Mariia Aleksandrovych and Rebecca Heilweil

Using Machine Learning to Predict New York City Agency Response Times

- Appendix

Project Overview: We have pursued an *application project.* We aimed to test several models, including a linear regression, a gradient boosting-based regression, two types of neural networks, and a ResNet to investigate how effectively we can predict response times for 311 calls based in New York City. We anticipate that significant features could be the particular city agency responding, such as the Sanitation Department or the New York Police Department, the location of the caller, and the description described by the caller. By analyzing these features, we aim to employ various models to predict the time it takes to resolve these service requests.

Notes on Dataset: Our data is the extremely rich data provided by the Open Data repository run by New York City's government. This is a large data set that can be easily tested and which includes more than 35 million reports about 311 phone calls over the past decade. This data also includes dozens of features, which is useful for having many options to select from.

Notes on Data Selection and Processing: We take several steps to prepare our data. First, we split this extremely large dataset into a smaller selection, about 500,000 reports, that we can work with. Unfortunately, we found that our computers could not handle larger file sizes and still allow us to effectively experiment with various models. Hypertuning was only possible on a smaller set of reports in a several code file.

We also use a "Report Created" and a "Report Resolved" time feature in order to create a new "Time Elapsed" or "Hours Taken" feature. Because we have the time of a report's creation and resolution down to the minute, we decided to calculate "Hours Taken" as a metric. Within the data set we are working with, the average report takes 72 hours to resolve, though it can be resolved within minutes or possibly dozens of days. Notably, the maximum value for hours taken was 1618 hours and the standard deviation was 178 hours — which shows that our our data includes a lot of variation.

Hours makes sense to use, since minutes would be too precise and days would not capture the nuance we're looking for.

We turned text-based categories, like Descriptors, into numerical-based categories since the descriptions were standardized. While not required for Gradient Boosting Regression, we used a scaler to normalize data used for other models, such as Linear Regression.

Data Error Discovery: While working through our project, we discovered a curious error in our data. A number of our samples were showing a negative "Hours Taken" metric, which doesn't make logical sense. We sought to investigate whether this was a trend throughout the data or whether there was a more focused, particular error. Ultimately, the issue was limited to one

particular agency, the Department of Transportation, and one particular issue: Street Light Condition complaint types. These included street light reports with a variety of problems ("Descriptor"), including potholes and exposed lamp post wires.

More than 2,000 reports in our sample data had this error. Notably, we dropped this data to avoid the issue impacting our larger training, though, in doing so, potentially underrepresented the Department of Transportation in our model training.

Notes on Gradient Boosted Regression: For this study, we initially started by evaluating a model with the following parameters: minimum samples = 2, N_estimators = 100, learning rate = 0.1. We initially achieved an MSE of about 16,000 with this (using in-sample data). However, we conducted parameter tuning and achieved the following results. *Please note we did our hyperparameter tuning in a separate file, which is also available in the GitHub.

	N_Estimators	Learning Rate	Depth	R2
0	300	0.30	7	0.732853
0	300	0.10	7	0.711626
0	300	0.30	5	0.705243
0	100	0.30	7	0.703394
0	50	0.30	7	0.679129
0	100	0.10	7	0.667795
0	300	0.10	5	0.658996
0	100	0.30	5	0.657224
0	50	0.10	7	0.642289
0	300	0.30	3	0.635714
0	50	0.30	5	0.626837
0	300	0.01	7	0.619298
0	100	0.10	5	0.613099
0	100	0.30	3	0.595482
0	50	0.10	5	0.594711
0	300	0.10	3	0.594522
0	300	0.30	2	0.589239

We select 300 for N_Estimators, .3 for Learning Rate, and 7 for depth, based on the highest R2 scores. This achieves better results for MSE in the full model.

Mariia Aleksandrovych and Rebecca Heilweil

Notes on Neural Networks Models:
Three neural networks were trained for the study: a 1-layer neural network (NN), a 2-layer NN, and a Deep ResNet.

1-layer NN:

Architecture: Input layer (fully connected, 128 neurons, ReLU activation, L2 regularization) with an input shape of (, 7) for seven features. DropOut Layer (20%) follows, and the output layer has a single neuron with linear activation.

Training: Adam optimizer, a learning rate of 0.001 for the first 10 epochs, followed by an exponential decay. Batch size is 32, with a validation split of 80%, using Mean Squared Error as the loss function.

2-layer NN:

Architecture: Input layer (fully connected, 64 neurons, ReLU activation, L2 regularization) with input shape (, 7). DropOut Layer (20%) precedes a hidden layer (fully connected, 32 neurons, ReLU activation, L2 regularization). Another DropOut Layer (20%) is followed by the output layer with a single neuron and linear activation.

Training: Similar to the 1-layer NN, with Adam optimizer, learning rate schedule, batch size of 32, and 80-20 training-validation split.

Deep ResNet:

Architecture: Input layer (fully connected, 64 neurons, ReLU activation, L2 regularization) with input shape (, 7). Residual blocks consist of two Dense layers (64 neurons, ReLU activation, L2 regularization) with Batch Normalization and Dropout (20%) layers. A Dense layer (fully connected, 32 neurons, ReLU activation, L2 regularization) and DropOut Layer (20%) precede the output layer (single neuron, linear activation).

Training: Similar to the previous networks, utilizing Adam optimizer, learning rate schedule, batch size of 32, and 80-20 training-validation split with Mean Squared Error as the loss function

Notes on Feature Selection:

First, we drop the large majority of our features, since many are related to relatively niche issues and aren't actually designed to address most calls (Some are only related to taxis, for instance). We also drop the EventID feature, which should have no influence on our model. We drop Zip Code, for example, because we already have a Borough and two other location features available. Ultimately, we settle on including these features: Longitude, Latitude, Borough Descriptor, Complaint Type, and Responding Agency.

Mariia Aleksandrovych and Rebecca Heilweil

Notes on Model Selection:

Baselines. Two versions of linear regression, including a lasso regression, are used as a baseline to evaluate our models. We also use Mean Squared Error to compare errors across different models. This project's objective is to evaluate and identify the most effective model for predicting the resolution time of service requests within New York City's agencies.

Final Model Selection and Conclusion:

The assignment asks us to prioritize the novelty of the approach and results. We believe we've introduced a relatively novel approach — and shown that neural networks can at least offer promise and sometimes comparable results to Gradient Boosted Trees, after some parameter tuning, in processing this data. While there has been some data analysis of NYC 311 calls, we may be the first to dig into using these diverse types of AI tools to make these predictions.

Ultimately, we showed that GBT could outperform linear regression on both in-sample and out-of-sample data. All of our models, except for ResNet, outperformed Linear Regression.

MSE Performance on In-Sample Data Cross Validated

Mean Squared Error across 5 folds: 32006.286915976634 +/- 562.4579811363096

Linear Regression - MSE: 28904.17261398608 +/- 346.5560153708919
Lasso Regression - MSE: 28918.12591391247 +/- 343.8195874035514
GB Regression - MSE: 4985.333993772696 +/- 86.11027934727582
Simple 1 layers NN- MSE: 20861.137890625 +/- 600.5459880750432
2 layers NN - MSE: 16587.6294921875 +/- 381.237108979759
3 layers DResNet - MSE: 59389.391796875 +/- 84009.95419371294

MSE Performance on Out-of-Sample Data

MSE for Baseline Model: 71167.0791123987
MSE for Linear Regression Model: 66329.90822792749
MSE for Lasso Model: 66449.24024717495
MSE for GBT Model: 64077.68799608705
MSE for NN 2 layers Model: 63790.3143967668
MSE for NN 3 layers Model: 65652.52003491706
MSE for DResNet 3 layers Model: 84365.82704451475

*We ran this several times and got different results, but the GBT and 2-layer NN always tended to outperform the others. The file size is too big to effectively shuffle.

Future work:

First, our study highlighted that one agency has a systemic problem of misfiling their reports, potentially poisoning this public data resource. A first step from this study would involve flagging the issue to Open Data NYC and investigating. For example, this data is used by the NY City Council.

It is possible that we could re-evaluate our features to see if previously-dropped features being incorporated might ultimately improve the model. We might also explore AdaBoost and CatBoost-based regressions to see if they improve the performance. We could adjust the second Neural Network model more, as well.

Critically, we didn't look at the type of action taken on the call and only measured our prediction by the response. NYC Open Data shows that this data is theoretically available and is grouped into "Fixed," "Ambiguous," and "Violations Issued." Looking at this type of response — through a classification rather than regression framework — could be ground for future study.