MBA 540
Final Project

Clustering and Classification Analysis for NYC Traffic Accidents

Ahmed Shata, Yancheng Xiang, Michael McCannell, and Nabil Islam

Stony Brook University

Introduction

This report analyzes the available data regarding New York City traffic accidents to determine the variable that most directly affects traffic accident severity. A severity index was determined based on the number of individuals injured and the number of individuals killed for each instance. Moreover, the attributes considered in this analysis include location, the time of day, day of the week, main contributing factors for each accident, and the types of vehicles present in each accident. With this data set, 60,000+ data points were clustered based on location information and classified using different models to determine the optimal classification model for the data set.

Data Description

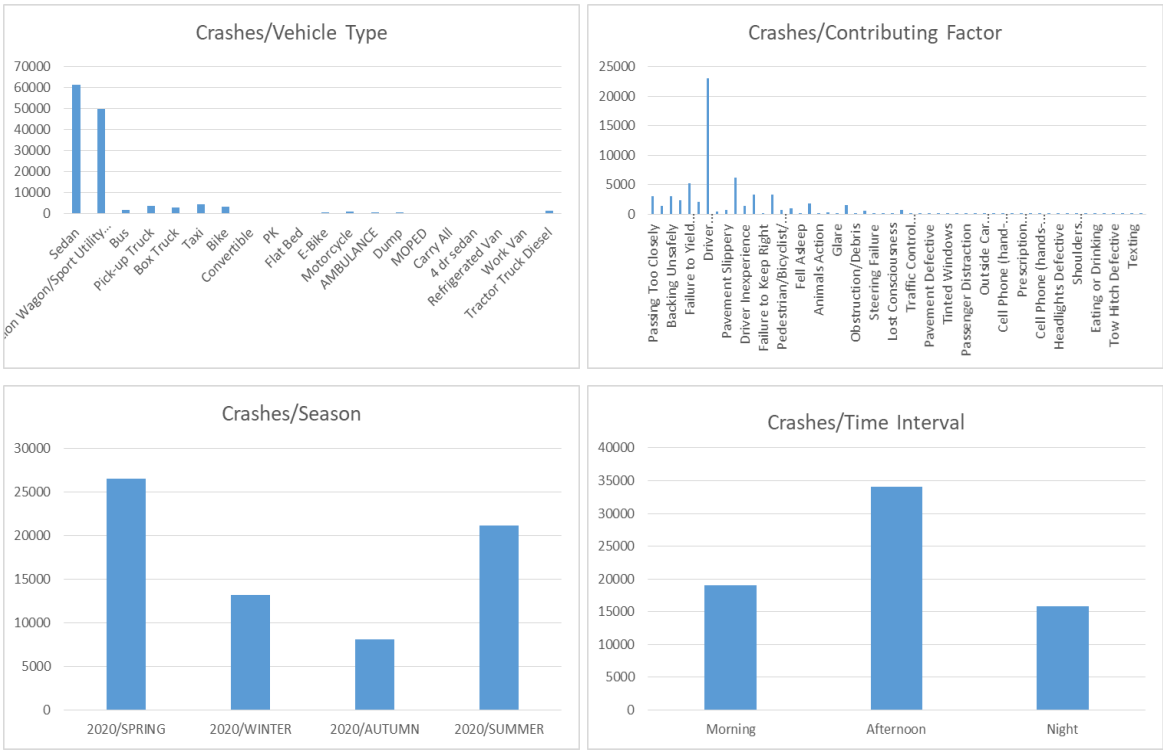
The data selected for processing includes the New York City Traffic Accidents from January 1, 2020 up to August 29, 2020. This data set consists of multiple variables ranging from location of the accident to the main contributing factor of the accident. Five independent variables (attributes) were focused on in the data analysis and are listed in Figure 1.

Figure 1: Data Set Variables

Variable	Type	Contents
Crash Date/Season	Categorial	Represents the date that the crash occurred
Crash Time/Interval	Continuous	Time Crash Occurred
Contributing Factors to Accident	Categorial	Individual Factors that Contributed to Crash
Vehicle Type	Categorial	Make of Vehicle involved in Crash
Clusters Based on Location	Categorial	Clusters of Locations of Crashes

In Figure 1, the five independent variables demonstrated that we will be processing. Each variable's individual type and contents is shown in the figure. A closer look at each variable in relation to the number of NYC traffic accidents and the distributions of the raw data collected are shown in Figure 2. Moreover, the lack of a normal distribution throughout the raw data is readily apparent.

Figure 2: Number of Accidents/Selected Attributes

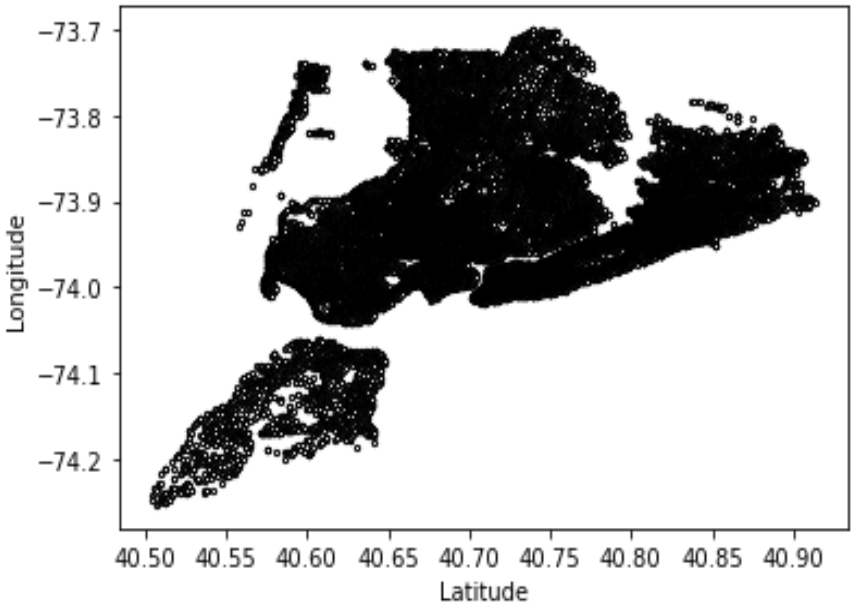


In the figure above we see the initial results of each individual variable from the raw data. From these initial findings we can clearly see the most prevalent factor in each data set. First it is shown that vehicle type has two main vehicles correlated with accident frequency. The Sedan and the Station wagon/Sport Utility Vehicle. Second, in contributing factors the raw data demonstrates that Distracted Driving is by far the largest contributor to the cause of an accident. The third variable Crash/Season has closer results in each season than other variables in the raw data, but demonstrates that the spring season has the largest amount of crashes. The final variable in this figure is the Crash/Time Interval, which demonstrates the largest amount of crashes occurring in the afternoon. These variables will be used to continue our classification and clustering to develop our final prediction.

Clustering

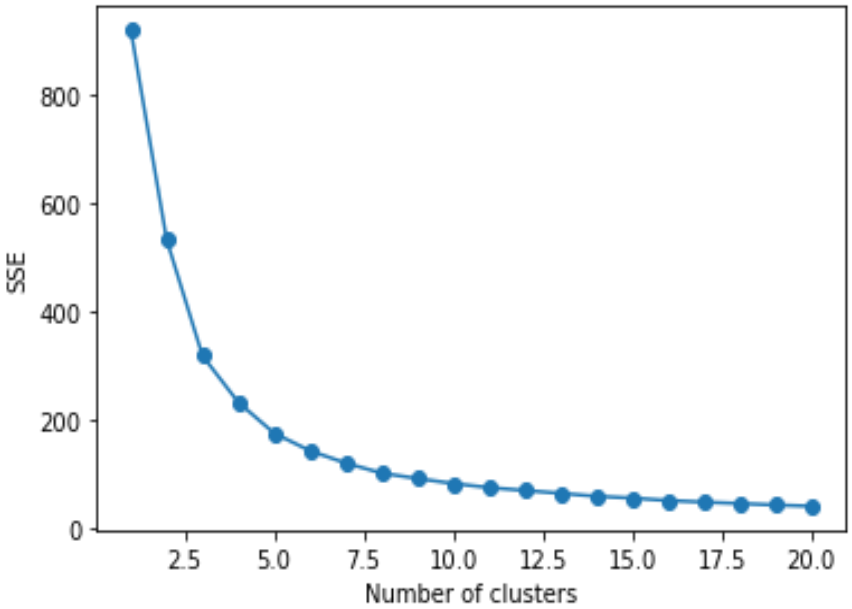
The data set was preprocessed in Microsoft Excel to delete all data entries with missing location information. Latitude and longitude values were used to initially plot the remaining data points. Figure 3 shows the data points plotted and outlining the general shape of New York City. Moreover, this plot illustrated the relative locations of various data points and influenced the decision to cluster data by geographic location.

Figure 3: Plot of all NYC Accidents



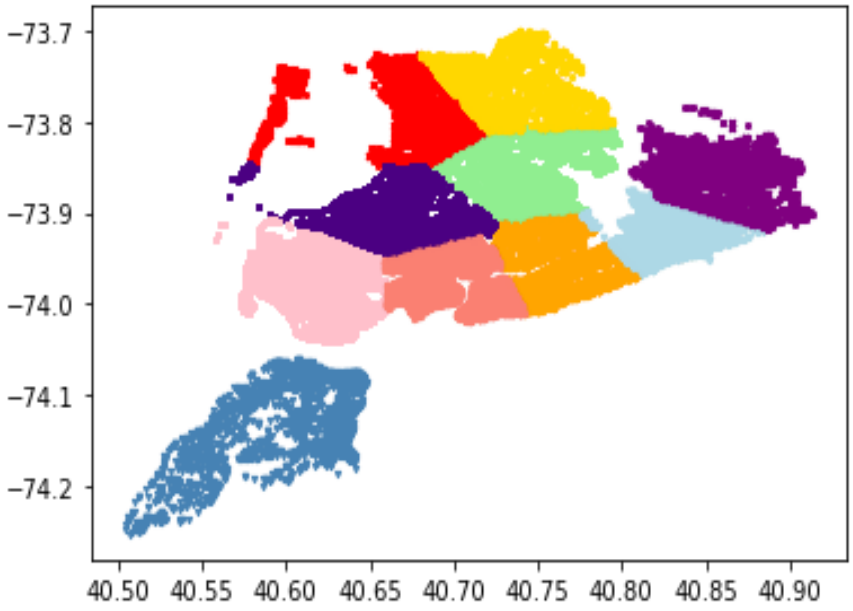
A loop was created to determine the optimal number of clusters to minimize SSE and complexity. The results of this loop were plotted in Python and can be seen in Figure 4. Visual analysis of this plot determined that 10 clusters would be ideal for classification purposes as SSE did not substantially decrease with the addition of more clusters, and thereby increased complexity.

Figure 4: Results of Loop to Determine Optimal Number of Clusters



Once 10 clusters were established, the results were plotted to demonstrate the clusters in unique colors. This plot can be seen in Figure 5. The assigned cluster for the data point was exported as a .csv file and appended to the existing data. The code used to develop the clusters and generate Figures 3, 4, and 5 can be found in Appendix A.

Figure 5: Clustered Data Plotted



Data Preprocessing

In order to process the data smoother, some simplifications and optimizations should be done before we process the data in Weka. We first determined that we will see which classifier is best at predicting the severity of the traffic accidents that happen in NYC. Five contributors had been selected as the independent variables: crash season, crash time interval, contributing factors to accident, vehicle type and clusters based on location. Since the original data are numerical and discrete data that will add extra work for algorithm to calculate, we converted some numerical data into categorical. We converted date into seasons, and converted time into three time intervals: “morning”, “noon”, and “night”. All those preprocessing was done in excel with formulas.

After the initial categorizing, we now need to build up the severity index. Since there’s no information about how bad those injuries were, we had to establish the severity index based on the injury number involved in each accident. We noticed that the max number of injuries in one accident was 15 and fatalities was 4, so we divided 15 into 3 intervals and gave each interval

different weight, from smallest to largest. Any accident involving at least one fatality will be categorized with the largest weight. However, we soon found out the limit of this index system, which is that it was too specified and will add an unnecessary workload for the algorithm. The J48 decision tree also cannot process numeric data, thus we had to turn the index into categorical value.

We then optimized our severity index by weighting accidents involving no injury or fatality to 0 points; accidents involving 1-5 injuries have 1 point; accidents involving 5-15 injuries have 2 points; accidents involving 1 fatality have 10 points and adding 10 points per fatality. Then fitted them into “minor accident”, “accident involves minor injury”, “accident involves medium injury” and “major or fatal accident” based on their accumulated points. Accidents with 0 points fell into the first category. Accidents with 1 point, which means 1-5 people were injured in the accident, fell into the second category. Accidents that involved 5-15 injuries fell into the third category. Accidents with accumulated points greater than 10 fell into the “major or fatal accident” category, which indicates that one or more people were killed in the accident.

The last step was simplification, we removed all redundant location-related data, since we had our clustering, and removed all pre-convert data. By preprocessing the dataset and establishing the severity index, our dataset was ready to be classified in Weka.

Results From Different Classification Methods

We used a total of five different classifiers for our dataset, they were ZeroR, J48 decision tree, K-nearest neighbor, ensemble method bagging and SVM. We collected several important metrics from the outcome, including True Positive rate, False Positive rate, precision, recall, F-measure, MCC, ROC area and PRC area. All the classifications were conducted in Weka, as we changed some parameters in order to find the satisfying result. All the reported metrics of the classifiers were a weighted average of the 10 folds cross-validation.

The first classifier was ZeroR, which was the most basic classifier and could provide us what the baseline of the classification should look like. We classified the dataset with batch size equals 100 and had an unsatisfied outcome. The correctly classified instances rate was 72.7131 percent, indicating the worst outcome we could get from any other classifiers. However, the confusion matrix was unpleasant, with all classes but “minor accident” has been classified into the first class. All accuracy percentages were missing except the “minor accident” class in the detailed summary. Although it did not prove much useful information, ZeroR at least gave us a general understanding of how our dataset looks in classification, which gave us a better direction for future classifying.

The second classifier was the J48 decision tree. We set batch size to 100, confidence factor = 0.25, collapse tree = true, and pruning = true. The correctly classified instances went up to 76.3663 percent, meaning it was more accurate than the basic ZeroR classifier. There were a total of 1917 leaves and the size of the tree was 1943. The confusion matrix was slightly better with some second classes falling into where they belong, however, we believe that overall inaccuracy in the confusion matrix was because of the relatively small cases of the last two classes compared with the entire 60,000 cases. Figure 6 shows the outcomes of an SUV or station wagon being involved in an accident. Accidents involving mopeds or motorcycles tend to get people injured (we believe they usually were the rider).

Figure 6: J48 Pruned Tree

```
VEHICLE TYPE CODE 2 = Station Wagon/Sport Utility Vehicle
|   VEHICLE TYPE CODE 1 = Station Wagon/Sport Utility Vehicle: MINOR ACCIDENT (9927.35/2311.47)
|   VEHICLE TYPE CODE 1 = Sedan: MINOR ACCIDENT (10301.35/2549.02)
|   VEHICLE TYPE CODE 1 = Moped: ACCIDENT INVOLVES MINOR INJURY (36.58/11.21)
|   VEHICLE TYPE CODE 1 = Box Truck: MINOR ACCIDENT (415.32/49.3)
|   VEHICLE TYPE CODE 1 = Taxi: MINOR ACCIDENT (777.24/223.21)
|   VEHICLE TYPE CODE 1 = Bus: MINOR ACCIDENT (304.0/67.63)
|   VEHICLE TYPE CODE 1 = Chassis Cab: MINOR ACCIDENT (21.23/1.14)
|   VEHICLE TYPE CODE 1 = Pick-up Truck: MINOR ACCIDENT (582.55/106.93)
|   VEHICLE TYPE CODE 1 = Van: MINOR ACCIDENT (128.2/23.64)
|   VEHICLE TYPE CODE 1 = Garbage or Refuse: MINOR ACCIDENT (42.28/5.67)
|   VEHICLE TYPE CODE 1 = PK: MINOR ACCIDENT (29.74/6.33)
|   VEHICLE TYPE CODE 1 = Dump: MINOR ACCIDENT (61.62/10.54)
|   VEHICLE TYPE CODE 1 = 4 dr sedan: MINOR ACCIDENT (55.75/11.46)
|   VEHICLE TYPE CODE 1 = Stake or Rack: MINOR ACCIDENT (4.04/0.03)
|   VEHICLE TYPE CODE 1 = Tractor Truck Diesel: MINOR ACCIDENT (145.48/16.19)
|   VEHICLE TYPE CODE 1 = Ambulance: MINOR ACCIDENT (107.33/4.15)
|   VEHICLE TYPE CODE 1 = BOX TRUCK: MINOR ACCIDENT (1.01/0.01)
|   VEHICLE TYPE CODE 1 = Flat Bed: MINOR ACCIDENT (39.91/3.31)
|   VEHICLE TYPE CODE 1 = escavator: MINOR ACCIDENT (0.35/0.0)
|   VEHICLE TYPE CODE 1 = Tractor Truck Gasoline: MINOR ACCIDENT (19.26/0.48)
|   VEHICLE TYPE CODE 1 = Motorcycle: ACCIDENT INVOLVES MINOR INJURY (175.43/63.85)
```

The third method we used was an ensemble method called bagging. With bag size to 100 percent and batch size to 100, bagging has the most accurate confusing matrix among all five methods, yet its overall accuracy was only 72.2979 percent, slightly better than ZeroR. Bagging was also the only method that gave us a complete Detailed Accuracy By Class table, with all metrics calculated.

The fourth method was the K-nearest neighbor (KNN). With all batch sizes set to 100, K-nearest neighbors performed slightly differently when we set different nearest-neighbor numbers. KNN with 1 nearest neighbor performed no better than ZeroR, it had 72.8424 percent accuracy for correctly classified instances. However, it has the most complete Detailed Accuracy By Class table among other KNN, the main reason was its small size. The second KNN with 100 nearest neighbors has the highest accuracy, up to 74.6051 percent. It also has the highest ROC area among all methods, indicating that it was a balanced method for classifying our dataset. The accuracy of the following two KNN started to decrease, with 73.4014 percent for 500 neighbors

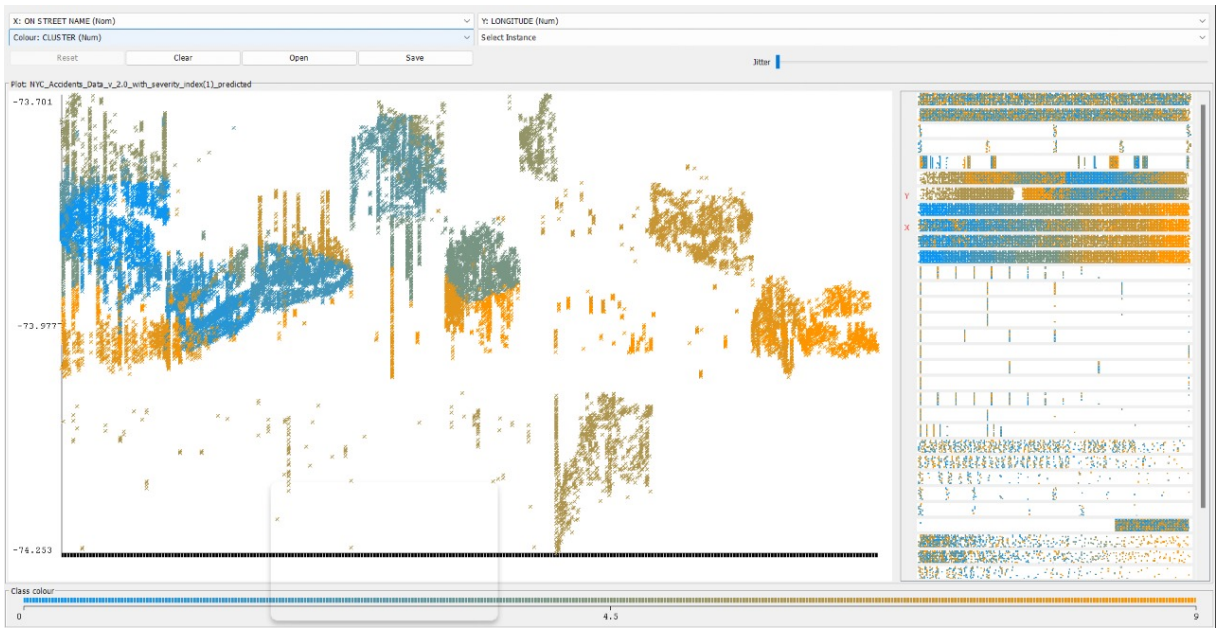
and 72.9019 percent for 1000 nearest neighbors. KNN with the 100 neighbors was the most accurate KNN classifier.

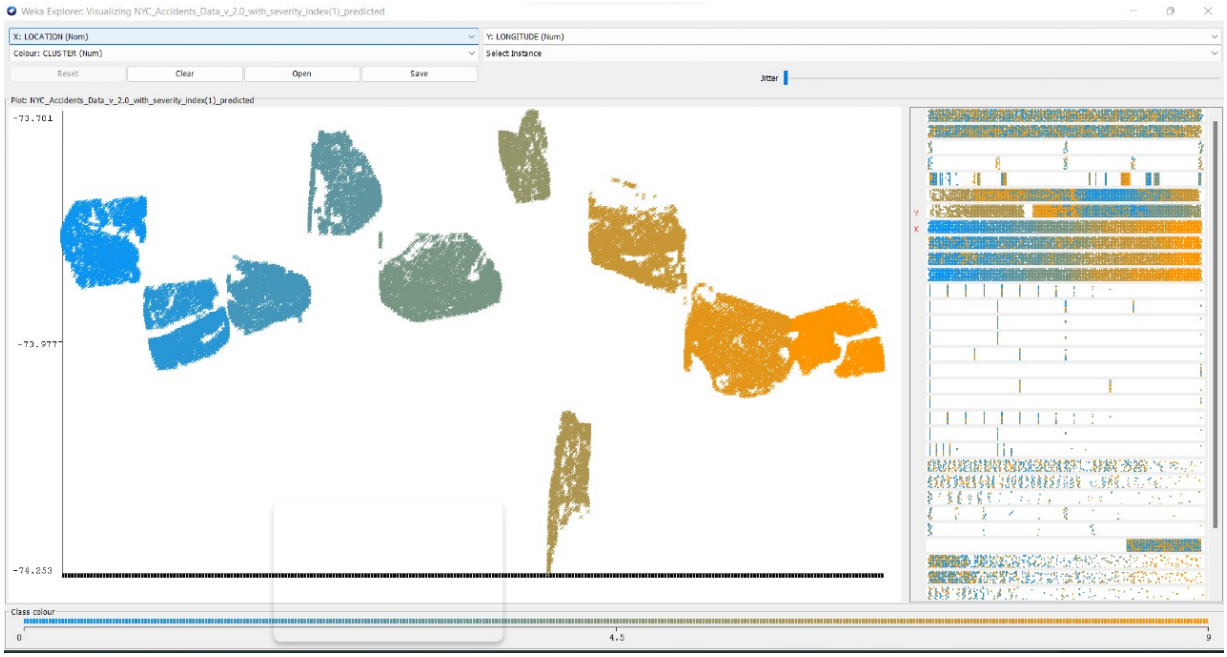
The fifth and the last method was SVM, which required a lot of computation time. We set it up with the following parameters: batch size = 100, cache size = 40, kernel type was radial basis function. We tried with different kernel types and turned out the current type of kernel type provides the highest accuracy of 75.5198 percent among other types. SVM has the second-highest accuracy below the J48 decision tree. We also tweaked the dataset a little bit by removing some of the independent variables, the change in the accuracy was around 1 to 3 percent for each classifier, thus we did not expand much on that.

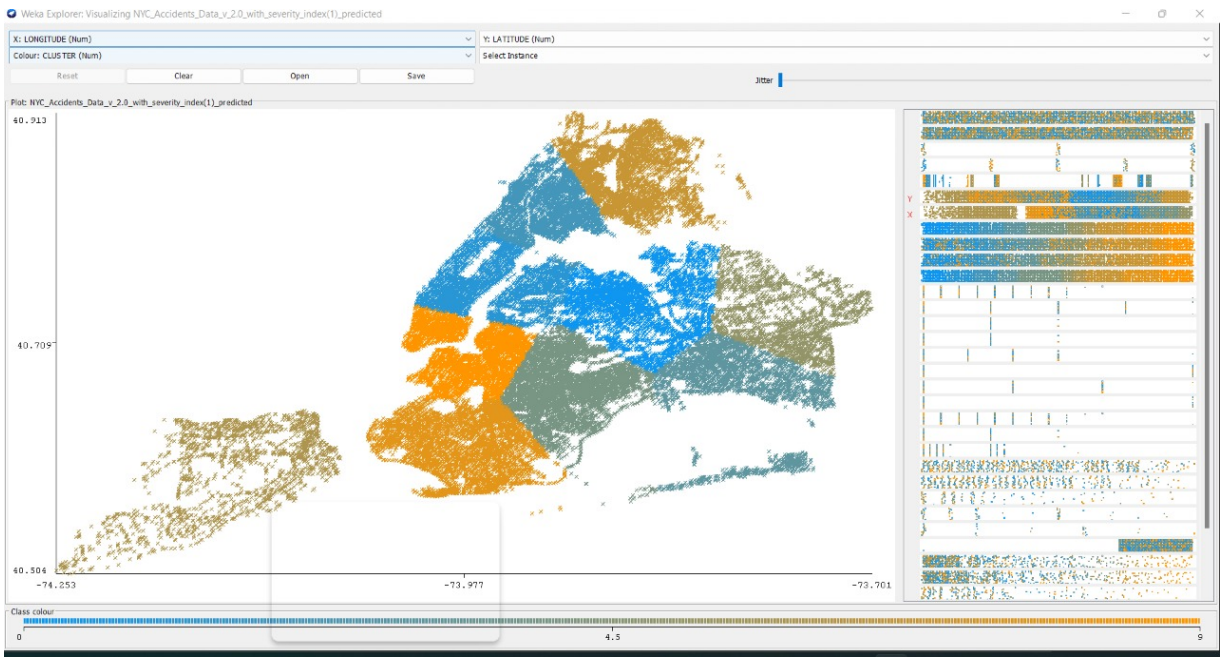
Figure 7: Best Classifier Based on Comparison

	Accuracy (%)	TP Rate	FP Rate	Precision	Recall	F-measure	ROC Area	PRC Area
ZeroR	72.7121	0.727	0.727	0.727	0.727	?	0.5	0.601
Decision Tree	76.3663	0.764	0.596	0.775	0.764	?	0.654	0.719
Bagging	72.2979	0.723	0.537	0.693	0.723	0.699	0.659	0.713
KNN = 1	72.8424	0.728	0.556	0.694	0.728	0.697	0.643	0.696
KNN = 100	74.6051	0.746	0.665	0.744	0.746	?	0.704	0.752
KNN = 500	73.4014	0.734	0.705	0.733	0.734	?	0.701	0.749
KNN = 1000	72.9019	0.729	0.721	0.729	0.729	?	0.696	0.746
SVM	75.5198	0.755	0.636	0.752	0.755	?	0.56	0.638

From Figure 7 we can see that over accuracy was consistent among different classifiers, although we also have a high false-positive rate. This indicates that all those methods could not build up a perfect connection between X and Y. We also noticed that most of them have missing F-measure, maybe because the proportion of the dataset was too big compared to relatively small accidents that involves injuries.


X: ON STREET NAME (Nom)
Y: LONGITUDE (Num)
Colour: CLUSTER (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: NYC_Accidents_Data_v_2.0_with_severity_index(1)_predicted
-73.701
-73.977
-74.253
Class colour
0
4.5
9


Weka Explorer: Visualizing NYC_Accidents_Data_v_2.0_with_severity_index(1)_predicted
X: LOCATION (Nom)
Y: LONGITUDE (Num)
Colour: CLUSTER (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: NYC_Accidents_Data_v_2.0_with_severity_index(1)_predicted
-73.701
-73.977
-74.253
Class colour
0
4.5
9



Here we focus on the temporo-spatial prediction of the risk of accident throughout New York City from the dataset. Basically, each color has their representation regarding on each cell on the type of accidents that have happened pertaining that duration (clusters) with crush intervals and instances.

Conclusion

We tried several different classifiers for our dataset, the outcome was not as ideal as we thought it would be, but we still got some useful results from them. Overall, Bagging has the most complete Detailed Accuracy By Class table. While the Decision tree has the highest accuracy, its false-positive rate is also lower than most of the other methods. K-nearest neighbor when K = 100 neighbors has the highest ROC area.

We also generated a deep learning model for predicting the risk of traffic accidents in New York City. This is based on clustering, classification and prediction on the numbers of cars in the accident

- The stated leading cause of collisions in NYC is driver inattentiveness
- The variability of the counts observed in the traffic volume data suggest non uniform measurement, and therefore could result in inaccurate counts.
- Aggregating by hour does not take into consideration fluctuations in traffic patterns, for example seasonal or monthly changes in normal traffic.
- Rideshare companies could use information like this as an additional layer of data when generating routes to avoid areas of increased risk of accident.

Appendix A

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
import pandas as pd

# import data set
df = pd.read_csv (r'/Users/ahmedshata/Desktop/MBA 540/NYC Accidents Data.csv')

# declare variables
latitude = df.iloc[:,4]
longitude = df.iloc[:,5]
X = df.iloc[:,4:6]

# plot data set
plt.scatter(
  latitude, longitude,
   c = 'white', marker = 'o',
   edgecolor = 'black', s = 5
)
plt.xlabel('Latitude')
plt.ylabel('Longitude')

from sklearn.cluster import KMeans
# calculate distortion for a range of number of cluster
SSE = []
for i in range(1, 21):
   km = KMeans(
      n_clusters = i, init = 'random',
      n_init = 20, max_iter = 300,
      tol = 1e-04, random_state = 0
   )
   km.fit(X)
   SSE.append(km.inertia_) # compute the SSE and save it in "SSE"

plt.plot(range(1, 21), SSE, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
```

```
plt.show()

from sklearn.cluster import KMeans
km = KMeans(
    n_clusters = 10, init = 'random',
    n_init = 10, max_iter = 300, random_state = 0
)
y_km = km.fit_predict(X)

# plot the clusters
plt.scatter(
    X[y_km == 0].LATITUDE, X[y_km == 0].LONGITUDE,
    s = 5, c = 'lightgreen',
    marker = 's', edgecolor = 'lightgreen',
    label = 'cluster 1'
)

plt.scatter(
    X[y_km == 1].LATITUDE, X[y_km == 1].LONGITUDE,
    s = 5, c = 'orange',
    marker = 'o', edgecolor = 'orange',
    label = 'cluster 2'
)

plt.scatter(
    X[y_km == 2].LATITUDE, X[y_km == 2].LONGITUDE,
    s = 5, c = 'lightblue',
    marker = 'v', edgecolor = 'lightblue',
    label = 'cluster 3'
)

plt.scatter(
    X[y_km == 3].LATITUDE, X[y_km == 3].LONGITUDE,
    s = 5, c = 'red',
    marker = 'x', edgecolor = 'red',
    label = 'cluster 4'
)

plt.scatter(
    X[y_km == 4].LATITUDE, X[y_km == 4].LONGITUDE,
    s = 5, c = 'indigo',
    marker = 's', edgecolor = 'indigo',
```

```
    label = 'cluster 5'
)

plt.scatter(
    X[y_km == 5].LATITUDE, X[y_km == 5].LONGITUDE,
    s = 5, c = 'gold',
    marker = 'o', edgecolor = 'gold',
    label = 'cluster 6'
)

plt.scatter(
    X[y_km == 6].LATITUDE, X[y_km == 6].LONGITUDE,
    s = 5, c = 'steelblue',
    marker = 'v', edgecolor = 'steelblue',
    label = 'cluster 7'
)

plt.scatter(
    X[y_km == 7].LATITUDE, X[y_km == 7].LONGITUDE,
    s = 5, c = 'purple',
    marker = 'x', edgecolor = 'purple',
    label = 'cluster 8'
)

plt.scatter(
    X[y_km == 8].LATITUDE, X[y_km == 8].LONGITUDE,
    s = 5, c = 'pink',
    marker = 's', edgecolor = 'pink',
    label = 'cluster 9'
)

plt.scatter(
    X[y_km == 9].LATITUDE, X[y_km == 9].LONGITUDE,
    s = 5, c = 'salmon',
    marker = 'o', edgecolor = 'salmon',
    label = 'cluster 10'
)

import csv
import pandas as pd
# convert array to column vector
df = pd.DataFrame(y_km)
```

```
# output to csv
df.to_csv('cluster.csv')
```

Classification

Run information

Scheme: weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Relation: NYC_Accidents_Data_v_2.0_with_severity_index(1)

Instances: 68872

Attributes: 34

- CRASH DATE
- CRASH TIME
- Crash Time Interval
- BOROUGH
- ZIP CODE
- LATITUDE
- LONGITUDE
- LOCATION
- ON STREET NAME
- CROSS STREET NAME
- OFF STREET NAME
- NUMBER OF PERSONS INJURED
- person Injury index
- NUMBER OF PERSONS KILLED
- person killed index
- NUMBER OF PEDESTRIANS INJURED
- NUMBER OF PEDESTRIANS KILLED
- NUMBER OF CYCLIST INJURED

NUMBER OF CYCLIST KILLED

NUMBER OF MOTORIST INJURED

NUMBER OF MOTORIST KILLED

SEVERITY INDEX

CONTRIBUTING FACTOR VEHICLE 1

CONTRIBUTING FACTOR VEHICLE 2

CONTRIBUTING FACTOR VEHICLE 3

CONTRIBUTING FACTOR VEHICLE 4

CONTRIBUTING FACTOR VEHICLE 5

COLLISION_ID

VEHICLE TYPE CODE 1

VEHICLE TYPE CODE 2

VEHICLE TYPE CODE 3

VEHICLE TYPE CODE 4

VEHICLE TYPE CODE 5

CLUSTER

Test mode: 10-fold cross-validation

Classifier model

IB1 instance-based classifier using 1 nearest neighbour(s) for classification

Time taken to build model: 0.06 seconds

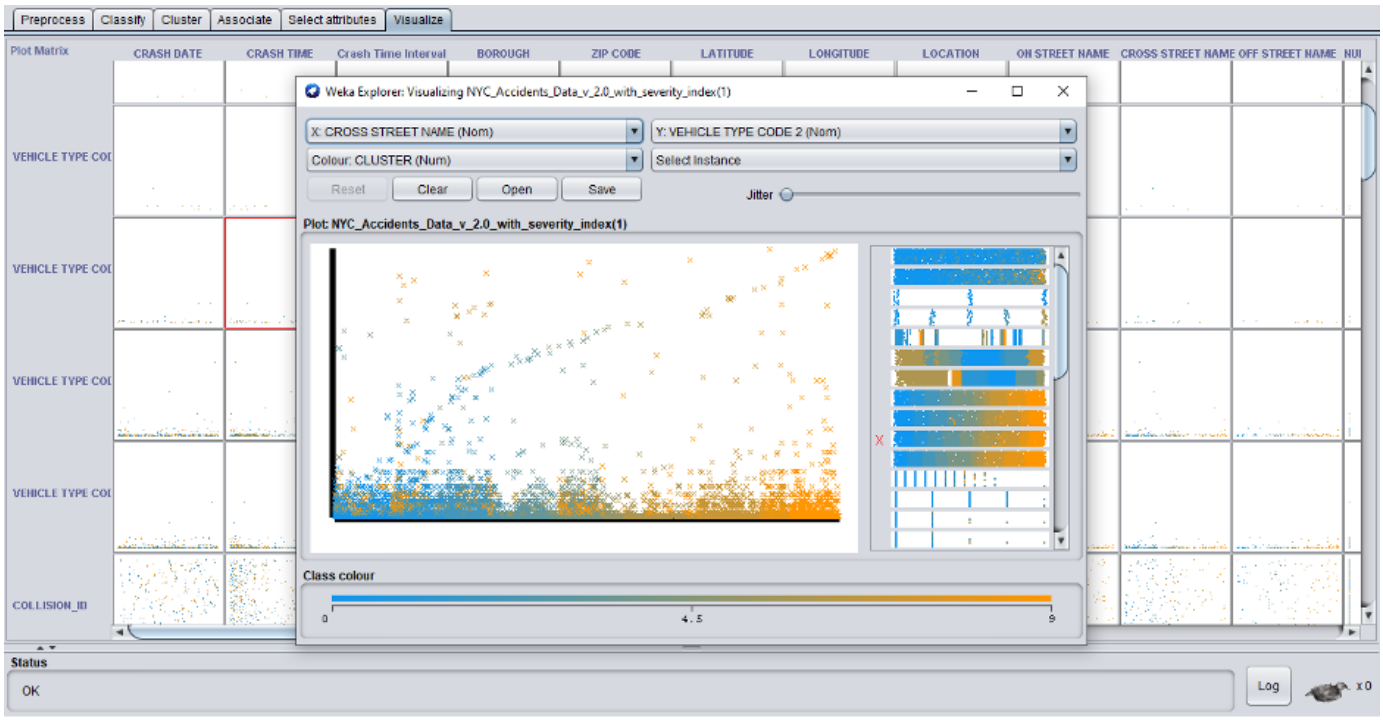
Cross-validation

Summary

Correlation coefficient	0.5901
Mean absolute error	1.4509
Root mean squared error	2.6974
Relative absolute error	55.5975 %

Root relative squared error 90.4688 %

Total Number of Instances 68872



Cluster

kMeans

Number of iterations: 9

Within cluster sum of squared errors: 498823.6933926288

Initial starting points (random):

Cluster 0: 3/14/2020,19:06:00,Afternoon,QUEENS,11434,40.67896,-73.774,'POINT (-73.774 40.67896)','BELT PARKWAY','3 AVENUE','168-44 127 AVENUE',0,0,0,0,0,0,0,0,0,0,0,'Backing Unsafely',Unspecified,Unspecified,Unspecified,Unspecified,4300512,Sedan,'Station Wagon/Sport Utility Vehicle',Sedan,Sedan,Sedan,3

Cluster 1: 4/19/2020,17:10:00,Afternoon,QUEENS,11435,40.709026,-73.81333,'POINT (-73.81333 40.709026)','85 DRIVE','143 STREET','772 EDGEWATER ROAD',0,0,0,0,0,0,0,0,0,0,0,'Driver Inattention/Distraction',Unspecified,Unspecified,Unspecified,Unspecified,4309646,Sedan,Sedan,Sedan,Sedan,Sedan,3

Missing values globally replaced with mean/mode

Final cluster centroids:

		Cluster#	
Attribute	Full Data	0	1
	(68872.0)	(31773.0)	(37099.0)
CRASH DATE	1/18/2020	1/18/2020	3/6/2020
CRASH TIME	0:00:00	0:00:00	0:00:00
Crash Time Interval	Afternoon	Afternoon	Afternoon
BOROUGH	BROOKLYN	BROOKLYN	BROOKLYN
ZIP CODE	10916.2397	10910.1985	10921.4137
LATITUDE	40.727	40.7219	40.7314
LONGITUDE	-73.9109	-73.9169	-73.9058
LOCATION	INT (-73.91282 40.861862)	POINT (-73.98453 40.696033)	POINT (-73.91282 40.861862)
ON STREET NAME	BELT PARKWAY	BELT PARKWAY	BELT PARKWAY
CROSS STREET NAME	3 AVENUE	3 AVENUE	3 AVENUE
OFF STREET NAME	772 EDGEWATER ROAD	772 EDGEWATER ROAD	772 EDGEWATER ROAD
NUMBER OF PERSONS INJURED	0.3627	0.3268	0.3934
person Injury index	0.3339	0.3013	0.3617
NUMBER OF PERSONS KILLED	0.0019	0.0021	0.0017
person killed index	0.0186	0.0208	0.0167
NUMBER OF PEDESTRIANS INJURED	0.057	0.0605	0.054
NUMBER OF PEDESTRIANS KILLED	0.0007	0.001	0.0005
NUMBER OF CYCLIST INJURED	0.0449	0.0369	0.0518
NUMBER OF CYCLIST KILLED	0.0001	0.0001	0.0001
NUMBER OF MOTORIST INJURED	0.2608	0.2295	0.2877

NUMBER OF MOTORIST KILLED	0.001	0.0009	0.0011
SEVERITY INDEX	0.3524	0.3221	0.3784
CONTRIBUTING FACTOR VEHICLE 1	Unspecified	Unspecified Driver	Inattention/Distraction
CONTRIBUTING FACTOR VEHICLE 2	Unspecified	Unspecified	Unspecified
CONTRIBUTING FACTOR VEHICLE 3	Unspecified	Unspecified	Unspecified
CONTRIBUTING FACTOR VEHICLE 4	Unspecified	Unspecified	Unspecified
CONTRIBUTING FACTOR VEHICLE 5	Unspecified	Unspecified	Unspecified
COLLISION_ID	4305770.6113	4305804.6327	4305741.4741
VEHICLE TYPE CODE 1	Sedan	Sedan	Sedan
VEHICLE TYPE CODE 2	Sedan	Sedan	Sedan
VEHICLE TYPE CODE 3	Sedan	Sedan	Sedan
VEHICLE TYPE CODE 4	Sedan	Sedan	Sedan
VEHICLE TYPE CODE 5	Sedan	Sedan	Sedan
CLUSTER	4.3543	5.8054	3.1116

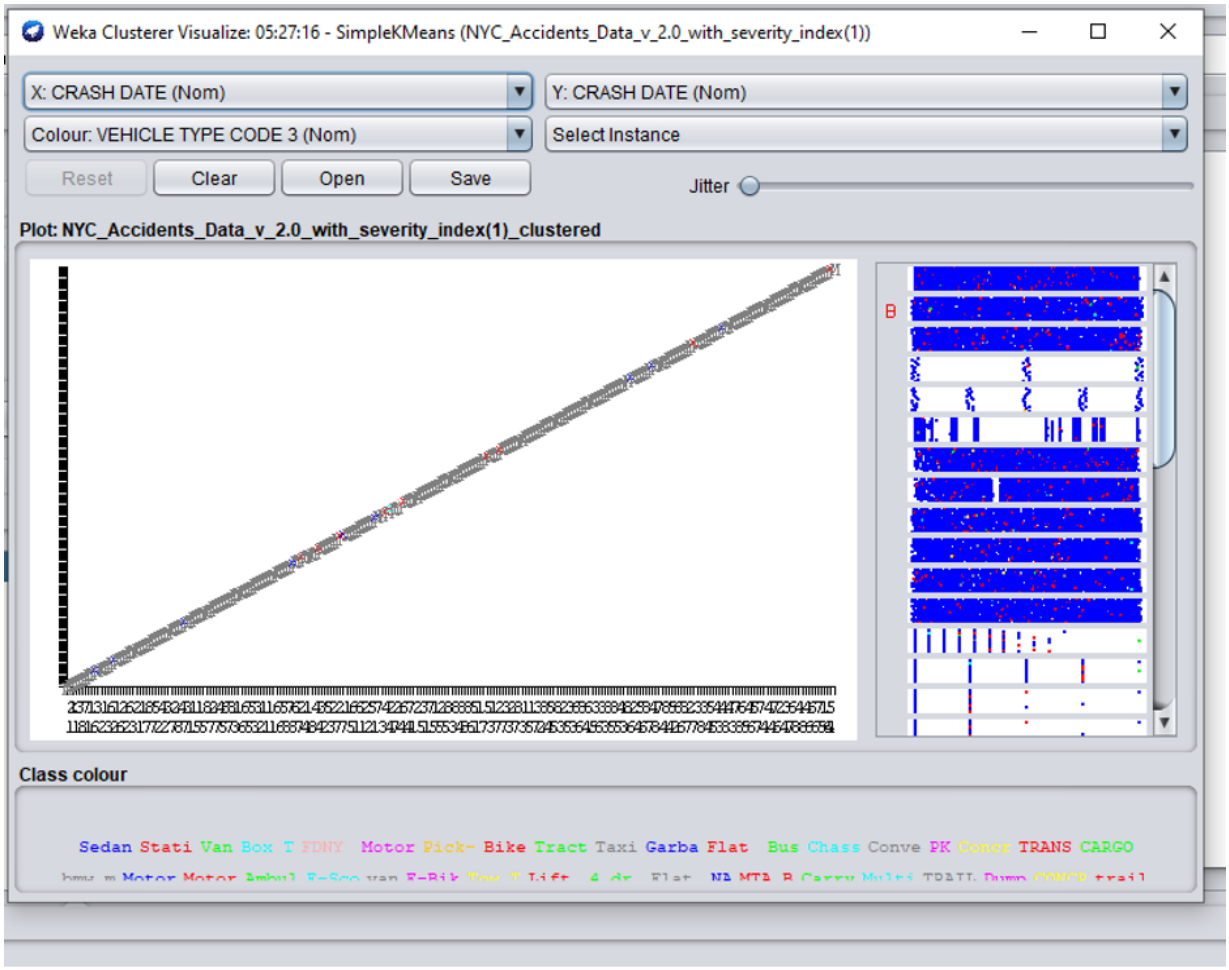
Time taken to build model (full training data) : 5.19 seconds

Model and evaluation on training set

Clustered Instances

0 31773 (46%)

1 37099 (54%)



Clustering model

CascadeSimpleKMeans:

cascade> mean CH: [20467.41 14160.09 11792.88 10549.86 8219.24 7340.54 6414.73 5838.17 5425.02]cascade> k (yields highest mean CH): 2

cascade> seed (highest CH for k=2) : -215436171

Time taken to build model (full training data) : 361.28 seconds

Model and evaluation on training set

Clustered Instances

0 20013 (29%)

1 48859 (71%)