ORIE 4741 Final Report: Predicting 911 Call Urgency in New York City

Authors: Richard Zheng (rz98), Iris Li (icl5), Peter Gribizis (pjg222)

December 13, 2020

1 Introduction and Overview

Emergency medical services (EMS) are a cornerstone of public health and safety in urban cities, saving countless lives every day. In large cities, EMS receive a sizable volume of 911 calls, with the largest of cities (New York City) averaging approximately 4,000 emergency calls placed per day. With the volume of emergency calls placed having risen significantly since the beginning of the COVID-19 pandemic, our team set out to better understand the nature of these 911 calls.

Our goal was to develop a predictive model that takes specific EMS call information (time of day, location, call type, etc.) as inputs to then predict the severity level of a given call, as well as the time it would take emergency services to reach a given caller if needed. Ideally, the insights and predictions garnered from this model could have a direct impact on an individual's well-being, as oftentimes 911 dispatch has to deal with life threatening situations and can not always receive the information necessary to deploy the most relevant resources.

The desired impact of this model is to generate valuable insights for dispatch operators to help inform them how to best allocate resources when ambiguous emergency calls are placed and received.

2 Data Description

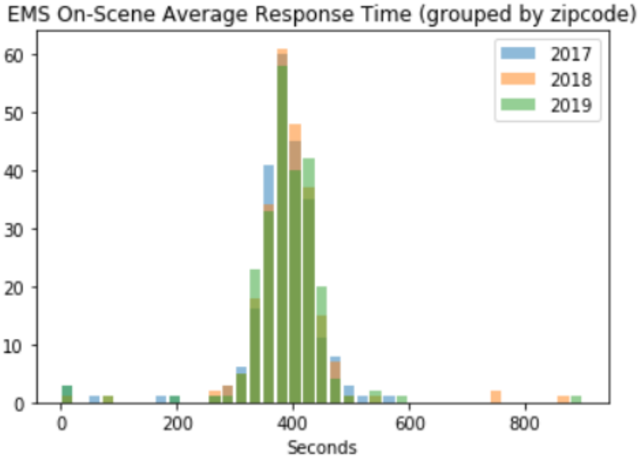
The dataset at the core of our analysis, publicly available via NYC OpenData, was EMS Dispatch Data. This dataset has 20 million EMS call entries, as recorded by the EMS Computer Aided Dispatch System from 2008 to present, and captures EMS calls placed within New York City's five boroughs. Of the 31 initial features, we filtered out and captured the most relevant ones (some features were simply different variations of the ones we selected, while others were columns indicating specific call types that applied to 1% of the data).

	Category	Description
1	Initial Call Type	268 unique symbols. (Ex. UNC for unconscious patient)
2	Final Call Type	Same description as 'Initial Call Type'.
3	Initial Severity Level Code	Scale from 1-8, 1 being most severe.
4	Final Severity Level Code	Same description as 'Initial Severity Level'.
5	Incident Disposition	A code indicating the final outcome of the incident. (Ex 82 for transporting patient)
6	Time Stamps	The time the incident was created in the dispatch system. (Year, Month, Day, Time)
7	Response Times	Dispatch, Incident, Travel. (In seconds)
8	Location Data	237 unique Zipcodes, 5 Boroughs, etc.
9	Special Indicators	Reopen, Special Event, Standby

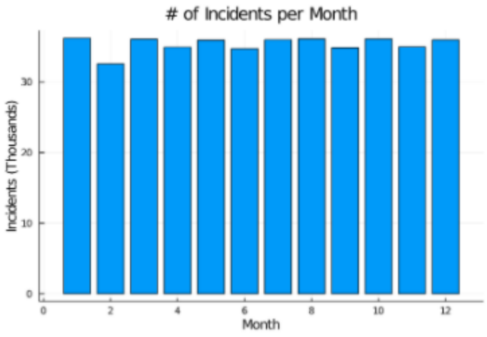
Table 1: Description of features by category

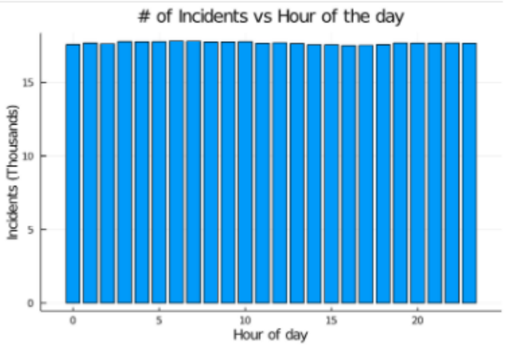
3 Exploratory Data Analysis

Due to the significant size of the dataset, we decided to focus on a subsample of the data from 2019. To ensure we were not overfitting to 2019 and that our model would be general applicable, we checked the average incident response time (the time it takes from a 911 call being placed, to the EMS arriving on-scene) for prior years to confirm the distribution, grouped by zipcode, was consistent among different years.



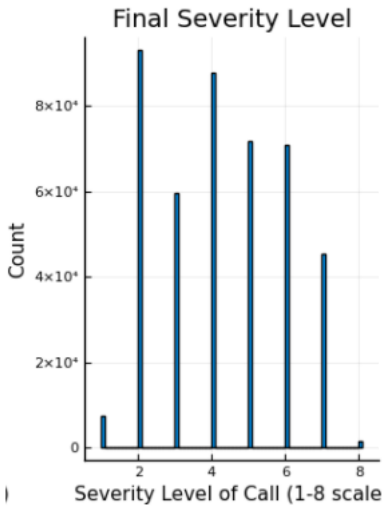
On this dataset, approximately 8.25% of samples had unknown initial call types, and 6.8% had unknown final call types. The data also had two features that indicated whether the calculation of the dispatch response time and incident response times were valid. We dropped the samples that were labeled as invalid. After that, only around 2% of data was missing. We further subsampled on the 2019 data, as over a million samples remained. To ensure we did not eliminate any effects that were due to seasonality, we took a standard random sample of 50 calls per every hour so that every time period was equally represented.

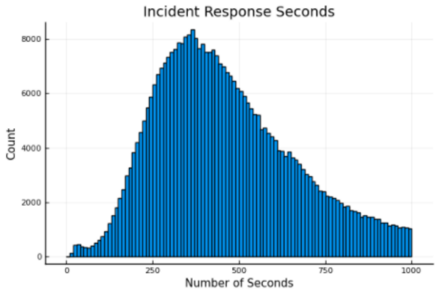




The two features we were looking to predict were the final severity level of EMS calls, and the average incidence response time (when applicable). The distribution for the final severity level looks somewhat uniform when not taking into account levels 1 and 8, as these indicators are reserved for rare call types.

A histogram of incident response seconds shows that the distribution is slightly right-skewed with most times falling around the average of 461 seconds. With an accurate model, the hope would be to further skew the histogram right, reducing the number of calls where incidence response time falls above the mean by better allocating resources when a call is placed.

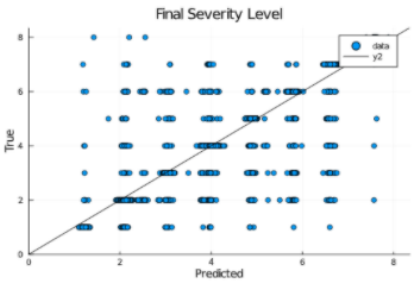


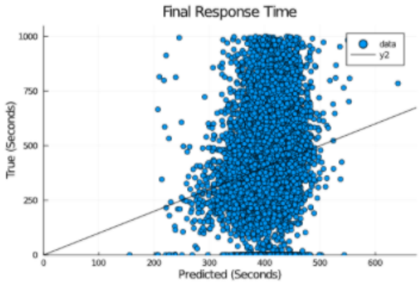


To process our features into a more usable form, we transformed some columns of data we believed to be most relevant into a form where it was easier to assess their performance and impact on our model. We extracted datetime information from the timestamp of all the calls, creating Year, Month, Week, Day, and Hour as real number features. We encoded the initial call type and zipcodes as one-hot vectors since these were categorical variables.

4 Initial Modeling Attempt - Linear Approximation

For a first pass attempt at trying to predict the response time and severity of calls, we created a least squares linear approximator using an 80/20 split of our data. We wanted to see if there were any easy-to-visualize trends given the input features we had available. We fit our data using quadratic loss and no regularization, resulting in the following plots:





From this attempt we observed that our model fairly accurately captured final severity level but struggled to fit incident response time. Additionally, we observed the presence of large amounts of outliers, which suggested that our model could be improved using a linear loss function instead, such as L1 Loss.

Initial Linear Approximation Testing Error		
	Quadratic Loss	L1 Loss
Final Severity Level	0.72757	0.32402
Incident Response Seconds	238.64	195.43

From this initial testing, we made the decision to use L1 loss going forward.

5 Gathering Additional Data

From our initial data exploration, we decided to further augment our data in hopes of finding features that could be more relevant to predicting call urgency. Especially in terms of response time, there are a significant number of outliers. Even after experimenting with models discussed later in the paper, we still experienced performance issues.

Thus, we decided to search for additional data that may help improve our predictions. We decided to join our dataset with the population density and demographic data on zipcode, since these may be potential factors affecting the response time of 911 calls. In addition, we also looked at historical data from the EMS dataset by looking at the average number of calls and average incident response time per zipcode in 2017 and 2018 to see whether that can help predict the incident response time in 2019. A list of the features we added are as follows:

- Population - A count of population by zipcode
- Population Density - Population per square mile
- Demographic Data - Minority proportion of population
- Number of Calls (2017 and 2018) - Calls made by zipcode in 2017/2018
- Average Incident Response Seconds (2017, 2018) - Average incident response time per zipcode in 2017/2018

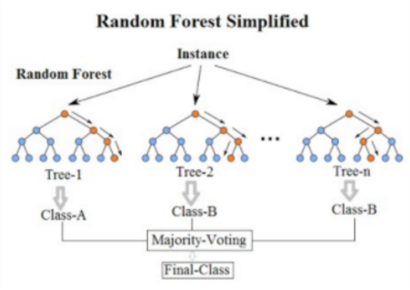
We ran another linear approximator with the new features. We used the same settings as in our initial data exploration, measuring L1 loss with no regularization.

Linear Approximation Testing Error	
Final Severity Level	0.31608
Incident Response Seconds	188.13

As we had hoped, the new features helped the performance of predicting incident response time, although it did not have a significant impact on predicting final call severity. Thus, we decided to use the augmented dataset in our models moving forward.

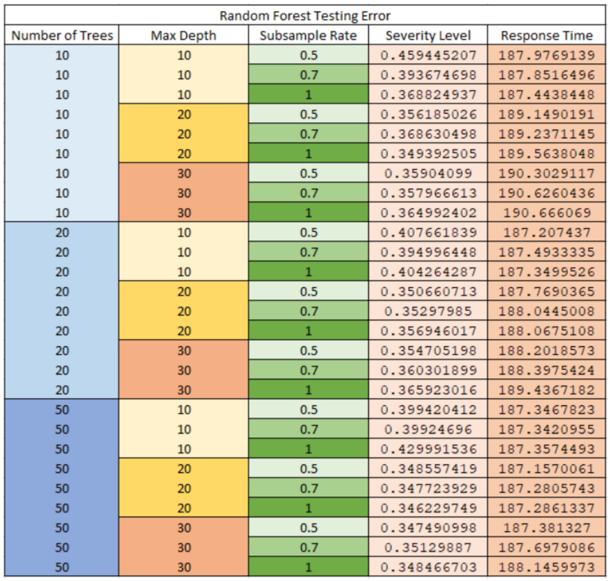
6 Random Forest

In addition to exploring linear models, we wanted to try the performance of nonlinear models which could have more complicated structures and therefore lead to better performance. One such model we considered is random forest, which creates groups of trees, each with a random subset of features from the dataset. At each layer of the tree, the node is split along the feature that minimizes the entropy of the following subtree. After all the trees are created, the model will predict the severity level and incident response time by taking a majority vote among all the predictions made by each tree in the forest.



One drawback to using the random forest model is that there are many hyperparameter configurations that are essentially black-box. For our model, we tried varying the number of trees from [10, 20, 50], max depth of the trees from [10, 20, 30], and partial subsample rate from [0.5, 0.7, 1.0]. For the full permutation of results, see the Random Forest Testing Error table at the end of this section. We found the most performant combination of hyperparameters to be 50 trees, a max depth of 20, and a subsampling rate of 0.7.

Random Forest Testing Error				
	Number of Trees	Max Depth	Subsample Rate	Testing Error
Final Severity Level	50	20	0.7	0.34772
Incident Response Seconds	50	20	0.7	178.28



Random Forest Testing Error				
Number of Trees	Max Depth	Subsample Rate	Severity Level	Response Time
10	10	0.5	0.459445207	187.9769139
10	10	0.7	0.393674698	187.8516496
10	10	1	0.368824937	187.4438448
10	20	0.5	0.356185026	189.1490191
10	20	0.7	0.368630498	189.2371145
10	20	1	0.349392505	189.5638048
10	30	0.5	0.35904099	190.3029117
10	30	0.7	0.357966613	190.6260436
10	30	1	0.364992402	190.666069
20	10	0.5	0.407661839	187.207437
20	10	0.7	0.394996448	187.4933335
20	10	1	0.404264287	187.3499526
20	20	0.5	0.350660713	187.7690365
20	20	0.7	0.35297985	188.0445008
20	20	1	0.356946017	188.0675108
20	30	0.5	0.354705198	188.2018573
20	30	0.7	0.360301899	188.3975424
20	30	1	0.365923016	189.4367182
50	10	0.5	0.399420412	187.3467823
50	10	0.7	0.39924696	187.3420955
50	10	1	0.429991536	187.3574493
50	20	0.5	0.348557419	187.1570061
50	20	0.7	0.347723929	187.2805743
50	20	1	0.346229749	187.2861337
50	30	0.5	0.347490998	187.381327
50	30	0.7	0.35129887	187.6979086
50	30	1	0.348466703	188.1459973

7 AutoML Pipeline

We decided to try as many other models as possible to confirm whether other models would work better, or if we needed more data. We used the AutoMLPipeline package developed by IBM to perform transformations on the data and train and cross-validate models for comparison. Running the code for the transformations (one hot encodings, pca), training, and cross validation took a long time, so we ran it on a randomly selected smaller sample size (20,000 samples vs 400,000+ samples) with 3 fold cross validation.

Using the AutoML Pipeline, we attempted using various additional models to fit our data:

- Ridge Regression - Adds a l2 norm regularizer to least squares, which mitigates problems with collinear features.
- Gradient Boosting Regression - Fits several smaller decision tree regression models to create an additive model.
- KNeighbors Regression - Locally interpolates the data to make predictions.

We compare LogisticRegression, Ridge, DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor, and KNeighborsRegressor with default SKLearn hyperparameters. The mean absolute error (L1 error) results are shown below.

[1]

AutoML L1 Testing Error					
	R.R.	D.T.R.	R.F.R.	G.B.R.	K.N.R.
Final Severity Level	0.30733	0.38993	0.36403	0.30971	0.91225
Incident Response Seconds	189.55	265.63	197.57	187.59	202.63

AutoML L1 Testing Error After PCA					
	R.R.	D.T.R.	R.F.R.	G.B.R.	K.N.R.
Final Severity Level	0.30732	0.37948	0.35284	0.31463	0.36135
Incident Response Seconds	189.44	263.12	196.49	187.59	203.20

Comparing these models, we see that Ridge regression and Gradient Boosting regression were the most effective, and worked about the same as or better than the two models we started off with, despite using the default hyperparameters.

After running PCA to reduce the dimensionality of our numeric features, the predictions for Final Severity Level improved a bit. This is likely because some features are meaningful while others are not, and PCA got rid of the not meaningful features. Overall, the improvement wasn't significant enough to predict as accurately as we had hoped. The mean absolute error of the best model was about 187 seconds, while the average response time was about 400.

Even after trying different hyperparameters for Ridge Regression (lambda = 1-5) and Gradient Boosting Regression (loss functions l2, l1, huber; max tree depth 2-6), there weren't any significant improvements.

8 Conclusion

In our semester long project, we used various methods we learned in class, such as one hot encoding, linear regression, and principal component analysis in order to predict the urgency of 911 calls. Our best model has a very reasonable error for final severity level of 0.31 and decent error for incident response time of 178 seconds. We noticed that the models we tested all produced error results in a similar ballpark as our linear model, which surprised us as we expected the non-linear models to be more performant due to their increased complexity. However, we did see that merging in features from additional related datasets did have a positive effect on incident response time, which led us to believe that we may see much better performance if we examined better correlated features in other potential datasets.

Our goal in taking this project was to hopefully find some correlation between incoming call metrics that can easily identify how much EMS resources a call would require. To this end we believe we have some success. We were able to identify the severity level of a call with an error rate of 0.31, which is quite accurate. By knowing the severity level of a call, EMS are able to estimate what level of emergency response is required, for example urgent ambulance rides are practically required for level 1 and 2, but the EMS do not need to rush at all for level 8 and can let other calls take priority. Similarly for incident response time, if the EMS know approximately how long calls will take, they can estimate the throughput of calls each day and allocate 911 operators appropriately. Although our models were not as performant on incident response time as we hoped, an error rate of 178 seconds is still significantly better than chance. In addition, since we saw a noted improvement in predicting response time by adding zipcode related features into our dataset, we concluded that putting more resources into EMS services at the zipcodes with highest historical response times would be a great way to lower the time it takes patients to receive urgent care when required.

With regards to our model becoming a potential “Weapon of Math Destruction”, this would only happen if our model downscaled severity level predictions, and did not suggest a proper utilization of EMS resources. Otherwise, we do not foresee the outcomes put forward by our project altering the current behavior of the EMS systems in place.

[1] Model acronyms for both tables are as follows: Ridge Regression, Decision Tree Regression, Random Forest Regression, Gradient Boosting Regression, K Neighbors Regression

In terms of fairness, our predictor worked best when we added in zipcode based metrics (calibrating between different locations differently). This meant that the predictions at each individual zipcode were more accurate, though it was not fair for predicting NYC as a whole.

(Same results as above: initial and second attempts at linear approximation)

Linear Approximation Testing Error		
	Without Location Metrics	With Location Metrics
Final Severity Level	0.32402	0.31608
Incident Response Seconds	195.43	188.13