NYC EMS Response Time Project

Jay Valdillez, Elana Stettin, Megan Conlon, Morgan Walker
EECS 349 - Northwestern University

Task Overview

Our task is to predict the response time of an ambulance dispatched to a location in New York City based on attributes such as zip code, severity level, and incident dispatch area. This project is important because it will predict how long it will take Emergency Medical Services to reach a location in New York City and also could show a correlation between response times and socioeconomic factors. This could help the city re-allocate resources to improve response times for certain areas of the city. The data set we are using is the EMS Incident Dispatch Data from the City of New York OpenData repository. This data set has 32 attributes and 4.83 million examples.

I. Methods and Classifiers

We initially took our large data set and filtered it using the valid incident response time indicator to remove the data that did not have valid incident response time. Then, we selected attributes that were important or potentially relevant to predicting an EMS response time. These specific features are explained more thoroughly in the Data Set section of the report.

We were interested in how well we could predict a response time by using continuous labels vs. discretized labels. Our original data set had a specific response time for each example, which we used for our continuous label analysis. From our data set of continuous labels we employed two methods to bin the data to make discretized EMS response time labels. Our first method separates the data by making sure that each bin has the same number of examples. As a result, the range of response time varies for each bin. Our other method separated the bins by doing equal intervals of response times, so each bin had a varied number of examples in them.

Using the continuous and the discrete fixed time data sets, we attempted to predict the EMS response time by using Random Forest, AdaBoost, MultiLayer Perceptrons, and Gaussian Naive Bayes. From these results we hope to be able to compare how much data is lost with discretion using different classifiers.

Additionally, we were interested in how smaller data sets performed in comparison to larger data sets. We felt as though this would be useful if a city wanted to analyze a smaller area within a city or only had a small amount of data available. The information we gained also allowed us to run our classifiers on fewer examples. In order to note the differences between the data set sizes we trained and tested up to 10,000 examples using 10-fold cross validation on Random Forest, Gaussian Naive Bayes, Support Vector Machine, Multi-Layer Perceptron, Linear Regression, and AdaBoost classifiers and plotted the corresponding learning curves using the Scikit python package. The data sets that we used for analyzing the learning curves were our two discrete data sets - fixed time and fixed examples.

II. Data Set

Size: The data set had a total of 4 million examples. However, we were unable to run that many examples so we randomly partitioned the data into smaller subsets. We created learning curves to understand how much information gain we receive with more data. We were able to successfully graph learning curves over 10,000 examples. When we discretized the data, we also varied the number of bins that we used to sort the data (15, 30, and 50 bins for each method - number of examples and time).

Features: Initial call type, Initial severity level, Held indicator (indicates if a unit could not be assigned immediately), Borough (county-level administrative divisions of NYC), Atom (smallest subset of a borough where incident was located), Incident dispatch area, Zipcode, Police Precinct, City Council District, Community District, Community School District, Congressional District, Special Event Indicator (indicates that the incident was a special event, ex: NYC Marathon)

Label: Incident Response Seconds

III. Results

Discrete vs. Continuous Data:

Naive Bayes: Running our discretized data through Weka's Naive Bayes classifier, we obtained a 93.44% accuracy, compared to only .35% correctly classified instances for the continuous data set. This discrepancy is large, but makes sense because Naive Bayes performs better on discrete data.

Random Forest: Random Forest gave us a higher correlation coefficient of .481 for the continuous data set, compared to .24 for the discrete data set. We were surprised by this because decision trees perform better on discretized data.

Multi-layer Perceptron: Multilayer perceptron performed better on the continuous data set with a .451 correlation coefficient, compared to .235 for discrete. This makes sense because neural networks perform better with nonlinear continuous data with a lot of input features.

AdaBoost: AdaBoost performed better with the discrete data with 96.01% accuracy, compared to only .927% accuracy with the discretized data. Again, this is a large discrepancy, but it makes sense because AdaBoost is made up of multiple weak decision trees and decision trees theoretically perform better on discrete data.

Learning curves:

We saw that the learning curves plateaued so we inferred that more than 10,000 examples does not continue to provide information.

The learning curves show that using fixed time intervals to create each bin was a much more successful strategy than using a fixed number of examples to create each bin. This could be because the vast majority of examples are categorized into a fewer number of bins. Additionally, we found that the number of bins also affected the accuracy of the models. Overall, a smaller number of bins was more accurate because there were fewer options for classifying the data, so the likelihood of selecting the correct bin was higher. Using a smaller number of bins, however, is less informative about the response time because each bin has a larger range of time for the response times.

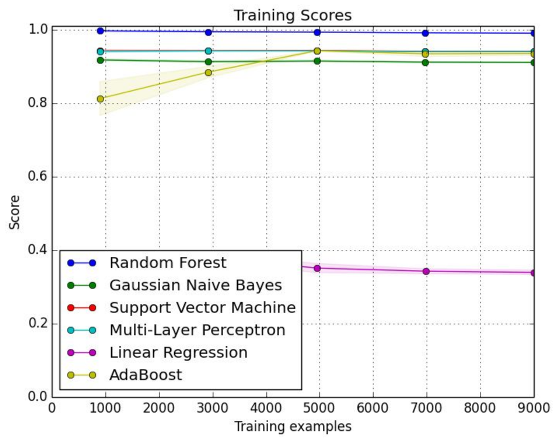
Random Forest: The random forest algorithm consistently scored the highest in the training curves using a fixed number of bins. This is because this algorithm uses multiple decision trees to train the examples and outputs the most common class value of the trees. When trained on every example, decision trees have a tendency to overfit the training data, which would explain the lower cross validation scores.

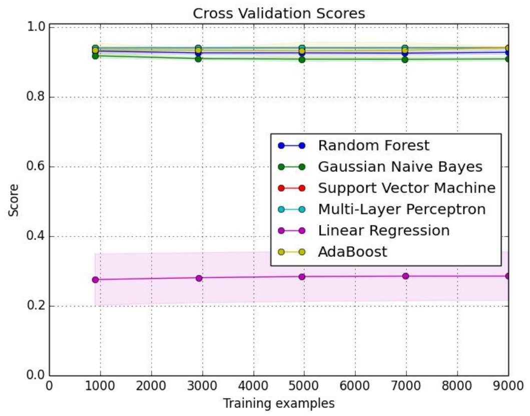
Gaussian Naive Bayes: Gaussian Naive Bayes is a supervised learning method that assumes the continuous values in each class are distributed according to a Gaussian distribution. This method scored well and consistently in training, but not as well as Random Forest, multilayer perceptrons, and support vector machines. This may be because discretizing the continuous values by binning them may lose discriminative information.

Multi-Layer Perceptron: Multilayer perceptrons are another method of supervised learning. This method performed in the middle compared to our other algorithms and showed minimal improvement with an increasing number of examples.

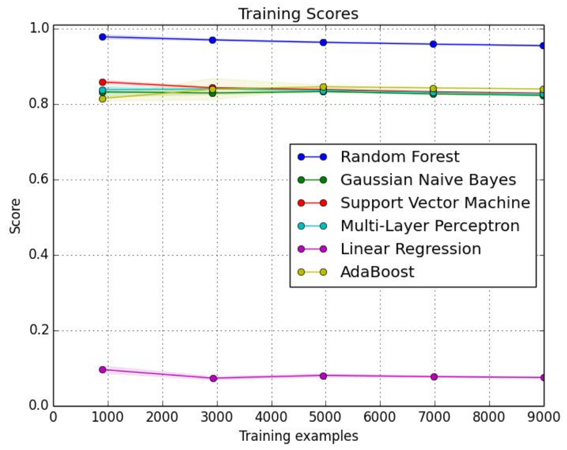
AdaBoost: Adaptive boosting (adaboost) fits a classifier on the dataset and then fits copies of the classifier on the same dataset with weights adjusted for the incorrectly classified instances. Adaboost's training curves tended to show gradual improvement with increasing examples but sometimes performed worse with more than 5000 examples.

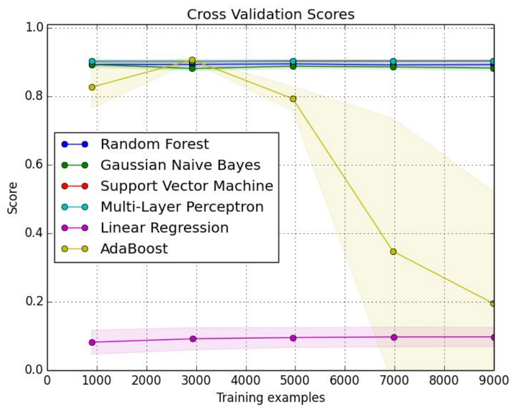
Training and 10-fold Cross Validation for 15 response time intervals:



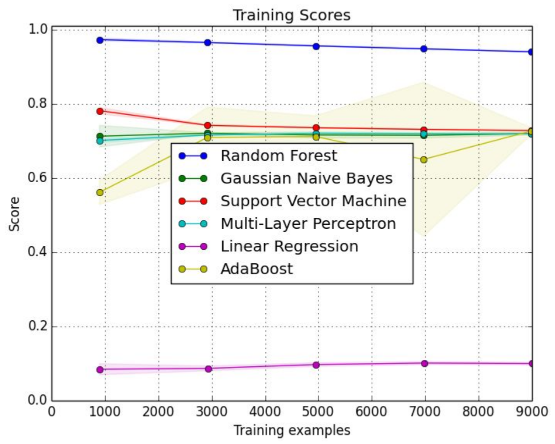


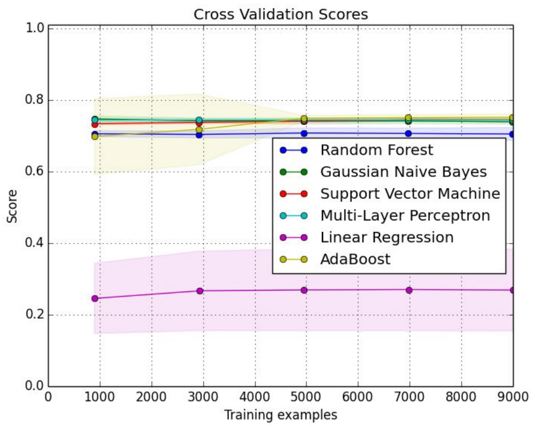
Training and 10-fold Cross Validation for 30 response time intervals:



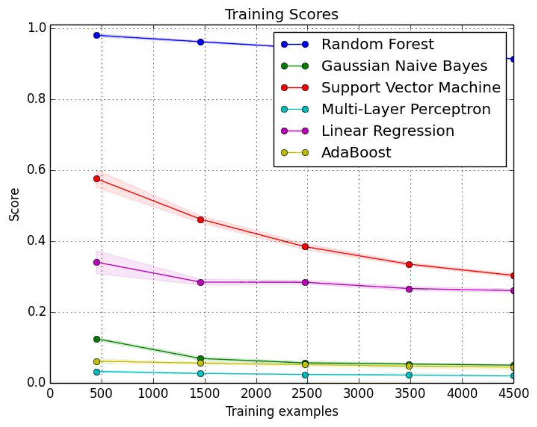


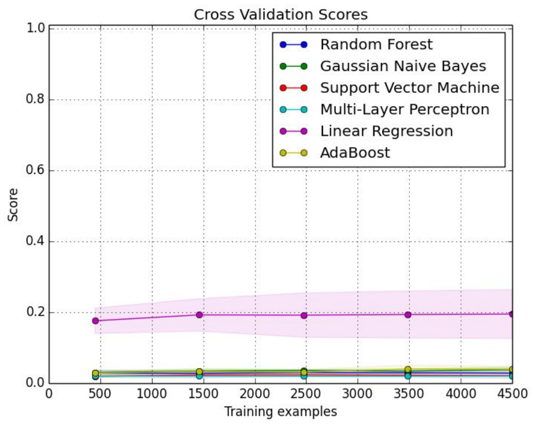
Training and 10-fold Cross Validation for 50 response time intervals:



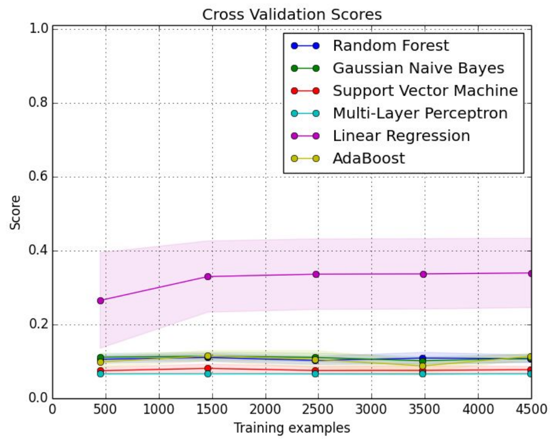


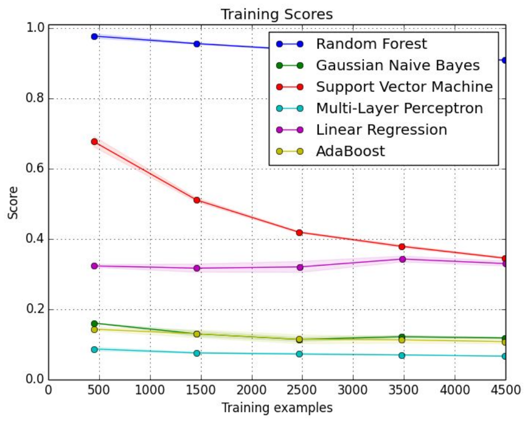
Training and 10-fold Cross Validation with 100 examples per bin:





Training and 10-fold Cross Validation with 333 examples per bin:





<u>Relating Average EMS Response Time in Zip Code to Average Household Income:</u>

We calculated the average EMS response times for each zip code. We could not find very reliable and consistent data for all of the average household incomes for each zip code. However with the information we did find, when looking at the zip codes with the fastest average response times compared to the slowest, we could not draw any conclusions between the average income and the response time.

IV. Future Work

We could look into other social observations that may contribute to slower response times in certain areas. Atom was actually more indicative than zip code, which makes sense because it encompasses a smaller and probably more representative area of New York. If we had access to other data such as the socioeconomic status of the atom, we could draw more conclusions.

This information is not only important for city planning and response improvements, but it may actually have some social applications as well. For example, we could join our data set with income based on zipcode, and from this we could analyze which zipcodes have the fastest response times and potentially theorize why this would be the case. With many attributes in our original data set, the possibilities of joining data to learn more information is expansive.

In the future, we could work on developing models to predict continuous response times instead of predicting a range of time that the instance's response would fall within. We could also use more detailed and project-specific models, rather than generalized algorithms which may not fit the data very well when incorporating all of the features we included. We could use a larger number of examples when we increase the bin-size for the learners that used a fixed number of bins.

V. Work Distribution

Elana cleaned the data set, converted attributes to numerical data, partitioned the data, and wrote code to classify with bins based on equal time intervals.

Morgan classified the data into bins based on a fixed number of examples per bin, helped with the learning curves function, and compared zip code and household income.

Megan wrote code to plot the learning curves, developed the project webpage, analyzed results and helped write the report.

Jay built models in Weka and analyzed the results, helped write the report, and also helped with developing project webpage.