Final Report for Capstone 1

NYC Crime Data 2019

Data Cleaning and Wrangling:
https://github.com/elizabeamedalla/Capstone-1/blob/master/Data%20Cleaning.ipynb
Data Exploration and Analysis:
https://github.com/elizabeamedalla/Capstone-1/blob/master/Data%20Analysis.ipynb
Statistical Inference:
https://github.com/elizabeamedalla/Capstone-1/blob/master/Statistical%20Inference.ipynb
In-depth Analysis (Machine Learning)
https://docs.google.com/document/d/1kzavchF91PqLY1fOkf86K_H3pNTocWX_ItQsBh6TqBE/edit#
Powerpoint Presentation (rough draft: need further review)
https://docs.google.com/presentation/d/1phNOvOprBJ-kH0ToQARHMCALQBZR0owFK8lWkyAQRyk/edit#slide=id.p

Problem Statement

NYPD wants to learn more about the crime statistics in NYC, so they hire Sherlock Holmes and Dr. Watson to investigate. They want to know what type of crimes are most prominent in certain boroughs and districts, the frequency of the crimes, and the demographics of individuals that are vulnerable to be the victim of the crime. NYPD wants to be data-driven in establishing appropriate measures on how to monitor and lessen prominent crimes in certain areas and to predict what crimes will likely to happen in certain places.

Sherlock Holmes's goal is to be able to predict what crime is most likely to happen in an area and put police on patrol in those areas. With the help of the data consultant, by using machine learning, they will be able to predict the nature of crime and establish cautionary measures to lessen the crime reports. This analysis is useful for NYPD hoping to put a stop to more crimes in NYC. It's also useful for citizens to be more aware of the criminal activity happening in their local areas.

Data Cleaning and Data Wrangling

The dataset used came from the NYC OpenData project, which provided a dataset of reported crimes from 1990-2019. What I later found is that the majority of the data is from the year 2019. This dataset contains approximately 500,000 rows of data, where each row is one instance of a reported crime. The dataset contains information about each crime such as the Date, Time,

Borough, District, Latitude/Longitude, Victim and Suspect Age-Range/Sex/Race, Crime Type and Severity. The data is almost completely categorical data, and there is very little numerical data contained within this dataset. This dataset was downloaded from [NYPD Complaint Data Current (Year To Date)] into a .csv file.

After downloading the dataset into a CSV file, I began the process of cleaning the data. Data cleaning is the process of detecting and correcting corrupt or inaccurate records from a dataset. In a new iPython notebook, I imported the data as well as the pandas library to begin cleaning the data. My process for cleaning the data involved filtering out unnecessary columns, renaming columns, filling in NaN values, converting dates to Datetime format, and checking that there are no duplicates and each row is unique.

First I selected the columns needed for the analysis. There were 35 columns originally. I cut it down to 18 columns. These columns contain data such as the NYC Borough/District, Date/Time and Type of crime reported, and Age-Group/Sex/Race of both the suspect and the victim.

```
]: df = df[['CMPLNT_NUM', 'ADDR_PCT_CD', 'BORO_NM', 'CMPLNT_FR_DT', 'CMPLNT_FR_TM', 'LAW_CAT_CD', 'LO
C_OF_OCCUR_DESC', 'OFNS_DESC', 'SUSP_AGE_GROUP', 'SUSP_RACE', 'SUSP_SEX', 'VIC_AGE_GROUP', 'VIC_RA
CE', 'VIC_SEX', 'X_COORD_CD', 'Y_COORD_CD', 'Latitude', 'Longitude', 'Lat_Lon']]
df.info()
```

Second I renamed the columns. This was so that column names could be more human-readable. **Third** I filled in NaN values. Some of the NaN values I filled in included Borough, (todo:)

Rename columns and fill in NaN values.

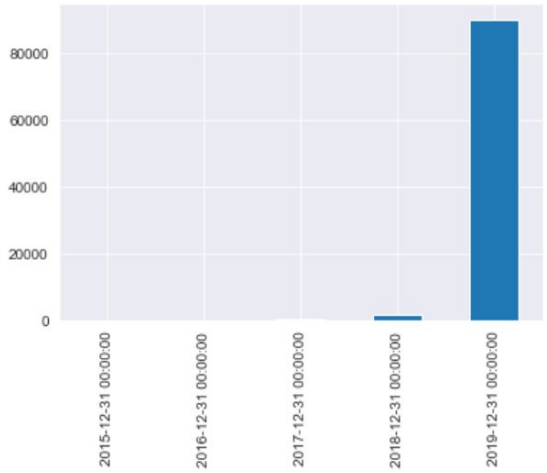
```
3]: # Rename columns.
df = df.rename(columns={"BORO_NM": "BOROUGH", "CMPLNT_FR_DT": "DATE", "CMPLNT_FR_TM": "TIME", "LAW
_CAT_CD": "OFFENSE LEVEL", "ADDR_PCT_CD": "DISTRICT", "OFNS_DESC": "OFFENSE_NAME"})

# Fill in NaN values on BORO_NM
df["BOROUGH"].fillna("UNKNOWN", inplace = True)
df.head()
```

Fourth I converted all the date columns to datetime format. I also added new columns for year, month and date. It could come in handy in later exploration. **Fifth I checked for duplicate row entries** in the “CMPLNT_NUM” column. I chose this column because I knew that each value in this column should be a unique ID.

After doing basic cleaning on the data, I plotted the total crimes each year in Queens to peek at the data and see if there are any abnormalities. **I found out that the majority of crimes reported in the dataset are in 2019.** This led me to focus only on the data available in 2019. **I filtered my data to only include data from 2019.**

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d19f310>
```



After filtering my data to 2019 only, I aimed to group similar offense names. The initial number of offense names were 64 total, but many specific offenses could be grouped into similar categories, such as Harassment, Assault, Fraud, Theft. I ended up grouping all 64 offense types into 20 categories, then renaming those offenses as such.

After cleaning and wrangling the data, I had a dataframe of offenses reported in 2019 for all boroughs in NYC, with NaN values cleaned and offense types grouped together for easier analysis. Finally the data was saved to a CSV file for the next notebook.

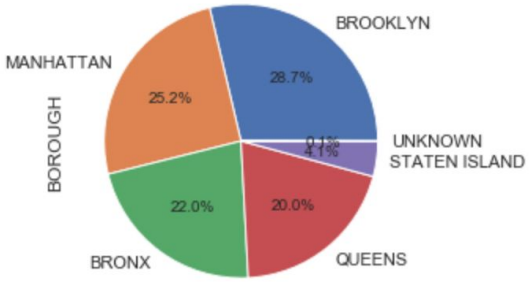
Exploratory Data Analysis

The initial approach that I did was to see the demographics of the crime by borough and districts. the nature of crimes that happen in NYC in each borough. Various crimes could be confined to different boroughs and districts.

I performed some data exploration and visualization with the hope of uncovering some interesting insights along the way.

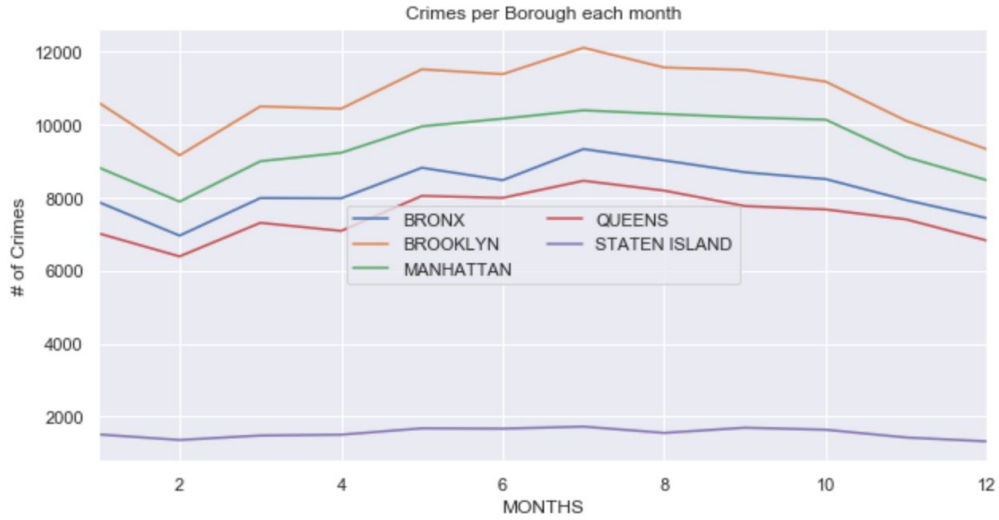
Display amount of Crimes per borough in the data set

After displaying the number of crimes per borough all 2019, I found that Brooklyn had the highest percentage of reported crimes at 28.7% of total crimes reported in 2019. However all boroughs except for Staten Island had a percentage above 20%. Staten Island had a percentage of only 4.1%.



Plot frequency of crimes for each Borough in 2019.

I wanted to see the trend of crimes reported every month in each borough. Is there a peak season for crimes? When is the crime reported lowest? To see the bigger picture, I made a multi-line time-series plot for the crime reports in all the boroughs each month.

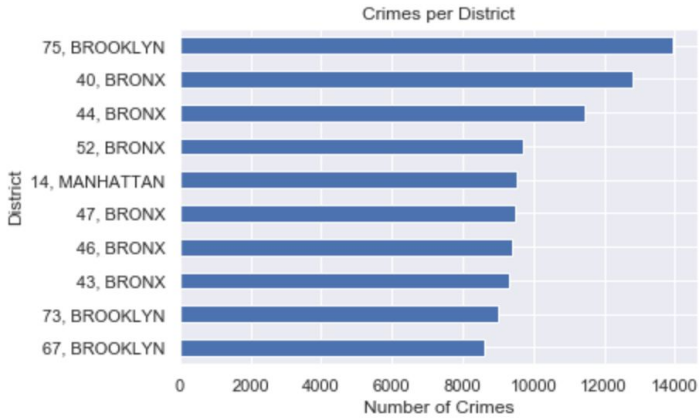


After plotting the crimes each month per borough, I noticed that Brooklyn had indeed consistently had the most reported crimes each month compared to the other boroughs, which clarifies how Brooklyn had the largest percentage of crimes overall.

I also noticed a consistent trend for all boroughs, where crime occurrences peaked in the summer and dipped significantly in the winter. The highest peak was in June and the lowest dip was in February.

Visualizing the Top 10 Districts with most crimes.¶

After analysing crime data about each borough, I wanted to look more granularly at each District to see the top 10 districts that had the highest reported crimes.



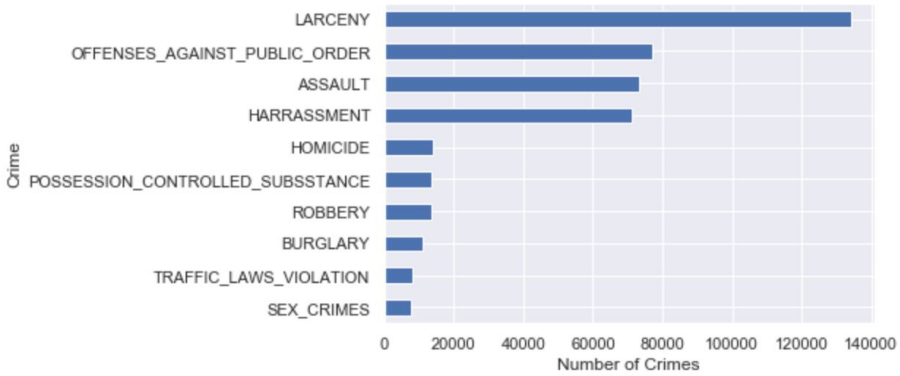
After plotting the top 10 districts with Crimes reported, I found out that District 75 in Brooklyn had the highest number of reported crimes at around 14000 in 2019.

I found that the next 6 of the next 7 highest districts are in the Bronx, and one is in Manhattan. District 40 in the Bronx seems to have the highest crimes reported for that borough.

An interesting analysis I found is that although the borough of Brooklyn has the highest reported crimes overall, only 3 out of the top ten districts for crimes reported in 2019 are in Brooklyn, with one of them being rank #1 and the other two Brooklyn districts at rank #9 and #10.

Plot the Top 10 Crimes reported in 2019

After analyzing crimes by location, I wanted to see the nature of the top 10 most frequent crimes reported overall in 2019.



According to findings, Larceny was the most reported crime in all of NYC in 2019 by a lot, with over 130,000 reports. Offenses such as Assault and Harassment came next, with a huge difference in numbers, at around just under 80,000 reports.

I found that the large majority of crimes reported included Larceny, Assault and Harassment, and that there is a mix of violent and non-violent crimes reported.

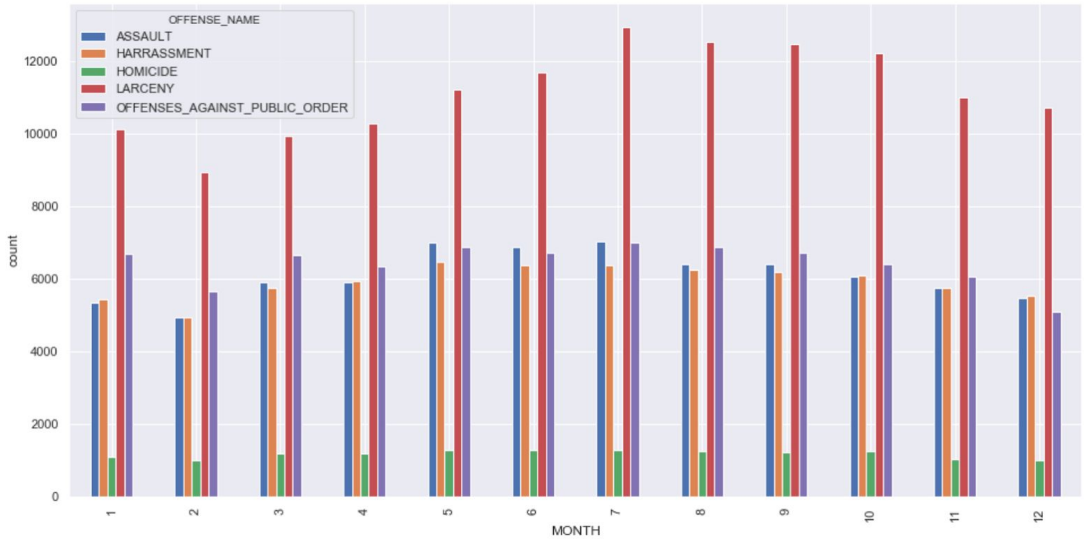
Comparing Felony vs Misdemeanor vs Violation Frequency

I wanted to see the difference in offense severities each month in 2019. Crimes could be classified as either Misdemeanor, Violation or Felony.

The findings show that Misdemeanors are the most common. This aligns with the data that crimes like Larceny and Harassment were the most common - most of which are usually classified as Misdemeanors.

Make a Stacked Bar Graph of Top 5 Crimes Each month in 2019

I would like to see the distribution of crimes reported each month but due to the large variety of crime types, it would be a lot of data to plot. We can instead plot 5 most frequent crimes reported each month.



Plotting the top 5 crimes reported each month provided interesting findings. Once again it is shown that Larceny consistently had the largest amount of reports each month and Homicide consistently had the fifth highest amount of reports each month, both by a significant distance from Assault, Harassment, and Offenses Against the Public Order. Each month however, the frequency of crimes for Assault, Harassment and Offenses against the Public order differed but stayed within the same range, with each of the three trading ranks each month.

The gap between the 4th highest and 5th highest occurring crime each month is significant, and the gap between the 1st and second most common crime each month is significant.

Data Analysis Conclusions:

After Initial Data analysis, I came across the following conclusions:

- Brooklyn has the highest crime reported with 29% of total reports. That makes sense since Brooklyn is the most populated borough according to (https://guides.lib.jjay.cuny.edu/c.php?g=288385&p=1922495).
- The trends of the crime reports in all boroughs are consistent throughout the year. All boroughs have the same months of high and low reports. No overlapping of data trends had occurred.
- Brooklyn as being the one with the highest reports is also consistent to be the highest portion in overall's monthly total in 2019.
- Interestingly, District 75 in Brooklyn had the most crimes reported followed by District 40, 44 and 52 which are in the Bronx. Manhattan has the second-highest reports overall.
- All districts in NYC have crimes reported.
- Theft and Harassment were racing with each other throughout the entire month
- The top 10 most crimes reported include larceny, harassment, and assault.
- The most frequent offense level of all crimes reported is a misdemeanor and the general trend of felonies and violations follows it throughout the whole year.

- During hot weather, reports are more frequent compared to the cold season. Reports drop drastically as holidays approach.

Statistical Inferencing

The Chi-Square Test of Independence determines whether there is an association between categorical variables (i.e., whether the variables are independent or related). It is a nonparametric test.

The test statistic for the Chi-Square Test of Independence is denoted X2, and is computed as:

$$\chi^2 = \sum_{i=1}^{R} \sum_{j=1}^{C} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

This test utilizes a contingency table to analyze the data. A contingency table (also known as a *cross-tabulation*, *crosstab*, or *two-way table*) is an arrangement in which data is classified according to two categorical variables. The categories for one variable appear in the rows, and the categories for the other variable appear in columns. Each variable must have two or more categories. Each cell reflects the total count of cases for a specific pair of categories. (Source: https://libguides.library.kent.edu/SPSS/ChiSquare)

Question 1: NYC Census Population vs. NYC Crime Victim Sample Data

We can use the **Chi-square goodness-of-fit test**. It tests whether the distribution of sample categorical data matches an expected distribution. The **null hypothesis** for this goodness-of-fit test is that the population mean of the 2019 NYC Census data by race **is the same** as the sample mean of NYC 2019 crime victims by race. The **alternative hypothesis** is that the population mean of the 2019 NYC Census data by race **is different** from the sample mean of NYC 2019 crime victims by race.

The chi-square formula is:

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

O = the frequencies observed

E = the frequencies expected

$\sum$ = the 'sum of'

In this formula, observed is the actual observed count for each category from your dataset, and expected is the expected count based on the distribution of the population for the corresponding category.

For my population data, I attained the data for the NYC Census by race in 2019 from an outside source. **I used the count of crime victims by race as my sample data**. Then I created a crosstab for both datasets.

I used the crosstab table for my sample as my **observed** dataframe, then calculated my **expected dataframe** as the product of the ratios of the population data and the size of the sample data. Then I calculated the chi-squared stat using these variables. I calculated it by hand, and I also used the function *scipy.stats.chisquare()* to calculate the statistic.

```
: # You can carry out a chi-squared goodness-of-fit test automatically using the scipy function
  stats.chisquare(f_obs= observed,   # Array of observed counts
                  f_exp= expected)   # Array of expected counts
```

```
: Power_divergenceResult(statistic=array([57318.7342417]), pvalue=array([0.]))
```

I found that my chi-square statistic is 57318.7342417 and my p-value is 0. I have tried to round the number to the nearest ten-thousandth but still getting 0.

From this, we conclude since our chi-squared statistic exceeds the critical value and the p-value having zero, we'd reject the null hypothesis that the two distributions are the same. Information regarding p-value for being 0.000 can be verified here.
https://www.statology.org/here-is-how-to-interpret-a-p-value-of-0-000/ .

Question 2: Victim Race Population Independency

For this question, we can use the **Chi-square Independence Test.** Independence is a key concept in probability that describes a situation where knowing the value of one variable tells you nothing about the value of another. Our **null hypothesis** for this test is that the victim race population is not related to the crimes reported in NYC. Our **alternative hypothesis** is that The victim race population is related to the crimes reported in NYC.

The chi-squared independence test uses the same chi-squared formula, and the main difference is how the observed and expected values are derived. For the independence test, we are using random sample distributions for our observed and expected data.

To calculate our observed dataset, we will first create a probability distribution of our crime victims and crime types based on our dataset. The probability distribution for both victim race and crime types will follow the distribution of each in the actual dataset. Next we will create a crosstab of the random Victim Race and Crime Type data.

victim_race	AMERICAN INDIAN/ALASKAN NATIVE	ASIAN / PACIFIC ISLANDER	BLACK	BLACK HISPANIC	WHITE	WHITE HISPANIC	All
offense							
ARSON	1	0	1	0	0	1	3
ASSAULT	18	229	757	106	478	530	2118
BURGLARY	0	17	77	16	37	59	206
DRIVING_UNDER_INFLUENCE	0	1	4	0	1	2	8
FORGERY	0	1	2	1	2	4	10
FRAUD	1	13	34	9	33	41	131
HARRASSMENT	15	197	755	122	444	470	2003
HOMICIDE	2	35	124	17	68	91	337
KIDNAPPING	0	1	3	0	1	0	5
LARCENY	17	276	988	158	624	641	2704
MURDER	1	0	2	0	1	3	7
OFFENSES_AGAINST_PUBLIC_ORDER	12	155	560	105	348	388	1568
POSSESSION_CONTROLLED_SUBSSTANCE	0	1	1	0	1	2	5
POSSESSION_WEAPON	0	1	10	0	9	5	25
ROBBERY	3	50	127	26	90	80	376
SEX_CRIMES	2	28	72	12	53	58	225
SOCIAL_RELATED_CRIMES	0	0	5	3	6	7	21
THEFT	1	6	13	1	3	12	36
TRAFFIC_LAWS_VIOLATION	0	24	84	7	44	50	209
UNCLASSIFIED	0	1	1	0	1	0	3
All	73	1036	3620	583	2244	2444	10000

Next, we have to calculate the expected dataframe based off the observed 2-way table. To get the expected count for a cell, multiply the row total for that cell by the column total for that cell and then divide by the total number of observations. We can quickly get the expected counts for all cells in the table by taking the row totals and column totals of the table, performing an outer product on them with the np.outer() function and dividing by the number of observations.

	BLACK	WHITE HISPANIC	WHITE	ASIAN / PACIFIC ISLANDER	BLACK HISPANIC	AMERICAN INDIAN/ALASKAN NATIVE	All
LARCENY	0.0219	0.3108	1.086	0.1749	0.6732	0.7332	3.0
ASSAULT	15.4614	219.4248	766.716	123.4794	475.2792	517.6392	2118.0
HARRASSMENT	1.5038	21.3416	74.572	12.0098	46.2264	50.3464	206.0
OFFENSES_AGAINST_PUBLIC_ORDER	0.0584	0.8288	2.896	0.4664	1.7952	1.9552	8.0
ROBBERY	0.0730	1.0360	3.620	0.5830	2.2440	2.4440	10.0
HOMICIDE	0.9563	13.5716	47.422	7.6373	29.3964	32.0164	131.0
SEX_CRIMES	14.6219	207.5108	725.086	116.7749	449.4732	489.5332	2003.0
TRAFFIC_LAWS_VIOLATION	2.4601	34.9132	121.994	19.6471	75.6228	82.3628	337.0
BURGLARY	0.0365	0.5180	1.810	0.2915	1.1220	1.2220	5.0
FRAUD	19.7392	280.1344	978.848	157.6432	606.7776	660.8576	2704.0
THEFT	0.0511	0.7252	2.534	0.4081	1.5708	1.7108	7.0
SOCIAL_RELATED_CRIMES	11.4464	162.4448	567.616	91.4144	351.8592	383.2192	1568.0
POSSESSION_WEAPON	0.0365	0.5180	1.810	0.2915	1.1220	1.2220	5.0
FORGERY	0.1825	2.5900	9.050	1.4575	5.6100	6.1100	25.0
DRIVING_UNDER_INFLUENCE	2.7448	38.9536	136.112	21.9208	84.3744	91.8944	376.0
MURDER	1.6425	23.3100	81.450	13.1175	50.4900	54.9900	225.0
POSSESSION_CONTROLLED_SUBSSTANCE	0.1533	2.1756	7.602	1.2243	4.7124	5.1324	21.0
ARSON	0.2628	3.7296	13.032	2.0988	8.0784	8.7984	36.0
KIDNAPPING	1.5257	21.6524	75.658	12.1847	46.8996	51.0796	209.0
UNCLASSIFIED	0.0219	0.3108	1.086	0.1749	0.6732	0.7332	3.0
All	73.0000	1036.0000	3620.000	583.0000	2244.0000	2444.0000	10000.0

Now that we have the expected dataframe, we can calculate the chi-squared statistic, the critical value, and the p-value.

```
: # Compute a critical value
crit = stats.chi2.ppf(q = 0.95, # Find the critical value for 95% confidence*
                      df = 24)   # *


# Compute a p-value
p_value = 1 - stats.chi2.cdf(x=chi_squared_stat,  # Find the p-value
                             df=8)

print("Critical value:", crit)
print("P value:", np.round(p_value, 3))
```

```
Critical value: 36.41502850180731
P value: 0.0
```

From this, we can see that **the critical value is 36.41502850180731** and **the p-value is 0.0.** Since the p-value is 0, and having a way higher chi-squared statistics, we can reject the null hypothesis.

As with the goodness-of-fit test, we can use scipy to conduct a test of independence quickly. We use stats.chi2_contingency() function to conduct a test of independence automatically given a frequency table of observed counts:

```
stats.chi2_contingency(observed = observed)
```

```
(31.157932571320597,
 0.0001316936897183417,
 8,
 array([[3.42960289e-02, 4.47653430e-01, 1.51805054e+00],
        [1.72166065e+01, 2.24722022e+02, 7.62061372e+02],
        [1.61191336e+00, 2.10397112e+01, 7.13483755e+01],
        [8.57400722e-02, 1.11913357e+00, 3.79512635e+00],
        [5.14440433e-02, 6.71480144e-01, 2.27707581e+00]]))
```

The output shows the chi-square statistic, the p-value and the degrees of freedom followed by the expected counts. As expected, given the high p-value, the test result **does detect a significant relationship between the variables**.

In-depth Analysis (Machine Learning)

If a model accuracy is higher than 50%, it means that using ML to predict crime type is better than random chance.
Jupyter notebook:
https://github.com/elizabeamedalla/Capstone-1/blob/master/In-depth%20Analysis%20(Machine%20Learning).ipynb

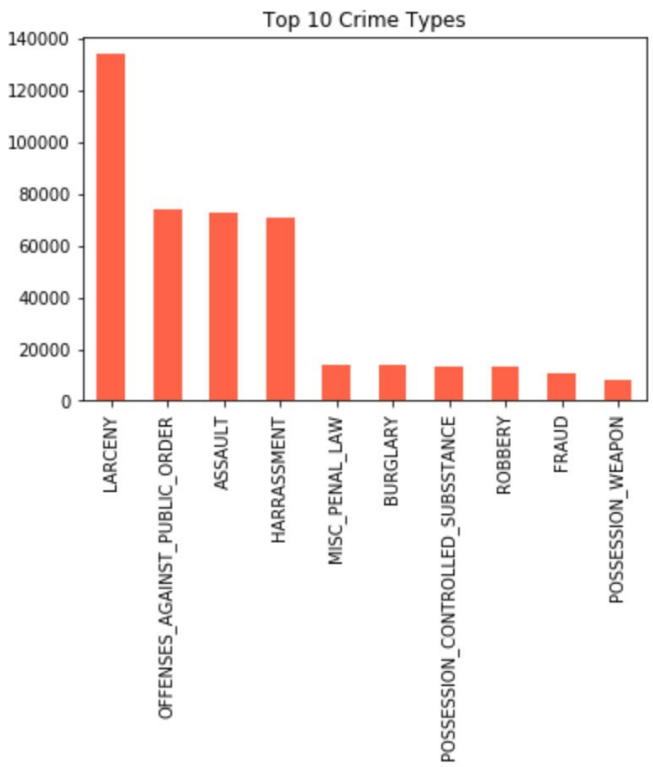
Procedure

What we will do is focus on the top 4 most occurring crimes and try to use machine learning to predict those. We will use 70,000 samples from each of the top 4 crimes then split the data into training and testing data.

Then we will fit a logistic regression and random forest classifier on the data, investigate the accuracy, then perform hyperparameter tuning to compare which classifiers achieve better predictions on what crimes may occur in different areas of NYC.

Inspecting the Data

Since we are trying to use machine learning to predict the crime type, we visualized the number of occurrences of each crime type in the data set to see if it was balanced.



When looking at the top 10 crimes, we can see that there is a heavy skew towards the top 4 crimes - the top 4 crimes have greater than 70,000 occurrences, while the rest of the crimes have less than 15,000 occurrences. We don't have enough uniform data to make a prediction on all the classes.

We can conclude here that we will have to resample our data. We can try undersampling or oversampling, but undersampling would risk losing important data, while oversampling would risk overfitting.

What we decided on doing instead, is to simply try to predict the top 4 most occurring crimes, and resample our dataset so that it only contains 70,000 rows each for the top 4 crimes. This will allow for predicting on a more uniform dataset with less data loss.

Feature Engineering

In this step, we inspect and select different features to use for training the dataset and predicting the outcome with.

At first, I chose a handful of features such as BOROUGH, DISTRICT, CRM_ATPT_CPTD_CD, PREM_TYP_DESC and VIC_SEX but my accuracy of Logistic Regression was only about

20%. I realized I needed to use more features to get greater accuracy. After trial and error, I chose the following features:

features = ['BOROUGH', 'DISTRICT', 'MONTH', 'VIC_AGE_GROUP', 'VIC_RACE','VIC_SEX', 'SUSP_AGE_GROUP', 'SUSP_RACE', 'SUSP_SEX']

These features yielded a testing accuracy of over 50% on each classifier.

One feature I wanted to use was OFFENSE LEVEL, but I chose not to for a few reasons.
1) Because a crime such as theft can be multiple offense levels (Felony or Misdeameanor).
2) Choosing OFFENSE_LEVEL alone gave me 50% accuracy with Logistic Regression, which immediately told me that it is easy to overfit using this feature.

Preparing Data

To prepare my data, first I resampled the data as described above, using only the rows containing the top 4 crimes and pulling only 70,000 rows each. This created a new dataset with 280,000 rows.

I then used the features I defined above to create the X dataframe containing the data I will use to train and predict with. I used the CRIME_TYPE column for my Y dataframe, as the prediction target.

I one-hot encoded all categorical variables in my X dataframe using pd.get_dummies(). I numerically encoded my Y dataframe using pd.factorize(). This is because The Y dataframe needs to be a single column as the target, and one-hot encoding would make it 4 columns.

Finally, I converted my X and Y dataframes into Numpy arrays, and split them into training and testing datasets using train-test-split.

Logistic Regression

Our first classification method was performed with logistic regression. Logistic regression models are able to make predictions through the logistic function.

I wanted to aim for above 50% accuracy scores on both my training and testing data. This means that although it’s not reliable enough, logistic regression is still better than using random chance to predict crimes.

Vanilla Logistic Regression Accuracy

First, I wanted to test out how accurately a basic Linear Regression with default settings would predict on my data. I ended up with a training accuracy of 0.532 and a testing accuracy of 0.537. Since the scores are similar, I can assume it is not overfitting on my training data.

Logistic Regression Hyperparameter Tuning

I wanted to tune the hyperparameters to see if I could get a little better accuracy out of my model. I ran a GridSearchCV on the logistic regression model and found out that using an l1 penalty with a liblinear solver were the best hyperparameters.

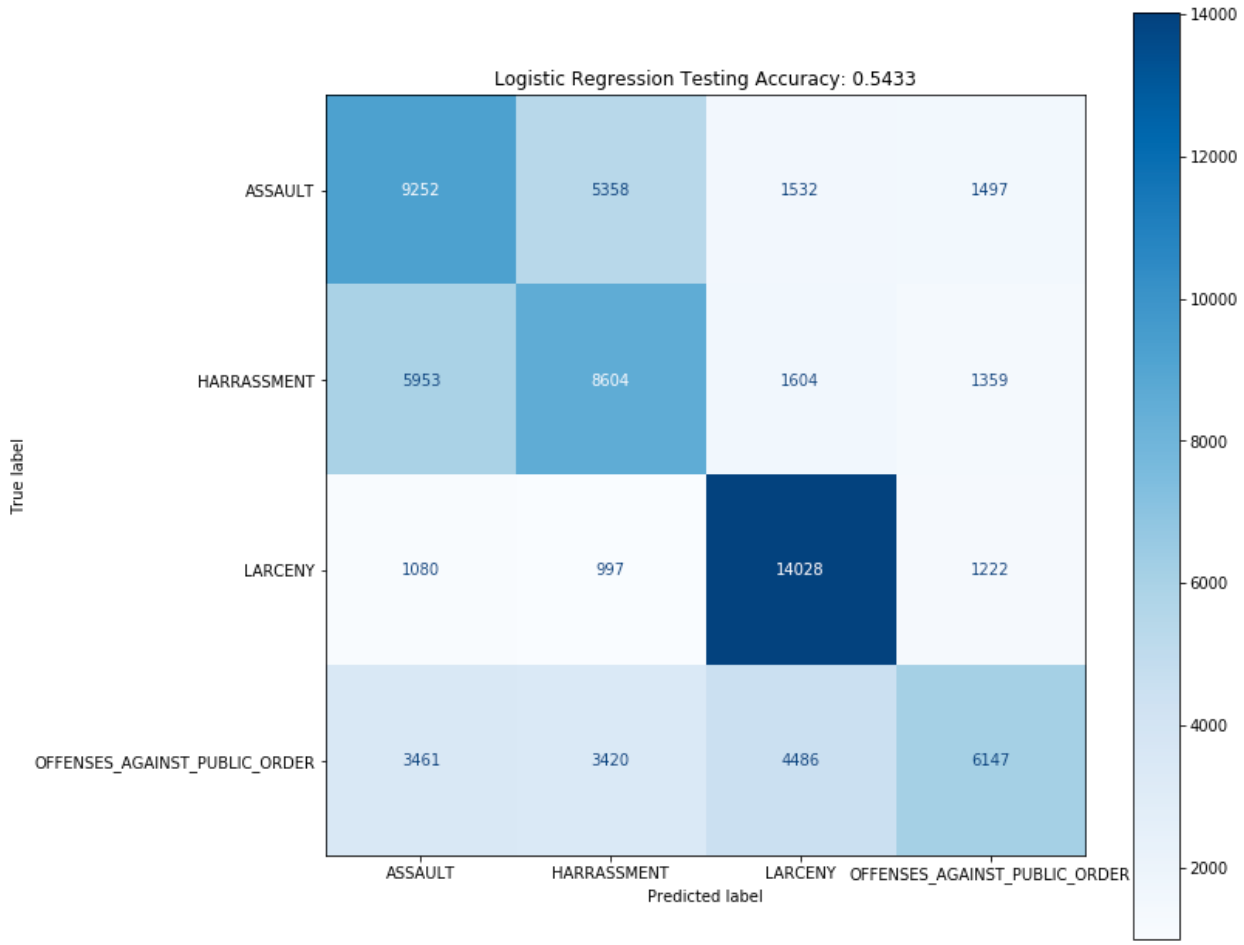
In this analysis, logistic regression was performed with an *l1* penalty in order to include a regularization method which works to prevent overfitting by decreasing the weights of the features to a small number (we initiated 4 features). Additionally, *liblinear* (A Library for Large Linear Classification) can work with *l1* penalties and which is said to minimize the cost function more quickly for large datasets.

Hypertuned Logistic Regression Accuracy

After applying the best estimators using hyperparameter tuning, our scores got higher. I achieved a training accuracy of 0.548 and a testing accuracy of 0.543, which was better than the vanilla Logistic Regression.

Confusion Matrix (Logistic Regression)

Confusion matrix was applied to test the performance of the hypertuned logistic regression model that we made. The confusion matrix shows the True Positives, False Negatives, False Positives and True Negatives for each class being predicted and its true result.



The confusion matrix showed a diagonal trend, meaning that the features are not biased in classifying crimes. The diagonal trend indicates that in the majority of cases, when crimes are misclassified, they were incorrectly classified into the nearby crime. Since we have 9 features and some are somewhat related, the misclassification was already expected. You can see some misclassifications in ASSAULT and HARASSMENT.

Random Forest

This function includes parameters for stratified cross-fold validation and supports code for hyperparameter tuning for a random forest model. Because we are working with a multi-classification problem, we evaluate our logistic regression models with one-vs-one classification.

Vanilla Random Forest Accuracy

I wanted to test out how accurately a basic Random Forest with default settings would predict on my data.

I ended up with a training accuracy of 0.853 and a testing accuracy of 0.508. It looks like the training accuracy is drastically greater than the testing accuracy, which indicates overfitting. We can fix this using hyperparameter tuning.

Random Forest Hyperparameter Tuning

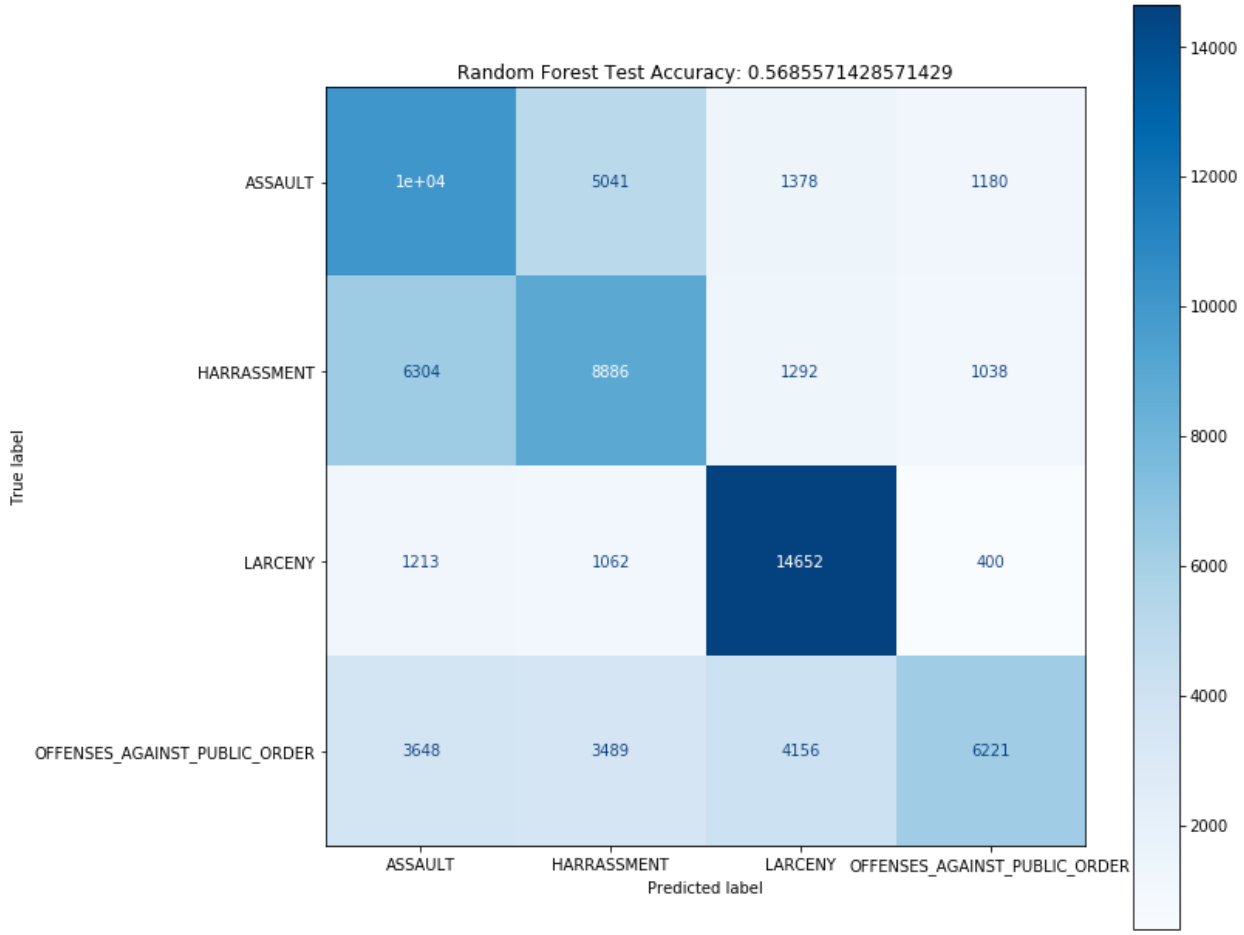
Hyperparameter tuning our Random Forest model will allow us to try to prevent the overfitting happening in the default Random Forest model. I ran a GridSearchCV on the Random Forest model and found out that the best hyperparameters were **criterion='entropy', max_depth=15, min_samples_leaf=10, n_estimators=70, random_state=12345.**

In this analysis, **entropy** calculates the homogeneity of the sample data. If the sample data is completely homogeneous the entropy is zero and if the sample is equally divided it has entropy of one. The **max_depth**, 15 is the maximum number of features random forest considers to split a node. The opposite does in our min_samples_leaf, the minimum number of samples required to split an internal node. Another significant estimator is **n_estimators** hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking the averages of predictions, here, we'll choose 70 random trees.

Hypertuned Random Forest Accuracy

After applying the best estimators using hyperparameter tuning, the testing accuracy got higher and the overfitting was fixed. I achieved a training accuracy of 0.5863 and a testing accuracy of 0.5695, which showed that the model was no longer overfitting to the training data.

Confusion Matrix (Random Forest)



The confusion matrix showed a diagonal trend, meaning that the features are not biased in classifying crimes. The diagonal trend indicates that in the majority of cases, when crimes are misclassified, they were incorrectly classified into the nearby crime. Since we have 9 features and some are somewhat related, the misclassification was already expected. You can see some misclassifications in ASSAULT and HARASSMENT.

Random Forest Hyperparameter Tuning

Found best parameters to be:

rf_best = RandomForestClassifier(criterion='entropy', max_depth=15, min_samples_leaf=10, n_estimators=70, random_state=12345)

In this analysis, **entropy** calculates the homogeneity of the sample data. If the sample data is completely homogeneous the entropy is zero and if the sample is equally divided it has entropy of one. The **max_depth**, 15 is the maximum number of features random forest considers to split a node. The opposite does in our min_samples_leaf, the minimum number of samples required to split an internal node. Another significant estimator is **n_estimators** hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking the averages of predictions, here, we'll choose 70 random trees.

Hypertuned Random Forest Accuracy

Training Accuracy: 0.5857476190476191
Testing Accuracy: 0.5746428571428571

Looks like the accuracy score got more consistent when we applied the best parameters from our hyperparameter tuning.

After comparing my accuracy scores in both logistics regression and random forest, we have better results if we use random forest than logistic regression. Both models, given the accuracy of more than 50%, will still give us better results in predicting crimes than random choice.

Conclusions

The main goal of the project is to be able to aid the detectives and NYPD to do crime analysis on crime data in NYC in 2019 and to create a model that would help them to predict crimes better.

To understand the statistics and nature of crime in NYC in 2019, various data exploration techniques applied using matplotlib and Seaborn libraries. I was able to create time series, bar and pie graphs for plotting the frequencies, and plot the crimes based on the location of crime occurrence. Next, I applied statistical inference. The data we have consists of categorical value. The most efficient way to apply inferential statistics for our data is through the Pearson Chi-squared Test. Lastly, I applied Logistic Regression and Random Forest for machine learning to create a model that would predict crimes better than chance alone.