New York City Collision

By Kazi Siam, Ming Hin Cheung

Motivation

It was November 21, 2020, when I got a call from one of my friends who found himself in a car accident. I rushed to the scene and witnessed one of the hideous accidents I have ever seen. His friend was driving the car, and my friend was in the passenger seat. They were driving at 97-mph in a 25-mph limit. They were both intoxicated. They plowed into a 52-year-old man who was driving his car through a green light at the intersection. Fortunately, nothing happened to my friend. However, that 52-year-old man was pronounced dead at the scene. My friend's friend was charged with vehicular manslaughter. This is just one of the many vehicular accidents that take place in New York City.

According to the New York State Department of Health, there were 228,047 car accidents in all of New York City in 2018. That's about 19,000 car accidents per month,

4,750 car accidents per week, 678 car accidents per day, 28 car accidents per hour, or one-car accident every 2 minutes.

NYPD keeps track of all the motor vehicle collisions that happen in New York City. In our project, we will analyze one of the popular datasets in NYC Open Data called Motor Vehicle Collisions. Through our analysis, we will see which months and/or years most car accidents happen. We will also see which borough has more car accidents. We will also build a model that can predict in which borough the accident will happen based on the information about the accident.

It is interesting because people should be careful when driving a car. Through our work, people would know which borough and month are the most dangerous to drive. People should be extra cautious when they are driving in dangerous places and times of the year. It only takes one moment of negligence to change someone's life forever.

Data

We are getting the dataset from NYC Open data. We're choosing it because it's one of the largest and most well-used open data platforms in the world for public use. There are multiple ways to access the dataset, such as CSV, Excel Spreadsheets, and API Endpoint. There are limitations on the data, there're many empty fields/missing data in our dataset, therefore, we need to do data cleaning.

Data Cleaning

The first step we had to do before building our model is to clean the data. We exported the Motor Vehicle Collisions crash dataset from NYC Open Data.

```
In [3]: df.head()
```

Out[3]:

	CRASH DATE	CRASH TIME	BOROUGH	ZIP CODE	LATITUDE	LONGITUDE	LOCATION	ON STREET NAME	CROSS STREET NAME	OFF STREET NAME	...	CONTRIBUTING FACTOR VEHICLE 2	C
0	04/14/2021	5:32	NaN	NaN	NaN	NaN	NaN	BRONX WHITESTONE BRIDGE	NaN	NaN	...	Unspecified	
1	04/13/2021	21:35	BROOKLYN	11217.0	40.68358	-73.97617	(40.68358, -73.97617)	NaN	NaN	620 ATLANTIC AVENUE	...	NaN	
2	04/15/2021	16:15	NaN	NaN	NaN	NaN	NaN	HUTCHINSON RIVER PARKWAY	NaN	NaN	...	NaN	
3	04/13/2021	16:00	BROOKLYN	11222.0	NaN	NaN	NaN	VANDERVORT AVENUE	ANTHONY STREET	NaN	...	Unspecified	
4	04/12/2021	8:25	NaN	NaN	0.00000	0.00000	(0.0, 0.0)	EDSON AVENUE	NaN	NaN	...	Unspecified	

5 rows × 29 columns

Fig. The first 5 rows in the row dataset

The Motor Vehicle Collisions data contains information from all police-reported motor vehicle collisions in NYC. It has about 600,000 rows and 29 columns. Each row represents a crash event.

Because the dataset size is too huge, we're going to take 50,000 random samples from it to ease our process and to make it efficient for this task.

Also, there are several features that we don’t need for our problem. We’re using the Python Pandas Data Frame library to clean our dataset.

#	Column	Dtype
0	CRASH DATE	object
1	CRASH TIME	object
2	BOROUGH	object
3	ZIP CODE	object
4	LATITUDE	float64
5	LONGITUDE	float64
6	LOCATION	object
7	ON STREET NAME	object
8	CROSS STREET NAME	object
9	OFF STREET NAME	object
10	NUMBER OF PERSONS INJURED	float64
11	NUMBER OF PERSONS KILLED	float64
12	NUMBER OF PEDESTRIANS INJURED	int64
13	NUMBER OF PEDESTRIANS KILLED	int64
14	NUMBER OF CYCLIST INJURED	int64
15	NUMBER OF CYCLIST KILLED	int64
16	NUMBER OF MOTORIST INJURED	int64
17	NUMBER OF MOTORIST KILLED	int64
18	CONTRIBUTING FACTOR VEHICLE 1	object
19	CONTRIBUTING FACTOR VEHICLE 2	object
20	CONTRIBUTING FACTOR VEHICLE 3	object
21	CONTRIBUTING FACTOR VEHICLE 4	object
22	CONTRIBUTING FACTOR VEHICLE 5	object
23	COLLISION_ID	int64
24	VEHICLE TYPE CODE 1	object
25	VEHICLE TYPE CODE 2	object
26	VEHICLE TYPE CODE 3	object
27	VEHICLE TYPE CODE 4	object
28	VEHICLE TYPE CODE 5	object

Fig. Checking Column names in the raw dataset

Hence, we must drop those features since they will be having no effect on our output. For example, we dropped the columns, Zip Code, on street name, cross street name, vehicle code… because these columns aren't relevant to our problem, which is predicting which borough this accident happened.

```
#Checking for missing values
df.isnull().sum()
```

```
CRASH DATE                          0
CRASH TIME                          0
BOROUGH                         15218
ZIP CODE                        15220
LATITUDE                         5809
LONGITUDE                        5809
LOCATION                         5809
ON STREET NAME                  10200
CROSS STREET NAME               17831
OFF STREET NAME                 42303
NUMBER OF PERSONS INJURED           1
NUMBER OF PERSONS KILLED            1
NUMBER OF PEDESTRIANS INJURED       0
NUMBER OF PEDESTRIANS KILLED        0
NUMBER OF CYCLIST INJURED           0
NUMBER OF CYCLIST KILLED            0
NUMBER OF MOTORIST INJURED          0
NUMBER OF MOTORIST KILLED           0
CONTRIBUTING FACTOR VEHICLE 1     153
CONTRIBUTING FACTOR VEHICLE 2    7195
CONTRIBUTING FACTOR VEHICLE 3   46599
CONTRIBUTING FACTOR VEHICLE 4   49292
CONTRIBUTING FACTOR VEHICLE 5   49823
COLLISION_ID                        0
VEHICLE TYPE CODE 1               286
VEHICLE TYPE CODE 2              8553
VEHICLE TYPE CODE 3             46712
VEHICLE TYPE CODE 4             49313
VEHICLE TYPE CODE 5             49827
```

Fig: Checking for missing values.

Then, it comes to the part of dealing with the missing values. Since some of the features contain a lot of missing values (including our target feature), we will have to fix them. Starting from our target feature, since it is a categorical feature and our dataset is huge, we're simply going to drop missing values in our target variable.

```
Mean of Latitude:  40.67402280233771
Median of Latitude:  40.71886135

Since both the values are approx. same, it means there are no outliers in latitude & longitude.
```

Fig. Finding outliers

We also checked the latitude and longitude to confirm there are no outliers in the dataset, meaning that the location is indeed in NYC,

After that, we're going to compute missing values for the test of our features. Since our data contains no outliers at all, we filled all the missing values using the mean strategy.

```
#Checking again for missing values
df.isnull().sum()
```

```
CRASH DATE                        0
CRASH TIME                        0
BOROUGH                           0
LATITUDE                          0
LONGITUDE                         0
NUMBER OF PERSONS INJURED         0
NUMBER OF PERSONS KILLED          0
NUMBER OF PEDESTRIANS INJURED     0
NUMBER OF PEDESTRIANS KILLED      0
NUMBER OF CYCLIST INJURED         0
NUMBER OF CYCLIST KILLED          0
NUMBER OF MOTORIST INJURED        0
NUMBER OF MOTORIST KILLED         0
```

Fig. Checking missing values

We confirmed there are no missing values after fixing it. Finally, we fixed the datatype of each column.

Exploratory Data Analysis

After cleaning our data, we made an EDA, which analyzes the cleaned dataset, and discovers the patterns, relationships within our dataset. We created data visualizations to help us build our model in predicting which borough did this accident happen is based on given information.

First, we made a heatmap to show the correlation of numerical features in our data.

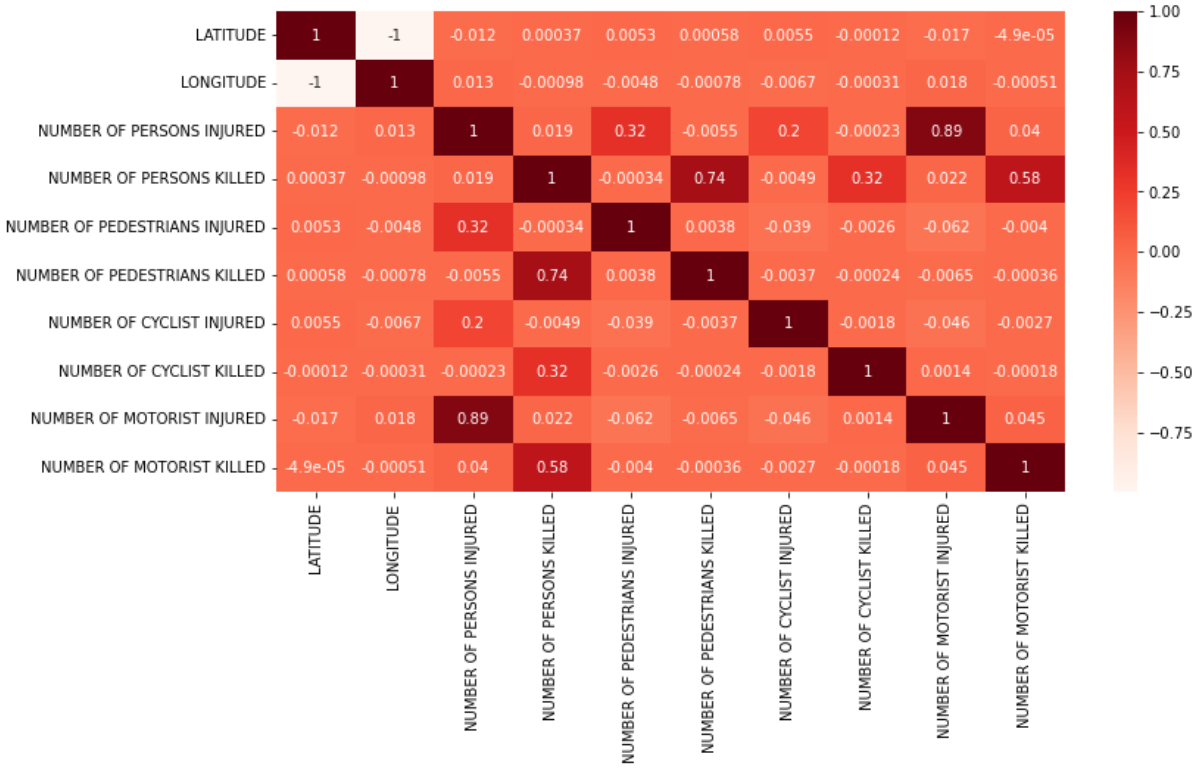

Fig. Heatmap on features correlation

- The Number closer to zero indicates a lower correlation (precise 0 implying no significant relation).
- The Number around +1 to one indicates a more positive correlation.
- The Number around -1 implies a more negative correlation.

A heatmap is a graphical representation where individual values of a matrix are represented as colors. A heatmap is especially useful in visualizing the concentration of

values between two dimensions of a matrix. This helps in finding patterns and gives a perspective of depth.

We found out that the number of motorists injured has a strong correlation with the total number of persons injured with a 0.89, meaning that in an accident, motorists are mostly the ones who are injured if it's involved, instead of others (pedestrians, cyclists).

We also found out that the number of people killed has a strong correlation with the number of pedestrians killed, meaning that pedestrians are the ones that have a higher death rate than drivers in an accident.

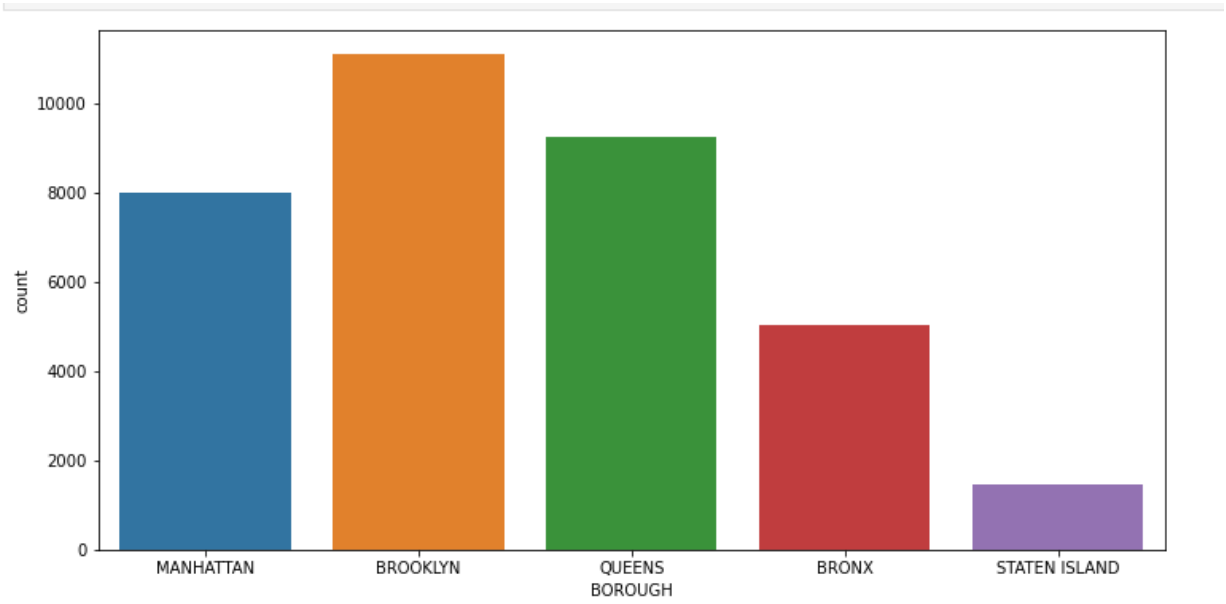

Fig. The number of vehicular collisions in boroughs

Brooklyn and Queens had the greatest number of collisions, and State Island had the least.

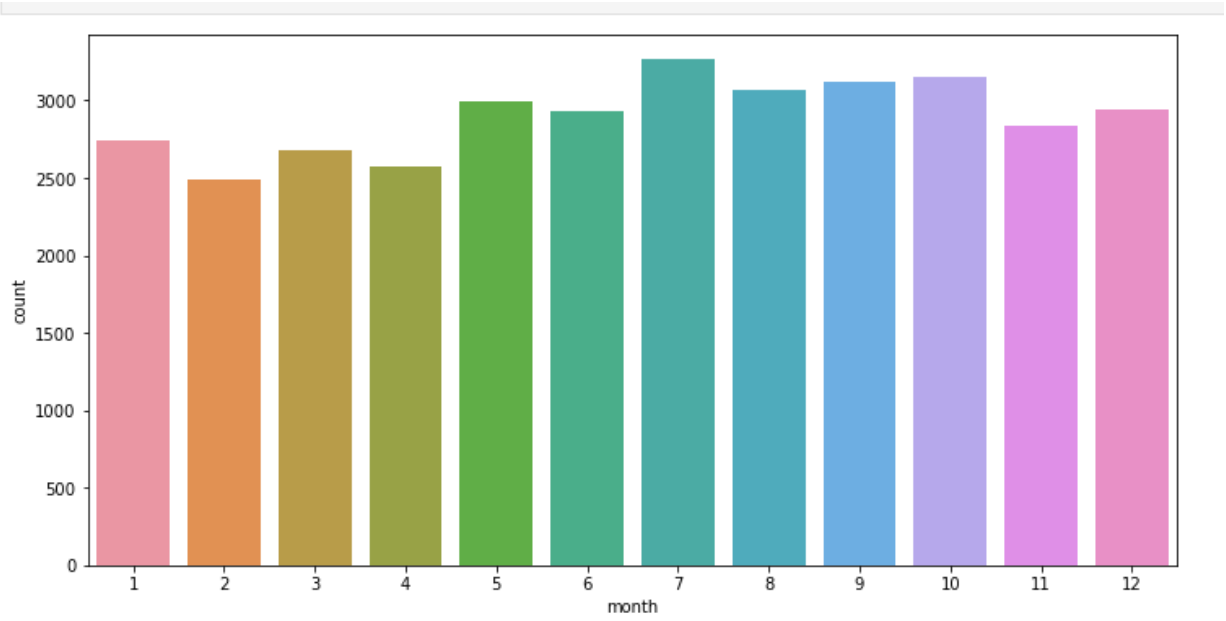

Fig. The number of collisions by a month bar chart

As we can see here, the greatest number of collisions occurred in July overall which is numbered as 7 in the bar chart.

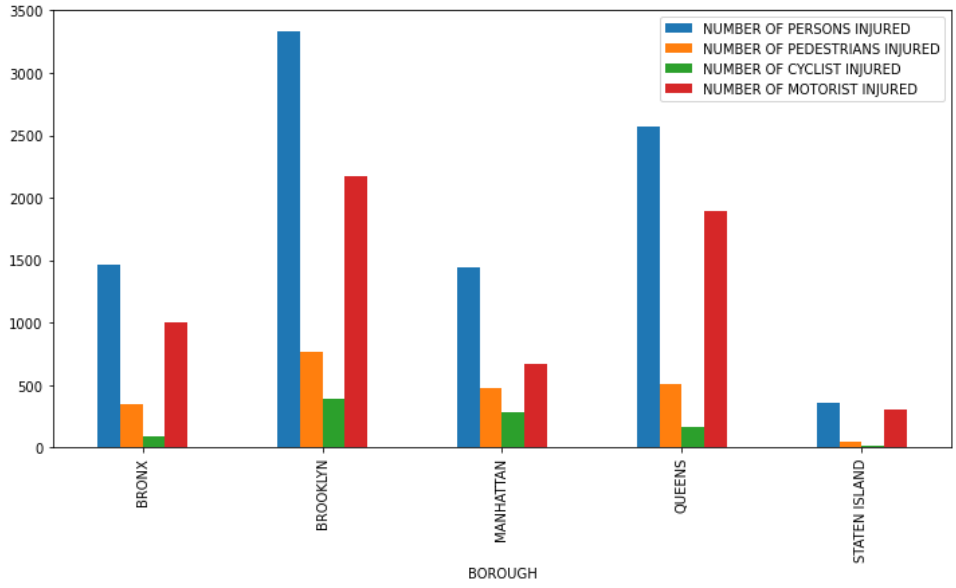

Fig. Boroughs with number of injuries

The above visualizations tell us that Brooklyn and Queens have the highest total number of injuries, in contrast to the lowest, Staten Island.

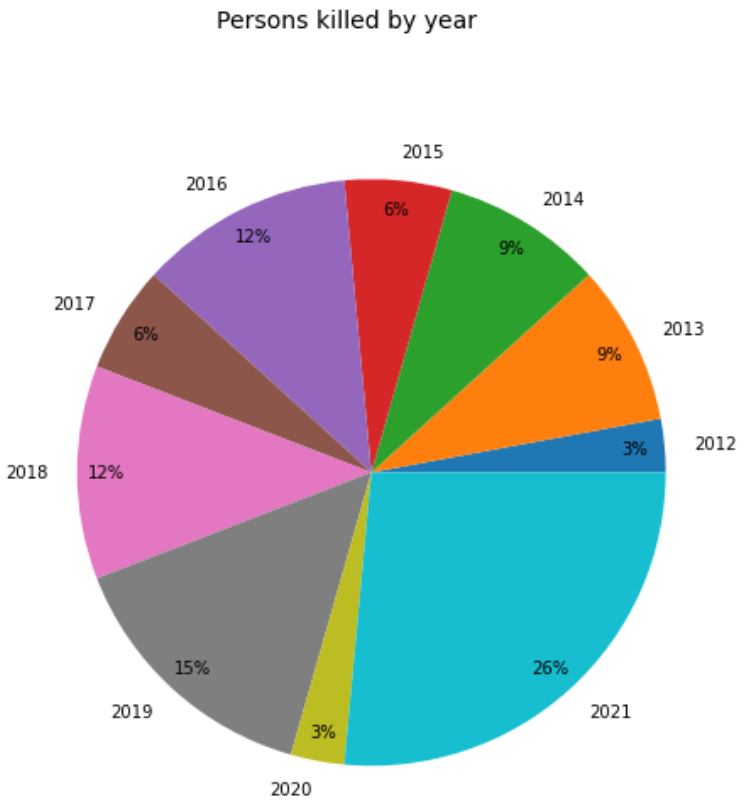

Fig. Pie chart for people killed by year

From the above pie chart, we can see that the highest percentage of deaths occurred in 2021. Surprisingly, it's this year, but the year hasn't ended yet.

Models

We build various models to predict in which borough did the accident happen. We use Logistic Regression, Neural Network, and K-Nearest Neighbors. As one can infer from the models we are using, all our models are classification models. We are using classification models because our question falls into "Which Category?". It is a supervised classification problem.

Evaluation

We evaluate our models using accuracy score, confusion matrix, and classification reports.

Logistic Regression

Accuracy Score:

```
#Making a class predictions for the training set
y_pred_train = logreg.predict(X_train)
print(accuracy_score(y_train, y_pred_train))
```

```
0.5578996549741231
```

```
#Making a class predictions for the testing set
y_pred_test = logreg.predict(X_test)

#Checking our model accuracy
print(accuracy_score(y_test, y_pred_test))
```

```
0.5605232892466935
```

Fig: Accuracy Score on Logistic Regression

From the pictures above, we can say that our Logistic Regression Model will predict the right borough 56% of the time. We can also infer from the pictures that since both the training and test accuracy are close and not low, greater than 50%, the model is neither overfitting nor underfitting.

Confusion Matrix:

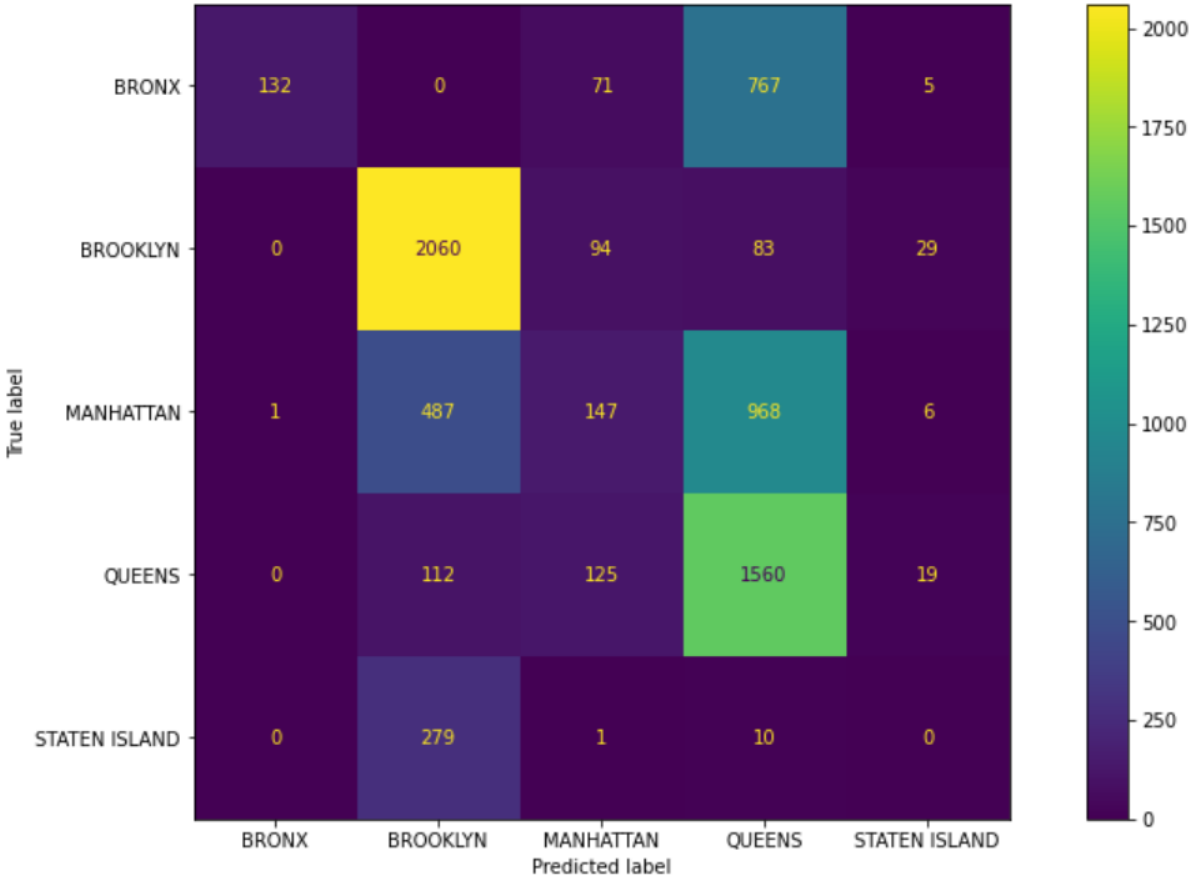

Fig: Confusion Matrix on evaluation our model

We use a Confusion Matrix to evaluate our Logistic Regression Model. It's a performance measurement for machine learning classification problems where output can be two or more classes. It is a table with different combinations of predicted and actual values, measuring a confusion matrix provides better insight into our model.

Explanation

On the first column on the confusion matrix, our model predicts 133 accidents are from the Bronx. However, it correctly predicts that 132 accidents are from the Bronx and falsely predicts that 1 accident is from Manhattan. For Brooklyn, it predicts that 2938 accidents are from Brooklyn. However, it accurately predicts that 2060 accidents are from Brooklyn and inaccurately predicts the rest of the accidents.

Classification Report

	precision	recall	f1-score	support
BRONX	0.98	0.30	0.46	1245
BROOKLYN	0.68	0.89	0.77	2682
MANHATTAN	0.30	0.09	0.14	2020
QUEENS	0.49	0.87	0.63	2349
STATEN ISLAND	0.00	0.00	0.00	368
accuracy			0.58	8664
macro avg	0.49	0.43	0.40	8664
weighted avg	0.55	0.58	0.51	8664

Fig: Classification report of our model

Precision

Precision shows us that from all the classes we have predicted as positive, how many are actually positive. Precision should be as high as possible. Here, Bronx has an extremely high precision value while Staten Island has an extremely low precision value.

F1- score

F-score helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more. Here, Brooklyn has a very high F1-value value while Staten Island has an incredibly low F1-score value.

Recall

Recall is how many of the true positives were recalled (found), i.e. how many of the correct hits were also found. Here, Brooklyn has a very high recall value while Staten Island has an incredibly low recall value.

Support

Support is the number of samples in data for that variable, the performance measures are better when the support is higher. Here, for Staten Island, the support is very low, so the performance measures are poor.

Confidence level:

We don't feel confident about its performance. It has a low accuracy score, 56%. Our Logistic Regression model did better than our Neural Network model but did worse than K-Nearest Neighbors.

Neural Network

Accuracy Score:

```
#Making a class predictions for the training set
y_pred_train = nn.predict(X_train)
print(accuracy_score(y_train, y_pred_train))
```

```
0.3171362852213916
```

```
#Making a class predictions for the testing set
y_pred_test = nn.predict(X_test)

#Checking our model accuracy
print(accuracy_score(y_test, y_pred_test))
```

```
0.3257619321449109
```

Fig: Accuracy Score on Neural Network

From the pictures above, we can say that our Neural Network Model will predict the right borough 32% of the time. We can also infer from the pictures that since the training accuracy is very low, less than 50%, the model is underfitting.

Confusion Matrix:

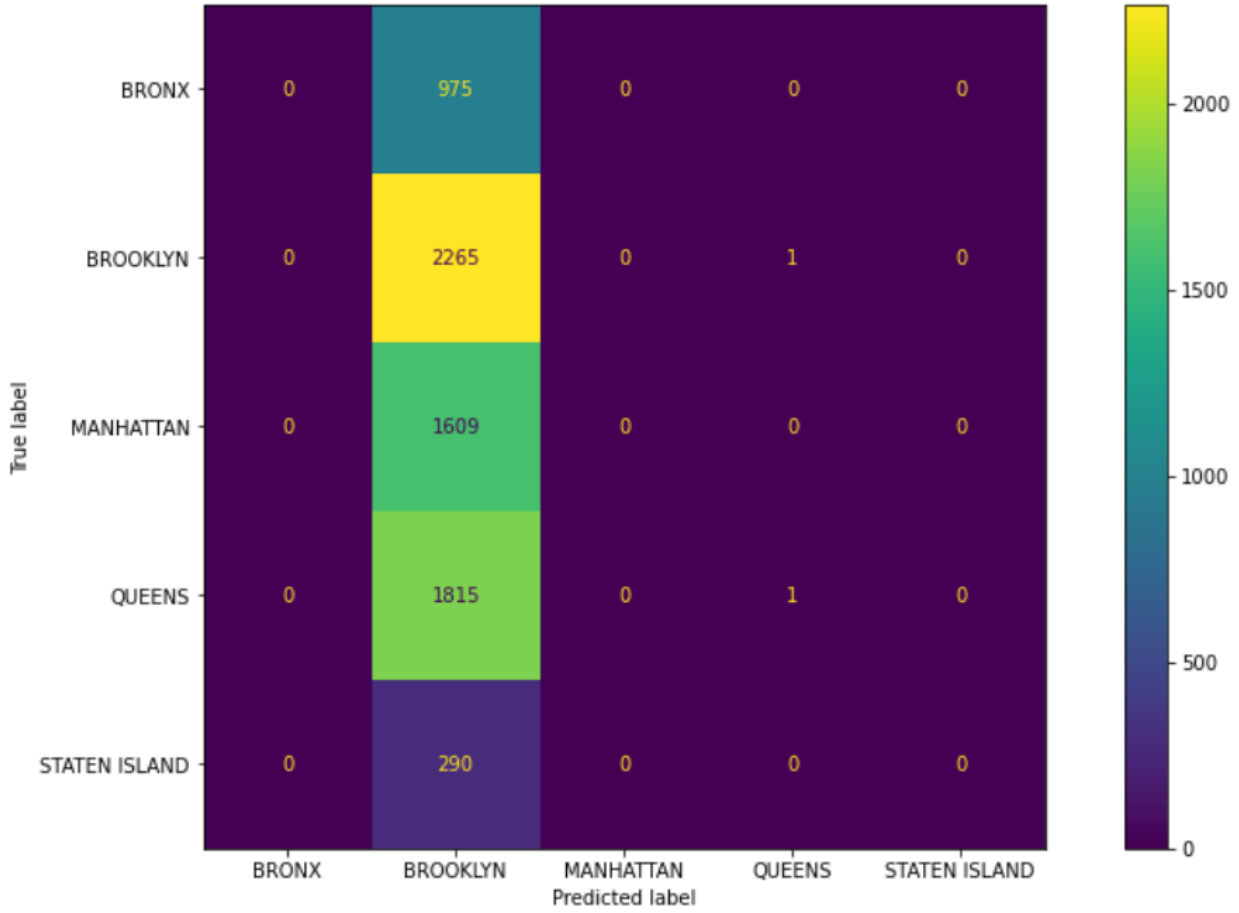

Fig: Confusion Matrix on evaluation our model

*** Note**: Definition of Confusion Matrix is given in the Confusion Matrix section of the Logistic Regression Model.

Explanation

On the second column on the confusion matrix, our model predicts 6954 accidents are from Brooklyn. However, it correctly predicts that 2265 accidents are from Brooklyn and falsely predicts that 975 accidents are from Bronx, 1609 accidents are from Manhattan, and 290 accidents are from Staten Island.

Classification Report

	precision	recall	f1-score	support
BRONX	0.00	0.00	0.00	975
BROOKLYN	0.33	1.00	0.49	2266
MANHATTAN	0.00	0.00	0.00	1609
QUEENS	0.50	0.00	0.00	1816
STATEN ISLAND	0.00	0.00	0.00	290
accuracy			0.33	6956
macro avg	0.17	0.20	0.10	6956
weighted avg	0.24	0.33	0.16	6956

Fig: Classification report of our model

*** Note**: More explicit definitions of Precision, Recall, F1-score, and Support are given in the Classification Report section of the Logistic Regression Model.

Precision, F1- score, and Recall:

Precision, Recall, and F1-score for each y-variable value are different performance measures for classification. Here, for Brooklyn, the Precision value is 0.33, the Recall value is 1 and the F1-score is 0.49.

Support

Support is the number of samples in data for that variable, the performance measures are better when the support is higher. Here, for Staten Island, the support is very low, so the performance measures are poor.

Confidence level:

We don't feel confident about its performance. It has an extremely low accuracy score, 32%. Our Neural Network model did worse than both of our other models, Logistic Regression and K-Nearest Neighbors.

K-Nearest Neighbors

Accuracy Score:

```
#Making a class predictions for the training set
y_pred_train = k_neighbors.predict(X_train)
print(accuracy_score(y_train, y_pred_train))
```

```
0.9777530189764232
```

```
#Making a class predictions for the testing set
y_pred_test = k_neighbors.predict(X_test)

#Checking our model accuracy
print(accuracy_score(y_test, y_pred_test))
```

```
0.9751293847038528
```

Fig: Accuracy Score on K-Nearest Neighbors

From the pictures above, we can say that our K-Nearest Neighbors Model will predict the right borough 98% of the time. We can also infer from the pictures that since both the training and test accuracy are close and not low, the model is neither overfitting nor underfitting.

Confusion Matrix:

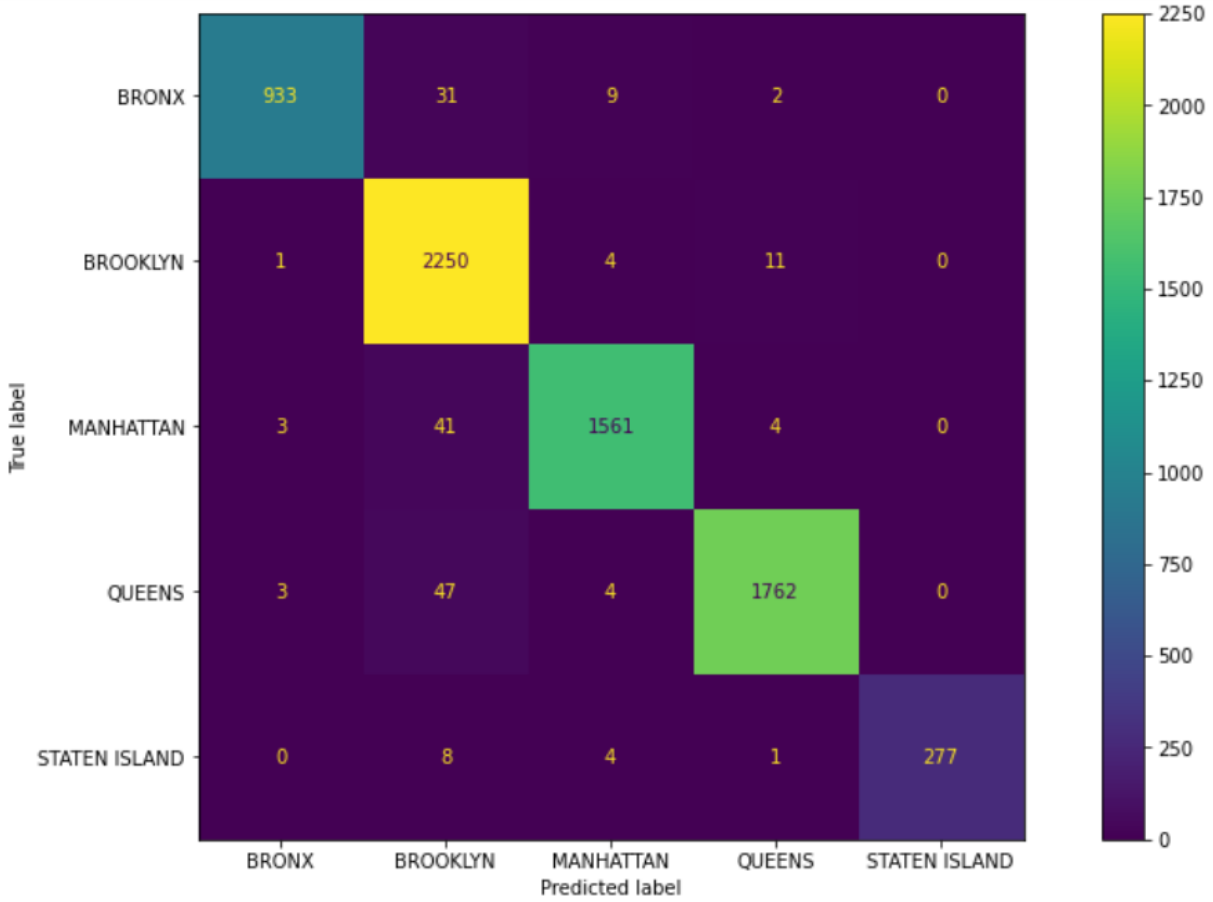

Fig: Confusion Matrix on evaluation our model

*** Note**: Definition of Confusion Matrix is given in the Confusion Matrix section of the Logistic Regression Model.

Explanation

On the first column on the confusion matrix, our model predicts 940 accidents are from the Bronx. However, it correctly predicts that 933 accidents are from the Bronx and falsely predicts that 1 accident is from Brooklyn, 3 accidents are from Manhattan, and.3 accidents from Queens. For Brooklyn, it predicts that 2377 accidents are from Brooklyn. However, it accurately predicts that 2250 accidents are from Brooklyn and inaccurately predicts the rest of the accidents.

Classification Report

	precision	recall	f1-score	support
BRONX	0.99	0.96	0.97	975
BROOKLYN	0.95	0.99	0.97	2266
MANHATTAN	0.99	0.97	0.98	1609
QUEENS	0.99	0.97	0.98	1816
STATEN ISLAND	1.00	0.96	0.98	290
accuracy			0.98	6956
macro avg	0.98	0.97	0.98	6956
weighted avg	0.98	0.98	0.98	6956

Fig: Classification report of our model

*** Note**: More explicit definitions of Precision, Recall, F1-score, and Support are given in the Classification Report section of the Logistic Regression Model.

Precision, F1- score, and Recall:

Precision, Recall, and F1-score for each y-variable value are different performance measures for classification. Here, for Brooklyn, the Precision value is 0.95, the Recall value is 0.99 and the F1-score is 0.97.

Support

Support is the number of samples in data for that variable, the performance measures are better when the support is higher. Here, for Staten Island, the support is very low, so the performance measures are poor.

Confidence level:

We feel greatly confident about its performance. It has an extremely high accuracy score, 98%. Our K-Nearest Neighbors Model did the best compared to both of our other models, Logistic Regression and Neural Network.

Future Work

For future work, we hope to improve our model performance by tuning our models and tuning hyper-parameters. We will look at other supervised classification models to see which model would give the best performance. We will also look at other datasets, perhaps weather and demographics of borough datasets. We will analyze if the weather and demographics of the borough play a crucial role in the number of vehicular collisions. In conclusion, the project needs more work to get the complete picture, but we are extremely happy with what we have accomplished.