windows. Moreover, only roughly 10% of the incidents were breakdowns. This is an important note because it drove a lot of our decision-making.

Classifying Delays vs. Breakdowns

The first problem we tackled was classifying delays and breakdowns. For this task, we also dropped the delay time column because this was an illicit feature that would not be available at inference time, and long delay times were heavily correlated with breakdowns.

Baselines

Given the data imbalance, we knew that a dummy classifier would perform well, so it was imperative that we choose a good baseline. This would allow us to show that we truly did make improvements. Using the dummy classifier functionality in sci-kit learn, we made the model predict the most frequent class, which in this case was 'delay'. For our baseline, the dummy classifier achieved an accuracy metric of 91.09% on testing data but performed poorly on all of our other testing metrics.

Models

Our first model was a pipeline that performed logistic regression. The first steps in the pipeline were to scale the data with Standard Scaler (as our data is in different scales), and then apply the RandomOverSampler from the imbalanced learn library (to help balance out the data for better predictions). After using grid search to fine-tune the logistic regression, we only achieved an accuracy of 89.45% on the testing data, but far better results in other metrics. We subsequently increased the model's complexity so that we could achieve comparable accuracy to our baseline. Since the data was predominantly binary and categorical, decision tree-based methods like random forests seemed very natural. In addition to the new model, we also incorporated degree 2 feature expansion into our preprocessing in an attempt to capture the interaction between different features in our data. (given the binary data, this was particularly useful, since feature expansion essentially acts as an AND operator). Our fine-tuned random forest model achieved an accuracy of 91.56% on our testing data and improved across all other metrics we tested.

Evaluation

When evaluating our models, it was important to stay away from accuracy metrics because they are not useful in the face of drastic imbalances. Therefore, we focused on optimizing and evaluating these models with precision, recall, and F1 metrics. We observed substantial gains in performance with random forest models. (See Appendix A for all charts and performance figures.)

Predicting Delay Windows

The second problem we tackled was delay-time prediction. After binning our data into four delay windows, this became a multiclass classification problem. Because the delay times followed a natural order, we adopted a one-vs-one approach. We compared it to a one–vs-rest approach in our base models, and they confirmed that one-vs-one was the right framework for the problem. With the one-vs-one approach, we had to exercise caution with data leakage in the splitting phase: We carefully split data into training and testing sets, *before* grouping them into classes and feeding them into the binary classifiers. (If we had grouped before the split, we would have taken 6 random splits, so it would be possible that some of the binary classifiers observed data that was in the holdout set of another classifier, leading to leakage in the combined model.)

Baselines

Noting 40% of values in the 0-30 minute interval and a correlation between long delays and breakdowns, we achieved a macro-baseline accuracy just above 0.5. As a second baseline, we used logistic regression models with degree-2 feature expansion and dataset normalizations, tuned with 5-fold cross-validation. We considered Naive Bayes as a potential candidate but dismissed it due to the mix of categorical and real-valued features. Evaluation results for our baseline logistic regression models are in the appendix.

Models

We explored a lot of different models for this problem. We built various pipelines for different models such as XGBoost and even deep neural networks featuring dense and dropout layers, which we trained and tuned extensively with cross-validation and grid search. However, once again, random forests outperformed all other models across all metrics by a large margin. We believe that this is because the vast majority of our columns were Bernoulli variables (i.e. 0 or 1), so the decision tree-based methods were a very good match for our data, as well as the fact that decision trees are very naturally suited for the multiclass problem. After settling with random forests, we extensively tuned our hyperparameters with the same grid search and cross-validation approach. Using our results, we constructed pipelines for each pair of classes and then combined them into one multiclass model.

Evaluation

Because of data imbalances, we optimized the individual classifiers and micro-averaged scores. Since we assembled the one-vs-one construction manually (as opposed to implementing a pre-assembled model such as OneVsOneClassifier in sci-kit learn), we were able to look under the hood to see which models were doing good and which weren't. Hence we tried to optimize individual classifier scores in the belief that this would lead to the greatest improvements in the combined model. We looked at five key metrics: accuracy, precision, recall, F1, and roc/auc curves for the individual classifiers and the micro averages in the combined model. The metrics in the appendix tables show that the baselines were able to distinguish short delays (< 1-hour) from long delays (>1 hour), but were unable to distinguish between .5-hour and 1-hour delays or between 1.5-2 hour delays and max delays. Our final models were able to detect the discrepancies between the two, which is why building the model manually was so powerful. In the end, we also used a sci-kit learn pre-assembly due to its superior efficiency. Scores for the manually and pre-assembled models are shown in the appendix.

Time Series Analysis for Delay Bin Prediction

This analysis emphasized temporal and locational data as the primary predictors of future delays. This would be particularly useful in helping city departments perform resource allocation and optimize logistics across the various boroughs.

Data Preparation

We used the same data processing techniques that we used for predicting delay times. However, we altered the granularity of the data to months in order to make it directly usable for time series analysis. The data set contained incidents distributed across all the NYC boroughs, and we decided to group the data by the boroughs and develop separate models for each one. Since different boroughs exhibited different delay and breakdown trends, we believed that this would be superior to applying a single model across all boroughs.

Baseline

Having a baseline model was essential because it allowed us to determine if our trained models truly had anything to offer. For delay bin forecasting, we built a model that simply predicted the majority delay bin. This baseline model failed in many boroughs, leaving substantial room for improvement, especially in the boroughs where the baseline accuracy was noticeably poor.

Time Series Model

One of the key challenges we faced in this phase was that time series models are generally used to forecast continuous outputs. However, we were trying to forecast a categorical output (i.e. the delay bins). In light of this, we decided to create a long short-term memory (LSTM) deep neural network. with the final layer being a dense network that outputs the probability of belonging to each delay bin. The general methodology of our model was to utilize a certain number of historical data entries to predict the next future entry. For the current implementation, the model uses ten entries from the past to predict the future delay bin.

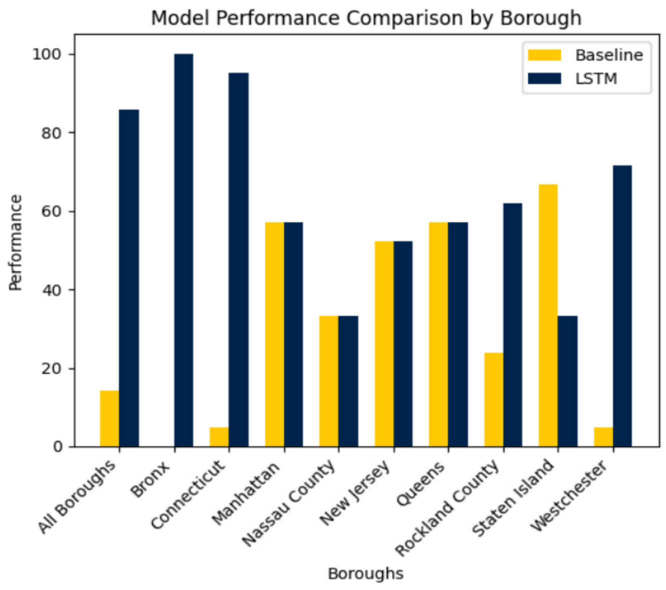
Some details about the model are as follows:

- LSTM layer with 50 neurons and Relu activation function.
- Dense layer with a softmax activation function
- Optimizer = Adam
- Loss = categorical cross entropy
- Epochs = 10
- Batch size = 32

One important note to make about this model is that, even while working on the test data, it takes its own predictions into account for future predictions. This allows us to get a more holistic view of the model's performance in a real-world scenario. Comparisons with the baseline are shown below, where the performance is a measurement of accuracy.

Evaluation

The LSTM model was generally successful in matching the few high-performance baselines. In addition to this, the model successfully boosted the performance in many of the problematic boroughs. This coincided with our hopes in the initial stages of developing this model.



Discussion and Conclusion

Overall, we were very pleased with the outcome of our project. Most of our models exceeded our hopes by a substantial amount. These models have tremendous real-world applications and could be extended to other areas, cities, and transportation systems. Given more time, we would have liked to integrate our models into one pipeline and test whether or not our breakdown vs delay predictor was able to improve the scores of the delay-time predictors in the absence of a breakdown-delay feature. We also would have liked to explore the possibility of increasing the granularity of our delay time frames to 15-minute intervals and extending the maximum timeframe. Lastly, we would have liked to see how these techniques transferred to other bus systems and in other cities. We all learned a lot about time series modeling and evaluation criteria, and we got a taste of more realistic machine-learning projects with real data.

References

[1] Elizabeth Borseti, and Paul D. Berger. (2020). "SCHOOL BUS BREAKDOWNS IN NEW YORK CITY." International Journal of Research - Granthaalayah, 8(1), 336-349. https://doi.org/10.5281/zenodo.3673435.

[2] "Bus Breakdown and Delays," NYC Open Data, https://data.cityofnewyork.us/Transportation/Bus-Breakdown-and-Delays/ez4e-fazm (accessed Dec. 5, 2023).

[3] F. Pedregosa *et al.*, 'Scikit-learn: Machine Learning in Python', *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] G. Lemaitre, F. Nogueira, and C. K. Aridas, 'Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning', *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[5] The pandas development team, "pandas-dev/pandas: Pandas". Zenodo, Nov. 10, 2023. doi: 10.5281/zenodo.10107975.

[6] C. R. Harris *et al.*, 'Array programming with NumPy', *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020.

[7] J. D. Hunter, 'Matplotlib: A 2D graphics environment', *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[8] F. Chollet and Others, 'Keras', 2015. [Online]. Available: https://keras.io.

Code

All of our code for this project can be found on Github:

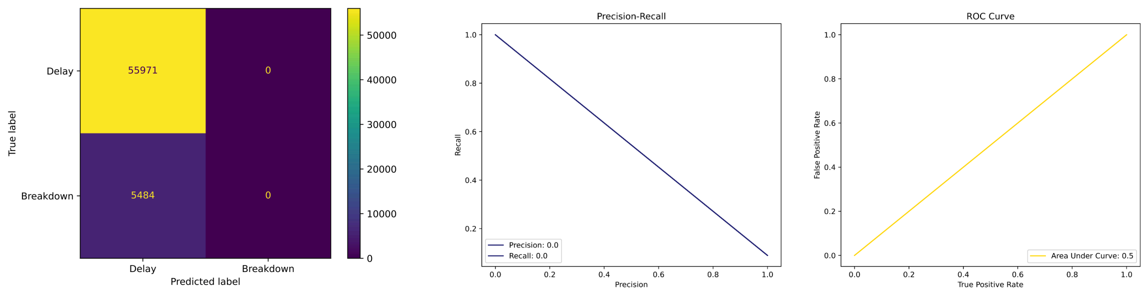
https://github.com/oaf9/Machine-Learning-Methods-for-Bus-Routes-and-Traffic

Appendix A

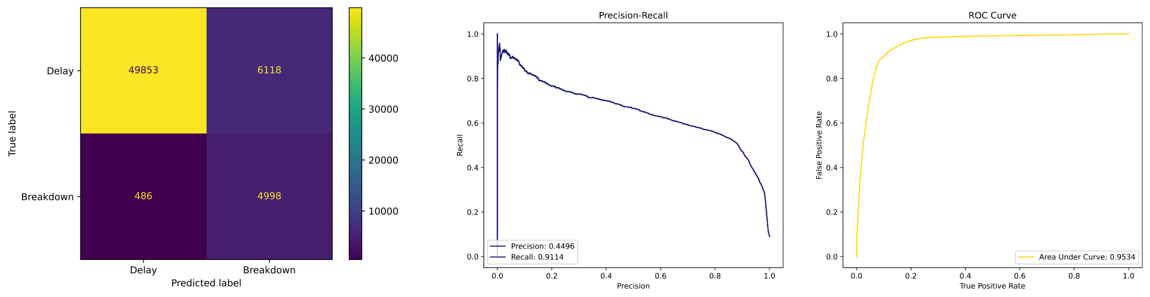
Results for Delay vs Breakdown Classifiers

Classifier	Accuracy	Recall	Precision	F1	AUC
Dummy	0.9108	0.0000	0.0000	0.0000	0.5000
Logistic Regression	0.8925	0.9114	0.4496	0.6022	0.9534
Random Forest	**0.9156**	**0.9201**	**0.5151**	**0.6605**	**0.9680**

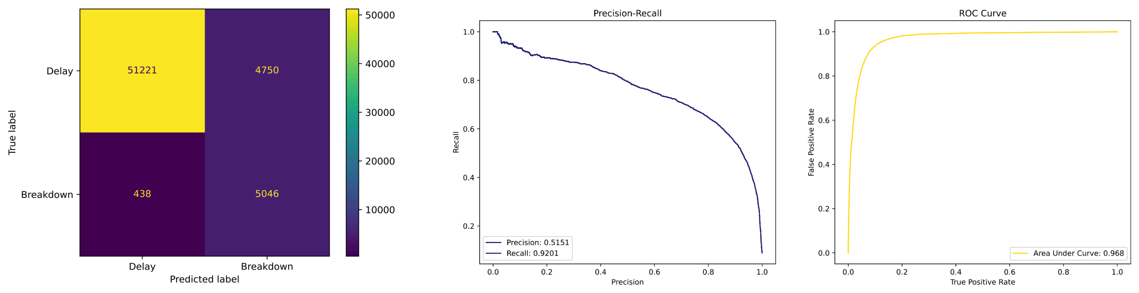
Dummy Classifier - Confusion Matrix, Precision-Recall and ROC Curves



Logistic Regression - Confusion Matrix, Precision-Recall and ROC Curves



Random Forest - Confusion Matrix, Precision-Recall and ROC Curves



Appendix B

Delay Window Prediction Results

Individual Estimator and Micro Averaged Scores for Logistic Regression Baselines

Classifier Pair	Accuracy	Recall	Precision	F1	AUC
0-30, 31-60	0.5529	0.5709	0.5536	0.5621	0.5528
0-30, 61-90	0.8641	0	0	0	0.5
0-30, max	0.9729	0.8999	0.9537	0.926	0.9449
31-60, 61-90	0.8653	0	0	0	0.5
31-60, max	0.9608	0.8281	0.9558	0.8874	0.7229
61-90, max	0.9305	0.9324	0.95	0.9411	0.7229
Avg. (Macro) (pre-assembled model)	.5372	0.49	0.48	0.49	.712
Avg. (Micro) (pre-assembled model)	0.574	.5354			

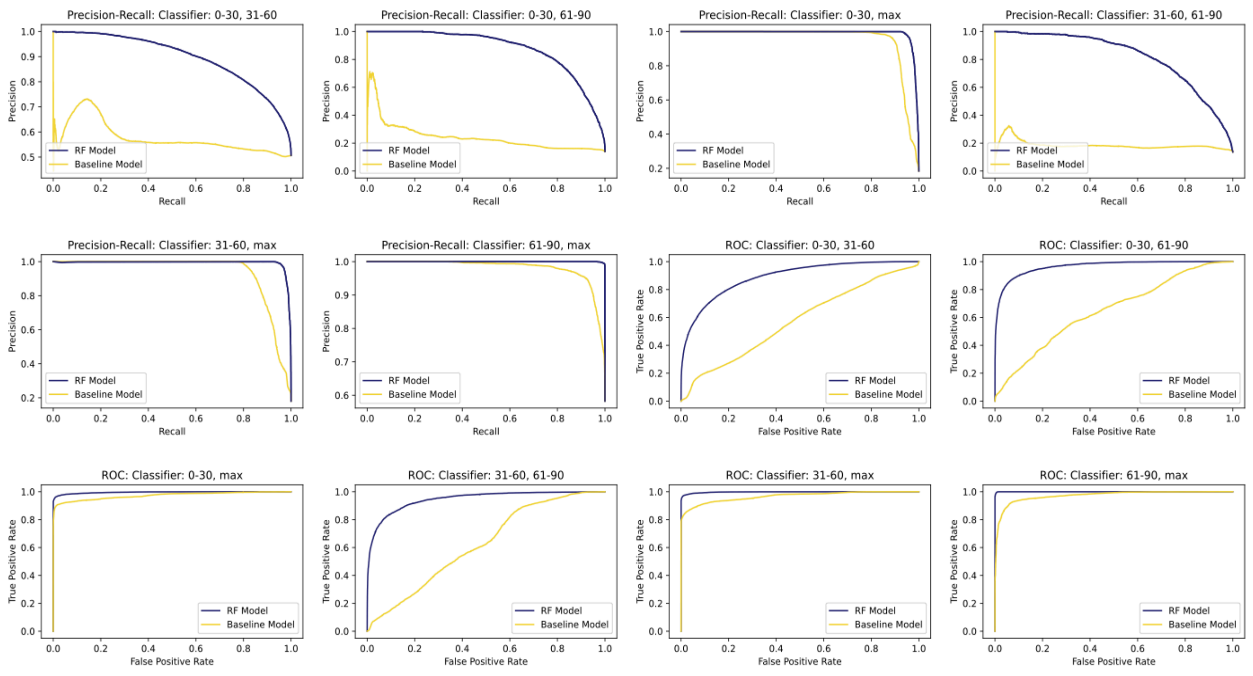
*(Avg. scores are from pre-assembled sci-kit learn models: while our custom assembly was great for analysis, it had efficiency issues.)
* Note that the baselines are effective against distinct classes, but poor at neighboring classes.

Individual Classifier and Micro Averaged Scores for the Combined Model

Classifier Pair	Accuracy	Recall	Precision	F1	AUC
0-30, 31-60	0.8017	0.7924	0.8115	0.8018	0.8893
0-30, 61-90	0.9427	0.6711	0.8911	0.7656	0.9655
0-30, max	0.9871	0.9337	0.9961	0.9639	0.9957
31-60, 61-90	0.9326	0.6053	0.8585	0.71	0.946
31-60, max	0.9887	0.9414	0.9956	0.9677	0.9978
61-90, max	0.9943	0.999	0.9914	0.9951	0.9999
Avg. (Macro) (pre-assembled model)	0.7924	0.77	0.84	0.80	.9392
Avg. (Micro) (pre-assembled Model)	0.7924	.0.7924			

*(Avg. scores are from pre-assembled sci-kit learn models: while our custom assembly was great for analysis, it had efficiency issues.)

Precision and Recall Curves For Baseline and Random Forest Classifiers


Precision-Recall: Classifier: 0-30, 31-60
Precision-Recall: Classifier: 0-30, 61-90
Precision-Recall: Classifier: 0-30, max
Precision-Recall: Classifier: 31-60, 61-90
Precision-Recall: Classifier: 31-60, max
Precision-Recall: Classifier: 61-90, max
ROC: Classifier: 0-30, 31-60
ROC: Classifier: 0-30, 61-90
ROC: Classifier: 0-30, max
ROC: Classifier: 31-60, 61-90
ROC: Classifier: 31-60, max
ROC: Classifier: 61-90, max
Precision
Recall
True Positive Rate
False Positive Rate
RF Model
Baseline Model