Assignment 6 - Term Assignment
Level - 4000

Olivia Davis

December 9, 2022

1 Abstract

With the need for quick response times for medical emergencies becoming more prevalent, data scientists have turned to analyzing EMS response times and predictors associated. And because this affects the health and safety of all, there is a large amount of data available that tracks the times from when 911 is dialed to when EMS teams arrive at the hospital with the patient. Analyzing the factors that affect the individual response windows is crucial, and could cut down overall EMS response time drastically as a result of learning what has commonly slowed these recorded times in the past. In addition, specifically looking into how busier days of the year, including holidays, might affect the response time could give insight as to how EMS teams should plan ahead in order to ensure that they deliver the fastest response time possible. The hypothesis of "Busier days of the year (such as Christmas, New Years, etc.) have longer 911 response times" is formed off of the logic that there will be significantly more traffic on these days, which would theoretically interfere with the travel time of the EMS vehicle.

The dataset I will be using to conduct this research is comprised of EMS response times, split into windows, with the corresponding dates. This data was collected for the entirety of New York City, split into their respective boroughs. I will be focusing specifically on the boroughs of Manhattan and Brooklyn, as to assess any drastic impact busier areas of New York might have on the response times. These boroughs are also popular with tourists, which makes this analysis even more important to conduct, and share with EMS response teams.

2 Data Description

The dataset I chose pertains to EMS Response times in the city of New York. I was interested in this data because of the many implications this research has towards human health and safety. The data is categorized by borough, and contains different windows to split up the overall response time. The First Activation refers to the time it takes the first unit to be on their way to the location of the incident. First On Scene tells the time it takes the first unit to arrive on scene. And First Hospital Arrival refers to the time it takes the first unit to arrive at the hospital. The response time ends when someone is First On Scene. Furthermore, the dataset also records the days of the calls, which are very important for the analysis that I am conducting on the data.

The data is sourced from the New York City Fire Department. Most of the data types contained in the dataset are either numeric or characters. The numeric types correspond mainly to the recorded seconds for the different response windows. Additionally, the file EMS_incident_dispatch_data_description_final.xlsx provides documentation for the dataset and describes each variable in depth (FDNY, 2022).

3 Exploratory Data Analytics

For the dataset training_final.RDS I performed a lot of clean up for invalid entries, as well as excluding some of the data that was not relevant to my analysis.

I began by going through the entire dataset and analyzing the graphs produced in the original file for loading the data. The following bar chart was produced from this file, and displays the EMS incidents by year with the year displayed on the x-axis and number of incidents on the y-axis.

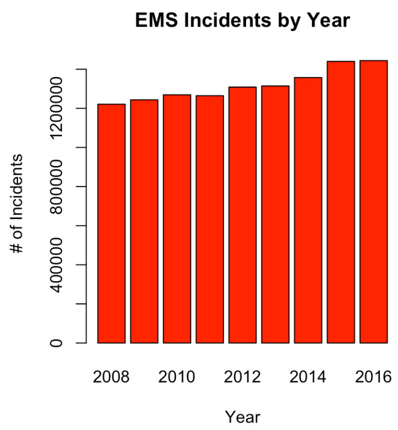

Figure 1: Bar chart of the EMS incidents by year

The second graph I found interesting was very relevant to my hypothesis. It displays the year on the x-axis and number of calls on the y-axis. It is split up by month and year, and you can see that the calls spike in particular months.

The next thing I focused on was cleaning the data. I went through every column I would be dealing with in my analysis, and replaced the NA values that were there previously. For the dates, I replaced the NA values with the current day and time. Since the day that replaced those values is not a special day, it shouldn't impact the results negatively. I then split the data by two boroughs, Manhattan and Brooklyn, assigning them to their own respective data frames. Going through each data frame, I deleted the columns that did not relate to my hypothesis, or had data types incapable of being converted to numeric. And with the remaining variables, I converted all of the values to be of type numeric.

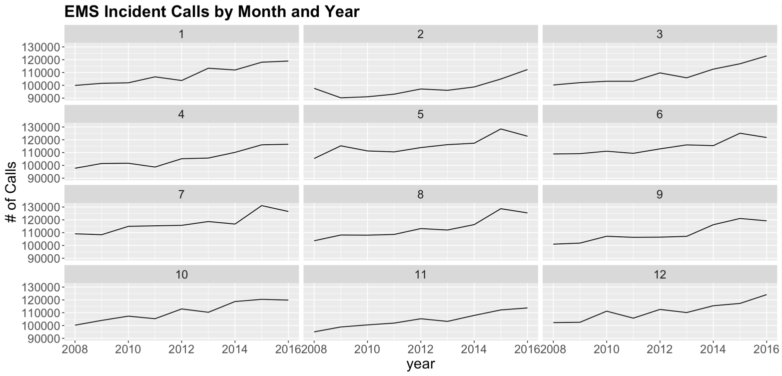

Figure 2: EMS Incident Calls by Month and Year

4 Analysis

To begin the analysis, I will take a look at more of the graphs produced initially in the training_data_load.RDS file. The plot shown below displays the EMS incidents by year with the year on the x-axis and the number of incidents on the y-axis. It is clear to see that as the years go on, there is a steady increase in EMS incidents, not only calls. The number of cases starts at 1,250,000 and ends close to 1,500,000 - this is almost a 250,000 case increase in 8 years. With this steady increase, the analysis of this data becomes even more prevalent.

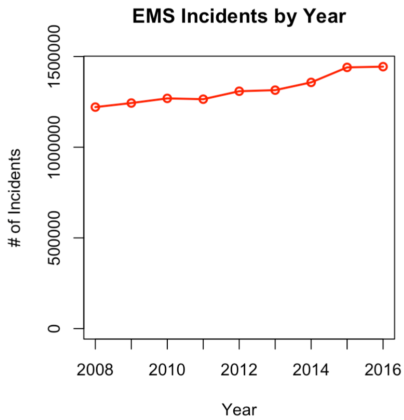

Figure 3: Scatterplot of the EMS incidents by year

The next plot displays a more detailed look at how many incidents are recorded by month, and these multiple line plots are being generated by year as well. The number of cases spike in the summer, but are more subdued in January and late fall, before spiking back up in December. This graph supports the hypothesis I previously stated, seeing as there are major holidays in both July and December that could elicit more cases, and thus, give a slower response time. And again, as the years go on, the number of incidents naturally increase.

The plot shown below shows the EMS incident dispositions next to a count of the incidents on the y-axis, while year is displayed on the x-axis. The incidents range from the patient being gone on arrival or being pronounced dead to the patient being treated and transported to the hospital. This gives a clear visual of the range in these cases, and shows that EMS incidents don't always have a response time, since some cases are cancelled, or unfounded. Additionally, the response time might not be valid if the

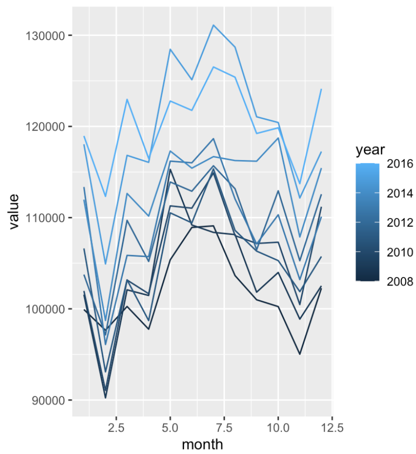

Figure 4: Graph of the EMS incidents by month and year

patient is gone upon the arrival of the EMS team.

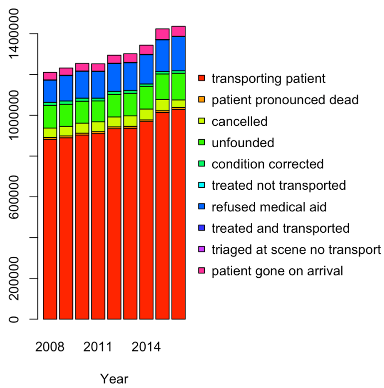

Figure 5: Bar chart of EMS incident dispositions

Previously mentioned, I did a lot of data cleaning in order to run the models without having any problems. Originally, I replaced the invalid entries and then attempted to complete the analysis with all the variables in the dataset. I quickly ran into many different errors associated with data typing. Thus, going back to the original data, I cleaned out a lot of the columns that wouldn't help me prove or disprove the hypothesis, such as:

- Initial/Final Severity Level Code - This is a very useful variable to be used in analysis, but this project exclusively focuses on the dates and times within this dataset, thus, it would not help to use this as a predictor or the predicted.
- Valid Incident Response Time Indicator - Indicates whether or not the calculations for Incident Response Seconds are valid or not. This variable is of data type "char", and is either a "Y" or "N". It was not completely necessary for the analysis, and cannot be directly converted to a numeric data type.
- Initial/Final Call Type - This variable is of type "Class 'labelled' num". It was not recognized as numeric in the models, and after failing to convert the type to numeric manually, I decided to exclude those two variables from the datasets.

The revised datasets only contain seventeen columns, and the dates were converted into numeric data types. As previously mentioned, the invalid entries in those variables were replaced with the current date and time. Additionally, the numeric value of the date is the number of seconds from an arbitrarily chosen date (1970 in this case) to the date previously held there.

A source of bias was introduced when I replaced the numeric invalid entries with 0s. I only remembered this mistake once I was finished modeling the data, but introducing 0's into the response window variables would indicate that they have a faster response than other entries, which is not the case. A better approach would have been to replace the invalid entries with the average, or base the value on similar rows throughout the dataset. This is an area that can be improved upon for future analysis.

5 Model Development and Application of model(s)

5.1 Random Forest

For the first model, I chose to implement Random Forest for regression in order to take a closer look at what factors influence the prediction of a variable. Using this model I can also look into the importance of each predictor, and thus measure how much impact two variables have on each other. After splitting the data into two distinctive data frames, I analyzed them separately by running the model twice. Starting with the data for Manhattan, the first model I devised has the predicted value as incident_dt, which refers to the date of the incident. The predictors I chose were the following: dispatch_response_seconds_qy, incident_response_seconds_qy, incident_travel_tm_seconds_qy. All of the predictors refer to different windows of the response time, and therefore are good for supporting my hypothesis because this format gives a date influenced by factors of time. The output of this first model is shown below.

```
Call:
 randomForest(formula = incident_dt ~ dispatch_response_seconds_qy +      incident_response_seconds_qy +
 incident_travel_tm_seconds_qy,      data = TrainSet, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 7.591482e+15
                    % Var explained: -11.54
```

Figure 6: Output snippet of model1

Then, I went on to measure the importance of the model and obtained the following results. The measure of %IncMSE for incident_response_seconds_qy was 50.18370, ranking highest of the three predictors, and its correpsonding IncNodePurity measure was 1.121038e+19. incident_travel_tm_seconds_qy had an %IncMSE of 49.60215 and IncNodePurity of 1.129314e+19 - the highest measure of IncNodePurity. And lastly, dispatch_response_seconds_qy has a %IncMSE measure of 47.50405, with IncNodePurity being 8.735956e+18.

I first used the built in varImpPlot() to visualize the results, but then graphed the %IncMSE measures in a more visually appealing way.

Notice in both of these graphs dispatch_response_seconds_qy ranks below the other two predictors, ranking it last in variable importance for this prediction model. Furthermore, since the measure of %IncMSE is the more reliable measure of variable importance and incident_response_seconds_qy ranks first in both graphs, this is the most influential variable for incident_dt in the model.

The results of this model indicate that the prediction of incident_dt is not independent of response window incident_response_seconds_qy. The more influence a variable of of the response time has over predicting the date of the incident, and vice versa, the more support is given to the hypothesis. Backing the relationship between the date and the corresponding times is the first step of validating a pattern between the two variables.

The procedure for implementing the second predictive model was quite similar. Still working with the data for Manhattan, the second model has the predicted value as incident_response_seconds_qy,

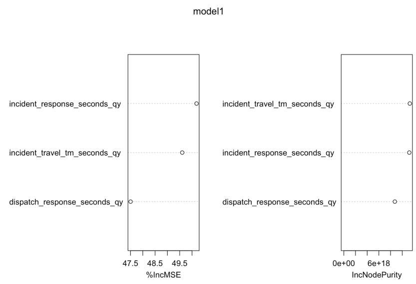

Figure 7: Using built in varImpPlot() to measure importance

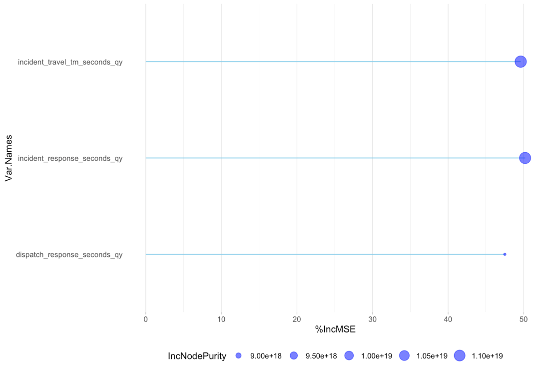

Figure 8: Taking a closer look at the %IncMSE measures for the variable importance of model1

which refers to the overall time from the start of the incident to when the first unit arrives on scene. The predictors I chose were the following: incident_dt, first_on_scene_dt, incident_close_dt. All of the predictors refer to different dates corresponding to the response time windows. Similar to the first run, this model's results will support my hypothesis because this format gives an overall response time influenced by the recorded date(s) of the incident. The output of this second predictive model is shown below.

```
Call:
 randomForest(formula = incident_response_seconds_qy ~ incident_dt +      first_on_scene_dt + incident_cl
ose_dt, data = TrainSet, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 463984.5
                    % Var explained: -29.39
```

Figure 9: Output snippet of model2

Again, I then went on to measure the importance of the model and obtained the following results. The measure of %IncMSE for incident_dt was 10.56838, ranking highest of the three predictors, and its correpsonding IncNodePurity measure was 1072317400. first_on_scene_dt had an %IncMSE of 10.29545 and IncNodePurity of 1110545453 - the highest measure of IncNodePurity. And lastly, incident_close_dt has a %IncMSE measure of 10.29643, with IncNodePurity being 1081665543.

The graph using built in varImpPlot() to visualize the results is shown below, as well as the second graph to focus on showing the %IncMSE measures clearly.

In the graphs above, using %IncMSE as the more reliable measure of varaible importance once again, we have that incident_dt is the most influential variable for predicting incident_response_seconds_qy in the model.

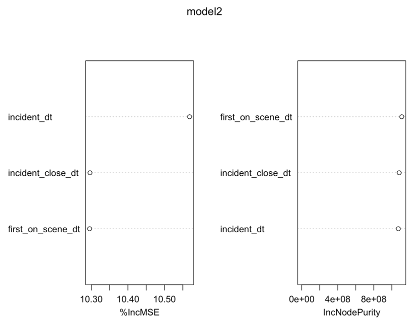

Figure 10: Using built in varImpPlot() to measure importance

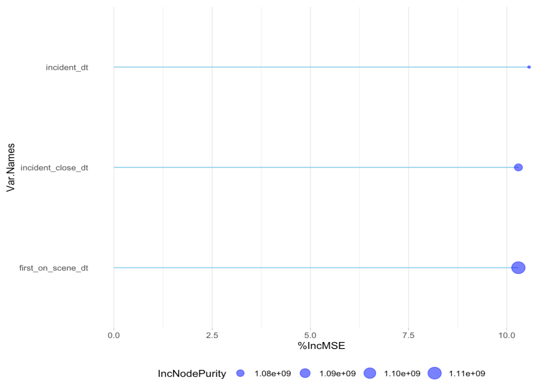

Figure 11: Taking a closer look at the %IncMSE measures for the variable importance of model2

Next, I move on to analyze the data for the borough of Brooklyn. I used the same predictive models in order to compare the results between the two boroughs. The output for the two predictive models is shown below:

```
Call:
 randomForest(formula = incident_dt ~ dispatch_response_seconds_qy +      incident_response_seconds_qy + incident_travel_tm_seconds_q
y,      data = TrainSet, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 7.498333e+15
                    % Var explained: -11.82
```

Figure 12: Output snippet of model1 for the Brooklyn borough

```
Call:
 randomForest(formula = incident_response_seconds_qy ~ incident_dt +      first_on_scene_dt + incident_close_dt, data = TrainSet, impo
rtance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 262301.3
                    % Var explained: -30.74
```

Figure 13: Output snippet of model2 for the Brooklyn borough

The outputs between the two boroughs for each model are very similar, and I have also graphed the importance of the variables for measures %IncMSE and IncNodePurity shown below.

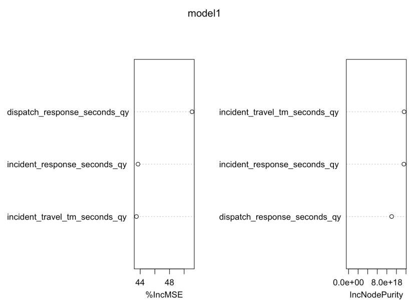

Figure 14: Using built in varImpPlot() to measure importance

By analyzing the graphs, it is clear that the boroughs have differing results for variable importance for the first model. With Manhattan, we have that incident_response_seconds_qy and incident_dt are the most influential variables for the predictors associated with models 1 and 2, respectively. While for the Brooklyn borough, dispatch_response_seconds_qy ranked highest in %IncMSE for model 1. And for model 2, the same results followed, with incident_dt ranking as the most influential variable in the model.

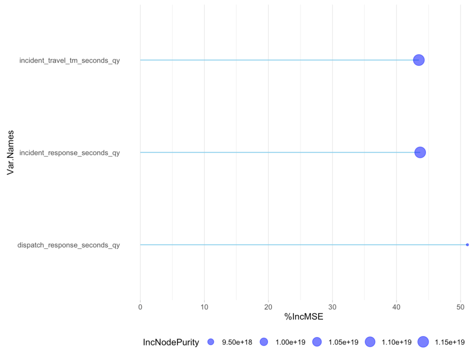

Figure 15: Taking a closer look at the %IncMSE measures for the variable importance of model1 for borough Brooklyn

5.2 KNN Regression

For the second model, I implemented KNN Regression to have more results regarding what factors influence the prediction of variables for the response windows, and their corresponding date(s). Using this model I further investigate the importance of each predictor, and the error rate for how accurate the predicted values of the variable are. I once again ran the model twice, once for each borough, and began with the Manhattan data frame first. The first step of my implementation was to check the variances of the related features of the dataset using the function var(). The output of the variances for select columns is shown below:

```
> var(randManhattanDF[,3])
[1] 141169.6
> var(randManhattanDF[,1])
[1] 295124.4
> var(randManhattanDF[,2])
[1] 491705.1
```

Figure 16: Output for the variances of Dispatch Response Seconds (1), Incident Response Seconds (2), and Incident Travel TM Seconds (3)

Now, the group of variables to be standardized as predictors are as follows: incident_dt, first_assign_dt, first_act_dt, first_on_scene_dt, first_to_hosp_dt, incident_close_dt. Thus, only this subset of the variables is to be standardized, excluding the variable that is going to be predicted - incident_response_seconds_qy. Similar to model 2 of the Random Forest data model, I am using the different dates given to predict the total response time variable. Once again, I examined the variance of the columns after the standardization, which was uniformly "1" for every column previously checked.

Then, the test set and train set are formed by obtaining a sample of the Manhattan data frame, and standardizing that sample within the already standardized Manhattan data frame, then taking another standardization excluding the sample.

Next, the predicted Incident Response Seconds is captured in the call to knnreg() in order to perform the regression on the model. The formula for the model is as shown:

$$\text{trainIncidentRS} \sim \text{incident\_dt} + \text{first\_assign\_dt} + \text{first\_act\_dt} + \text{first\_on\_scene\_dt} + \text{first\_to\_hosp\_dt} + \text{incident\_close\_dt}$$

using the trainData set as the data given to the function, with a k value of 10. Then, the predicted values can be found by using the predict() function and passing in the predicted Incident Response Seconds and testData. The output displaying these predicted values is shown below:

The error for this first run of KNN regression is found by taking the mean of the following difference:

```
> predictedValues
  [1]  588.8  513.9  531.0  404.9  431.9  599.1  563.3  490.0  635.7  511.6  688.2  413.0
 [13]  475.2  748.2  494.3  777.9  931.9  510.9 1003.6  361.2  427.5  539.1  510.8  475.3
 [25]  638.5 1184.3  433.9  578.9  335.7  650.1  717.4  495.5  654.1    0.0  405.8  585.0
 [37]  702.8  830.4  611.1 1491.6  582.7  477.6  482.4  729.7  725.3  578.0   68.0  510.4
 [49]  400.7  531.8  466.6  374.4  450.0  534.1 1114.5  610.8  625.7  253.6  568.5  705.8
 [61]  661.8  458.9  726.6    0.0  572.7  665.5  451.9  451.9  698.3  862.5  959.5  540.3
 [73]  575.2  352.0  428.2 1043.0  615.0  564.1  591.6  461.0  768.7  390.6  454.6    0.0
 [85]  474.7  721.8  488.9  505.9  492.9  497.9  489.6  472.8  690.7  541.0  456.8  499.9
 [97]  709.8  427.6  600.4  519.1  637.9  320.0  482.9  442.0  755.1 1057.7  466.9  549.5
[109]  502.2  359.1  366.0  827.1  453.4  655.8  427.3  406.5  569.9  477.0  430.0  442.8
[121]  438.0  606.7  337.3  585.2    0.0  485.0  463.6  437.5  550.0    0.0  551.6  427.6
[133]  323.5  507.9  525.7 1013.0    0.0  366.7  407.1  726.4  350.5    0.0 1221.4  560.4
[145]  550.3  427.7  478.0 1043.1  657.8  361.2  469.1  975.3    0.0  917.4  415.4  531.9
[157]  801.5  245.9    0.0  429.8  427.3  534.9  442.8  430.8 1038.9  530.9  512.3    0.0
[169] 1003.4  325.8  601.1 1422.2  364.1  505.3  519.1  556.5  392.5  502.1  502.5  598.3
[181]  437.6    0.0  796.6  468.5  448.1  538.1  569.9  995.7  618.8  653.1  428.6  669.7
[193]  456.8  436.6  665.0  618.7 1585.5  305.7  410.7  812.8  497.8  581.9  412.1  461.0
[205]  436.2  296.8  444.2  295.7  624.5  411.1  393.9  686.1 1057.7  497.3  486.2  389.8
[217]  542.3  774.8  836.2  535.7  620.4  439.9  879.2  397.4  588.8  474.9  700.1  582.4
[229]  433.9  425.4 1100.0  829.1  647.9  467.7  425.1  545.1  479.3  685.2 1559.1  586.5
[241]  429.7  573.7  436.8  787.3  337.7 1279.3  515.4  577.1  485.3  490.5  741.7  383.6
[253]  510.5  418.5  531.4 1018.2 1588.0  828.1  578.2  461.5  851.3  792.8  512.1  358.7
[265]  584.3  351.3 1156.6  309.8  416.6 1105.2  518.5  510.8  459.0  713.8  300.3  588.3
[277]  517.2  417.9  445.8  408.1  497.4  754.4  480.8  294.1  383.0  719.6  507.7  364.1
[289]  508.4  781.1  456.9  521.6  847.5  433.1  412.9  752.9  442.3  426.4  432.2  429.8
[301] 1062.3  460.7  547.6  411.6  609.2 3725.2  549.1  748.6  515.5  537.3  649.8  571.6
```

Figure 17: Plot of the predicted values for the Manhattan borough's Incident Response Seconds

testIncidentRS - predictedValues

The obtained error from this calculation is "10.88472"

Running the model a second time for the Brooklyn borough dataset, we have the following predicted values displayed below:

```
> predictedValues
  [1]  435.0000    0.0000  453.2000  475.3000    0.0000  664.0000  461.2000  564.2000
  [9]  541.8000  395.5000  434.3000  466.1000  509.0000  568.6000  496.7000    0.0000
 [17]  532.8000  412.0000  804.5000  359.0000  485.4000    0.0000  579.6000  462.6000
 [25]  349.9000    0.0000  430.3000 1541.3000  672.6000  674.7000  544.5000  335.7000
 [33]  422.7000  417.9000  534.7000  545.9000  391.4000  385.9000  402.3000  423.5000
 [41]  458.9000  623.6000  389.3000  442.3000  508.2000  614.9000  415.1000  409.3000
 [49]  392.0000  572.8000  789.5000  369.0000  498.5000  651.4000  416.8000  662.5000
 [57]  523.8000  503.9000  468.1000  336.9000    0.0000  445.8000  390.6000  343.3000
 [65]  963.7000  386.5000  433.6000  587.9000  701.9000  452.4000  655.6000  402.3000
 [73]  541.4000  343.1000  768.5000  529.8000  470.0000  394.9000  527.7000  509.1000
 [81]  379.5000  695.8000  401.0000  407.6000    0.0000  614.6000  438.7000  417.6000
 [89]  555.3000  384.5000  366.6000  516.1000  645.9000  371.7000  466.4000  348.4000
 [97]  461.5000 1220.8000  963.7000  611.9000  406.2000  464.3000  615.7000  590.0000
[105]  874.1000  321.6000  471.0000  384.7000  507.4000  326.9000  564.1000  388.1000
[113]  509.5000  413.0000  481.1000  517.7000  614.3000  412.7000  344.6000  571.0000
[121]  362.3000  322.6000  428.6000  459.5000  447.5000  434.1000  585.4000  525.7000
[129]  380.9000  541.2000  343.1000  313.0000 2226.2000    0.0000  354.6000  480.7000
[137] 2253.7000  487.6000  625.5000  455.8000  544.1000  657.6000  545.1000  419.0000
[145]  645.5000  373.1000  534.9000  518.1000  506.3000  380.7000 1017.4000  432.1000
[153]  355.5000  401.3000  525.6000  831.4000    0.0000  476.7000  527.8000  486.0000
[161]  498.1000  624.4000  373.2000  346.4000  542.4000    0.0000  365.1000  634.2000
[169]  331.8000    0.0000  562.7000  333.3000  371.2000  559.8000  348.2000  382.0000
[177]  739.3000  429.6000  483.5000  446.4000  490.6000  637.0000  454.4000  502.3000
[185]  546.9000  364.2000  406.1000  623.6000    0.0000  471.0000  435.5000  497.7000
[193]  561.3000  385.9000  600.2000  578.5000  439.0000    0.0000  451.6000  400.3000
[201]  346.3000  406.6000  500.9000  457.7000  353.4000  374.3000  550.3000    0.0000
[209]  502.1000  418.4000  432.3000  581.5000  568.3000  552.5000  547.3000  689.4000
[217]  647.9000  336.4000  519.1000  592.7000  450.5000  514.7000 1017.4000  507.1000
[225]  348.2000  363.5000  385.5000  541.1000  455.0000  370.1000  391.4000  731.7000
```

Figure 18: Plot of the predicted values for the Brooklyn borough's Incident Response Seconds

One thing to notice is that running this borough through KNN regression with the identical model as before gives a significantly lower error of "2.019571".

In order to test the regression further, I implemented another predictive model for both boroughs in order to compare the results. This second model has the predicted value as incident_dt, referring to the date of the incident. The predictors I chose were the following: dispatch_response_seconds_qy, incident_response_seconds_qy, incident_travel_tm_seconds_qy. The implementation follows the same steps as previously described, except this model only has three predictors, instead of six, and so the formula passed in to the knnreg() function is the following:

$$\text{trainIncidentDate} \sim \text{dispatch\_response\_seconds\_qy} + \text{incident\_response\_seconds\_qy} + \text{incident\_travel\_tm\_seconds\_qy}$$

The predicted values for the Incident Date are shown in the output below:

The obtained error from this calculation is "-502784.6"

Similarly, when run again for the Brooklyn borough, we obtain the following predicted values for Incident Date:

The obtained error for this run-through is "-3748254".

```
> predictedValues
  [1] 1332576563 1363495871 1335878049 1378194261 1366572160 1361758722 1363122394
  [8] 1389986048 1346106637 1361320319 1346562485 1318874156 1388956908 1335594054
 [15] 1383010573 1377316092 1320661653 1326563890 1372979813 1334681199 1368497368
 [22] 1332697758 1337832802 1362312729 1352522492 1308156445 1344631056 1339608267
 [29] 1395752705 1337336956 1353757285 1387839711 1404978444 1346415453 1326638481
 [36] 1305229650 1328148740 1346415453 1357201763 1344693472 1282799109 1343396736
 [43] 1355223612 1320625319 1304657806 1289843511 1343629936 1396787544 1334200136
 [50] 1335432727 1330518909 1352621068 1360693331 1344979101 1373649130 1316826462
 [57] 1354389601 1332079369 1336384245 1386464637 1351261898 1354703024 1338406282
 [64] 1381626778 1375645129 1264689485 1387153798 1329070867 1411574184 1325044594
 [71] 1340839178 1335387032 1358885877 1404978444 1336447833 1291264025 1322361998
 [78] 1292503900 1350204651 1340196972 1324615927 1304926916 1346415453 1367872491
 [85] 1368494544 1358172744 1356919218 1385786097 1393792977 1445939853 1302236558
 [92] 1365950921 1344464345 1380885480 1319607522 1338816362 1292427090 1331780661
 [99] 1343694025 1335555490 1372588953 1368197320 1372205082 1346415453 1301557243
[106] 1348678835 1381258112 1365910519 1372672222 1333345863 1363998067 1327583109
[113] 1387920509 1306781394 1389723469 1371142224 1373825485 1362979176 1365556404
[120] 1346929979 1420836369 1396604726 1375403016 1370836208 1345199355 1367895234
[127] 1390422450 1326317968 1335870480 1380755351 1297342018 1374112313 1348590012
[134] 1357096927 1307009024 1370291396 1366929066 1368865512 1319716590 1346260715
[141] 1365056472 1341890726 1344979101 1385457177 1310128473 1334042579 1354385924
[148] 1359286920 1377424986 1347641465 1288263984 1331453875 1368975110 1373816762
[155] 1383209513 1329756560 1379899175 1315699573 1331630295 1290424322 1321517672
[162] 1333124782 1320998711 1348121608 1359105233 1357196280 1348897486 1363807725
[169] 1346103472 1327655643 1353898104 1334029673 1374961901 1320950956 1349644107
[176] 1391949546 1395064148 1407123270 1325731833 1292675322 1323821550 1348930559
[183] 1352245537 1315441384 1355712091 1309861226 1337518994 1329580558 1330068157
[190] 1342385492 1375763098 1370523109 1366229249 1333724561 1369042338 1351416793
```

Figure 19: Plot of the predicted values for the Manhattan borough's Incident Date

```
> predictedValues
  [1] 1313635312 1367766513 1331159320 1355505356 1352025293 1380107670 1396164161
  [8] 1350670758 1301702124 1306239821 1287027286 1362609056 1322878494 1351712327
 [15] 1363796071 1330624951 1368452600 1335494036 1354446347 1328853885 1293113936
 [22] 1339356500 1289212156 1366947776 1343425363 1336714319 1379419791 1336981494
 [29] 1334107624 1298662623 1367411769 1351706098 1377073232 1318154208 1336714319
 [36] 1308229189 1344933913 1336649018 1357203219 1332559437 1317553731 1342068541
 [43] 1380650596 1375888752 1343328719 1366954681 1339616212 1326243064 1384239571
 [50] 1345717285 1378288351 1348048720 1283028386 1374916683 1345208785 1343296329
 [57] 1375964458 1320328241 1356889602 1301761145 1367766513 1339376206 1362332002
 [64] 1406259756 1322146885 1342900486 1323913524 1356688789 1329201953 1360446375
 [71] 1311144763 1374096994 1378333766 1273595583 1329610224 1355696265 1340779465
 [78] 1350879915 1358206340 1384416707 1309840141 1372833504 1301044752 1341905447
 [85] 1359708978 1333033272 1347403318 1344950143 1294354120 1355987700 1343378680
 [92] 1366001191 1335835147 1353065868 1345538025 1354283659 1339944025 1345496382
 [99] 1364801656 1351045086 1295186410 1341300712 1342303001 1343937818 1389188586
[106] 1350100556 1328115941 1378369720 1390117321 1332945713 1323132015 1354663124
[113] 1331650083 1350928574 1324760690 1390601631 1412174952 1352055434 1323461098
[120] 1299029479 1366868799 1329840160 1322063968 1310179964 1334783380 1401383196
[127] 1309282584 1380188265 1340452835 1314190282 1321556402 1338289000 1332569692
[134] 1333135403 1392160792 1359197425 1358588297 1344937439 1313955852 1351357241
[141] 1338546177 1374382552 1342520586 1301003011 1366191386 1391606724 1328937531
[148] 1304049491 1357940817 1338394723 1330230176 1340558870 1354981540 1329141705
[155] 1321062788 1355771434 1353411636 1342075373 1358662680 1376887030 1376021052
[162] 1306074683 1388513284 1409042626 1330370135 1326826377 1371166768 1350842735
[169] 1355575466 1367998656 1380820369 1374637610 1377037012 1373024871 1363303979
[176] 1335062174 1385677312 1314937823 1362037909 1327787064 1410002016 1360203080
[183] 1353651844 1377282145 1309396444 1359601089 1323656230 1322747134 1336714319
[190] 1319259107 1379641868 1319108820 1326416486 1365621506 1379652296 1357396273
[197] 1333514989 1369945727 1348599379 1309728042 1340522497 1336714319 1339010143
```

Figure 20: Plot of the predicted values for the Brooklyn borough's Incident Date

5.3 Optimization

I made a necessary change to the amount of data that was being used in order to optimize the implementation. With the full dataset, the implementation was not efficient, and even threw some errors pertaining to vector memory being exhausted. In order to fix this, after splitting the two boroughs into their respective data frames, I created a new dataset from each that only contained a sample of the original values. This sample was chosen randomly with the only criteria being that the number of rows taken be capped at 10,000. This gave an adequate amount of data to be used and analyzed, giving an accurate result, while still allowing for the most efficient implementation possible. I created these samples at the forefront of each data model. The following line of code was used to take a subset of the data:

$$\text{randManhattanDF} < - \text{manhattanDF[sample(nrow(manhattanDF), size=10000),]}$$

5.4 Results

The predictive models that I formulated for the Random Forest algorithm results indicated that the predicted variables were not independent of the predictors. Thus, for example with model 1, the Incident Date is dependent on the predictor incident_travel_tm_seconds_qy. And for model 2, the Incident Response Seconds is dependent upon incident_dt. This dependency shows that there is a correlation between the variables of the response time and the variables of those corresponding date(s). This correlation implies that one may influence the other. And if a variable of time influences what day is chosen in a predictive model, we have that a slower response time would be more likely to predict a day that had heavier traffic patterns. While I did not analyze traffic patterns and have no verification for this, the dependency between variables implies that the hypothesis is correct, since traffic patterns are already proven to be heavier on busier days of the year such as Christmas, New Years, 4th of July, etc. The same results follow with the expanded models I ran with the KNN regression algorithm. KNN's predicted values show a close fit to the actual values, validating the dependency between the variables once more.

6 Conclusions and Discussion

The goal of my analysis was to assess how busier days of the year affect 911 response times and, furthermore, to possibly address whether busier days correlate to slower response times. What I have definitively shown through this analysis is that the date and time variables do affect each other, since they have shown through their predictive models to have a strong dependency between them. As previously stated, a slower response time would be more likely to predict a day that had heavier traffic patterns. And the dependency between variables implies that the hypothesis is correct, since traffic patterns are already proven to be heavier on busier days of the year such as holidays, days with parades, etc. Upon first looking at the data, I predicted that the variables associated with time would have an impact on the response time variables, thus, this result was what I was expecting.

I chose to implement the models Random Forest and KNN regression models in order to investigate what factors influence the prediction of these time and date variables. With Random Forest, I looked into the importance of each predictor, and thus, measured how much impact each set of variables had on each other. Similarly, for KNN regression, I looked into more results for importance of each predictor, along with the error rate for how accurate the predicted values of the variable are.

As I went through the project, I focused more on the dependency and importance of predictors and variable to be predicted than I had originally thought I would. From this, I was able to establish a clear correlation between certain variables, which ended up aiding my project more than the route I would have taken otherwise.

If I were to continue this analysis, I would aim to establish a more clear result that proves the hypothesis. Part of why I was not able to accomplish this with the current project is due to my own inexperience. However, I did aim to accomplish as much as I could with my current skill set.

NOTE* The page requirements are met - I initially wrote the document in Google Docs, then copied it in to Latex.

7 References

Analysis of a Random Forests Model - Journal of Machine Learning Research. Apr. 2012, https://jmlr.org/papers/volume13/biau12a/biau12a.pdf.

Biecek, Przemyslaw. DALEX: Explainers for Complex Predictive Models in R. Nov. 2018, https://research.jmlr.org/papers/v19/18416.html.

Fire Department of New York City (FDNY). "EMS Incident Dispatch Data: NYC Open Data." EMS Incident Dispatch Data — NYC Open Data, 11 Apr. 2022, https://data.cityofnewyork.us/Public-Safety/EMS-Incident-Dispatch-Data/76xm-jjuj.

System Model for Prediction Analytics Using K-Nearest Neighbors Algorithm. Oct. 2019, https://www.researchgate.net/publication/338014147_System_Model_for_Prediction_Analytics_Using_K-Nearest_Neighbors_Algorithm.