# Predicting Car Accidents Recurrence in New York

**Takudzwa Chikwanda**
tchikwanda1@student.gsu.edu

Georgia State University

## Abstract

The main agenda of this research is to find a pattern in the recurrence of accidents and try to determine the model to use when trying to determine the number of vehicles that might get involved in a car crash given other conditions like weather, Lighting Conditions etc. On predicting the number of vehicles involved, Naive Bayes Classifiers were applied on the data set.

## 1 Introduction

The research is about discovering or rather detecting patterns in the occurrence of accidents. This is archived by making use of machine learning algorithms which detects the pattern or structure of data from the past occurrences of accidents. The algorithm takes the features of each instance as independent. The algorithm takes in labeled data.

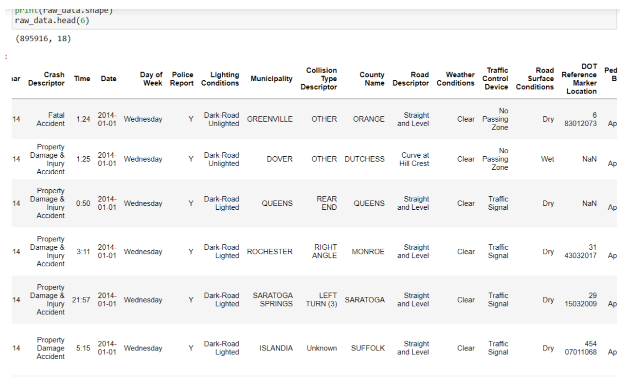
Naïve Bayes is a classification technique based on Bayes Theorem with an assumption of independence among predictors. Basically, Bayes theorem provides a way of calculating posterior probability from Marginal Likelihood, Likelihood and prior.

Also, time series ARIMA model is going to be used to forecast details about the number of vehicles that might be affected by car crashes during the course of the day

### 1.1 Problem

Since New York is a very busy town, there should not be much time wasted whenever an accident occurs rather, they should find ways of avoiding these accidents from happening by all means possible. In other words, they should be ready whenever car accidents happen or have ways of avoiding the accidents. There should be a software that helps New York by predicting the recurrence of these accidents so that they can either avoid the accident or they can clean the scene as quick as possible to avoid time lost. The main agenda of this research is to device an algorithm that would assist motorist in the city of New York by predicting the occurrences of an accident and also, inform the drivers on the possibilities so that they can be more careful.

## 2 Exploratory Data Analysis



```
print(raw_data.shape)
raw_data.head(6)
(895916, 18)
```

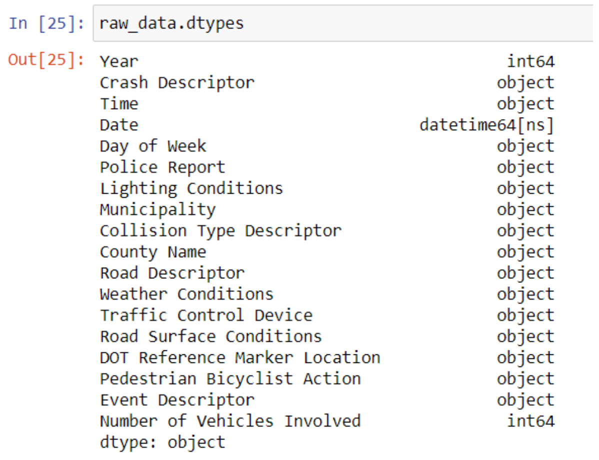
| ear | Crash Descriptor | Time | Date | Day of Week | Police Report | Lighting Conditions | Municipality | Collision Type Descriptor | County Name | Road Descriptor | Weather Conditions | Traffic Control Device | Road Surface Conditions | DOT Reference Marker Location | Ped B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | Fatal Accident | 1:24 | 2014-01-01 | Wednesday | Y | Dark-Road Unlighted | GREENVILLE | OTHER | ORANGE | Straight and Level | Clear | No Passing Zone | Dry | 6 83012073 | Ap |
| 14 | Property Damage & Injury Accident | 1:25 | 2014-01-01 | Wednesday | Y | Dark-Road Unlighted | DOVER | OTHER | DUTCHESS | Curve at Hill Crest | Clear | No Passing Zone | Wet | NaN | Ap |
| 14 | Property Damage & Injury Accident | 0:50 | 2014-01-01 | Wednesday | Y | Dark-Road Lighted | QUEENS | REAR END | QUEENS | Straight and Level | Clear | Traffic Signal | Dry | NaN | Ap |
| 14 | Property Damage & Injury Accident | 3:11 | 2014-01-01 | Wednesday | Y | Dark-Road Lighted | ROCHESTER | RIGHT ANGLE | MONROE | Straight and Level | Clear | Traffic Signal | Dry | 31 43032017 | Ap |
| 14 | Property Damage & Injury Accident | 21:57 | 2014-01-01 | Wednesday | Y | Dark-Road Lighted | SARATOGA SPRINGS | LEFT TURN (3) | SARATOGA | Straight and Level | Clear | Traffic Signal | Dry | 29 15032009 | Ap |
| 14 | Property Damage Accident | 5:15 | 2014-01-01 | Wednesday | Y | Dark-Road Lighted | ISLANDIA | Unknown | SUFFOLK | Straight and Level | Clear | Traffic Signal | Dry | 454 07011068 | Ap |

Figure 1: The image above shows data that is going to be used in this project

#### 2.0.1 Dataset description

The car crashes dataset can be found on https://catalog.data.gov/dataset/motor-vehicle-crashes-case-information-beginning-2009. The data can be obtained in both json format and csv file format. The dataset consists of data about motor vehicle crashes in New York. The occurrences in the data are from 1 January 2014 to 31 December 2016.

The data has the date and also the time that the accident occurred and this helps to determine weather there is some form of pattern in the recurrence of the accidents or not. The data also has lightning conditions when the accident occurred which is one factors that might affect the occurrence of an accident. One of the columns is Crash Descriptor which describes the severity of the accident, if either it was a fatal accident or it was a Property Damage Injury Accident.



```
In [25]: raw_data.dtypes

Out[25]: Year                                 int64
         Crash Descriptor                    object
         Time                                object
         Date                        datetime64[ns]
         Day of Week                         object
         Police Report                       object
         Lighting Conditions                 object
         Municipality                        object
         Collision Type Descriptor           object
         County Name                         object
         Road Descriptor                     object
         Weather Conditions                  object
         Traffic Control Device              object
         Road Surface Conditions             object
         DOT Reference Marker Location       object
         Pedestrian Bicyclist Action         object
         Event Descriptor                    object
         Number of Vehicles Involved          int64
         dtype: object
```

Figure 2: The image above shows data types of the car crashes dataset

## 2.1 Data Analysis

Dataset consists of 895 816 instances of accidents and each instance has 18 features/attributes. The data type for Year is int64, for Date is datetime64 and number of vehicles involved is int64, and the rest are of object type as seen in the Figure above. The data is incomplete and inconsistent. It contains a lot of unknown, other and blank values. The data is not clean itself hence data cleaning is required if we are to expect quality results. Of the 18 features, not all features are relevant in prediction of the occurrences

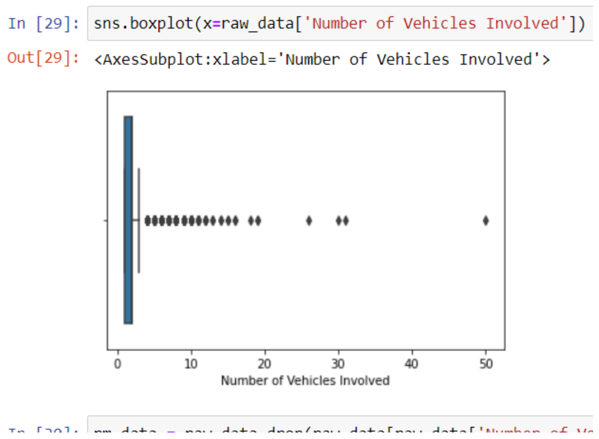
## 2.2 Visualizations

Performed a counting to see how many values gets up to 50 in the dataset This was due to the reason below hence needed to clean the data and remove outliers

## 2.3 Data Preprocessing

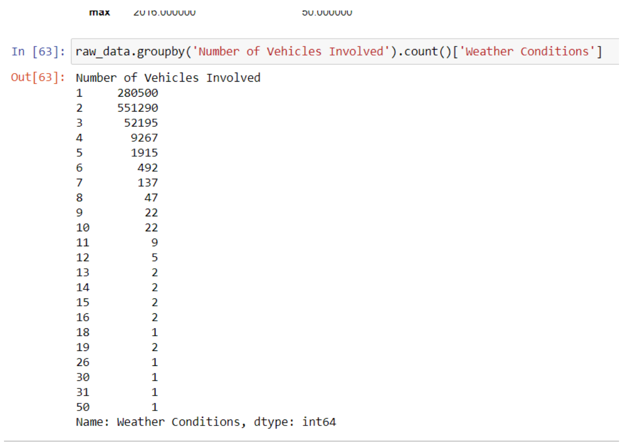
### 2.3.1 Drop columns and rows

The idea is to try and build a more generalized model that is more accurate and avoid overfitting. When building the model outliers will just add noise to model



```
In [29]: sns.boxplot(x=raw_data['Number of Vehicles Involved'])
Out[29]: <AxesSubplot:xlabel='Number of Vehicles Involved'>
```

Figure 3: The image above shows data types of the car crashes dataset



```
max      2016.000000                    50.000000

In [63]: raw_data.groupby('Number of Vehicles Involved').count()['Weather Conditions']

Out[63]: Number of Vehicles Involved
         1      280500
         2      551290
         3       52195
         4        9267
         5        1915
         6         492
         7         137
         8          47
         9          22
         10         22
         11          9
         12          5
         13          2
         14          2
         15          2
         16          2
         18          1
         19          2
         26          1
         30          1
         31          1
         50          1
         Name: Weather Conditions, dtype: int64
```

Figure 4: The image above shows data types of the car crashes dataset

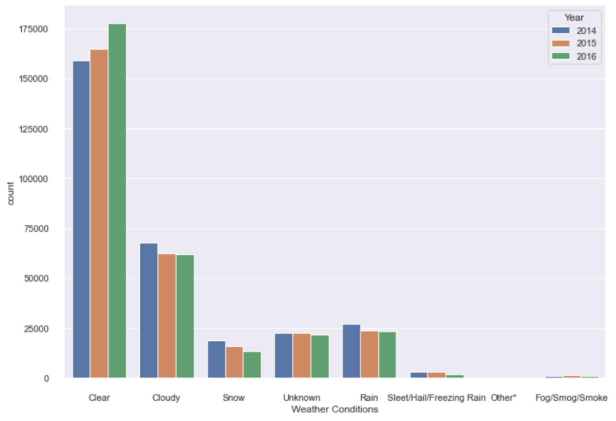


Figure 5: The image above shows weather for the accidents occurrences by year

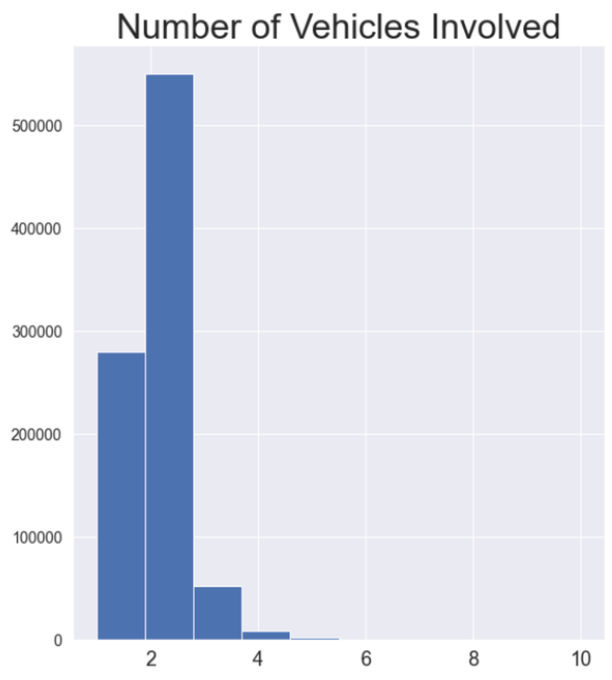


Figure 6: The image above shows weather for the accidents occurrences by year

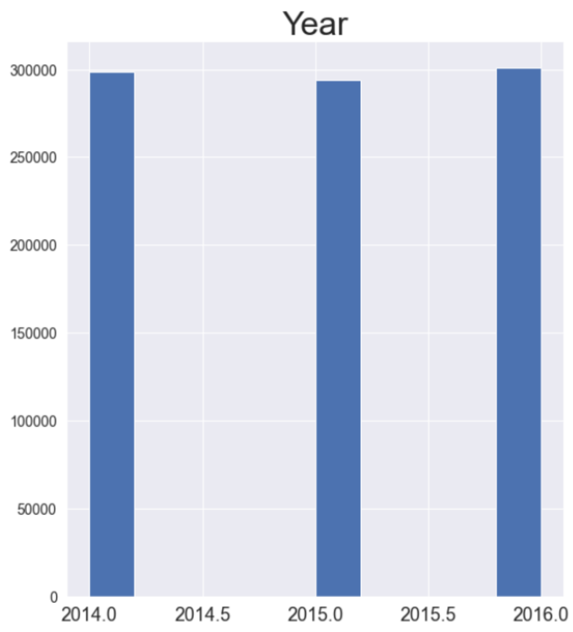
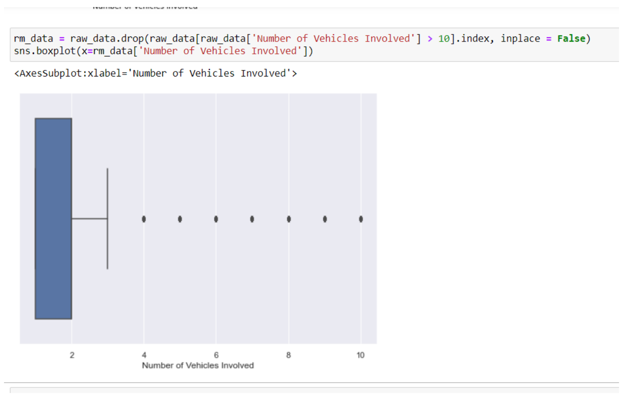


Figure 7: The image above shows weather for the accidents occurrences by year

instead of helping it create a more generalized model. Dropped outliers from the Number of Vehicles involved. The figure below shows boxplot of the Number of Vehicles involved after dropping all instances that has the Number of Vehicles involved more than 10.

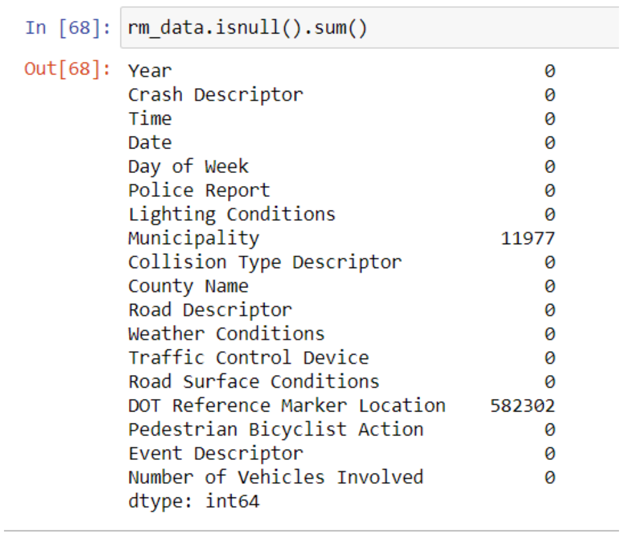


```
rm_data = raw_data.drop(raw_data[raw_data['Number of Vehicles Involved'] > 10].index, inplace = False)
sns.boxplot(x=rm_data['Number of Vehicles Involved'])
```

```
<AxesSubplot:xlabel='Number of Vehicles Involved'>
```

Figure 8: Boxplot of the Number of Vehicles involved

DOT Reference Marker Location has more that half of its values empty. Trying to use the mode on DOT Reference Marker Location will just affect the accuracy of the model.The best solution is just to drop the column. Day of Week and Police Report are not very important factors to consider when trying to predict Number of Vehicles involved in a car crash. The two columns had to be dropped.



```
In [68]: rm_data.isnull().sum()

Out[68]: Year                                0
         Crash Descriptor                    0
         Time                                0
         Date                                0
         Day of Week                         0
         Police Report                       0
         Lighting Conditions                 0
         Municipality                    11977
         Collision Type Descriptor           0
         County Name                         0
         Road Descriptor                     0
         Weather Conditions                  0
         Traffic Control Device              0
         Road Surface Conditions             0
         DOT Reference Marker Location  582302
         Pedestrian Bicyclist Action         0
         Event Descriptor                    0
         Number of Vehicles Involved         0
         dtype: int64
```

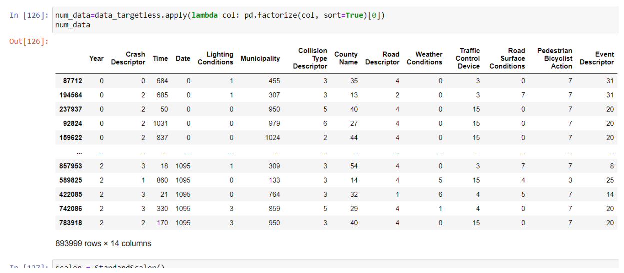
Figure 9: The image above shows the counts of the null values of the dataset per feature

Also dropped the County Name and Municipality to avoid noise when trying to build the model because the two columns does not really impact Number of Vehicles involved.

Lastly as part of data preprocessing had to check for duplicates. There were quite a significant number of duplicates which needed to be handled. Hence dropped the duplicates.

#### 2.3.2 Change to numerical data

For us to be able to build the model using the data, there is need for it to be numeric so that the algorithms can work on the data. The code in the figure below transforms the data from text to numerical just so that the model can be built easily.



```
In [126]: num_data=data_targetless.apply(lambda col: pd.factorize(col, sort=True)[0])
          num_data
Out[126]:
```

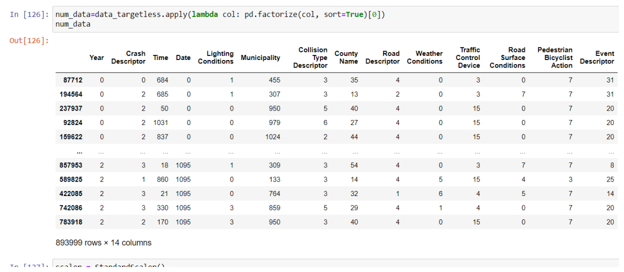
| | Year | Crash Descriptor | Time | Date | Lighting Conditions | Municipality | Collision Type Descriptor | County Name | Road Descriptor | Weather Conditions | Traffic Control Device | Road Surface Conditions | Pedestrian Bicyclist Action | Event Descriptor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **87712** | 0 | 0 | 684 | 0 | 1 | 455 | 3 | 35 | 4 | 0 | 3 | 0 | 7 | 31 |
| **194564** | 0 | 2 | 685 | 0 | 1 | 307 | 3 | 13 | 2 | 0 | 3 | 7 | 7 | 31 |
| **237937** | 0 | 2 | 50 | 0 | 0 | 950 | 5 | 40 | 4 | 0 | 15 | 0 | 7 | 20 |
| **92824** | 0 | 2 | 1031 | 0 | 0 | 979 | 6 | 27 | 4 | 0 | 15 | 0 | 7 | 20 |
| **159622** | 0 | 2 | 837 | 0 | 0 | 1024 | 2 | 44 | 4 | 0 | 15 | 0 | 7 | 20 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **857953** | 2 | 3 | 18 | 1095 | 1 | 309 | 3 | 54 | 4 | 0 | 3 | 7 | 7 | 8 |
| **589825** | 2 | 1 | 860 | 1095 | 0 | 133 | 3 | 14 | 4 | 5 | 15 | 4 | 3 | 25 |
| **422085** | 2 | 3 | 21 | 1095 | 0 | 764 | 3 | 32 | 1 | 6 | 4 | 5 | 7 | 14 |
| **742086** | 2 | 3 | 330 | 1095 | 3 | 859 | 5 | 29 | 4 | 1 | 4 | 0 | 7 | 20 |
| **783918** | 2 | 2 | 170 | 1095 | 3 | 950 | 3 | 40 | 4 | 0 | 15 | 0 | 7 | 20 |

893999 rows × 14 columns

Figure 10: The image above shows the counts of the null values of the dataset per feature

#### 2.3.3 Normalize data

After converting the data to numerical values it is important that the data is normalized in order to be able to get more accurate results from the Naive Bayes algorithms. This means that the data will now follow a distribution in which the mean is 0 and standard deviation is 1. The values in the table are known as the z-scores. This is achieved using StandardScaler from the scikit-learn which then normalizes the data. The figure below shows the normalised data.



```
In [126]: num_data=data_targetless.apply(lambda col: pd.factorize(col, sort=True)[0])
          num_data
Out[126]:
```

| | Year | Crash Descriptor | Time | Date | Lighting Conditions | Municipality | Collision Type Descriptor | County Name | Road Descriptor | Weather Conditions | Traffic Control Device | Road Surface Conditions | Pedestrian Bicyclist Action | Event Descriptor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **87712** | 0 | 0 | 684 | 0 | 1 | 455 | 3 | 35 | 4 | 0 | 3 | 0 | 7 | 31 |
| **194564** | 0 | 2 | 685 | 0 | 1 | 307 | 3 | 13 | 2 | 0 | 3 | 7 | 7 | 31 |
| **237937** | 0 | 2 | 50 | 0 | 0 | 950 | 5 | 40 | 4 | 0 | 15 | 0 | 7 | 20 |
| **92824** | 0 | 2 | 1031 | 0 | 0 | 979 | 6 | 27 | 4 | 0 | 15 | 0 | 7 | 20 |
| **159622** | 0 | 2 | 837 | 0 | 0 | 1024 | 2 | 44 | 4 | 0 | 15 | 0 | 7 | 20 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **857953** | 2 | 3 | 18 | 1095 | 1 | 309 | 3 | 54 | 4 | 0 | 3 | 7 | 7 | 8 |
| **589825** | 2 | 1 | 860 | 1095 | 0 | 133 | 3 | 14 | 4 | 5 | 15 | 4 | 3 | 25 |
| **422085** | 2 | 3 | 21 | 1095 | 0 | 764 | 3 | 32 | 1 | 6 | 4 | 5 | 7 | 14 |
| **742086** | 2 | 3 | 330 | 1095 | 3 | 859 | 5 | 29 | 4 | 1 | 4 | 0 | 7 | 20 |
| **783918** | 2 | 2 | 170 | 1095 | 3 | 950 | 3 | 40 | 4 | 0 | 15 | 0 | 7 | 20 |

893999 rows × 14 columns

Figure 11: The image above shows the counts of the null values of the dataset per feature

## 3 Methods

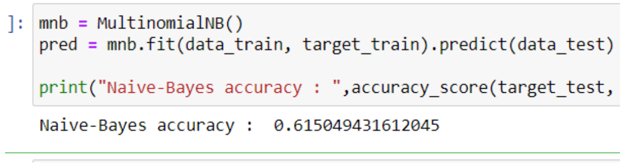
The agenda for this is to try and determine the number of vehicles involved given a set of conditions. In addition, will map time series to forecast the number of vehicles involved in an accident on a daily basis.

### 3.1 Naive Bayes

The first method to be applied on the given dataset is Naive Bayes. Naive Bayes algorithm is based on Bayes rule and makes a strong assumption that the attributes are conditionally independent, given the class. While this independence assumption is often violated in practice, Naive Bayes nonetheless often delivers competitive classification accuracy. Based on that we applied Naive Bayes classifiers on the data.

#### 3.1.1 Multinomial

First Multinomial Naive Bayes was applied on the dataset to check how the algorithm will perform on the given dataset.

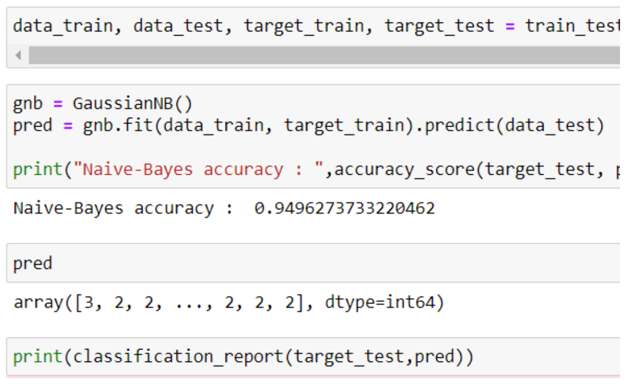


```
]: mnb = MultinomialNB()
   pred = mnb.fit(data_train, target_train).predict(data_test)

   print("Naive-Bayes accuracy : ",accuracy_score(target_test,

   Naive-Bayes accuracy :  0.615049431612045
```

Figure 12: The image above shows the fitting and the transformation on MultinomialNB Classifier

#### 3.1.2 Gaussian

Secondly applied Gaussian Naive Bayes algorithm on the dataset to check if it would perform better than the Multinomial Naive Bayes. The Gaussian Naive Bayes is more accurate when the numpy data is not normalized unlike the multinomial which requires the data to be normalized for it to function proper.



```
data_train, data_test, target_train, target_test = train_test

gnb = GaussianNB()
pred = gnb.fit(data_train, target_train).predict(data_test)

print("Naive-Bayes accuracy : ",accuracy_score(target_test, p

Naive-Bayes accuracy :  0.9496273733220462

pred

array([3, 2, 2, ..., 2, 2, 2], dtype=int64)

print(classification_report(target_test,pred))
```

Figure 13: The image above shows the Gaussian Naive Bayes Algorithm on action

### 3.2 Cross validation

Applied cross validation using Gaussian Classifier.

### 3.2.1 K- fold

Firstly the data was split into 10 folds which were then used to crossvalidate the Gaussian Naive Bayes model. This was repeated 20 times and plotted the box plot to show the accuracy of the model during its cross validation.

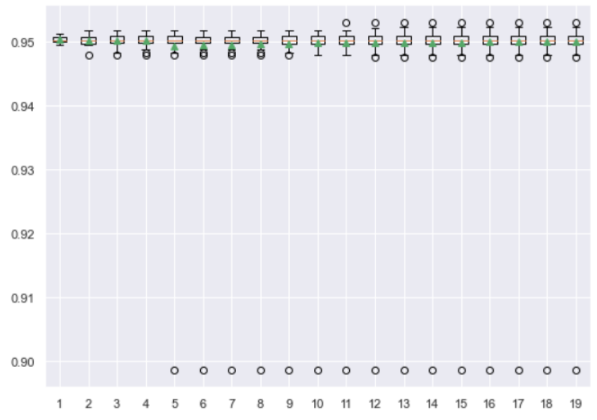


Figure 14: The image above shows the Gaussian Naive Bayes Algorithm on action

## 3.3 Time Series

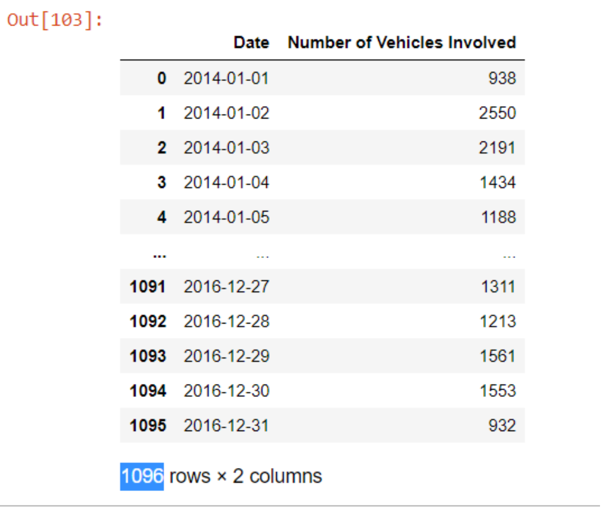
The goal was to forecast the number of accidents that might occur on a particular day in the future. Before applying the time series algorithms on the data, firstly had to group the sum of the Number of vehicles involved by date. This change affected the dataset as it reduced the data from 882024 rows to 1096 rows as there were only 1096 days between 1 January 2014 and 31 December 2016.

**The data** was then split randomly into train and test dataset so that we may be able to train and test the accuracy of the model after forecasting.

Had to plot a graph so that we may be able to visualize and understand the train data before applying the ARIMA model. ARIMA is an acronym that stands for Auto-Regressive Integrated Moving Average. It is a class of model that captures a suite of different standard temporal structures in time series data.

### 3.3.1 Future Prediction

The blue line on the graph of figure 17 shows the predictions of the forecast for the number of vehicles involved whilst the red line shows the plot of the test data.



Out[103]:

| | Date | Number of Vehicles Involved |
|---|---|---|
| **0** | 2014-01-01 | 938 |
| **1** | 2014-01-02 | 2550 |
| **2** | 2014-01-03 | 2191 |
| **3** | 2014-01-04 | 1434 |
| **4** | 2014-01-05 | 1188 |
| **...** | ... | ... |
| **1091** | 2016-12-27 | 1311 |
| **1092** | 2016-12-28 | 1213 |
| **1093** | 2016-12-29 | 1561 |
| **1094** | 2016-12-30 | 1553 |
| **1095** | 2016-12-31 | 932 |

1096 rows × 2 columns

Figure 15: The image above shows the Sums of Number of Vehicles involved grouped by date

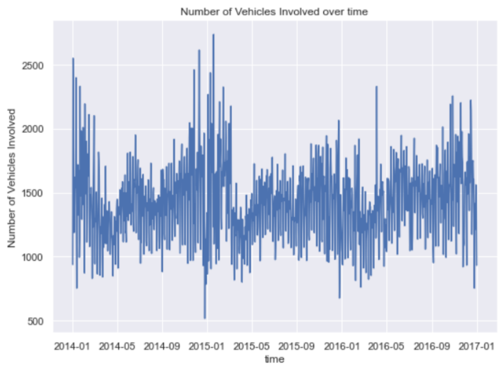


Figure 16: The image above shows the plot of training data for the time series

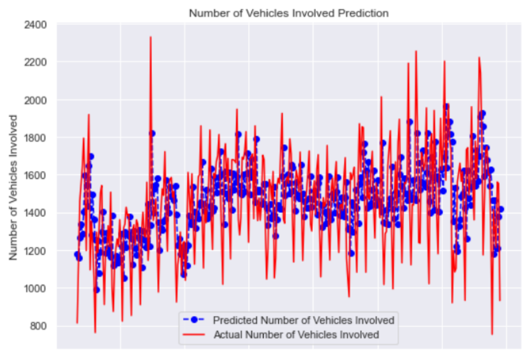


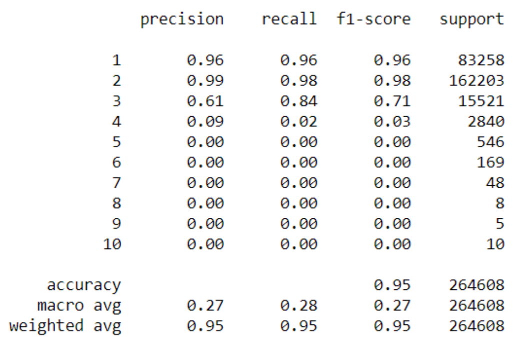
Figure 17: The image above shows the plot for the forecast of the time series

## 4 Results

These are results obtained from the two Algorithms which were applied to the data set.

### 4.1 Naive Bayes

The Gaussian Naive Bayes has proven to be more accurate on the particular data set as compared to Multinomial Naive Bayes as shown below.



```
              precision    recall  f1-score   support

           1       0.96      0.96      0.96     83258
           2       0.99      0.98      0.98    162203
           3       0.61      0.84      0.71     15521
           4       0.09      0.02      0.03      2840
           5       0.00      0.00      0.00       546
           6       0.00      0.00      0.00       169
           7       0.00      0.00      0.00        48
           8       0.00      0.00      0.00         8
           9       0.00      0.00      0.00         5
          10       0.00      0.00      0.00        10

    accuracy                           0.95    264608
   macro avg       0.27      0.28      0.27    264608
weighted avg       0.95      0.95      0.95    264608
```

Figure 18: The image above shows the accuracy of the Gaussian Naive Bayes algorithm

### 4.2 Time Series

The reduction of the number of rows from 882024 to 1096 affected the accuracy of the time series as it had only 1096 instances to use in training and testing.However there is some bit of trends that can be noticed from the data set. Even though the model does not the peaks as the the test data set, the model could follow through the red line which is the test target data set. The ARIMA model had a Mean Squared Error of 70194.80683904634.

### 4.3 Conclusion

In conclusion, the model to inform the drivers in the city of New York to inform them that on a particular we are expecting so much vehicle crashes according to the model so that they maybe careful. The increase in the number of accidents might be due to the weather or lighting condition or Road Surface Conditions so that drivers may need to be more careful during these conditions. Further analysis can be done on the data set using many other different algorithms