Predicting Motor Vehicle Collisions in New York City

by Alex Fung, Viswesh Krishnamurthy, Tony Lee, Patrick Osborne

Abstract

Technological progress in the world has unarguably improved the quality of life for the average person in many ways. The age of the automobile has shaped the way in which work, play and live our lives. Roadways, buildings, cities and entire countries have been designed to accommodate motor vehicles. As automobile technology has advanced making cars faster and capable of more advanced maneuvers, so has our concern with the safety of these vehicles. Entire disciplines such as traffic management are devoted to optimizing numerous factors to ensure the safe and efficient movement of people and goods. As we move into the age of data, all stakeholders in the automobile industry must effectively collect and utilize the wealth of information available to better meet their goals if progress is to continue. In this project, we take the position of a law enforcement agency, the New York City Police Department, as they seek to best utilize their resources in the context of responding to traffic collisions in the city.

Background

At the end of 2017 in New York City, there were 1,923,041 cars registered to residents of the city. (https://nyc.streetsblog.org/2018/10/03/car-ownership-continues-to-rise-under-mayor-de-blasio/) This already-significant number does not include the heavy flow of vehicles of those who visit the city or are simply passing through. By contrast, the New York City Police Department (NYPD) budgets for a headcount of 35,822 uniformed officers (http://council.nyc.gov/budget/wp-content/uploads/sites/54/2017/03/056-NYPD.pdf - page 4), distributed across 77 police precincts (geographic divisions of the city). On-duty officers/traffic enforcement agents are allocated to each precinct to enforce traffic laws and handle emergency and administrative response to traffic incidents (such as collisions). NYC has been collecting traffic data, including specific data on vehicle collisions since 2014 to support "Vision Zero" , a traffic safety initiative which has the goal of eliminating traffic fatalities. (https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95/data)

Objective

The objective of our analysis is to develop a supervised, prediction model using Machine Learning techniques and the CRISP-DM framework (cite textbook) on the available collision data to predict whether there will be a collision in a specified police precinct at a specified time. The intent of predicting this data is to inform the NYPD's optimal assignment of limited officers and resources across the 77 police precincts.

Data Analysis

The data set that supports this analysis is sourced from the NYC Open Data project. The title of the data set is "Motor Vehicle Collisions – Crashes". It contains entries for every collision recorded within New York City limits by NYPD agents beginning July 1st, 2012 up to the present day. There are approximately 1.65 million entries in the data set.

Data Dictionary

Data dictionary sourced from https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95/data - “MVCollisionsDataDictionary_20190813_ERD.xlsx”.

Table 1: Data Dictionary - Motor Vehicle Collisions – Crashes

Feature	Feature.Description
COLLISION_ID	Unique record code generated by system
ACCIDENT_DATE	Occurrence date of collision
ACCIDENT_TIME	Occurrence time of collision
BOROUGH	Borough where collision occurred
ZIP CODE	Postal code of incident occurrence
LATITUDE	Latitude coordinate for Global Coordinate System, WGS 1984
LONGITUDE	Longitude coordinate for Global Coordinate System, WGS 1984
LOCATION	Latitude , Longitude pair
ON STREET NAME	Street on which the collision occurred
CROSS STREET NAME	Nearest cross street to the collision
OFF STREET NAME	Street address if known
NUMBER OF PERSONS INJURED	Number of persons injured
NUMBER OF PERSONS KILLED	Number of persons killed
NUMBER OF PEDESTRIANS INJURED	Number of pedestrians injured
NUMBER OF PEDESTRIANS KILLED	Number of pedestrians killed
NUMBER OF CYCLIST INJURED	Number of cyclists injured
NUMBER OF CYCLIST KILLED	Number of cyclists killed
NUMBER OF MOTORIST INJURED	Number of vehicle occupants injured
NUMBER OF MOTORIST KILLED	Number of vehicle occupants killed
CONTRIBUTING FACTOR VEHICLE 1	Factors contributing to the collision for designated vehicle
CONTRIBUTING FACTOR VEHICLE 2	Factors contributing to the collision for designated vehicle
CONTRIBUTING FACTOR VEHICLE 3	Factors contributing to the collision for designated vehicle
CONTRIBUTING FACTOR VEHICLE 4	Factors contributing to the collision for designated vehicle
CONTRIBUTING FACTOR VEHICLE 5	Factors contributing to the collision for designated vehicle
VEHICLE TYPE CODE 1	Type of vehicle based on the selected vehicle category
VEHICLE TYPE CODE 2	Type of vehicle based on the selected vehicle category
VEHICLE TYPE CODE 3	Type of vehicle based on the selected vehicle category
VEHICLE TYPE CODE 4	Type of vehicle based on the selected vehicle category
VEHICLE TYPE CODE 5	Type of vehicle based on the selected vehicle category

Initial Data Exploration and Cleaning

To begin our exploration of the data set, we'll look at a summary table for each feature. This may help inform which we would like to keep and which to remove during feature selection.

[1] “

Row 1:”

CRASH.DATE	CRASH.TIME	BOROUGH	ZIP.CODE	LATITUDE	LONGITUDE
01/21/2014: 1161	16:00 : 24074	:499865	Min. :10000	Min. : 0.00	Min. :-201.36
11/15/2018: 1065	17:00 : 23613	BRONX :160097	1st Qu.:10304	1st Qu.:40.67	1st Qu.: -73.98
12/15/2017: 999	15:00 : 23171	BROOKLYN :355543	Median :11206	Median :40.72	Median : -73.93
05/19/2017: 974	18:00 : 21767	MANHATTAN :273153	Mean :10828	Mean :40.69	Mean : -73.87
01/18/2015: 961	14:00 : 21258	QUEENS :305510	3rd Qu.:11237	3rd Qu.:40.77	3rd Qu.: -73.87
02/03/2014: 960	13:00 : 19732	STATEN ISLAND: 49124	Max. :11697	Max. :43.34	Max. : 0.00
(Other) :1637172	(Other):1509677		NA's :500110	NA's :199425	NA's :199425

[1] “

Row 2:”

LOCATION	ON.STREET.NAME	CROSS.STREET.NAME	OFF.STREET.NAME
: 199425	: 323553	: 555996	:1411905
POINT (0 0) : 1113	BROADWAY : 16258	3 AVENUE : 9846	772 EDGEWATER ROAD : 375
POINT (-74.038086 40.608757) : 670	ATLANTIC AVENUE : 14401	BROADWAY : 9685	110-00 ROCKAWAY BOULEVARD : 244
POINT (-73.9845292 40.6960346): 586	3 AVENUE : 11761	2 AVENUE : 8425	2800 VICTORY BOULEVARD : 221
POINT (-73.98453 40.696033) : 574	NORTHERN BOULEVARD : 11460	5 AVENUE : 7052	: 183
POINT (-73.91282 40.861862) : 545	BELT PARKWAY : 11302	7 AVENUE : 6634	2100 BARTOW AVENUE : 158
(Other) :1440379	(Other) :1254557	(Other) :1045654	(Other) : 230206

[1] “

Row 3:”

NUMBER.OF.PERSONS.INJURED	NUMBER.OF.PERSONS.KILLED	NUMBER.OF.PEDESTRIANS.INJURED	NUMBER.OF.PEDESTRIANS.KILLED
Min. : 0.0000	Min. :0.000000	Min. : 0.00000	Min. :0.000000
1st Qu.: 0.0000	1st Qu.:0.000000	1st Qu.: 0.00000	1st Qu.:0.000000
Median : 0.0000	Median :0.000000	Median : 0.00000	Median :0.000000
Mean : 0.2634	Mean :0.001174	Mean : 0.05092	Mean :0.000641
3rd Qu.: 0.0000	3rd Qu.:0.000000	3rd Qu.: 0.00000	3rd Qu.:0.000000
Max. :43.0000	Max. :8.000000	Max. :27.00000	Max. :6.000000
NA's :17	NA's :31		

[1]

”

Row 4:”

NUMBER.OF.CYCLIST.INJURED	NUMBER.OF.CYCLIST.KILLED	NUMBER.OF.MOTORIST.INJURED	NUMBER.OF.MOTORIST.KILLED
Min. :0.00000	Min. :0.00e+00	Min. : 0.000	Min. :0.000000
1st Qu.:0.00000	1st Qu.:0.00e+00	1st Qu.: 0.000	1st Qu.:0.000000
Median :0.00000	Median :0.00e+00	Median : 0.000	Median :0.000000
Mean :0.02062	Mean :8.34e-05	Mean : 0.192	Mean :0.000452
3rd Qu.:0.00000	3rd Qu.:0.00e+00	3rd Qu.: 0.000	3rd Qu.:0.000000
Max. :4.00000	Max. :2.00e+00	Max. :43.000	Max. :5.000000

[1] “

Row 5:”

CONTRIBUTING.FACTOR.VEHICLE.1	CONTRIBUTING.FACTOR.VEHICLE.2	CONTRIBUTING.FACTOR.VEHICLE.3	CONTRIBUTING.FACTOR.VEHICLE.4
Unspecified :599544	Unspecified :1193786	:1536917	:1621113
Driver Inattention/Distraction:308381	: 222550	Unspecified : 98931	Unspecified : 20912
Failure to Yield Right-of-Way : 94090	Driver Inattention/Distraction: 75420	Other Vehicular : 1949	Other Vehicular : 364
Following Too Closely : 82937	Other Vehicular : 27258	Driver Inattention/Distraction: 1424	Following Too Closely : 237
Backing Unsafely : 62951	Failure to Yield Right-of-Way : 14284	Following Too Closely : 1298	Driver Inattention/Distraction: 175
Other Vehicular : 52108	Following Too Closely : 14009	Fatigued/Drowsy : 853	Fatigued/Drowsy : 170
(Other) :443281	(Other) : 95985	(Other) : 1920	(Other) : 321

[1] “

Row 6:”

CONTRIBUTING.FACTOR.VEHICLE.5	COLLISION_ID	VEHICLE.TYPE.CODE.1	VEHICLE.TYPE.CODE.2
:1637609	Min. : 22	PASSENGER VEHICLE :715236	PASSENGER VEHICLE :537550
Unspecified : 5361	1st Qu.:1038275	SPORT UTILITY / STATION WAGON :313500	:273620
Other Vehicular : 96	Median :3457776	Sedan :172288	SPORT UTILITY / STATION WAGON :237846
Following Too Closely : 51	Mean :2808228	Station Wagon/Sport Utility Vehicle:140852	Sedan :128269
Fatigued/Drowsy : 41	3rd Qu.:3868829	TAXI : 50670	Station Wagon/Sport Utility Vehicle:108877
Driver Inattention/Distraction: 40	Max. :4280357	VAN : 26540	(Other) :357127
(Other) : 94		(Other) :224206	NA's : 3

[1] “

Row 7:”

VEHICLE.TYPE.CODE.2	VEHICLE.TYPE.CODE.3	VEHICLE.TYPE.CODE.4	VEHICLE.TYPE.CODE.5
PASSENGER VEHICLE :537550	:1507965	:1593350	:1632527
:273620	PASSENGER VEHICLE : 63655	PASSENGER VEHICLE : 24743	PASSENGER VEHICLE : 5476
SPORT UTILITY / STATION WAGON :237846	SPORT UTILITY / STATION WAGON : 33161	SPORT UTILITY / STATION WAGON : 14375	SPORT UTILITY / STATION WAGON : 3138
Sedan :128269	Sedan : 11698	Sedan : 2598	Sedan : 668
Station Wagon/Sport Utility Vehicle:108877	Station Wagon/Sport Utility Vehicle: 9730	Station Wagon/Sport Utility Vehicle: 2102	Station Wagon/Sport Utility Vehicle: 548
(Other) :357127	UNKNOWN : 3285	TAXI : 1681	TAXI : 246
NA's : 3	(Other) : 13798	(Other) : 4443	(Other) : 689

Based on what we know about the data set from the specifications at NYC Open Data and the data dictionary, we have decided to perform some initial cleaning steps.

Our analytics problem is to predict whether there will be a collision at a specific time (time including a time of the day, day of the year and calendar year). In this context, we will first look at the "CRASH.DATE" graph. Thinking about the scheduling of police resource, we assume that this happens in advance, on a hour-by-hour and day-by-day basis. We assume that resources are not scheduled on a year-by-year basis due to uncertainty in staffing, budget, etc. We therefore examine the data to see whether we should include the year at all. Including the year would treat the data set as a time-series, years ranging from 2012-2020. Alternatively, we could drop the year and group all occurrences on the same day in the same bin, possibly enhancing our prediction.

To decide, we plot the dates and look for trends. If trends repeat annually, we will drop the year as this trend will be preserved when we combine. If the trend does not repeat annually (extends over the whole range of dates) then we will not combine year as we will lose this information when dropping year.

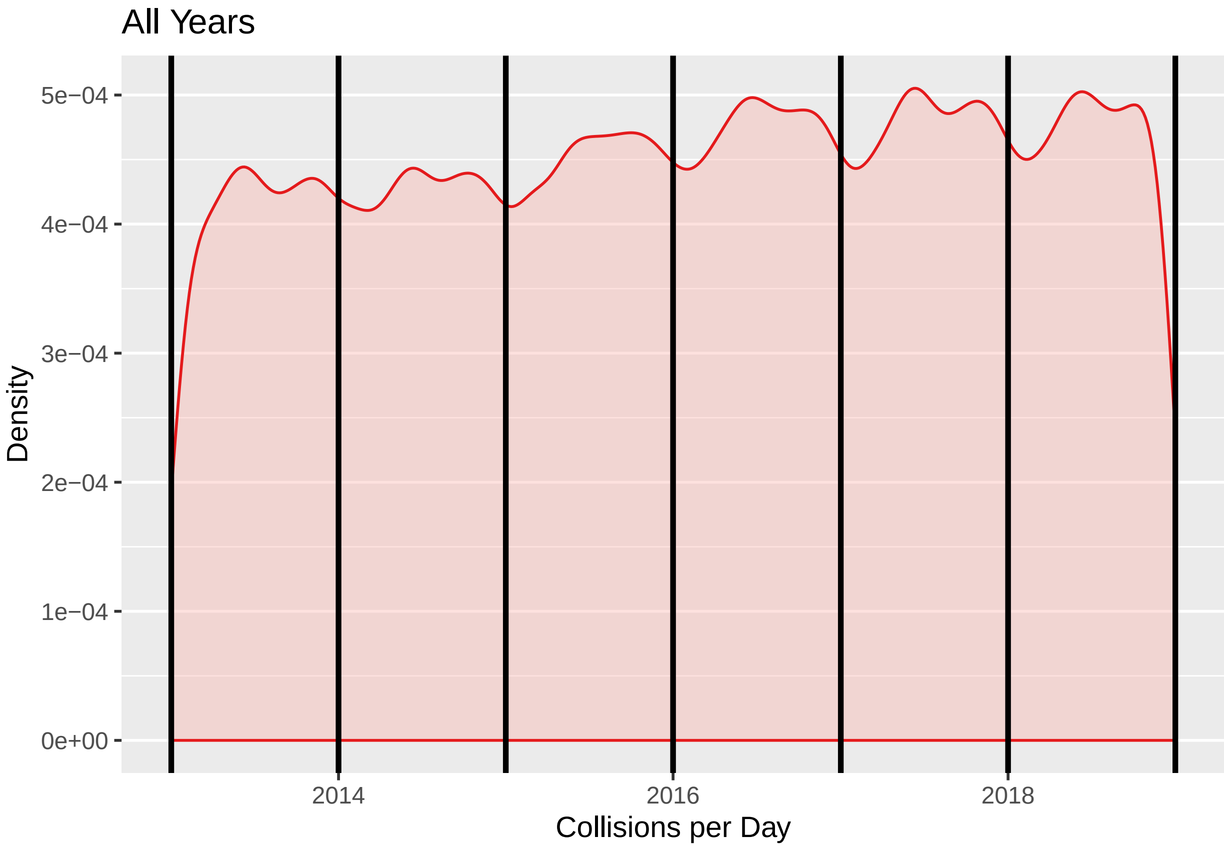


Figure 1: Collisions per Day - All Years

The vertical black lines in the "All Years" plot represent the start of each year. As you can see from the plot, there is a noticeable repeating trend in each year (between the black lines) with a decrease in collisions at the start of each year, followed by various other increase/decreases. As we are more interested in capturing this repeating annual trend than year-over-year changes, we will combine all data into a representation of one year. Additionally, we will drop years 2012 and 2020 (the first and last years in the data set) to avoid over/under-representing specific months in the combined-year data set. This leaves us with the "Years Combined" data set plotted below.

We use the following code to keep only data in 2013 onwards and in prior to 2020, as mentioned.

```
raw_crash_dates_df <- raw_crash_dates_df[raw_crash_dates_df$raw_crash_dates > "2013-01-01", ]
raw_crash_dates_df <- raw_crash_dates_df[raw_crash_dates_df$raw_crash_dates < "2020-01-01", ]
```

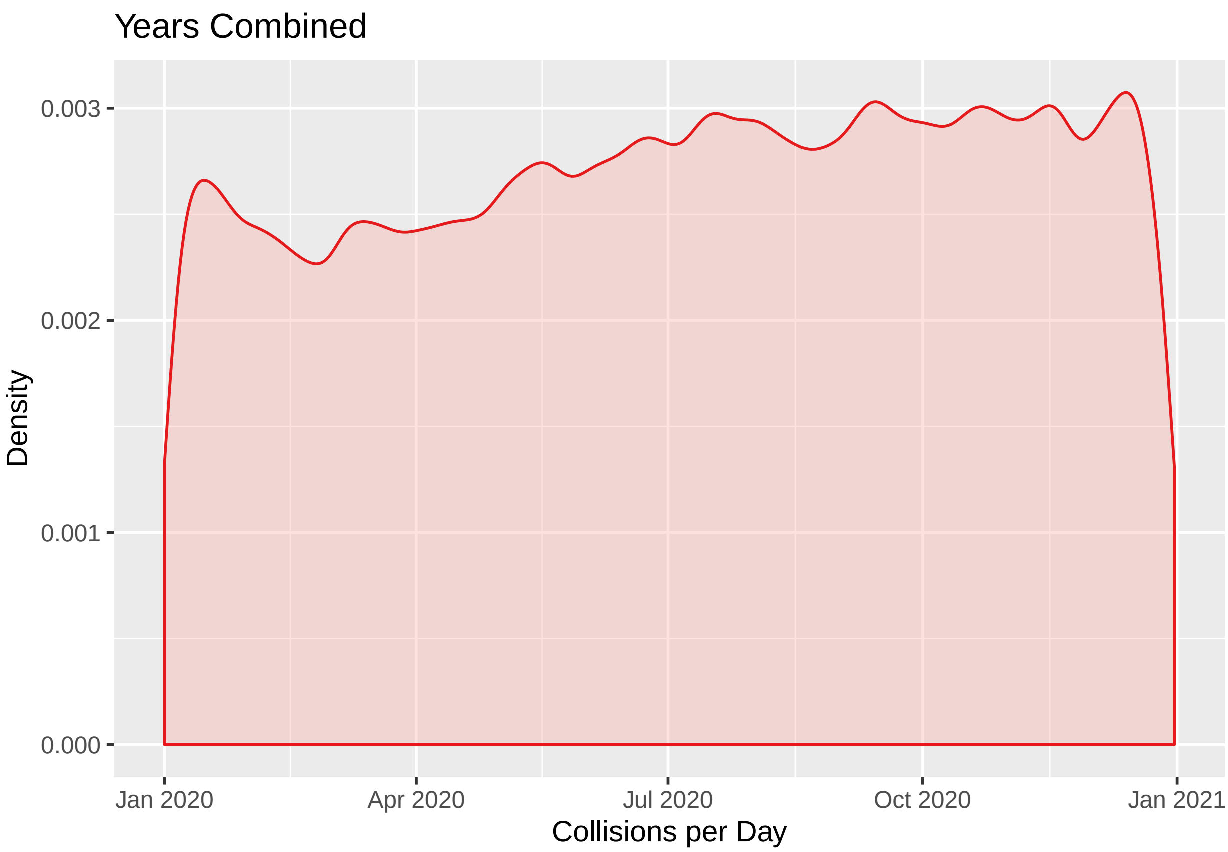


Figure 2: Collisions per Day - Years Combined

Feature Engineering

The business problem, in simple terms, is that the NYPD wants to deploy their police force optimally through predicting collisions. From that very statement, it is clear that the required fields are CRASH.DATE, CRASH.TIME, LATITUDE, LONGITUDE and collisions. The other fields are either redundant (borough & zip code which are redundant to longitude & latitude) or not directly relevant to the problem (street names, number of pedestrians, cyclists, motorists injured or killed. These are ancillary and a deeper level of data and not required to solve the problem). Depending on whether a classification model or a regression model was used, "Collisions" were treated as binary (did crash happen, Y/N?) or as a sum, respectively. Through further research, it was found that NYPD manages NY city by Precincts. A precinct is a district of a city or town defined for police purposes. The NYC open data website also has data about precincts and their shape files. These shape files were used to convert every Longitude, Latitude point into a precinct by finding out the shape polygon in which that Longitude, Latitude point fell. Further, for all classification models, the data was imputed with '0' collisions data where needed, as the original data contained only crashes.

Primary Data Cleaning

The Problem

While the Motor Vehicle Collisions – Crashes" dataset on NYC Open Data contains every motor vehicle collision within the City of New York, there is a singular problem with this dataset that prevents us from using it for the purpose of creating a supervised, binary classification model: it only contains positive observations (i.e. motor vehicle crashes). Therefore, to make this dataset usable for the intended supervision problem as stated, negative observations must be created to complement the dataset. In addition, a binary response variable would be created afterwards to identify the positive and negative observations.

However, the creation of negative observations itself is not a simple problem to solve. for our motor vehicle collision dataset. Each row of the dataset represents a specific motor vehicle collilsion at a particular point in time, as represented by the `CRASH DATE` and `CRASH TIME` columns, and a particular point in space, as represented by the `LATITUDE` and `LONGITUDE` columns. For every motor vehicle collision that has been captured by the dataset, there are potentially thousands, or even hundreds of thousands of collisions that did not happen at any given point in time or space. To put into perspective, let us look at the statistics to show a rough estimate that the likelihood of an average driver being a participant of a car crash. For example, of the 3.8 million commuters in New York City on a regular workday, approximately 27% of the commuters do so by car, truck, or van (https://edc.nyc/article/new-yorkers-and-their-cars). Assuming that half the commuters carpool (https://www.citylab.com/transportation/2019/01/commuting-to-work-data-car-public-transit-bike/580507/), and the rest drive solo, this would mean, at the very least, there are a bit more than half a million vehicles on the road on any given workday. Out of the half a million or so vehicles on New York City roads, there are only about 678 car crashes in New York each day (https://www.dandalaw.com/are-car-accidents-common-in-new-york-city/, but ideally we should calculate this from our dataset!!!). All these numbers point out that getting into a motor vehicle collision, even in a city with an unsafe road reputation like New York, is an unlikely event.

Consequently, to generate negative observations would yield a hugely imblanaced dataset, consisting mostly of negative observations, and very few positive observations. In addition, these negative observations would also have to have generated features, such as `CRASH DATE`, `CRASH TIME`, `LATITUDE`, `LOGITUDE`, etc. This itself is also another issue, as we cannot randomly generate those features without understanding the distribution of New York City traffic across different areas and times, since some boroughs of New York, such as Staten Island, will have less traffic, and therefore less motor vehicle accidents than other boroughs (we should ideally show graphs of the number of accidents per borough).

The Proposed Solution

An easier solution to the problem of generating negative observations is to: (a) group the positive observations by pre-determined datetime ranges, and pre-determined geolocation areas, (b) aggregate the number motor vehicle collisions into a column containing the counts of motor vehicle crashes given a pre-determined datetime range, and pre-determined geolocation area, and (c) via inference, generate negative observations from the grouped, and aggregated positive observations. By generalizing both the time and space of motor vehicle collisions, the generation of negative observations will not likely create a highly imbalnaced dataset that heavily skewers towards the negative class, but also there is now no need to generate those features whilst having to research the New York City traffic flow across different space-time groups. Nevertheless, it is still important to determine an appropriate datetime range, and geo-location areas to bin the observations; it must be taken into account these pre-determined datetime ranges, and geo-location areas should not only prevent extreme class imbalance towards either positive or negative class, but also be relevant to our business problem.

For this particular business problem, it would make sense to use New York Police Department precincts as the pre-determined geolocation areas group the positive observations by. This made sense given our business problem was from a NYPD point of view, as they could divert police resources from other precincts that are less likely to experience motor crashes to other precincts with potentially higher collision rates. We also experimented to use hourly bins, as we felt it would be useful for the police to accurately predict which precinct would likely have more motor vehicle accidents. Overall, using hourly bins and NYPD precincts allowed the dataset to have 77% negative observations, and roughly 23%

positive observations, which was good enough to have only a moderate imbalance.

Creation of the Precinct column

To map the NYPD precinct, the GeoJson containing all the MultiPolygons of NYPD precincts was downloaded from `https://data.cityofnewyork.us/Public-Safety/Police-Precincts/78dh-3ptz`. Each MultiPolygon consists of an array of Polygons, and in turn, each Polygon consists of an array of Latitude and Longitude coordinates. Looping through each positive observation in the dataset, the correct precinct would be assigned to each positive observation.

As an example:

CRASH DATETIME	LATITUDE	LONGITUDE	PRECINCT
06/09/2019 11:32	40.725210	-73.995860	5
06/09/2019 12:55	40.586680	-73.945114	61
06/09/2019 11:11	40.720626	-73.994971	5

Creation of a CRASH_BINARY column

We then create a `CRASH_BINARY` column for each positive observation. All of the positive observations will have a `CRASH_BINARY` value of 1, denoting a positive observation.

CRASH DATETIME	LATITUDE	LONGITUDE	PRECINCT	CRASH_BINARY
06/09/2019 11:32	40.725210	-73.995860	5	1
06/09/2019 12:55	40.586680	-73.945114	61	1
06/09/2019 11:11	40.720626	-73.994971	5	1

Round Date Time to nearest Hour

We round the `CRASH DATETIME` column to the nearest hour. It is always rounded down (i.e. if it is 11:59AM for example, it is rounded down to 11AM).

ROUNDEDCRASH DATETIME	LATITUDE	LONGITUDE	PRECINCT	CRASH_BINARY
06/09/2019 11:00	40.725210	-73.995860	5	1
06/09/2019 12:00	40.586680	-73.945114	61	1
06/09/2019 11:00	40.720626	-73.994971	5	1

Creation of Negative Observations

After assigning a precinct to each positive positive observation, negative observations must now be created. We first group the positive obserations by `PRECINCT` and `CRASH DATETIME`, and aggregate the groups by the sum by CRASH_BINARY

ROUNDEDCRASH DATETIME	PRECINCT	CRASH_BINARY SUM
06/09/2019 11:00	5	2
06/09/2019 12:00	61	1

We will also need to create negative observations. In this small example, there would be no crash accidents in Precinct 61 at 11AM, and the same could be said for Precinct 5 at 12AM. For the real dataset, this imputation of negative observations was done for all possible permutations of precincts (77 possible precincts) and hourly bins (24 hourly bins), but it is not shown in the example due to its length. Therefore, for each day, there are a total of 1848 possible observations, whether positive or negative. CRASH_BINARY SUM for these negative observations are assigned a numeric value of 0.

ROUNDEDCRASH DATETIME	PRECINCT	CRASH_BINARY SUM
…	…	…
06/09/2019 10:00	61	0
06/09/2019 11:00	5	2
06/09/2019 11:00	61	0
06/09/2019 12:00	5	0
06/09/2019 12:00	61	1
06/09/2019 13:00	5	0
…	…	…

Split Datetime into MONTH, WEEK, DAY, WEEKDAY, and HOUR

Lastly, for each datetime observation, the datetime was split into its corresponding month (month of the year), week (week of the year), day (day of the month), weekday (0 to 6, where 0 represents Monday, and 6 represents Sunday), and of course hour (in 24-hour time).

Lastly, for each datetime observation, the datetime was split into its corresponding month (month of the year), week (week of the year), day (day of the month), weekday (0 to 6, where 0 represents Monday, and 6 represents Sunday), and of course hour (in 24-hour time).

ROUNDEDCRASH DATETIME	PRECINCT	CRASH_BINARY SUM	MONTH
…	…	…	…
06/09/2019 10:00	61	0	9
06/09/2019 11:00	5	2	9
06/09/2019 11:00	61	0	9
06/09/2019 12:00	5	0	9
06/09/2019 12:00	61	1	9
06/09/2019 13:00	5	0	9
…	…	…	…

[*table continued below*]

WEEK	DAY	WEEKDAY	HOUR
…	…	…	…
36	6	4	10
36	6	4	11
36	6	4	11
36	6	4	12
36	6	4	12
36	6	4	13
…	…	…	…

Cleaned Data Exploration

Now that we've completed our data cleaning, let's explore the cleaned-up data in more detail. Below, we look at the denstity distribution of each feature (in terms of total occurrences in the data set).

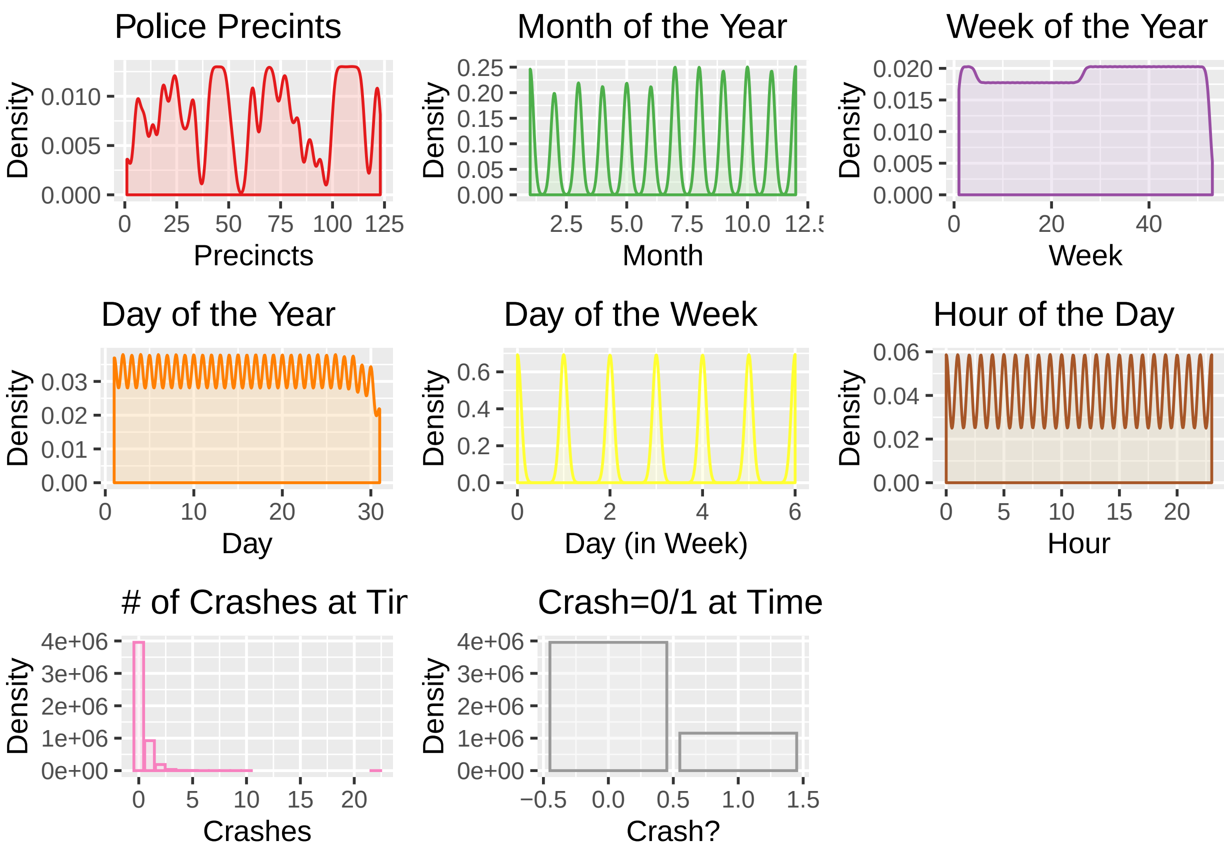


Figure 3: Plots of Selected Features

Density Plots

Models

As we worked through the data set, we were unsure as to which supervised machine learning models would best suit our business needs. As such, we have run, evaluated and compared the following 6 models:

- Decision Tree
- Gradient Boosting
- K-Nearest Neighbor
- Logistic Regression
- Random Forest
- Regression Tree (Decision Tree Alternate Output)
  - In this model we attempted to outut # of crashes rather than a binary variable for crash/no crash.

Decision Tree

The Decision tree algorithm is a simple tree based algorithm that allows for easy modeling and interpretation to the end business user. Though understood to be computationally complex, the model works well in the given context and helps identify if there will be a collision for a given Police Precinct.

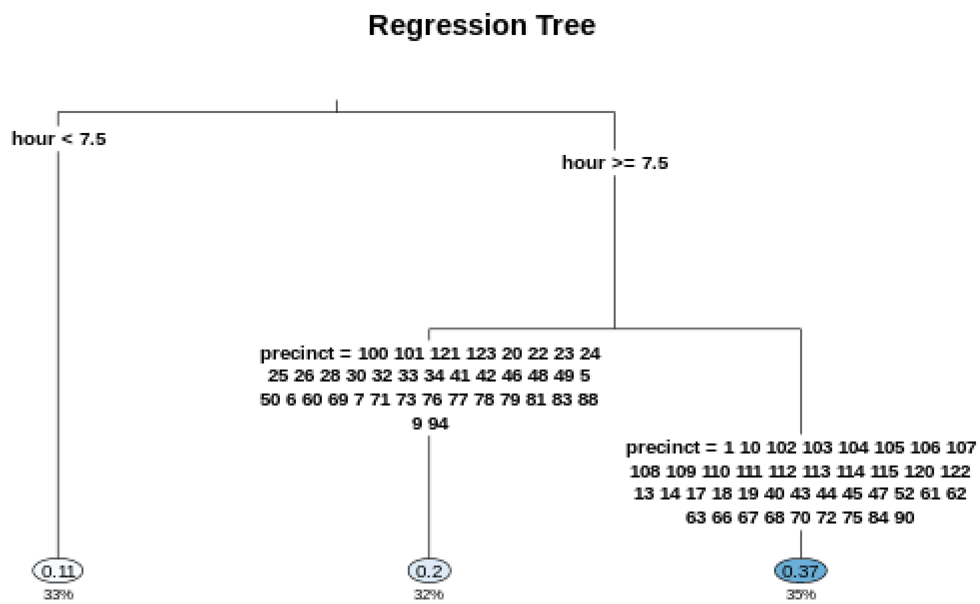


Figure: Decision Tree Visualization of Branches

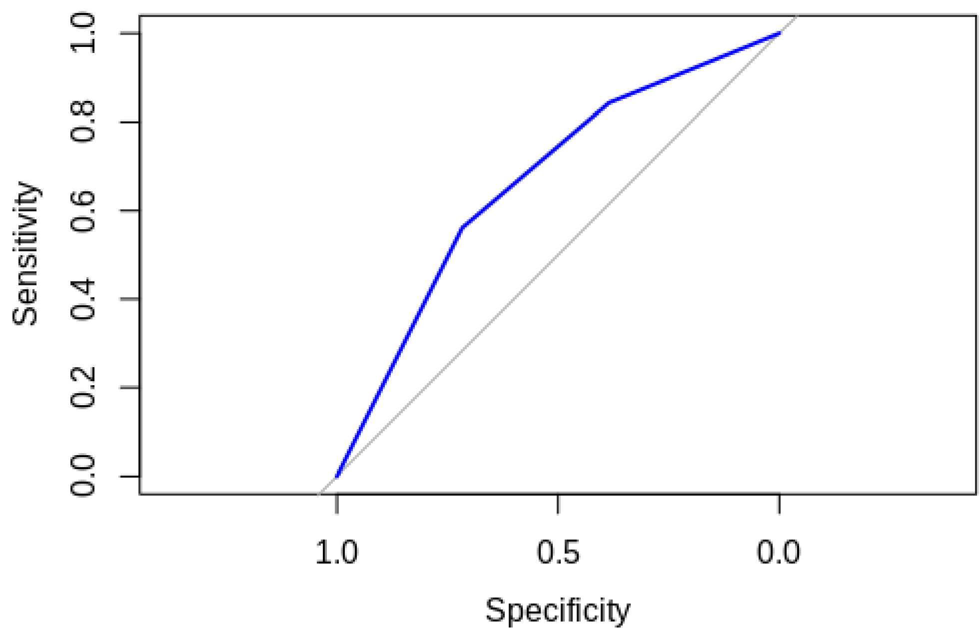


Figure: Decision Tree AUC/ROC Curve

Stochastic Gradient Boosting (GBM)

Stochastic Gradient Boosting is a method of supervised learning, where it continuously iterates over each tree one at a time to boost the performance of its weaker learners. Unlike other types of Gradient Boosting algorithms, Stochastic Gradient Boosting allows a base learner to draw samples randomly from the training set. There were a few hyperparamters to tune: `n.trees` denotes the number of trees, `interaction.depth` denotes the maximum depth of each tree, `n.minobsinnode` denotes the minimum number of observations in the final node of the trees, and `shrinkage` denotes the shrinkage, or learning rate for each tree. We used a grid search for finding the appropriate hyperparameters for gbm, with values of 10 and 20 for `interaction.depth`, and values of 50, 100, and 250 for `n.trees`. For the other hyperparameters, we left `n.minobsinnode` at 10, and `shrinkage` at 0.1. We found the gbm model performed the best, in terms of metrics, at a `interaction.depth` of 20, and a `n.trees` of 250, although doing so also increased the time to train the gbm model.

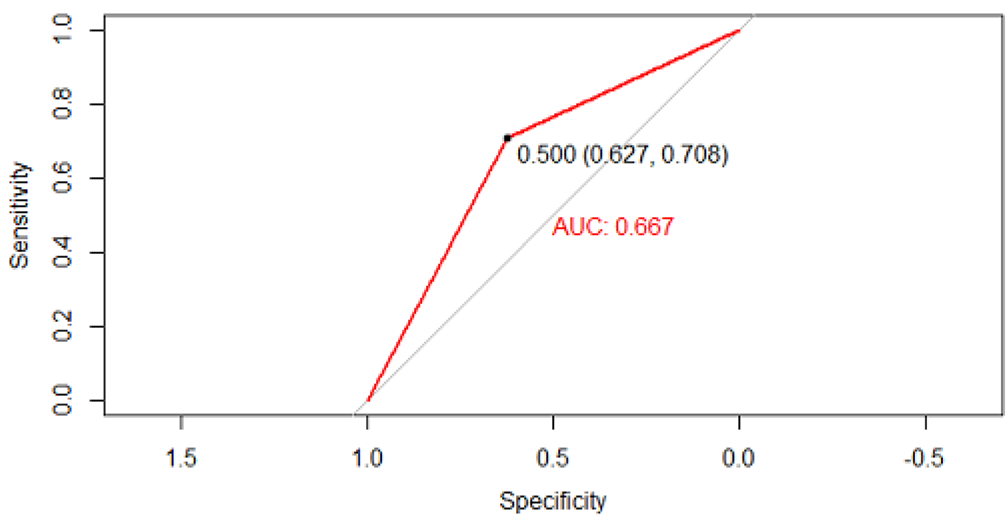


Figure: Stochastic Gradient Boosting (GBM) AUC/ROC Curve

Stochastic Gradient Boosting (GBM) Confusion Matrix and Statistics

```
          Reference
Prediction     no    yes
       no  326251  44978
       yes 194322 109046

               Accuracy : 0.6453
                 95% CI : (0.6441, 0.6464)
    No Information Rate : 0.7717
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2495

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6267
            Specificity : 0.7080
         Pos Pred Value : 0.8788
         Neg Pred Value : 0.3595
             Prevalence : 0.7717
         Detection Rate : 0.4836
   Detection Prevalence : 0.5503
      Balanced Accuracy : 0.6673

       'Positive' Class : no
```

Logistic Regression

Logistic regression is a useful model for the purposes for binary classification. For a parametric model, it does not have many assumptions to satisfy. The first assumption is that logistic regression requires a large dataset of observations to make accurate predictions. This is fulfilled in this situation quit easily: there are 5 million positive and negative observations in total to train a logistic regression model. The second assumption is that the observations are made independent of one another. This is true in this situation, as the positive observations were made from the grouped and aggregated sum of accidents, and the negative observations were inferred from the lack of positive observations in the dataset for a particular precinct and hourly bin. The third assumption is that the response variable must be binary, which is true in our case because it is either a 1 or 0. However, there are two other assumptions which we unfortunately did not have the time to ascertain their certainty, namely that the predictor variables are related to the logit function, and that there is minimal multicollinearity among the predictor variables.

Nevertheless, the logistic regression model showed an ROC curve that was not far from the two other best models, namely decision tree and gbm. One of the benefits of logistic regression is that there are no hyperparameters to tune, which made generating the model easier as there was no need to do validation.

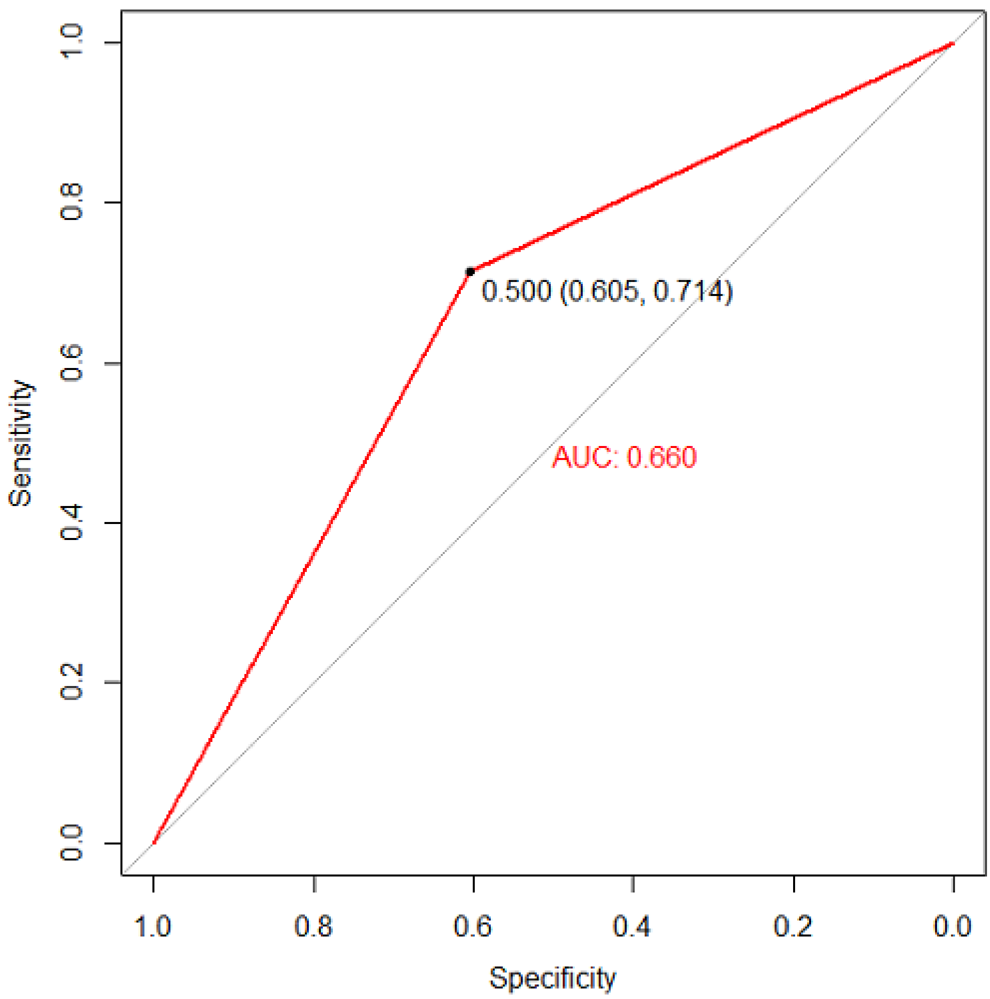


Figure: Generalized Linear Model AUC/ROC Curve

Generalized Linear Model Confusion Matrix and Statistics

```
          Reference
Prediction     no    yes
       no  314273  43937
       yes 204801 109738

               Accuracy : 0.6303
                 95% CI : (0.6291, 0.6314)
    No Information Rate : 0.7716
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2335

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6054
            Specificity : 0.7141
         Pos Pred Value : 0.8773
         Neg Pred Value : 0.3489
             Prevalence : 0.7716
         Detection Rate : 0.4671
   Detection Prevalence : 0.5325
```

```
      Balanced Accuracy : 0.6598

       'Positive' Class : no
```

Random Forest (rf)

Random Forests/Decision Trees is an ensemble learning method for classification and regression problems. Using a large number of individual decision trees, the one with the most votes is chosen as the prediction. There are two hyperparameters to tune: `ntree` and `mtry`. `ntree` denotes the number of trees, whereas `mtry` denotes the number of variables to use for splitting at each tree node. We decided to use a `ntree` of 250, as for a `ntree` size any larger than 250 resulted in a significantly longer training time, and memory overflow issues. We also used a `mtry` of 2, as this was the default and recommended number, which was calculated based on the rounded down square root of the cardinality of predictors (http://code.env.duke.edu/projects/mget/export/HEAD/MGET/Trunk/PythonPackage/dist/TracOnlineDocumentation/Documentation/ArcGISReference/RandomForestModel.FitToArcGISTable.html).

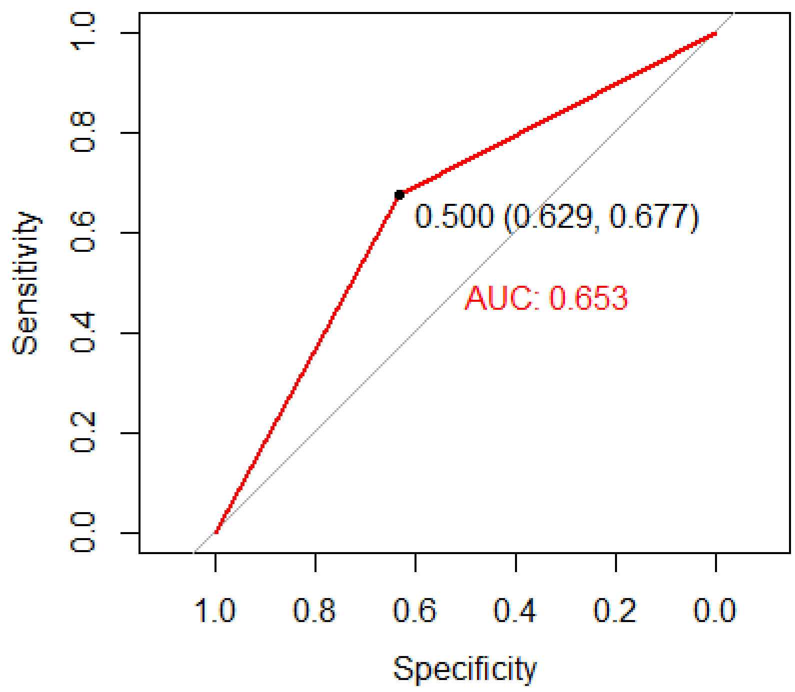


Figure: Random Forest AUC/ROC Curve

Random Forest Confusion Matrix and Statistics

```
          Reference
Prediction     no    yes
       no  128730  20709
       yes  76344  43680

               Accuracy : 0.6398
                 95% CI : (0.638, 0.6416)
    No Information Rate : 0.761
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2361
```

```
Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6277
            Specificity : 0.6784
         Pos Pred Value : 0.8614
         Neg Pred Value : 0.3639
             Prevalence : 0.7610
         Detection Rate : 0.4777
   Detection Prevalence : 0.5546
      Balanced Accuracy : 0.6531

       'Positive' Class : no
```

KNN (knn)

K-Nearest Neighbor is a prediction algorithm used for classification and regression problems. It works by taking each point in the dataset, and looks at k-nearest Neighbors to decide which class a particular point belongs to. A disadvantage of KNN is that the features have to be scaled and normalized. THis itself was not an issue for this particular model, as all features except `Precint` were numeric. There is only one hyperparamter to tune for KNN, which is the `k` number. `k` denotes the nearest number of neighouring points to calculate the Euclidean distance from. We tried a variety of `k` parameters, ranging from 3, 5, 7, 9, and 11, but changing the `k` parameter achieved negligble prediction improvements in our metrics.

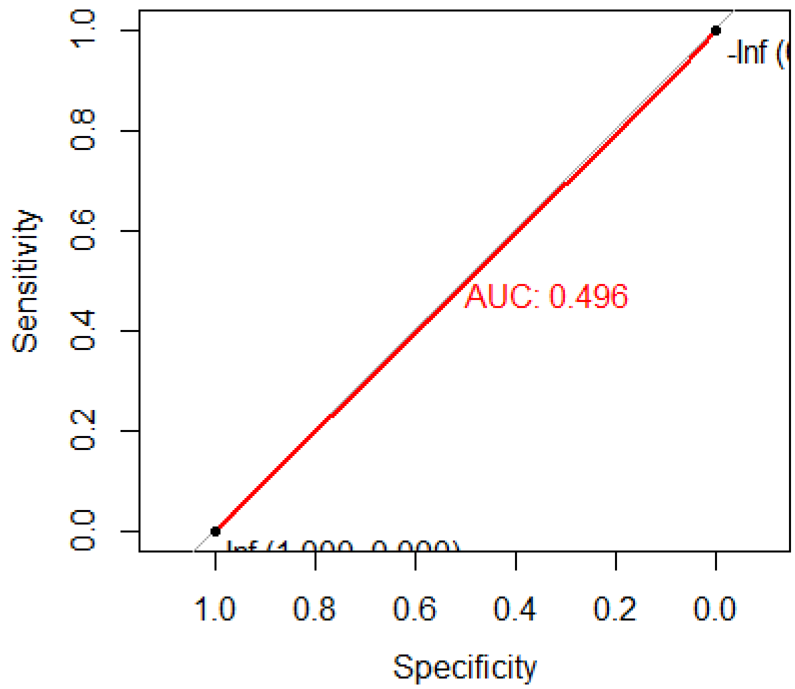


Figure: K-Nearest Neighbor (k=3) AUC/ROC Curve

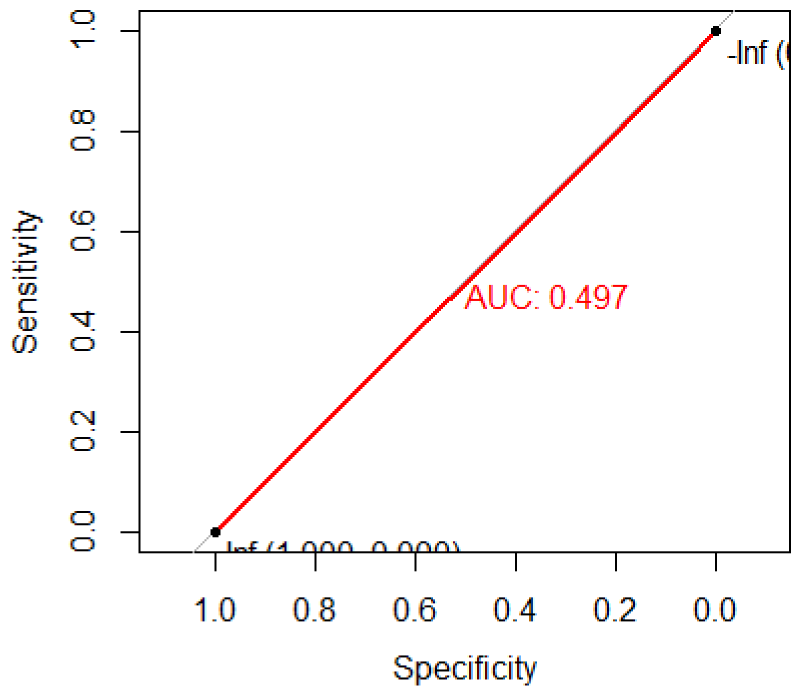


Figure: K-Nearest Neighbor (k=5) AUC/ROC Curve

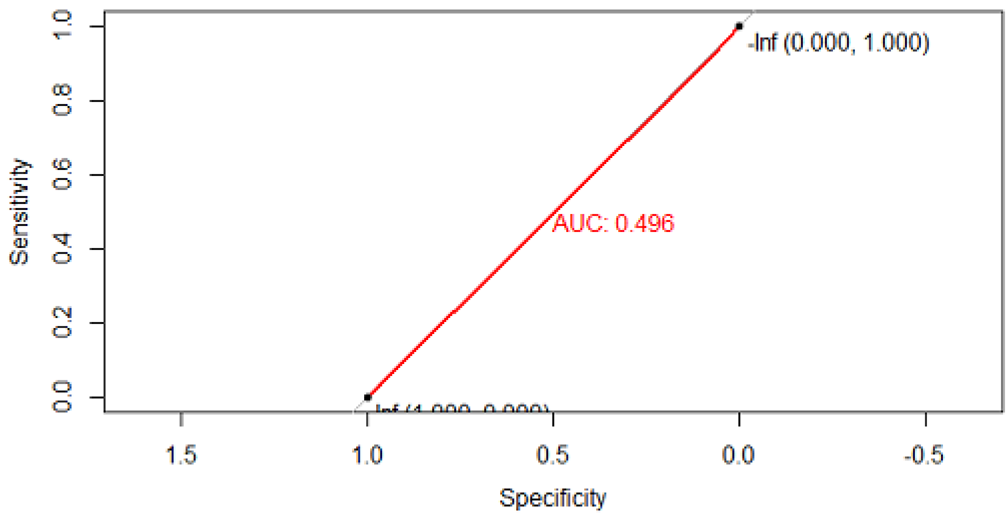


Figure: K-Nearest Neighbor (k=5) AUC/ROC Curve

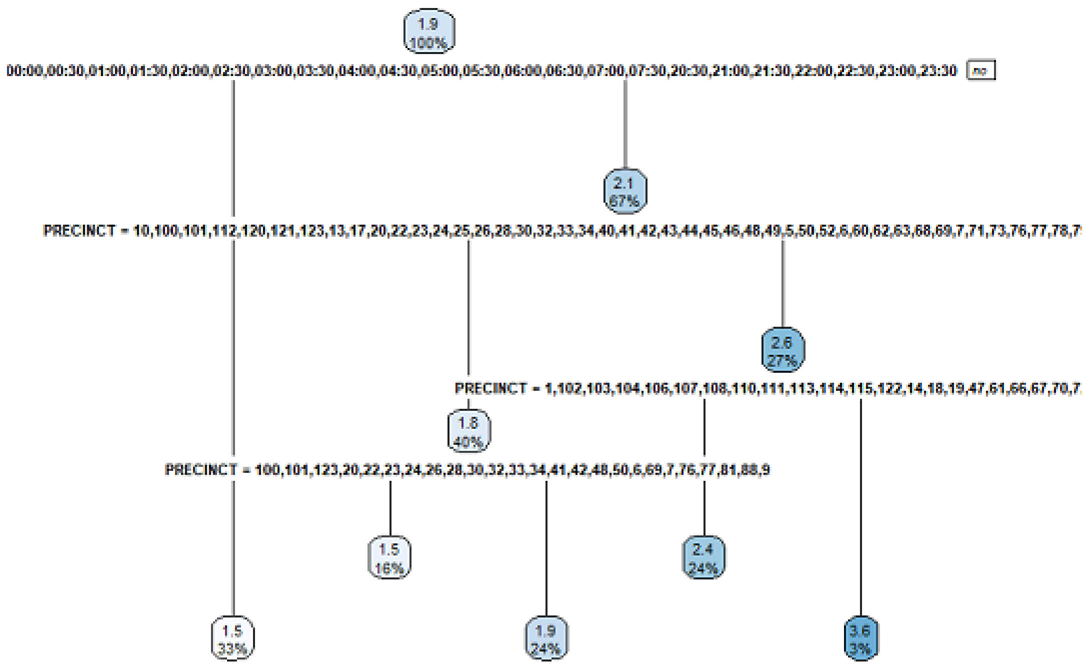
K-Nearest Neighbor Confusion Matrix and Statistics

```
          Reference
Prediction     no    yes
       no  179828  38370
       yes 126599  59570

               Accuracy : 0.592
                 95% CI : (0.5905, 0.5935)
    No Information Rate : 0.7578
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1493
```

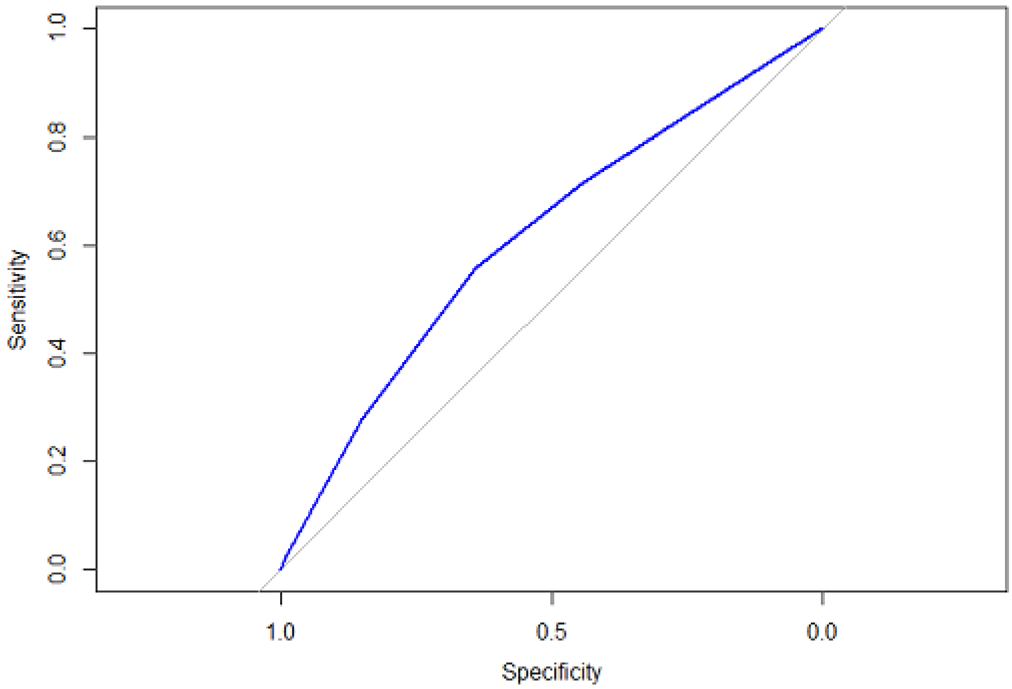
```
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.5869
            Specificity : 0.6082
         Pos Pred Value : 0.8242
         Neg Pred Value : 0.3200
             Prevalence : 0.7578
         Detection Rate : 0.4447
   Detection Prevalence : 0.5396
      Balanced Accuracy : 0.5975

       'Positive' Class : no
```

Regression Tree (Decision Tree Alternate Output)

Approaching the case as a prediction problem, leads us to consider regression algorithms. The features in consideration aren't linearly related and hence non parametric algorithms are used. Regression trees easily model non linearity and avoids the assumptions of normal regression methods. The regression tree helps determine the no.of collisions that can happen at a given Precinct.



Decision Tree Visualization of Branches



Decision Tree AUC/ROC Curve

Evaluation

In order to evaluate which model best suits our business case, we look at both the AUC/ROC Curve plots and consider other factors such as interpretability, time to run, and ease of maintenance.

The AUC/ROC Curve graph below displays a comparison of this metric for each of our models.

As you can see, Decision Tree and GBM score the best on this metric.

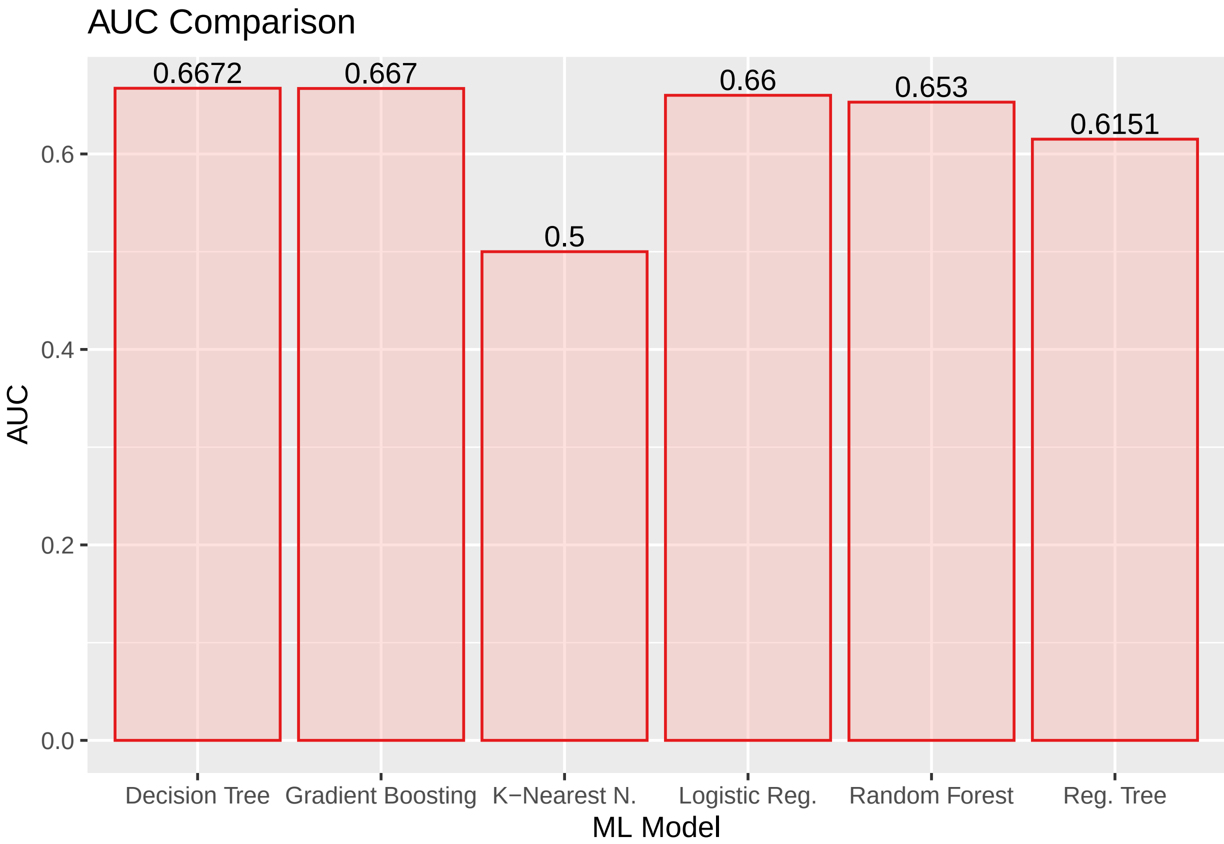


Figure 4: AUC Comparison

Model Selection and Conclusion

By looking at the metric Area Under Curve (AUC) for Receiving Operator Characteristic, the Decision Tree model ranked slightly higher than the Stochastic Gradient Boosted model by a very small margin when tested against the testing data. This indicates both models would likely perform similarly to one another when placed in a production environment. We must therefore take into account other factors, such as model interpretation, and model implementation in production.

In terms of the model interpretation, the Decision Tree model is better than the Stochastic Gradient Boosted model. A business user can easily follow the logic via the diagram produced by the Decision Tree model, which shows how the model came to arrive at its classification via each node of the tree. On the other hand, the Stochastic Gradient Boosted model is a fairly complex model to explain to business users. Although certainly not impossible to understand, since a GBM is simply an ensemble implementation of multiple trees, with weak classifieds being boosted at each iteration, it is hard to trace how the model arrived at its classification prediction as one would have to iteratively go through hundreds of trees.

The model implemented with Decision Tree algorithm is also easier to deploy into production. We were able to train a Decision Tree algorithm within 5-10 minutes on a full dataset of 5 million observations on average laptops with 8GB RAM. With multiple cross-validation and hyperparameter optimization, where one would have to train multiple models, one could validate and optimize decision tree algorithm easily within an hour. On the other hand, training a model implemented with Stochastic Gradient Boosting was a lot longer and harder. It took roughly 1.5 hours to train a GBM model with an above-average desktop with a 4th generation i7 processor and 16GB RAM. With multiple cross-validation and hyperparameter optimization, this would likely mean 12+ hours would be spent validating and optimizing a GBM model. Although the dataset itself is refreshed once everyday to include the nearest crash collisions of the day, and hence one could still train , validate, and optimize a GBM model within a 24 hour timeframe, the need for a relatively advanced desktop to run continuously to train and test and get model for more than 10 times the duration needed to train and test a decision free model for a negligible return means we would choose to implement the Decision Tree algorithm in production rather than GBM.

Document Style Attribution

This document was generated using a modified version of the "RJournal.sty" file provided by the The R Foundation at https://journal.r-project.org/submissions.html.

Inspiration, useful package suggestions and some sample code for the xtables and ggplot functionality has been reproduced from the example assignment 1 R Markdown file created by V2MSLabs (https://github.com/v2msLabs/ML1000-1/blob/master/source/main.Rmd).

The document can be regenerated in RStudio by Knitting the provided "R Markdown-Group 10-Assignment 1.Rmd" file with the provided "RJournal.sty" file in the same directory.

Alex Fung

Viswesh Krishnamurthy

Tony Lee

Patrick Osborne