Code Blue: Evaluation of Emergency Medical Service Incidents in NYC

Sophia Tsilerides (smt570), Ilana Weinstein (igw212), Amanda Kuznecov (anr431)
DS-GA 1001 - New York University

ABSTRACT

EMS travel times are constantly on the rise within New York City. Insights about the reasoning behind this can be uncovered by understanding historical information about these events and what is really driving the delayed response times. This paper attempts to predict EMS travel time while pinpointing inefficiencies in the system and determining what key factors contribute to slow response times.

1. BUSINESS UNDERSTANDING

New York City's Emergency Medical Services (EMS) has saved countless lives since the early 19th century, but as the city grows it has become more difficult for ambulances to reach those in need. In particular, the increase in response time has alarmed NYC officials. According to councilwoman Elizabeth Crowley, "Each and every year it is taking longer in life-threatening emergency situations for help to arrive" [1]. It is crucial for response times to remain low as they are directly correlated with chance of survival. Using data mining solutions, this paper attempts to identify constraints and suggest implementations of infrastructure policies, so that response times can improve and lives can be saved.

EMS travel time is of interest as it affects the outcome of the incident. The aim is to predict travel times in order to identify bottlenecks within the system. During this study, travel time predictions were comparatively observed as continuous and discrete values during modeling.

The data mining solution will help uncover patterns in the data, specifically why response time can be high. It is expected that some patterns will arise within dispatch areas and during certain days of the week or time of day. Proposed solutions could include increased staffing during certain times, reorganization and addition of dispatch stations, or road improvements in certain areas. An effective model will aid in the preciseness of the infrastructure analysis.

2. DATA UNDERSTANDING

2.1 Data Collection and Extracted Feature

The EMS Incident Dispatch dataset was provided by the Fire Department of New York City (FDNYC) and is available for public use on NYC OpenData [2]. The dataset contains over 8 million instances, with each instance containing data from the time the incident was created in the system, to the time it was closed. Each instance is represented by exactly 30 features, all of which were extracted.

2.2 Data Bias and Data Leakage

An instance within the dataset is defined as an EMS vehicle being called to transport a patient to the nearest hospital, which follows the timeline provided in Figure 1. Selection bias lies in some areas where citizens know it takes longer for the ambulance to arrive than it would take to get themselves to the hospital by their own means. Therefore, these instances are not represented in the data, although they represent the exact bottlenecks we are trying to detect. Another motivation behind self-transport lies with the hefty cost of EMS transport or proximity of hospitals to incident areas. These cases depend on the severity of the situation and result in missing instances and selection bias. It is important to note that selection bias should be avoided because with bias, results cannot be generalized.

Several features within the dataset occur after sampling time and were therefore removed to avoid data leakage. As shown in Figure 1, events which occurred after the units signaled they were on-scene, were removed. Data leakage

is key when understanding the data, because these features already know the outcome of the target variable. If included, they would overfit the model and skew the results.

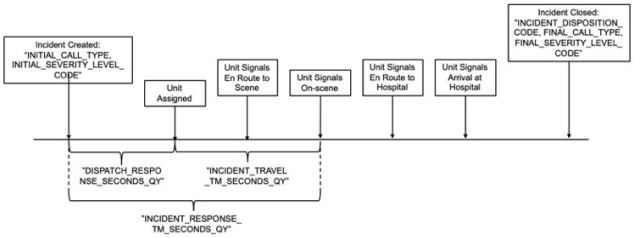

Figure 1. Timeline of an EMS instance.

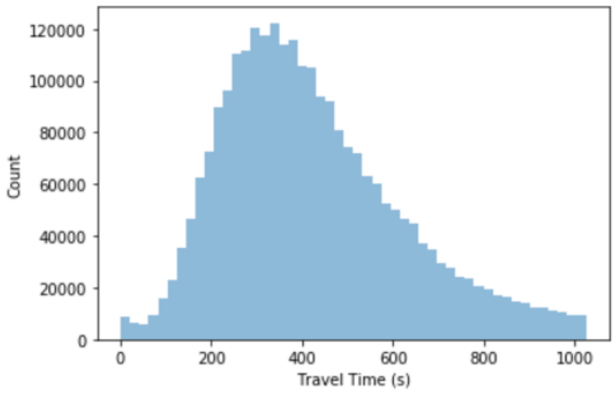

Figure 2. Histogram of travel time.

3 DATA PREPARATION

Due to the large volume of data and computational memory restriction, only records from 2017 and 2018 were utilized in the analysis, accounting for nearly 3 million instances.

3.1 Target Variable

For the purpose of this study, the selected target variable was *'INCIDENT_TRAVEL_TM_SECONDS_QY',* representing EMS vehicle travel time. This specifically represents travel time from the moment EMS has been assigned to the incident to the moment it arrives on-scene.

During the data exploration stage, many outliers were observed. It was recognized that some outliers were due to special circumstances such as large snowstorms in the city, however, there were also instances where recorded travel time was up to 24 hours. These extremes were removed as they were likely erroneous rather than circumstantial. Upon analyzing the descriptive statistics for travel time, outliers were calculated to be any travel time greater than 17 minutes based on the summation of the interquartile range and the third quartile. This caused 161,623 instances to be removed from the dataset, which compared to the size of the dataset was minimal, accounting for less than 6% of instances. The resulting distribution of travel times is shown in Figure 2.

3.2 Data Cleanup

Upon initial investigation of the dataset, all features within the dataset were categorized as either binary, categorical, numerical, or date-time objects.

Of all time-stamped objects within the dataset, *'FIRST_ACTIVATION_DATETIME',* which corresponds with the time at which the unit is en route, was judiciously chosen to aid in the prediction. The remaining time-stamped features were discarded because they were all relatively close to this feature, as well as highly correlated, and therefore would add little value to the analysis. From the *'FIRST_ACTIVATION_DATETIME'*, additional information was extracted, including the month, day of the week, and hour of the travel start time.

It was necessary to remove the time duration feature *'INCIDENT_RESPONSE_SECONDS_QY'* due to issues with collinearity [3]. The feature nearly predicts the target variable because it describes the entire EMS response timeline. Travel time is essentially calculated by subtracting the dispatch response time from the incident response time.

The meaning behind each binary feature was carefully considered during the data cleaning process. Multiple indicator variables allowed for the removal of additional instances. For example, two indicator variables represented whether the calculation components of travel times were valid. Instances with non-valid calculation components were removed from the dataset, which in turn, removed all instances where travel times were not reported, and therefore eliminated the need to perform feature engineering on missing travel time data. As well, indicator variables existed

for hospital transfers and special events. For the purpose of this study, these instances were removed.

Exploring the *'INITIAL_CALL_TYPE'* feature led to some important decisions during the cleaning process. Fake calls, canceled calls, and instances where patients were gone on arrival with no record of EMS travel were all discarded from the dataset because they did not provide any information about travel time.

3.3 Feature Engineering

Multiple feature engineering techniques were performed in order to prepare the dataset for modeling. Binary features were converted to '0' for 'N' values, and '1' for 'Y' values.

Selectively binning features extracted from datetime objects was considered, where month would be binned as time of year (ie. spring, summer, etc) and hour as time of day (ie. morning rush, afternoon, evening rush, etc), however this was later rejected because the methodology seemed too subjective and could potentially introduce bias into the model.

The description of each call type was carefully analyzed for the purpose of creating broader call type categories. This was accepted because the severity level of each incident was still captured in the incident's recorded severity level. The number of unique call types was reduced from 110 unique values to 38 categories.

Missing values were found within all location descriptor features. To treat missing values, the modes of each feature were calculated within each Incident Dispatch Area, and then applied to the missing values based on the instance's corresponding Incident Dispatch Area. Modes were used instead of means, because each location descriptor was a categorical value, and therefore the mean would not provide a useable result.

As the majority of the dataset was comprised of categorical features, which the model would incorrectly interpret as natural ordered relationships, it was necessary to convert the data using one-hot encoding [4]. This type of feature engineering was performed so that variables could be represented numerically for the models to learn from without placing significant weight on any specific category. Since some features consisted of the same numerical categorical values but with different meanings, a prefix was added to the column names to prevent any conflicts during the encoding process. A summary outlining the one-hot encoded features and their corresponding prefixes is provided in Table 1.

Table 1. Descriptions of highest importance features.

One-Hot Encoded Features	**Column Prefix**
'INITIAL_CALL_TYPE'	T
'INCIDENT_DISPATCH_AREA'	A
'BOROUGH'	
'ZIPCODE'	
'POLICEPRECINCT'	P
'CITYCOUNCILDISTRICT'	City
'COMMUNITYDISTRICT'	Comm
'COMMUNITYSCHOOLDISTRICT'	Schl
'CONGRESSIONALDISTRICT'	Con
'Activation_Month'	Month
'Activation_Day'	Day
'Activation_Hr'	Hour

After performing feature engineering, the final dataset consisted of 2,726,929 instances with 603 features. Working with the dataset proved to be challenging given its large size, therefore a random seed "2019" was added to the dataset and 50,000 rows were extracted to be used for the remainder of the analysis. The encoding and random seed sampling reduced the dataset from 490 MB to 61 MB.

4 MODELING AND EVALUATION

4.1 Feature Selection

Feature selection removed irrelevant features for the purpose of creating models with better performance, interpretability and quick run times. Through feature selection, 84 "important" features were identified. To avoid overfitting, feature selection was only performed on the training dataset.

Due to the nature of one-hot encoding, features were highly correlated with each other. Therefore, analyzing

correlation and covariance matrices would not be useful. Instead, other tools were utilized including L1-regularization, sklearn's Decision Tree and Random Forest Classifier with criterion 'entropy', and sklearn's SelectFromModel from the 'feature_selection' module for feature selection.

To perform L1-regularization, the Logistic Regression linear model from sklearn was used with a Lasso (L1) penalty, monitoring the c penalization. The 'SelectFromModel' object from sklearn observed L1-regularization zeroed out the coefficients of 445 variables.

For tree classifiers, the built-in feature importance attribute with implicit binning was utilized to compute the normalized mutual information of each feature [4]. Correlated features were given equal or similar importance, but overall reduced importance compared to the same tree built without correlated counterparts [5]. The decision tree selected 10 features to have feature importance greater than zero, while the Random Forest specified 84 features.

The results of the processes make sense. Prior to one-hot encoding, dispatch area was speculated to provide more specific information than borough and the two were 0.97 correlated, confirmed by feature selection. Zip code, Police Precinct, Community District and Community School District were all highly correlated amongst each other as well. Additionally, all months, days of the week, and hours of the day were selected as important features to keep in the model for predictions. Table 2 provides informative descriptions of some of the most important features.

Table 2. Descriptions of highest importance features.

Selected Features	**Descriptions**
'INITIAL_SEVERITY_ LEVEL_CODE'	The priority assigned at the time of incident creation.
'DISPATCH_RESPONSE_ SECONDS_QY'	The time elapsed in seconds between incident_datetime and first_assignment_datetime.
'HELD_INDICATOR'	Indicates that for some reason a unit could not be assigned immediately
'T10'	Call type corresponding to stroke.
'T1'	Call type corresponding to altered mental status.
'A10'	Incident dispatch area corresponding to K4 in Brooklyn
'10457'	Zip code in Bronx
'11207'	Zip code in East Brooklyn
'P79'	Police Precinct in Bedford Stuyvesant, Brooklyn
'P103'	Police Precinct in Jamaica, Queens

After using feature selection to better understand the data, baseline models were developed to test which of these features produced the best results for modeling travel times.

4.2 Baseline Modeling

For the continuous response time prediction problem, the historical average response time was used as the baseline prediction. The average baseline error was 222.32 seconds.

To determine whether the selected features created a good baseline model, a Random Forest regressor was developed with 1,000 randomly selected instances as shown in Figure 3.

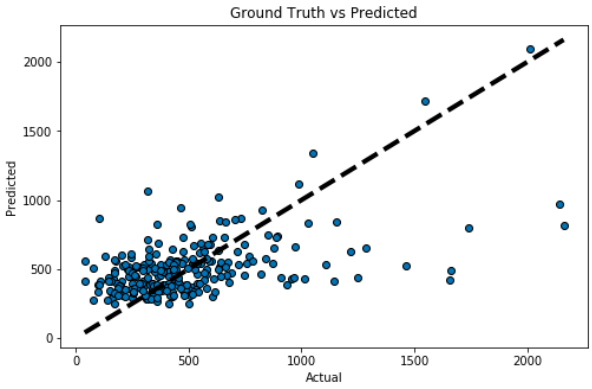

Figure 3. Results of baseline random forest model with 100 trees.

Without hyperparameter tuning, the resulting MSE was very high at 83,599 seconds. This is likely because tree-based algorithms give preference to features with high cardinality and are biased towards these variables. Several other linear models were created with hyperparameters and optimized through GridSearch and cross-validation in an attempt to

predict continuous travel times. However, model accuracies remained low, leading to the conclusion that regression algorithms could not accurately predict the travel time. MSEs resulting from these linear models are provided in Table 3.

Since no linear model could improve on the baseline, it was concluded that data science could not help predict a continuous distribution of response times for EMS vehicles. In order to realistically solve this problem, the target was modified to be a categorical feature, and the nature of the problem was changed from regression to classification.

Table 3. MSEs for continuous EMS travel times with regression algorithms.

Algorithm	**MSE (seconds)**
Linear Regression	35,901
Least Squares Regression with OLS Fitting	35,007
Linear Regression with L1-regularization	35,682
Decision Tree Regressor	66,996
Random Forest Regressor	36,795
Tree-Based XGBoost Regressor	96,070
Linear Regression-Based XGBoost Regressor	37,609
SGD Regressor	1.05E+21
Ridge Regression	35,512
Huber Regressor	42,778

A baseline was created for the classification problem by first discretizing the target variable into five bins. The second bin, which ranged from 205 to 410 seconds contained the greatest number of occurrences for every instance in the training set. By setting the predictions for the validation to this bin, the base rate accuracy was estimated to be 43.2%.

4.3 Modeling Approach

A sensitivity analysis was performed where the number of bins was varied with each model run to see how it would affect the prediction. During the modeling process, the trade-off between bias and variance on binning widths were explored. Bins were selected to be of equal width to represent equal durations of time. By reducing bin width, the amount of bias decreased because smaller bin widths provide a higher resolution of the travel time. Increasing the number of bins produced models with higher variance and lower accuracies due to overfitting on the training data.

Since the problem was inherently multiclass, sklearn's Multiclass and multilabel algorithms landing page was used to research models that would best solve the problem [6]. Models were fit using the following estimators: Random Forest, AdaBoost, Logistic Regression, Linear Discriminant Analysis, Support Vector Machine, and k-Nearest Neighbors. With sensitivity performed for each algorithm, a total of 20 models each were compared against one another.

4.4 Results & Prediction

4.4.1 Random Forest

Random Forests make good predictions based on an aggregation of individual tree performance which implicitly do bootstrap aggregation, variance reduction, and accuracy improvement [8]. As expected, the random forest estimator performed well out-of-bag. Hyperparameter tuning was performed to optimize the maximum tree depth and the minimum number of samples required to be at a leaf node. The minimum number of samples required to split an internal node is usually good at 2 and was left as the default setting [9]. Regardless, adding more complexity to the model decreased performance as shown in Figure 4.

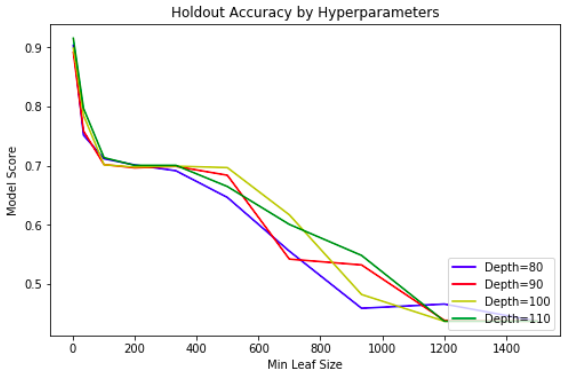

Figure 4. Hyperparameter tuning for random forest algorithm.

Model accuracy likely worsened with optimization because the large depth caused overfitting. For most depths, overfitting occurred when leaves became too small. As a result of hyperparameter tuning, the maximum tree depth and

the minimum number of samples required to be at a leaf node were left with the default setting in order to prevent any limitations.

4.4.2 AdaBoost

AdaBoost is similar to Random Forest in that they both tally up the predictions made by each decision tree within the forest to decide on the final classification. However, the predictions made by each decision tree in the AdaBoost model had a varying impact on the final prediction made by the model [10]. Because of this, average accuracy was computed amongst five cross-validation folds of AdaBoost runs. Additionally, AdaBoost iteratively corrects the mistakes of the weak classifier and improves accuracy by combining weak learners, making it not prone to overfitting [11]. Optimizing the AdaBoost model for the number of weak learners to train iteratively increased performance, but also increased training time exponentially. The AdaBoost model reached the highest accuracy of nearly 98% with 1100 'n_estimators'. Additionally, the confusion matrix of the model showed that the algorithm yielded low false positives and had a high recall of 0.98.

4.4.3 Logistic Regression

Logistic Regression is one of the most popular algorithms used for classification problems, and was therefore used in this prediction problem. However, the algorithm was developed to perform binary classification which can explain why the model accuracies produced were so low [7].

4.4.4 Linear Discriminant Analysis

Unlike Logistic Regression, which is more applicable to binary classification problems, the Linear Discriminant Analysis (LDA) classifier is widely used for multi-class classification problems, and thus directly applicable to this problem. For model fitting, singular value decomposition (SVD) or least-squares solution (LSQR) with or without shrinkage can be utilized [12]. In order to determine the best fit model for the problem, the solvers were tested for accuracy by modifying the number of bins. It was evident that the SVD solver yielded the highest accuracies as the number of bins increased.

4.4.5 Support Vector Machine

Support Vector Machines (SVM) are an extension of LDA as they also classify instances based on linear relationships between features [7]. One of the major drawbacks of SVM is that it is most sensitive to the points closest to the hyperplane, and therefore if these points change, the hyperplane re-aligns and completely alters the model. Default parameters were utilized for SVM as the change in accuracy was negligible when performing cross-validation. It was decided that the linear nature of SVM was not suitable for this problem.

4.4.6 k-Nearest Neighbors

The k-Nearest Neighbors (kNN) algorithm classifies data points by other points around it and their corresponding neighborhood. k-NN is largely based on the number of neighborhoods, and therefore an optimal k is defined as the balance between bias and variance. A high *k-value* leads to a low variance and high bias for which every instance gets the same prediction. A low k, on the other hand, has a high variance, low bias and is more prone to overfitting [13]. In order to evaluate the most suitable k for this problem, the error rate was computed for k-values ranging from 1 to 30 [14]. Additionally, k-NN is largely based on the weighting metric which can either be set to 'uniform', where all points in each neighborhood are weighted equally, or 'distance', where points are weighed by the inverse of their distance [15]. Hyperparameter tuning aided with determining which weighting metric was better-suited for this problem. Results from uniform weighting, Euclidean distance (p=2), and Manhattan distance (p=1) were compared against one another as shown in Figure 5.

All weighting metrics exhibited similar behavior that once k reached approximately 20, the error rate leveled off, indicating that k=20 was an optimal value for the number of neighborhoods. All metrics yielded similar model accuracies, however on average, the Manhattan distance metric

performed slightly better especially as the number of bins increased. For this reason, the k-NN model using the Manhattan distance metric with k=20 was utilized as the best-suited model.

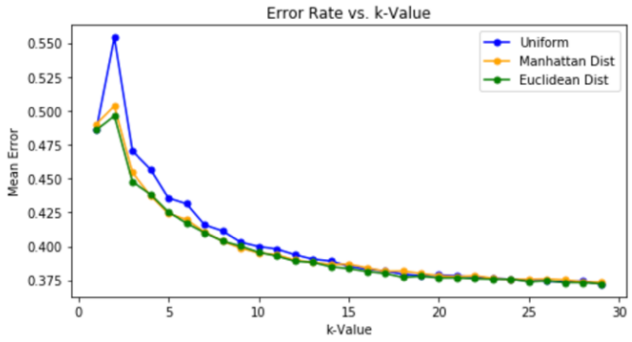

Figure 5. Hyperparameter tuning for *k*-value using weight functions.

4.4.7 Further Interpretation

The learning curves of all algorithms are provided in Figure 6, showing the relationship between sample size and accuracy for each selected number of bins. Accuracy was used as the evaluation metric for ease of comparing each data mining algorithm side-by-side. It represents the ability of the selected algorithm to correctly predict within which time interval the EMS travel time will fall in.

Bias can be shown in the learning curves. Larger training sets exhibited better performance of the classifiers. This would likely continue until a certain point where more data would not significantly improve the model.

It is evident that k-NN performed consistently well for all bin sizes, and that model accuracy did not change by much even with the smaller sample sizes. SVM also performed consistently well, following the model accuracies of k-NN quite closely, however upon increasing the bin size to 25, model accuracy reduced significantly. AdaBoost was also observed to be a good classifier, however not as good as the other two algorithms. The AdaBoost algorithm requires large sample sizes to achieve higher model accuracy, and as mentioned previously, the algorithm is very slow, therefore it is not feasible to use moving forward.

This makes sense as classification trees are more flexible than linear and logistic regression. Classification trees select a single attribute at a time whereas linear classifiers use a weighted combination of all the attributes. Additionally, a classification segments the instance recursively while a linear classifier places a single decision surface through the entire space.

As expected, a lower number of bins yielded higher accuracies, however, the results are based on much coarser travel time durations. For example, five bins correspond with making a prediction within a 205 second time interval, which is not as useful as a prediction with smaller bin widths. However, it is essential to understand that a larger number of bins corresponds with potentially overfitting the model and therefore it is not as reproducible or generalizable. In conclusion, to combat the bias-variance trade-off, it is best to use a combination of moderate bin sizes when using the models as a prediction tool. Therefore, the most informative and accurate model was k-NN with 25 bins as it was able to correctly predict travel time within 41 seconds, 74% of the time.

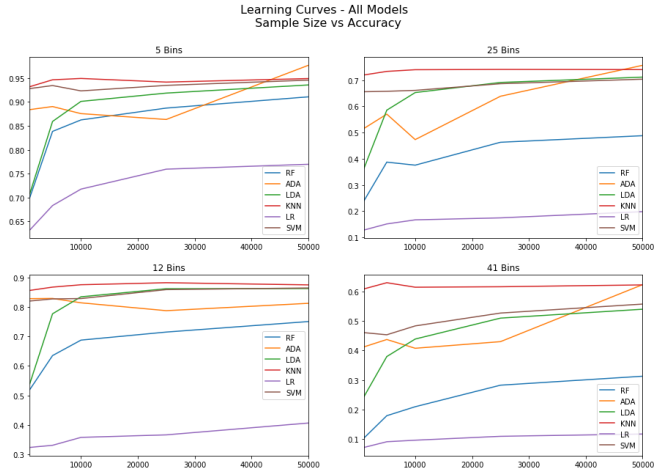

Figure 6. Learning curves for all algorithms.

SHAP interpretation, shown in Figure 7, aids with the interpretability of the selected model by displaying results across the entire population [16]. Looking at all the instances from the training data, it is evident how much each feature contributes to the target variable. It is not surprising that initial severity code is associated with higher and lower

predictions. It was identified as an important determining feature in feature selection as well.

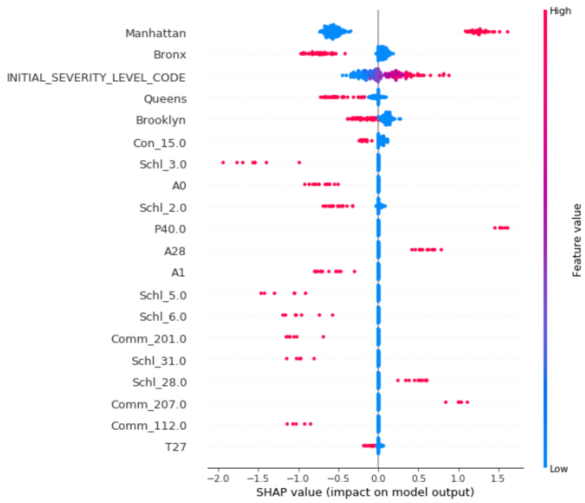

Figure 7. SHAP variable impact on model.

SHAP also has the capability of reviewing a single instance of a prediction and the contributions of the features to that prediction as shown in Figure 8. This randomly selected instance shows that zip code 11435 was one of the features that was associated with a higher prediction. This may be due to the fact that the incident occurred in Jamaica, Queens, which is considered to be a higher activity and poorer area in the city.

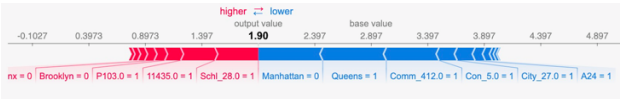

Figure 8. SHAP Variable Impact on a single instance.

For reference, a heat map of the predicted response time bins for the best algorithm is provided in Figure 9. The visualization was made with Tableau.

5 DEPLOYMENT

While the model results are predictive, there are descriptive implications that can assist the city's government with decreasing EMS response times. The results of the data mining work can be deployed for city planning and response improvement purposes. City policies that attempt to mitigate traffic congestion with dedicated lanes is an example of how

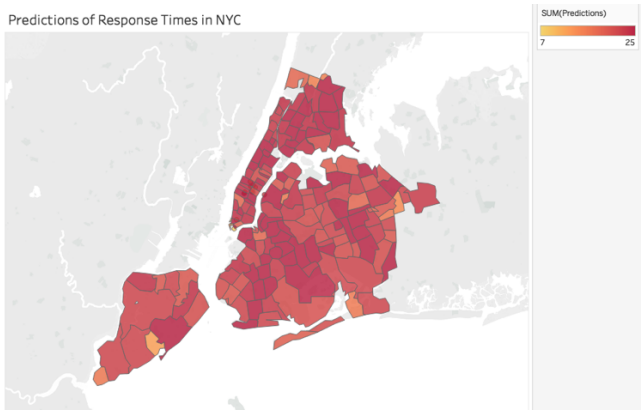

Figure 9. Predictions of EMS travel times in NYC.

to implement changes in areas that the model identifies as problematic. Initial severity levels were considered an important feature, so training operators about the importance of its accuracy can also be a result of deploying this model.

In an actual production system, an overprediction between the severity and response time must be closely monitored. There is a correlation between these two variables, however not all initial response times are reported accurately and are often changed after sampling time. Because of their post-sampling nature, they were excluded from the model because of data leakage concerns, but a curious official should monitor the effects of when the initial and final severity levels are not consistent.

It should be made aware that the results presented are generalizable only to standard EMS vehicle emergencies and do not include response times for special events or transfers between hospitals. Rush hour, weather, and special events that occur consistently throughout the year also affect the model and attempted to be addressed through one-hot encoding. Nevertheless, these are issues that should be made aware during deployment.

There are social implications of these results that require ethical considerations. For example, the feature zip code in our model is heavily tied to income. From analyzing zip codes that have the fastest response times, it can be hypothesized that these are higher-income areas. This can lead to disparate treatment of certain neighborhoods instead of treating all neighborhoods fairly. It is preferred that results

are used for group fairness instead of individual fairness so all areas in the city could be considered "safe" when it comes to EMS response times.

One issue with the proposed models is that it is more practical and informative for city officials to be able to predict the continuous response times instead of an arbitrary range of response times for EMS vehicles. To mitigate this risk, future applications of this work could use project-specific models rather than the generalized ones on sklearn. For learners that use a fixed number of bins, more data could potentially increase performance up to a point. Unfortunately, this was constrained by computational memory.

REFERENCES

[1] Durkin, Erin. "FDNY Ambulance Response Times Worsen despite De Blasio's Budget Boost." *Nydailynews.com*, New York Daily News, 9 Apr. 2018, www.nydailynews.com/new-york/ambulance-times-worsen-nyc-pol-budge-boost-article-1.2541573.

[2] Fire Department of New York City (FDNY). "EMS Incident Dispatch Data: NYC Open Data." *EMS Incident Dispatch Data | NYC Open Data*, 21 Mar. 2019, data.cityofnewyork.us/Public-Safety/EMS-Incident-Dispatch-Data/76xm-jjuj.

[3] James, Gareth, et al. An Introduction to Statistical Learning: with Applications in R. Springer, 2017.

[4] D'Alessandro, Brian. Intro to DS Module 9 - Data Prep for Modeling. October 2019, https://newclasses.nyu.edu/. PowerPoint Presentation.

[5] Dubey, Akash. "Feature Selection Using Random forest." Medium, Towards Data Science, 14 Dec. 2018, towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f

[6] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. https://scikit-learn.org/stable/modules/multiclass.html

[7] Provost, Foster, and Tom Fawcett. Data Science for Business: [what You Need to Know About Data Mining and Data-analytic Thinking]. Sebastopol, Calif.: O'Reilly, 2013. Print.

[8] D'Alessandro, Brian. Intro to DS Module 12 - Ensembles. November 2019, https://newclasses.nyu.edu/. PowerPoint Presentation.

[9] "3.2.4.3.1. Sklearn.ensemble.RandomForestClassifier." Scikit, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[10] Maklin, Cory. "AdaBoost Classifier Example In Python." Medium, Towards Data Science, 15 May 2019, towardsdatascience.com/machine-learning-part-17-boosting-algorithms-adaboost-in-python-d00faac6c464

[11] Navlani, Avinash. "AdaBoost Classifier in Python." DataCamp, DataCamp, 20 Nov. 2018, datacamp.com/community/tutorials/adaboost-classifier-python

[12] "Sklearn.discriminant_analysis.LinearDiscriminantAnalysis." Scikit, scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html.

[13] D'Alessandro, Brian. Intro to DS Module 13 - kNN. November 2019, https://newclasses.nyu.edu/. PowerPoint Presentation.

[14] Robinson, Scott. "K-Nearest Neighbors Algorithm in Python and Scikit-Learn." Stack Abuse, Stack Abuse, 15 Feb. 2018, stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/.

[15] "Sklearn.neighbors.KNeighborsClassifier." Scikit, scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.

[16] Dataman. "Explain Any Models with the SHAP Values — Use the KernelExplainer." Medium, Towards Data Science, 6 Nov. 2019, https://towardsdatascience.com/explain-any-models-with-the-shap-values-use-the-kernelexplainer-79de9464897a

CONTRIBUTIONS

Project work was distributed after thoroughly discussing ideas and performing exploratory analysis on the dataset in weekly meetings. The git repository can be accessed at: https://github.com/sophia-tsilerides/CodeBlue.git

Ilana cleaned the data set, converted attributes to numerical data, partitioned the data, developed and optimized Logistic

Regression and SVM models, analyzed results and helped write the report.

Amanda performed preliminary statistical analysis of the dataset and wrote code for consolidating call types and one-hot encoding features. She also wrote code to discretize the target variable into equal time intervals, developed and optimized Linear Discriminant Analysis and k-NN models, wrote the code for SHAP, analyzed results and aided in report writing.

Sophia wrote code and performed analysis for feature selection, ran baseline models, developed and optimized the Random Forest Classifier and AdaBoost models, plotted the learning curves, created the heat map, interpreted the SHAP figures, analyzed results and helped write the report.