The University of Texas at Austin

Chicago Crime Incident Analysis

Project Report

INF 397 – Statistical Analysis and Learning w/ Prof. Varun Rai
Spring 2018

TEAM MEMBERS:
Milind Siddhanti (mss4376)
Nimish Kate (nk8648)
Prachi Singh (ps28755)
Sanchit Singhal (ss84657)
Shreshtha Shukla (ss83452)

Introduction

As the mankind takes its steps towards digitization, the world eagerly follows. Information grows to become the most sought out resource. The ever-increasing presence of information can then be used to perform analysis and better our understanding of the world. In this constant race towards a better future, security is a important feature that presents an increasing concern. Cities are growing at a rapid rate, and so is the population. All these factors have led to an increase in crime, making it a priority for the authorities to identify crime-prone areas and take precautionary measures. To understand neighborhoods that need extra attention, it is imperative to identify zones with higher crime rates, predict recurring crimes, their frequencies, and analyze the time frames in which most crimes occur. A detailed analysis of the crimes would help the authorities understand the type of security measures that need to be put in place.

We attempt to explore the data to first find the most common types of crime, and sort them according to the neighborhoods. We attempt to understand the frequency of arrests for different crimes based on various features. By classifying crimes into whether or not arrests will occur, this project aims to assist law enforcement by providing a better understanding of arrest patterns in the city of Chicago. We will then analyze these findings and provide recommendations to local police authorities to improve policing practices and training.

Problem Statement

This report will provide a statistical analysis of criminal activities in Chicago and attempt to classify arrests based on features relating to the crime’s features.

Data & Preprocessing

The dataset for the analysis is borrowed from kaggle.com. It reflects reported incidents of crime (with the exception of murders where data exists for each victim) that occurred in the City of Chicago from 2001 to 2017. Data is originally extracted from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system. In order to protect the privacy of crime victims, addresses are shown at the block level only and specific locations are not identified.

The dataset is a total of 1.85 GB. The entire data is split into 4 groups of specific time spans consisting of over 6,000,000 records.

- 2001-2004 - 1923517 records
- 2005-2007 - 1872346 records
- 2008-2011 - 2688712 records
- 2012-2017 - 1456715 records

The available data had inconsistencies in records. There were missing values for certain features, and erroneous values for others. A few observations did not follow a consistent format of values. The data was cleansed to eradicate these issues. Data generation was also performed for missing data (records with missing geospatial information but a known location were transformed to generate the missing entries). Many records had spaces, quotations and other punctuation, which made organizing the data difficult. These contraptions were removed to make the data more robust.

Observations from the first file were removed from the refined dataset to reduce computational resource requirements - due to slow model training, a design decision was made to only analyze data from 2005 onwards. Also, it should be noted that that the year 2017, only had records for a few months and therefore significantly different aggregation numbers compared to the rest of the years.

The data has 22 columns, which will act as variables while building the models; they are listed below:

Features

ID - Case ID (Unique Identifier)
Case Number - Case number of the crime reported
Date - Date of the occurrence of the crime
Block - Area (block) of Chicago where the incident took place
IUCR - Illinois union crime reporting code
Primary Type - Top most classification of the crime
Description - Description of the crime
Location Description - Description of the location
Arrest - Arrested or not (True or False)
Domestic - Type of crime
Beat - Police jurisdiction
District - Police Jurisdiction
Ward - Jurisdiction of the crime scene
Community Area - Jurisdiction of the crime scene
FBI Code - FBI code defining crime category
X Coordinate - X coordinates of the crime scene location
Y Coordinate - Y coordinates of the crime scene location
Year - Year the case was filed
Updated On - Date/Time of the update on the crime case
Latitude - Location of the crime scene (Latitude)
Longitude - Location of the crime scene (Longitude)
Location - Location of the crime scene (Latitude+Longitude)

Data Exploration

An exploratory analysis of the cleansed data revealed the following:

- The number of crime incidents encountered by the Chicago crime department per year.

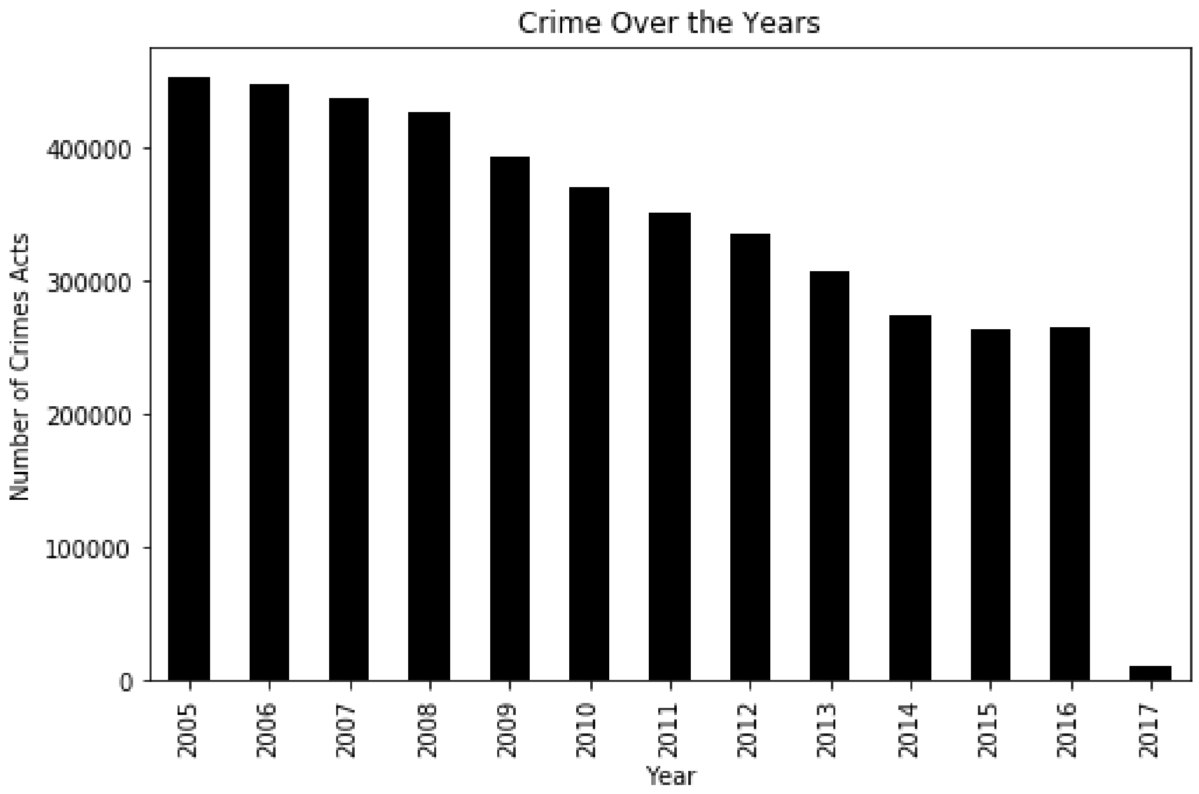


Fig. 1. Number of crimes reported each year

In Fig. 1, we can observe that the number of crimes reported in Chicago reduced gradually from 2005 to 2017. There was a significant drop in the crimes reported in the year 2009 and was almost constant in 2015 and 2016. As we do not have complete data for 2017, we cannot compare that with the data collected over the years.

- The relative number of crime incidents encountered by the Chicago crime department per month over the span of 13 years.

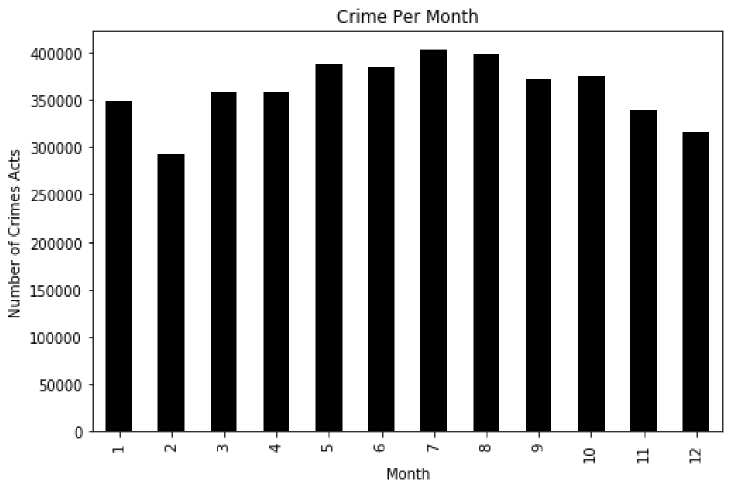


Fig. 2. Number of crimes reported per month

In Fig. 2, we can observe that the highest number of crimes were reported in July and the lowest in the month of February.

- The relative number of crime incidents encountered by the Chicago crime department per day of each month over the span of 13 years.

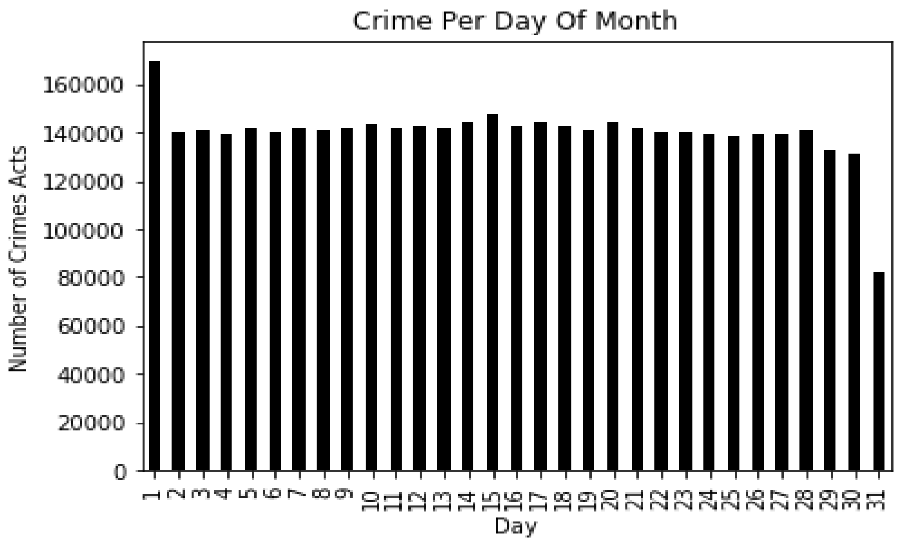


Fig. 3. Relative number of crimes reported every day per month

In Fig. 3, we can observe that the highest number of crimes were reported on the first of every month. There seems to be a drop in reported crime in the last three days of the month but the other days seem to have approximately the same number of reported crimes.

- The relative number of crime incidents encountered by the Chicago crime department every hour of the day over the span of 13 years.

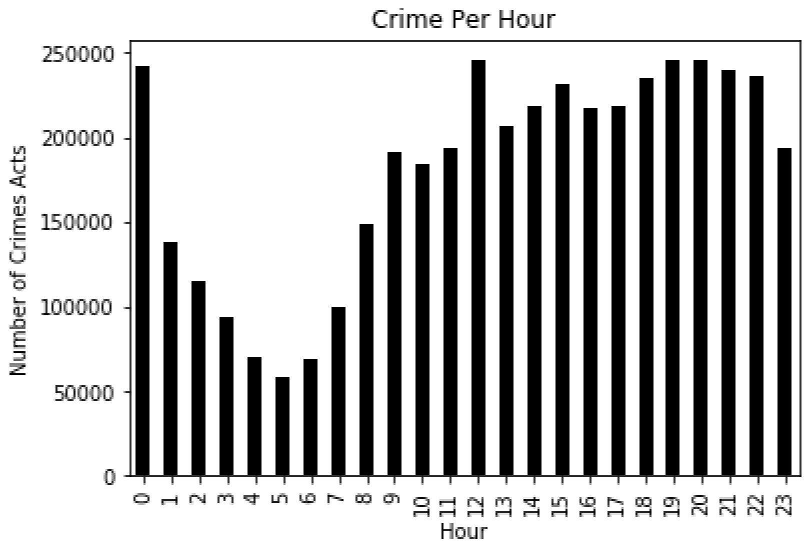
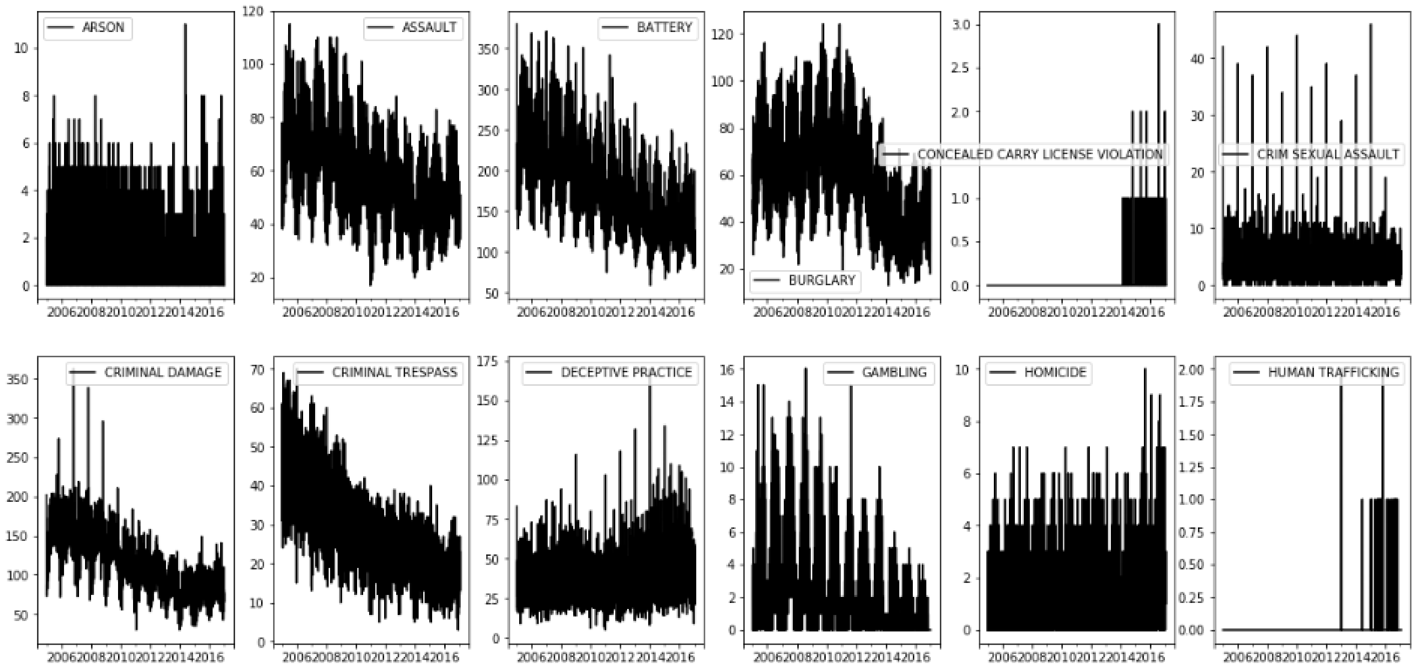


Fig. 4 Number of crimes reported every hour per day

In Fig. 4, we can observe that the highest number of crimes were reported around 12 PM every day. The crimes were significantly lower during the early hours of the day and increased during the evening and night time.

- Types of crime per month



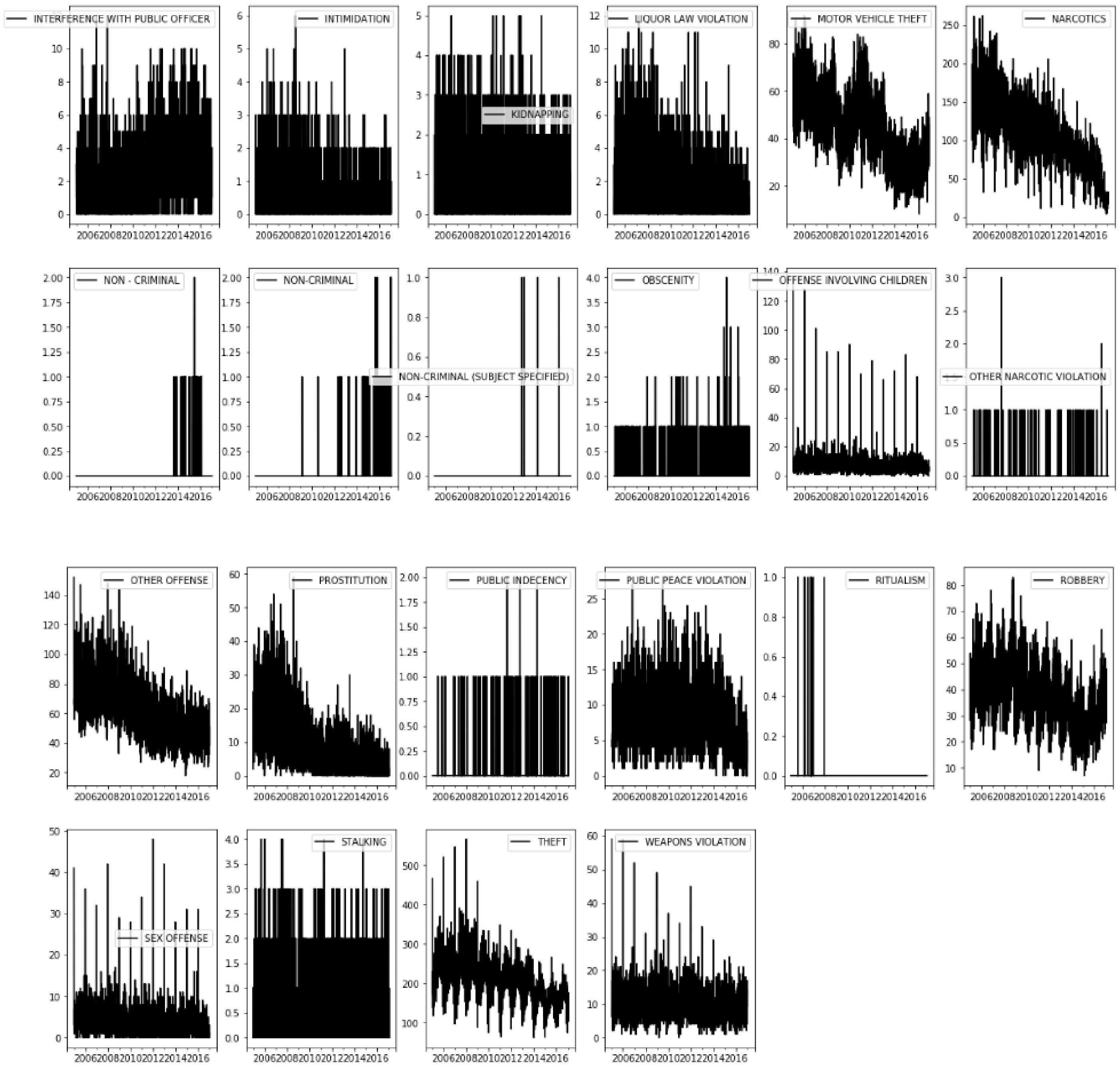


Fig. 5 Types of crimes reported per month

The number of crime incidents for the year 2017 are significantly less than those reported for the other years. A manual scan of the data revealed that the dataset only has records for January of 2017. We decided to exclude the data for the year 2017 noting its incomplete and incoherent nature. As it is shown in fig. 1, the crime reported was substantially higher in the year of 2008, and has steadily declined over the years. An exploratory analysis of this revealed that many crimes were reported at midnight on the first day of the year. This data appears to be unprecedented, and a possible explanation might be the way the crime department operates on the first day of the year. In order to reduce inaccuracies in the data, we shifted our analysis to focus on the years, rather than the time of a particular day.

Despite having a significantly large number of crime incidents from 2006 to 2010, the number of arrests did not increase to match the crime reports. The difference between the crime reports and arrests seems to perpetuate over the span of 16 years.

- The number of crime incidents reported for each type of crime.

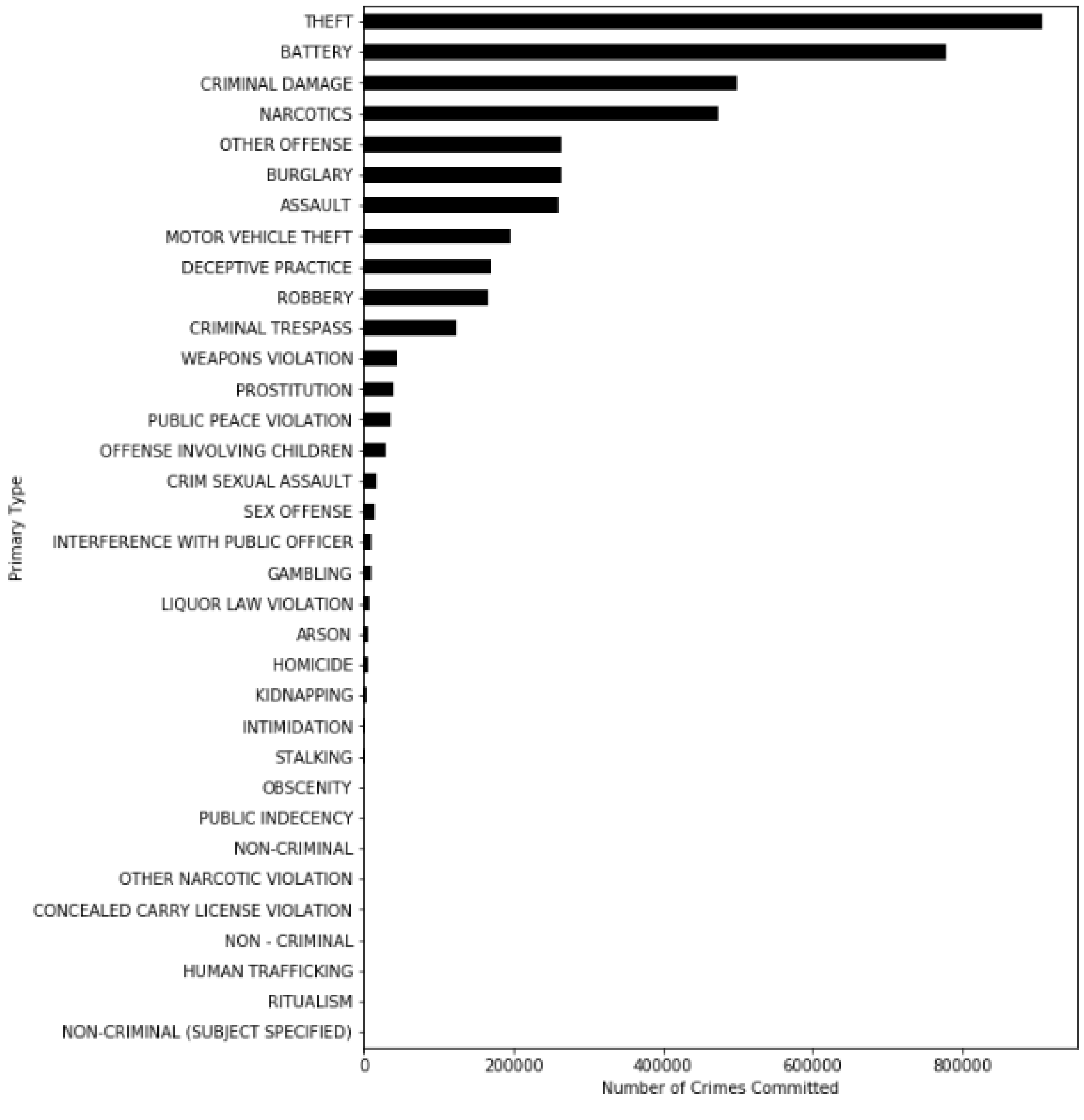


Fig. 6. Number of crimes reported for each type of crime.

The crimes for theft were the highest to be reported as high as 1,640,506, while domestic violence was the least reported crime with a number as low as 2. There appears to be a large difference between the incidents of different crimes reported. The lack of domestic violence crimes can be attributed to probable unwillingness to report domestic violence, not necessarily implying that domestic violence does not happen in the city of Chicago.

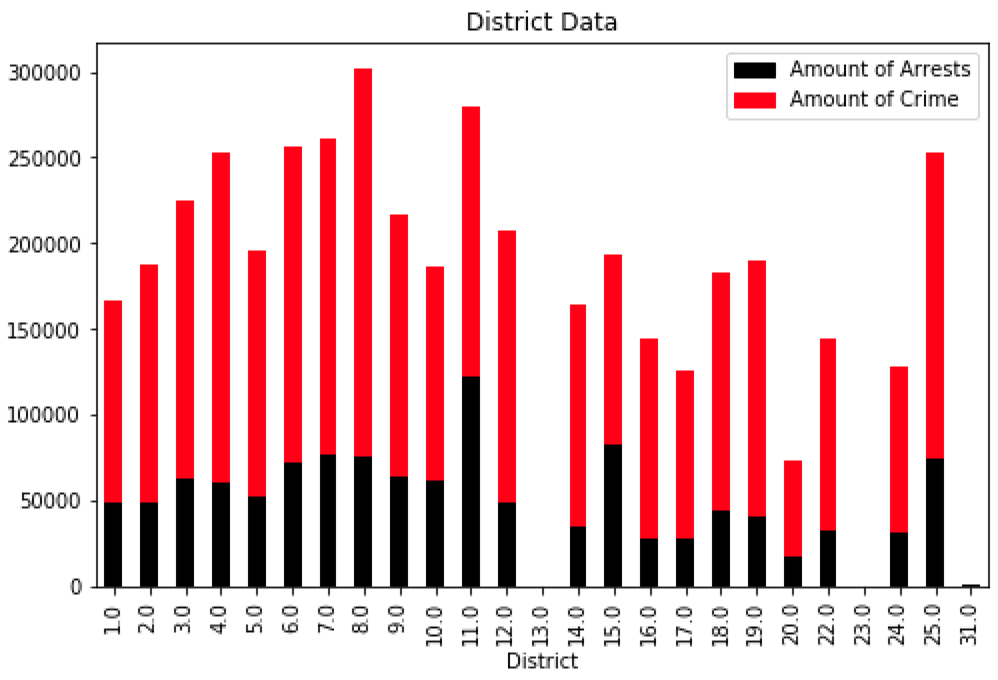


Fig. 7. Number of arrests made for the number of crimes reported district wise.

Fig. 7 shows the number of arrests that have been made in each district as compared to the number of crimes that have been committed. For District number 13, 23 and 31, there are hardly any number of crimes committed and arrests made. District 11 and 15 have a 50% arrest rate for the crimes committed.

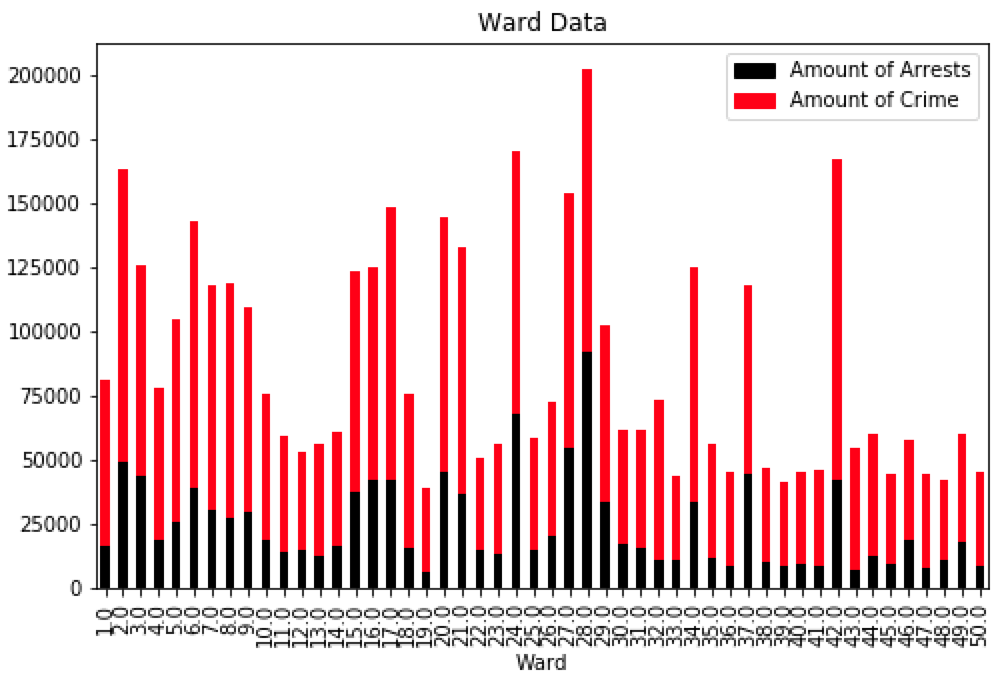


Fig. 8. Number of arrests made for the number of crimes reported ward wise.

Fig. 8 shows the number of arrests that have been made in each ward as compared to the number of crimes that have been committed. Ward numbers 28, 24 and 37 have a 50% arrest rate for the crimes committed. All other wards have about similar rates.

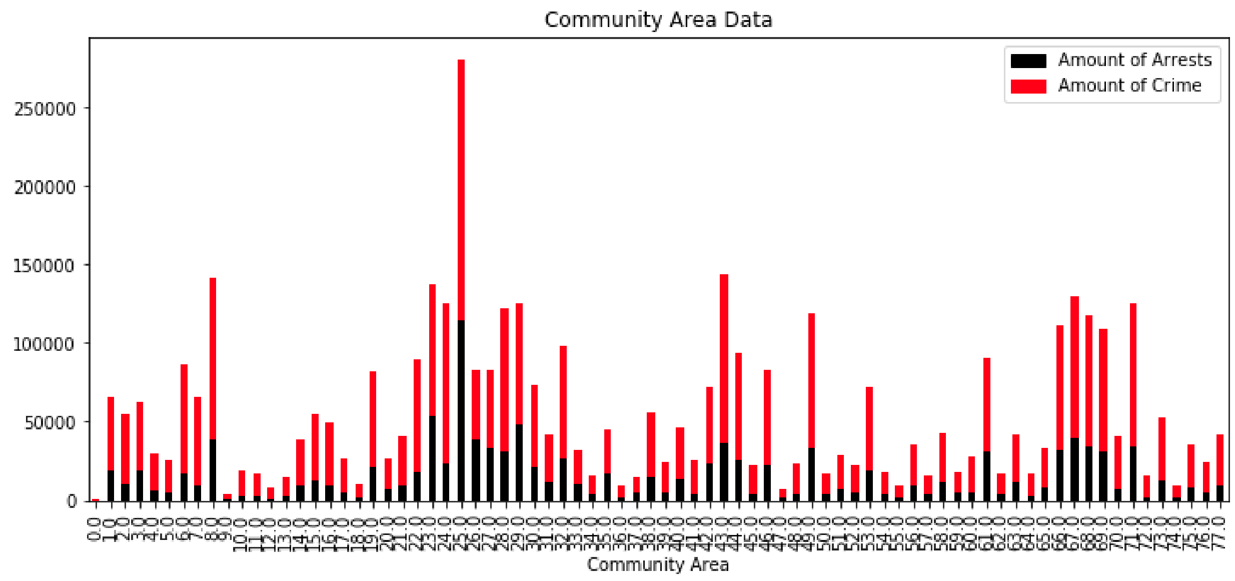


Fig. 9. Number of arrests vs the number of crimes reported community area wise.

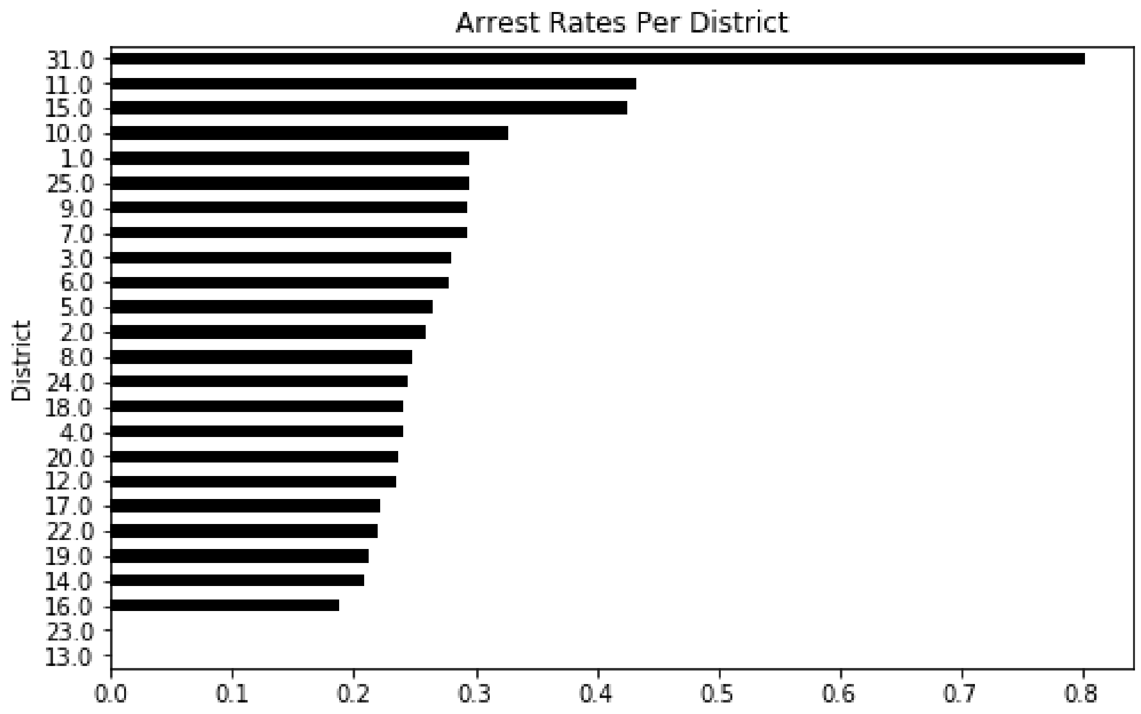


Fig. 10. Number of arrests made in districts in decreasing order.

Fig. 10 shows the arrest rates for each of the district. As seen from the data in Fig. 7, District 11 and 15 have about 50% arrest rate but District 31 has the highest arrest rate as high as 80%. This was not evident on Fig. 7 due to less number of crimes being committed in the district. District 13 and 23 have almost 0% arrest rates.

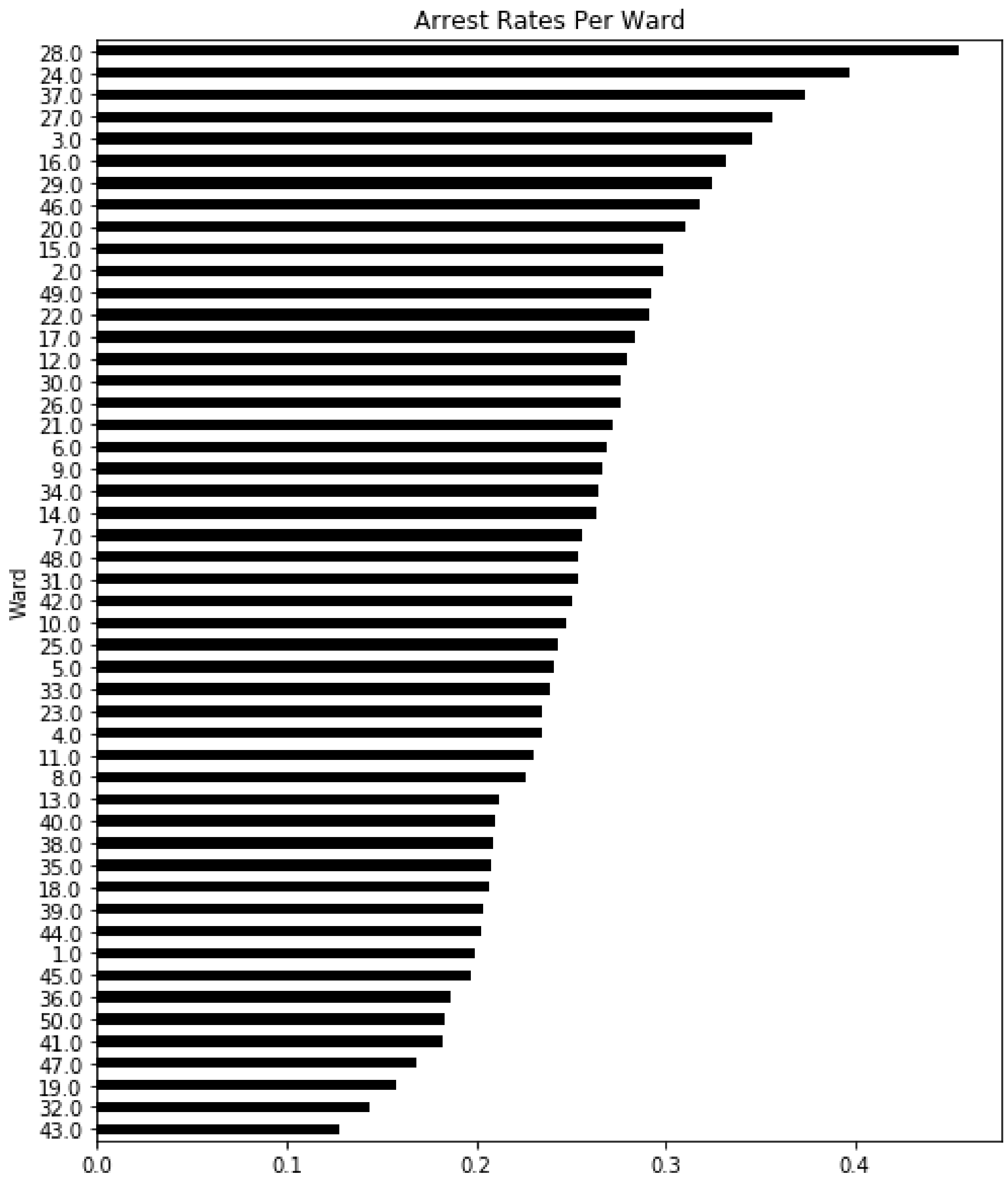


Fig. 11. Number of arrests made in wards in decreasing order.

Fig. 11 shows the arrest rates for each of the wards. As seen from the data in Fig. 8, Ward number 28 has about 50% arrest rate followed by Ward 24 and 37. On the contrary ward 43 has the lowest arrest rate as low as 12%. Rest all the wards have arrest rates between 15 and 40%.

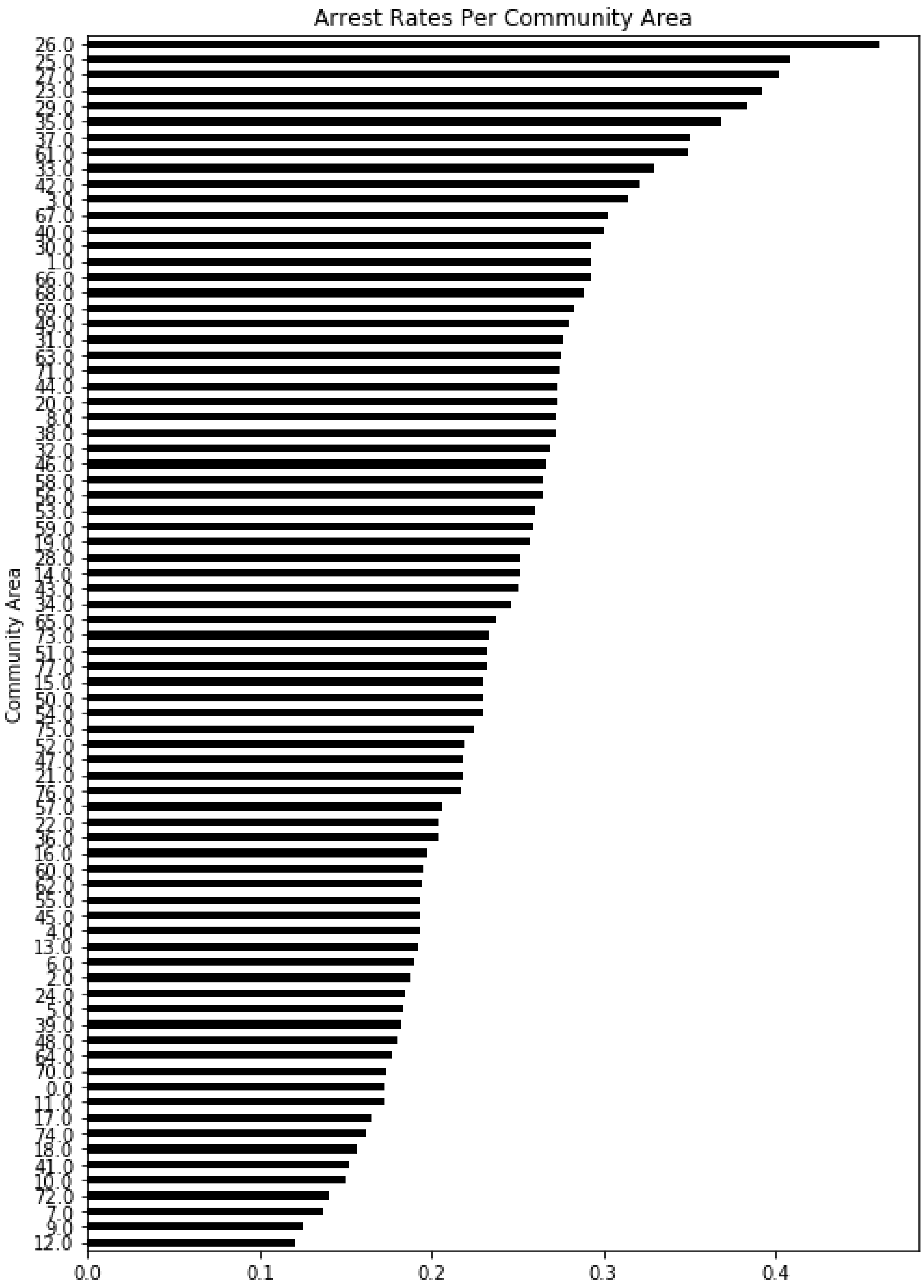
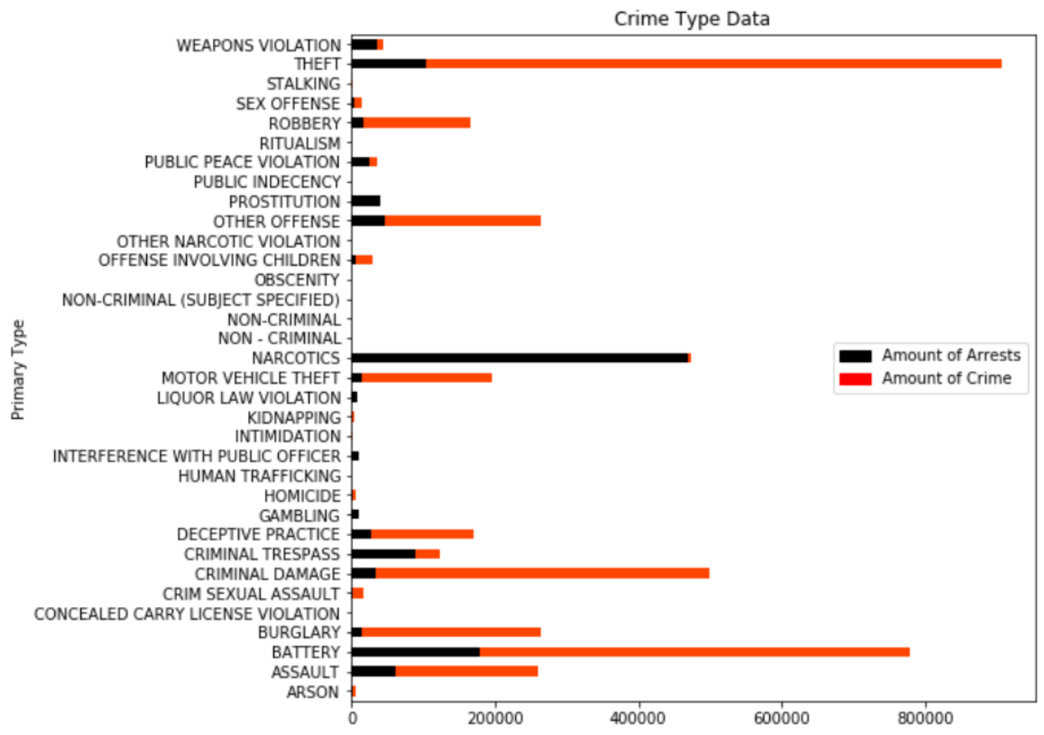


Fig. 12. Number of arrests made in wards in decreasing order.

- The number of arrests for the reported crime incidents.



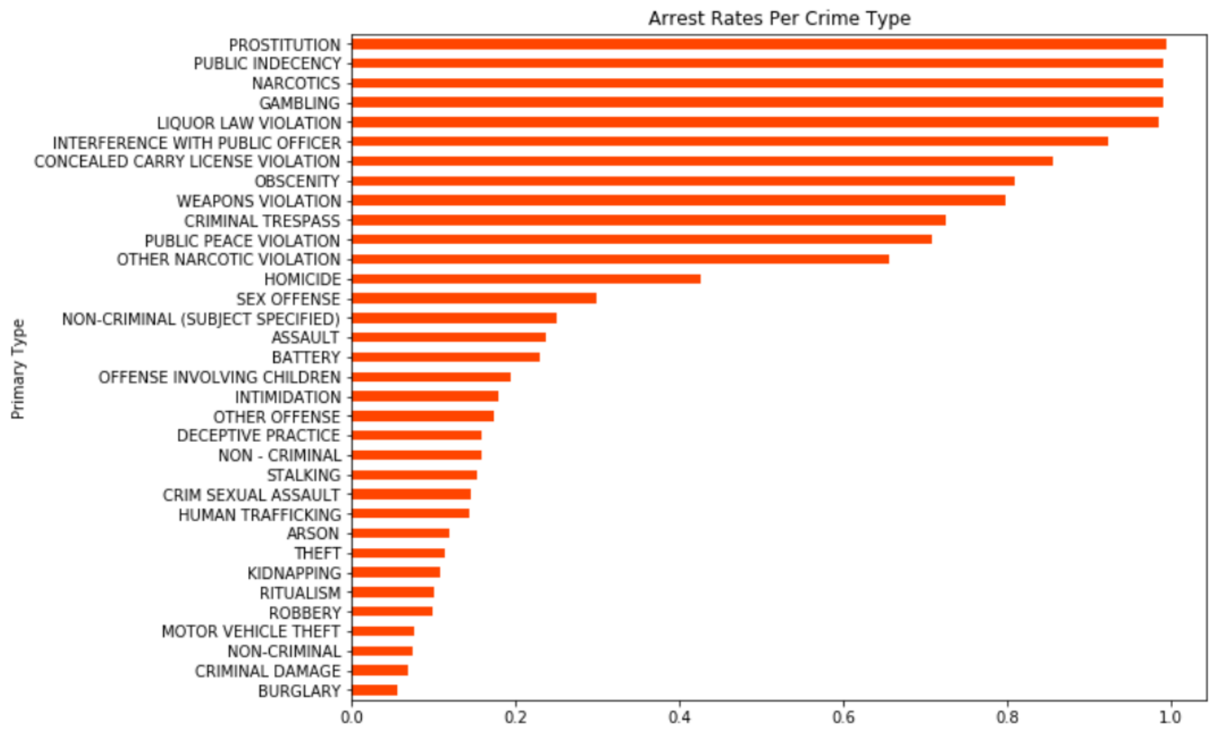


Fig. 13. Arrest vs Crime Rates

The preliminary analysis conducted provides decent insights about the crime and policing pattern in the city of Chicago. It also brings to attention certain major issues about the law enforcement and their understanding of criminal activity. We proceed to classify arrest patterns to better interpret the relationship between criminal acts and the resulting arrests.

Explored Methods

Our general strategy when modeling the dataset was to build various classifiers, and then evaluating the results using the accuracy on the test set. Confusion matrices were generated for each model to further understand the predictive performance for the framework. The steps below are in the order that they were performed :

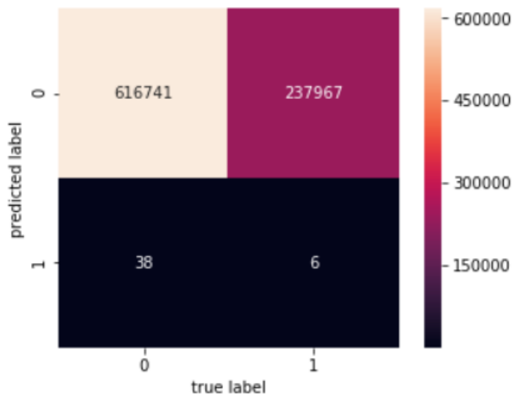
1. Principal Component Analysis

Before using the dataset on supervised learning methods, we decided to use unsupervised learning to help better explain the variance in the information. Therefore, we applied PCA to our dataset to derive a set of low-dimensional features from our variables that would potentially help us gain a clearer perspective of the relationships. We chose the number of components needed to describe the data on the basis of the cumulative *explained variance ratio* as a function of the number of components. Because we had already removed several redundant features, we observed that there was no change in the number of variables. The resulting output had 12 features which were then used in the subsequent approaches.

2. GLM Least Square Regression

Initially, we implemented the Linear Regression model on our dataset to predict the possibility of an arrest on the basis of the various parameters that were available. For this, we split our dataset into a training set containing 80% of our data and the remaining 20% as our test set. The probability was calculated based on the logic that if the probability of arrest is more than 50%, there would be an arrest, and for all the other cases (less than 50%), the crime would not lead to an arrest. We created a confusion matrix for our results and found that even though the test accuracy of the model was 72.15%, the model was predicting clearly wrong results that were biased. Because most of the records in our data are of the class 'not arrested', we think that the model was predicting the value on the basis of the mode of the data - classifying almost everything into the popular class. Thus, we concluded that the linear regression would not be a good model for our dataset.

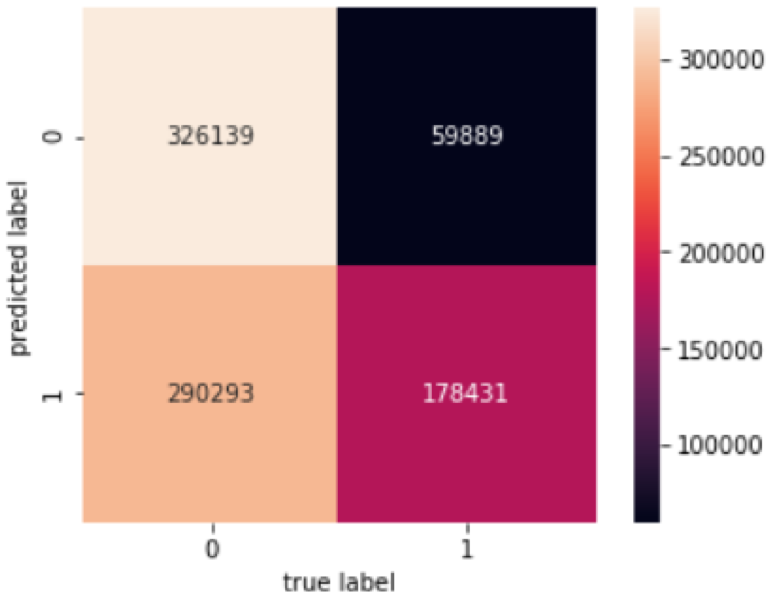
```
             precision    recall  f1-score   support

        0.0       1.00      0.72      0.84    854708
        1.0       0.00      0.14      0.00        44

avg / total       1.00      0.72      0.84    854752

Accuracy of Model is: 0.721551
```



3. <u>Logistic Regression</u>

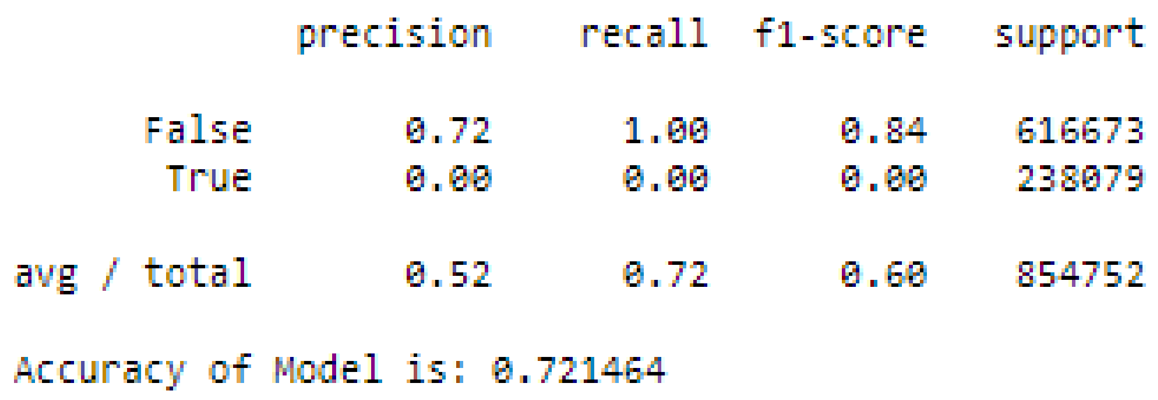
Next we considered Logistic Regression to judge whether the probability of the arrest belong to the class true or false for arrest. Again, we split our dataset into a training set containing 80% of our data and the remaining 20% as our test set. This model was only 47.88% accurate but the predictions were better than the ones we obtained from the linear regression model because it was not biased towards a false classification of arrest. As observed from the confusion matrix, this model had a relatively high number of correct classifications (false as false and true as true) and a low number of misclassifications of true arrests as false. Unfortunately, there were a high number of false arrests categorized as true which threw off the overall accuracy of the model. The team decided that although this binary classification approach was a valid framework for our data, we continued to assess other models.

```
[ 0.28300202  0.59447727  0.33211146]
Mean Cross Validation Accuracy for Logistic Regression: 0.4031969168842588
             precision    recall  f1-score   support

      False       0.53      0.84      0.65    386028
       True       0.75      0.38      0.50    468724

avg / total       0.65      0.59      0.57    854752

Accuracy of Model is: 0.478898
```

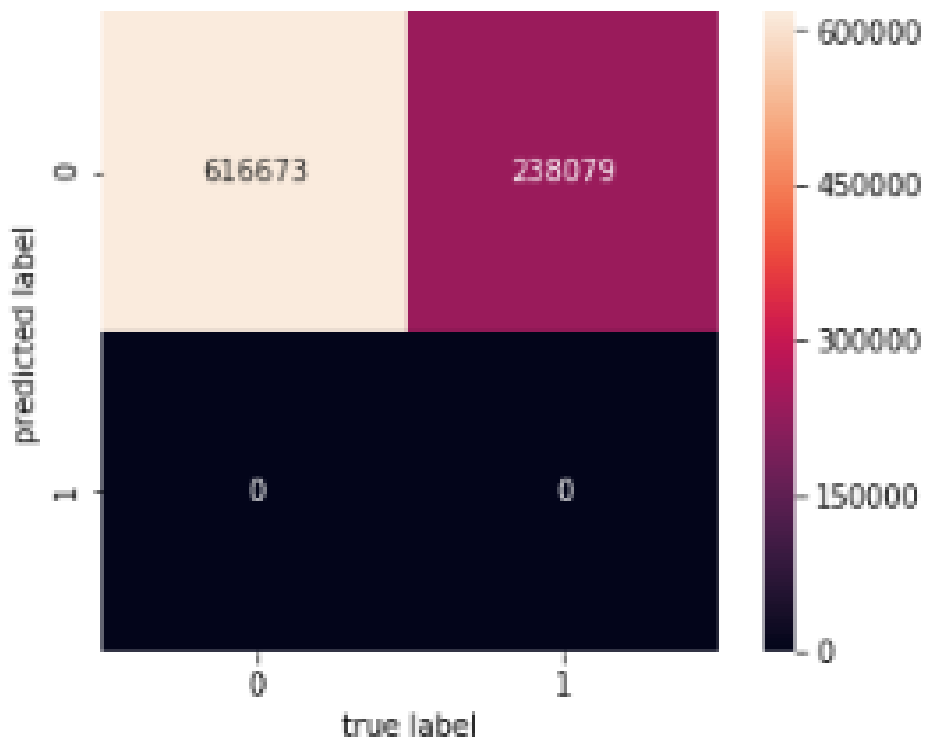


4. <u>Gaussian Naive Bayes</u>

Next, we chose the Naive Bayes classification model by assuming features in our dataset are independent of each other - this is mostly true although it can be argued that some of the geospatial data is correlated. Once more, we split our dataset into a training set containing 80% of our data and the remaining 20% as our test set. The accuracy of the model was 72% but, as seen, the bias is even more than the Generalized Linear Model as the model classified 100% of the test set as not arrests. We believe the reason this model did not perform well was because the Gaussian classifier, assumes a normal distribution and usually works better for continuous values; on the contrary, our dataset contained mostly discrete, nominal predictors and therefore was reason to discard the model.

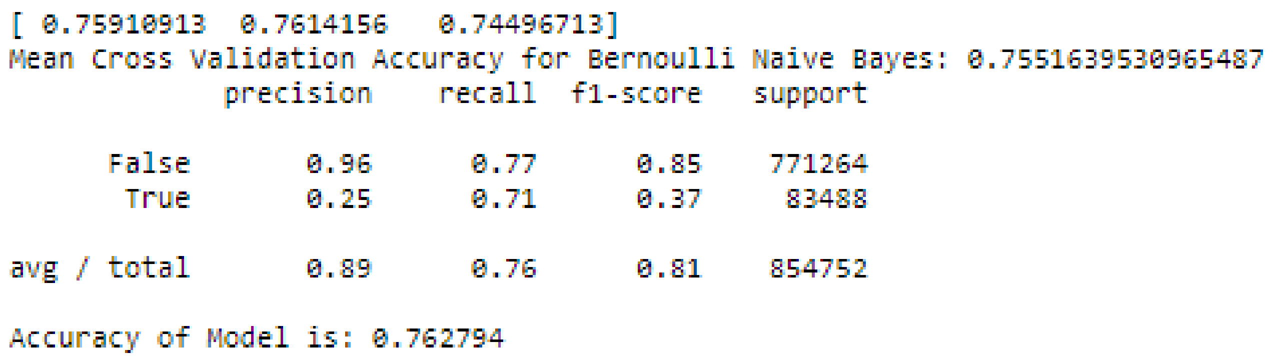


```
             precision    recall  f1-score   support

      False       0.72      1.00      0.84    616673
       True       0.00      0.00      0.00    238079

avg / total       0.52      0.72      0.60    854752

Accuracy of Model is: 0.721464
```

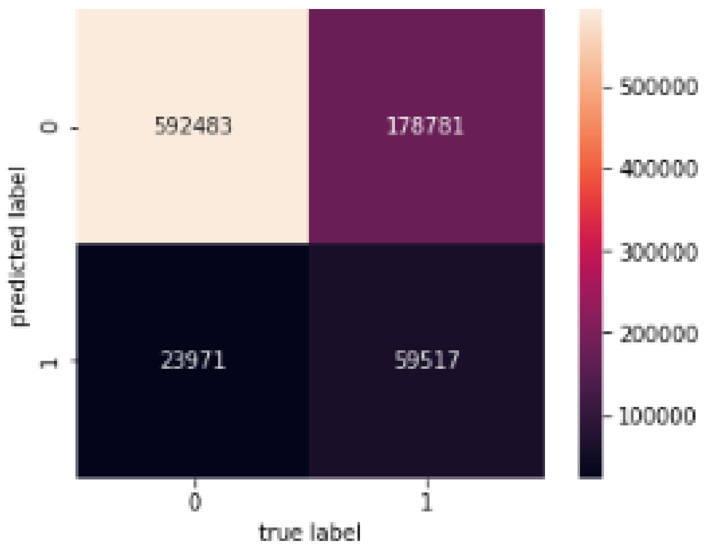


5. <u>Bernoulli Naive Bayes</u>

We also attempted to build Bernoulli Naive Bayes model - which works well when the variables are discrete (with the same assumption of independence). As with the previous models, we split our dataset into a training set containing 80% of our data and the remaining 20% as our test set. This model performed surprisingly well and produced an accuracy of 76.27%. Even with such a high accuracy level, when examining the confusion matrix, it can be seen that the model is still somewhat biased towards false classifications (most of the outcomes are predicted to be not arrests). That being said, this model is doing a fairly good job at not predicting false arrests as true which could be beneficial to our objective.

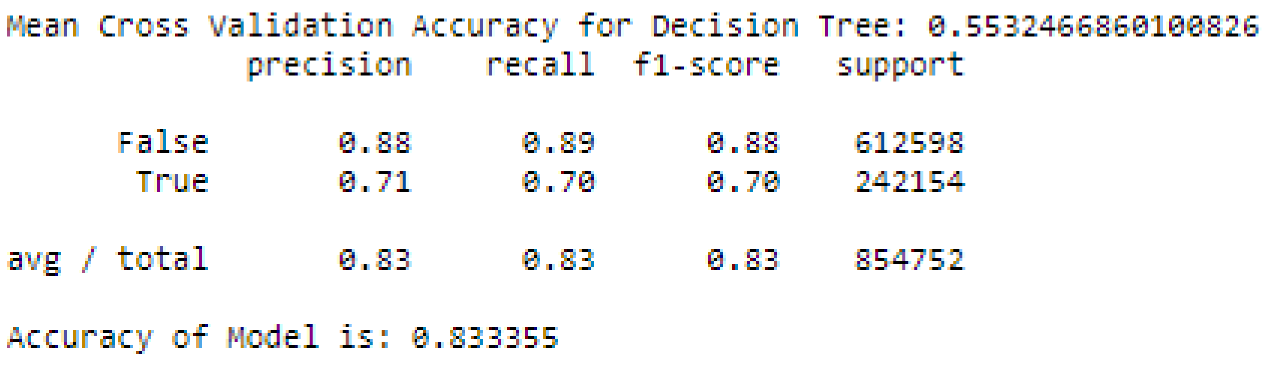


```
[ 0.75910913  0.7614156   0.74496713]
Mean Cross Validation Accuracy for Bernoulli Naive Bayes: 0.7551639530965487
             precision    recall  f1-score   support

      False       0.96      0.77      0.85    771264
       True       0.25      0.71      0.37     83488

avg / total       0.89      0.76      0.81    854752

Accuracy of Model is: 0.762794
```

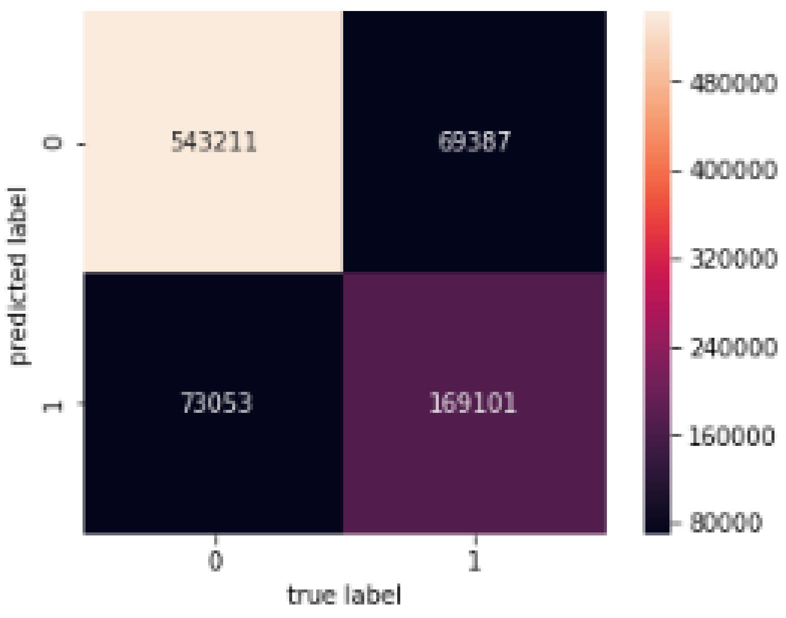


6. <u>Decision Tree</u>

We selected decision trees as our next model to predict the outcome. Again, we split our dataset into a training set containing 80% of our data and the remaining 20% as our test set. The accuracy of the model was 83.33% and produced the best results out of all the models built. Although computationally a little expensive, the model's confusion matrix was very good. The diagonal line formed by the true negatives that were classified as negative and true positives that were classified as positive were where most of the observations well. Both darker regions represent the misclassifications by the model - the fact that not a large percentage of records fell into this region suggested that this model was doing a good job at representing the relationships between the variables.



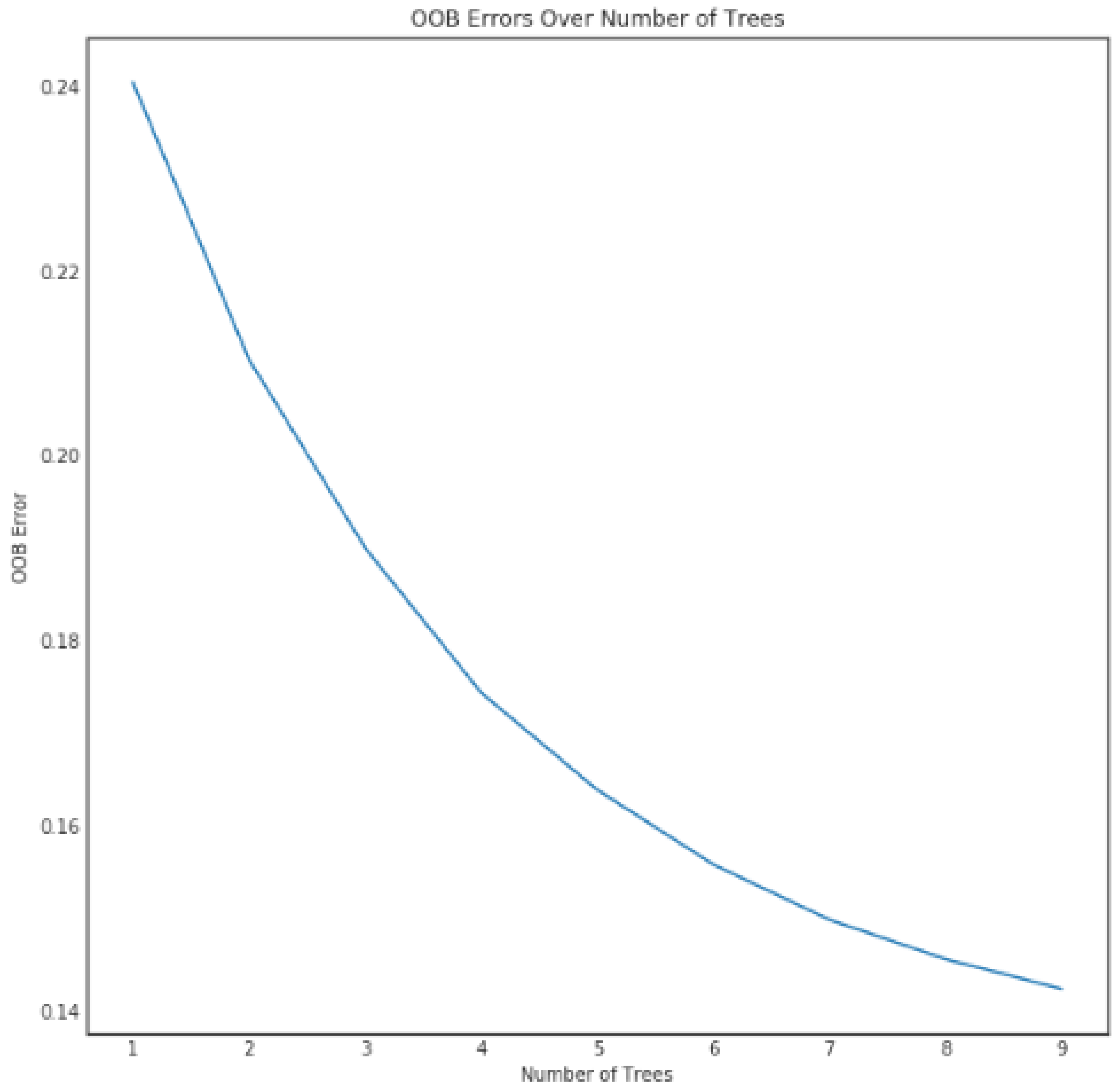
```
Mean Cross Validation Accuracy for Decision Tree: 0.5532466860100826
             precision    recall  f1-score   support

      False       0.88      0.89      0.88    612598
       True       0.71      0.70      0.70    242154

avg / total       0.83      0.83      0.83    854752

Accuracy of Model is: 0.833355
```



7. <u>Random Forest</u>

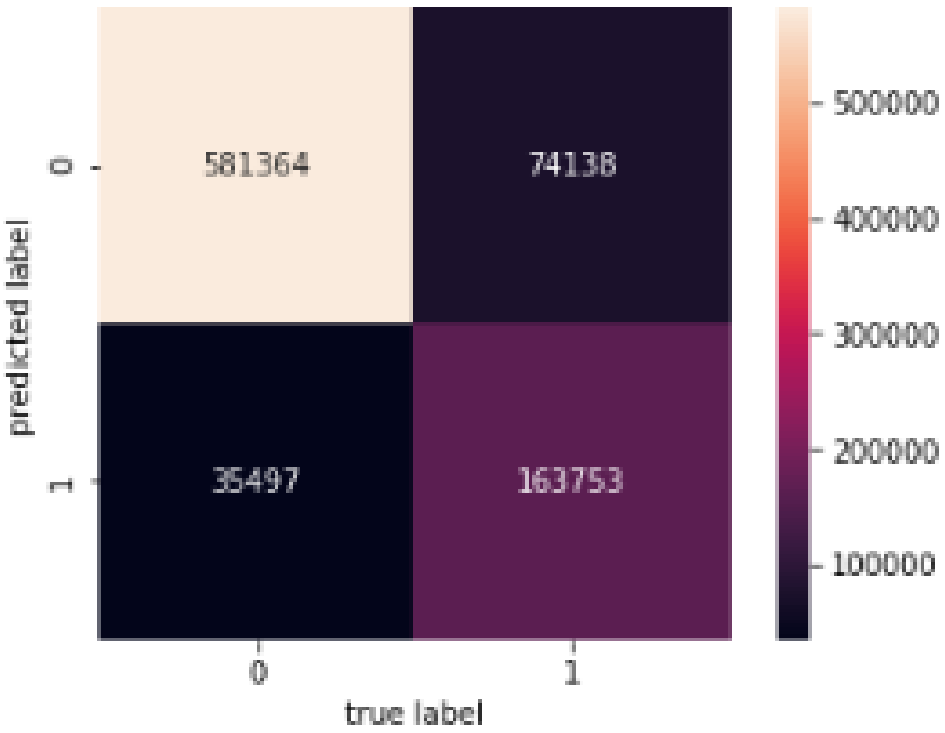
Given that the decision tree model performed the best out of all the previous frameworks, we next attempted to use ensemble learning to construct multiple decision trees and aggregate them into a random forest in order to produce a more optimized result. As with the previous models, we split our dataset into a training set containing 80% of our data and the remaining 20% as our test set.

Now that we had selected a framework and were working on optimizing the model, we ran a simulation of the random forest to analyze the Out-Of-Bag (OOB) errors rates for different number of trees. As seen from the figure below, the error rate falls as more number of trees are used. This is an expected result - but a design decision was made to select 9 trees as the optimal number. Building more trees than 9 provided not much gain in accuracy as compared to the computational resources and time needed to train the model.



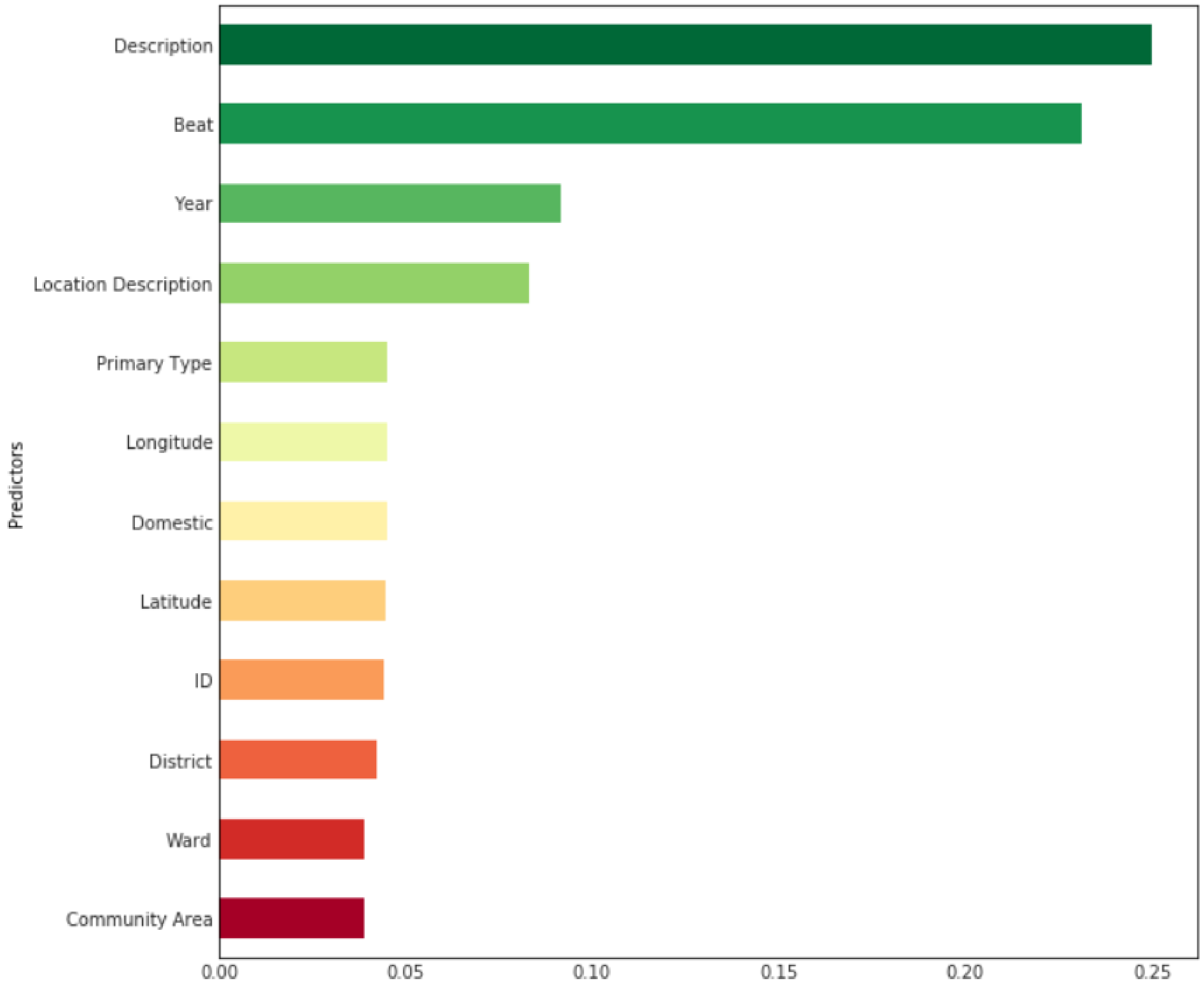
For the random forest with 9 trees, we got an accuracy of 87.17% - which was the highest level of correct classifications that were obtained from our experiments. As noted in the decision tree analysis, the confusion matrix displayed that most observations were being classified in the correct class.

```
             precision    recall  f1-score   support

      False       0.94      0.89      0.91    655502
       True       0.69      0.82      0.75    199250

avg / total       0.88      0.87      0.88    854752

Accuracy of Model is: 0.871735
```



Findings & Analysis of Models

Based on the analysis of the results we obtained from all the models, we can observe that the best model to predict the probability of arrest is the random forest. Because of its high accuracy and the correct prediction of the false arrests and the actual arrests, it was better than all the other models we implemented. As an additional benefit, the random forest and decision tree were more interpretable than some of the other models which allowed us to obtain the most important features when classifying arrest in the random forest.



We can observe that the Crime Description is the most important factor for deciding whether or not an arrest will take place. Obviously, this makes sense - the classification of arrest should be primarily based on the type of crime, as per policing protocol. We decided this was important to note as it implies that our model is considering the correct feature to calculate the possibility and produce the results.

The second most important factor was the Police Beat, which is surprising since the territory and the time of patrol of an officer should not affect the rate of arrest. After exploring the data again, we observed that the arrests were all concentrated in particular areas and were only related to certain Police Beats. This could be a potential problem for the Chicago Police Department as it could hint to many possible scenarios: performance of the other Police Beats, the possibility of some Police Beats being more aggressive than the others in their approach towards the offender, bias against certain communities, etc.

The third most important feature for the prediction was the year. As discussed in the data exploration, we concluded that as the year increased, the rate of arrest decreased as well. We can infer that the crime rate in the city of Chicago decreased in the span of 13 years and this was one of the reasons that the rate of arrest also went down. It’s also interesting to note that since the year is the third most important predictor: if we hold description of crime and beat constant, when you commit the crime can decide whether or not you will be arrested. Which implies that for the same crime in the same beat, one is less likely to get arrested for a crime now than in the past as the number of arrests are decreasing.

Another implication of our analysis is that the type of location played a more important role in deciding whether there would be an arrest or not as opposed to the physical, geographical location. For example, certain location descriptions such as streets and residences had a high number of arrests but which street or where the residence is did not play as important a role.

Lastly, the features that did not show much importance on the classification of arrest were fairly expected from our analysis above. Longitude, Latitude, Community, District, Ward all had little significance on our model. The team would like to note that most of these features are correlated to other variables which might need further analysis to finalize definitive findings.

Challenges

The biggest challenge we faced while working with our dataset was its quality. Our dataset consisted of approximately 6 million records and had inconsistencies and missing data. This limited our scope of working on questions we had planned on working on, in the initial stages of the project. We did initial work on answering two other questions to give better insights to Chicago Police Department which are as below:

- What better security measures can be implemented in which areas?

  We wanted to do a detailed analysis to identify if there were any specific type of crime that had high frequencies in some areas. On the basis of the location where these crimes were occurring, suggestions could be provided as to what type of security measures could be implemented in that district or block to avoid the high crime rate. For instance, if there are lots of mugging reported on a particular block, installing more surveillance cameras can help in reducing the number. Since the reported time was inconsistent and the addresses were masked, we did not reach to substantial conclusions.

- How to better assess incomplete 911 calls?

  We wanted to predict the type of crime that should be expected from a particular block or ward if the 911 caller is unable to give all the information. That would help the EMS team to estimate the amount of backup required.

Further, most of our data was either geographical data, date time and categorical data, which limited the implementation of models to only qualitative models. Because of the large size of the dataset and limited computational power, running the models was time consuming task. Some of the models took up to 10 hours of running time limiting our scope of exploration.

Weaknesses

The challenges discussed previously led to most of the weaknesses in our modeling and analysis. These areas could have led to inaccurate results and the team believed it was important to note these down:

- The geographical nature of the data, with multiple factors playing a role in the frequency and occurrence of crime, abstained the use of a general model for the entire data while predicting results.

- Lack of certain demographics (like unemployment, age, gender, alcoholism etc.) left dependent attributes from the database to be inadequate, which may have led to biased predictions (collinearity).

- Addresses have been masked to protect the privacy of crime victims, aggregating addresses to block level. This affected the accuracy of predictions.
- Computational and Time Resource available to us were limited. Therefore, a balance needed to be met between model complexity and accuracy.

Conclusion

An analysis on the relationships between crime and arrests for the city of Chicago can help city authorities plan, prepare, organize, and allocate available security measures to improve safety concerns. Our models revealed that, for the most part, the police department is doing a commendable job in assessing the classification of an arrest based solely on the crime itself. On the other hand, several predictors that were surprisingly impactful on the arrests proved to be the beat and the year of the crime. The team recommends a further, in depth analysis on the arrest rate related to these variables. A more thorough understanding of the systematic reduction in arrests over time would provide a clearer snapshot of the criminal activity in the city. It’s also imperative some investigation behind the distribution of arrest rates among the various beats is conducted to ensure the policing arounds these crimes is fair process. Before implementing any changes in policies that address these concerns we hope to continue analyzing the information and overcome the challenges and weaknesses we faced in our analysis to assist the city in their pursuit towards a safe future.

References

1. (What's Behind Chicago's Surge in Violence). Retrieved March 20, 2018, from http://www.chicagotribune.com/news/opinion/editorials/ct-chicago-crime-increase-causes-edit-0118-md-20170117-story.html

2. (Chicago Crime Data). Retrieved March 02, 2018, from (Chihttps://www.kaggle.com/currie32/crimes-in-chicago/data

3. (An Introduction to Statistical Learning). Retrieved March 02, 2018, from http://www-bcf.usc.edu/~gareth/ISL/

4. Boundaries - Police Beats (current) | City of Chicago | Data Portal. (n.d.). Retrieved March 02, 2018, from https://data.cityofchicago.org/d/aerh-rz74

5. C. (2017, January 28). Retrieved March 02, 2018, from https://www.kaggle.com/currie32/crimes-in-chicago/data

6. Chicago Police Department - Illinois Uniform Crime Reporting (IUCR) Codes | City of Chicago | Data Portal. (n.d.). Retrieved March 02, 2018, from https://data.cityofchicago.org/Public-Safety/Chicago-Police-Department-Illinois-Uniform-Crime-R/c7ck-438e

7. (Crime Type Categories). Retrieved March 02, 2018, from http://gis.chicagopolice.org/clearmap_crime_sums/crime_types.html

8. (Google Maps APIs). Retrieved March 02, 2018, from https://developers.google.com/maps/

9. (Chicago Crime Analysis). Retrieved March 02, 2018, from https://github.com/ajitkoduri/Chicago-Crime-Analysis

10. (Public Safety). Retrieved March 02, 2018, from https://github.com/dssg/publicsafety

Appendix

Find Python code used to work with the data below

In [1]:

```
# import required packages

import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
import os
import matplotlib.pyplot as plt
import pydotplus
import matplotlib.patches as mpatches

from sklearn.preprocessing import MinMaxScaler

from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.tree import export_graphviz

from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn import metrics
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score

from sklearn import linear_model
from sklearn.decomposition import PCA
```

In [2]:

```
## LOAD DATA
print('Loading Data')
```

```
Loading Data
```

In [3]:

```
# display parent directory and working directory

print(os.path.dirname(os.getcwd())+':', os.listdir(os.path.dirname(os.getcw
d())));
print(os.getcwd()+':', os.listdir(os.getcwd()));
```

```
C:\Users\sanch\OneDrive\Documents\Graduate School\UT-Austin - MSIS\INF 397
- Statistical Analysis and Learning: ['Data', 'Exams', 'Homework', 'Labs',
'Lectures', 'Project', 'SAL_Spring-2018_Syllabus.pdf', 'Textbooks']
C:\Users\sanch\OneDrive\Documents\Graduate School\UT-Austin - MSIS\INF 397
- Statistical Analysis and Learning\Project: ['.ipynb_checkpoints', '180209
, Project Timeline.docx', 'arrest_year_counts.xlsx',
'Chicago_Crimes_2001_to_2004.csv', 'Chicago_Crimes_2005_to_2007.csv', 'Chic
ago Crimes 2008 to 2011.csv', 'Chicago Crimes 2012 to 2017.csv', 'crime typ
e_counts.xlsx', 'Project Proposal PPT.pptx', 'Project Prototype v2.ipynb',
'Project Prototype v2.py', 'Project Prototype v5.ipynb', 'SAL code v10.4.py
', 'SAL Project Code v10.1.html', 'SAL Project Code v10.2.ipynb', 'SAL Proj
ect Code v11.1.py', 'SAL proposal.docx', 'SAL proposal.pdf',
'SAL_Project_Data Exploration-Models.html', 'SAL_Project_Data Exploration-M
odels_Evaluation_Final_Version.html', 'SAL_Project_Data Exploration-Models_
Evaluation_Final_Version.ipynb', 'well_A.csv', 'year_counts.xlsx']
```

In [4]:

```
# import data

Crime_5_7 = pd.read_csv('Chicago_Crimes_2005_to_2007.csv.',
                        na_values = [None, 'NaN','Nothing'], header = 0)
Crime_8_11 = pd.read_csv('Chicago_Crimes_2008_to_2011.csv.',
                        na_values = [None, 'NaN','Nothing'], header = 0)
Crime_12_17 = pd.read_csv('Chicago_Crimes_2012_to_2017.csv.',
                        na_values = [None, 'NaN','Nothing'], header = 0)
```

```
C:\Users\sanch\Anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:2785: DtypeWarning: Columns (21)
have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [5]:

```
Crime_Data = [Crime_5_7, Crime_8_11, Crime_12_17]
del Crime_5_7
del Crime_8_11
del Crime_12_17
```

In [6]:

```
## PRE-PROCESSING DATA
```

In [7]:

```
# comebine dataframes

Crime_Data = pd.concat(Crime_Data,axis = 0)

Crime_Data.info()
Crime_Data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6017770 entries, 0 to 1456713
Data columns (total 23 columns):
Unnamed: 0              int64
ID                      int64
Case Number             object
Date                    object
Block                   object
IUCR                    object
Primary Type            object
Description             object
Location Description    object
Arrest                  bool
Domestic                bool
Beat                    int64
District                float64
Ward                    float64
```

```
Community Area          float64
FBI Code                object
X Coordinate            float64
Y Coordinate            float64
Year                    int64
Updated On              object
Latitude                float64
Longitude               object
Location                object
dtypes: bool(2), float64(6), int64(4), object(11)
memory usage: 1021.5+ MB
```

Out[7]:

	Unnamed: 0	ID	Case Number	Date	Block	IUCR	Primary Type	Des
0	0	4673626	HM274058	04/02/2006 01:00:00 PM	055XX N MANGO AVE	2825	OTHER OFFENSE	HARASSMENT I TELEPHONE
1	1	4673627	HM202199	02/26/2006 01:40:48 PM	065XX S RHODES AVE	2017	NARCOTICS	MANU/DELIVER
2	2	4673628	HM113861	01/08/2006 11:16:00 PM	013XX E 69TH ST	051A	ASSAULT	AGGRAVATED: HANDGUN
3	4	4673629	HM274049	04/05/2006 06:45:00 PM	061XX W NEWPORT AVE	0460	BATTERY	SIMPLE
4	5	4673630	HM187120	02/17/2006 09:03:14 PM	037XX W 60TH ST	1811	NARCOTICS	POSS: CANNAB 30GMS OR LES

5 rows × 23 columns

In [8]:

```
# remove duplicates

Crime_Data.drop_duplicates(subset=['ID', 'Case Number'], inplace=True)
```

In [9]:

```
# remove data errors

Crime_Data.drop(['Unnamed: 0','Case Number','IUCR','FBI Code','Updated On',
'Location',
                 'X Coordinate','Y Coordinate','Location'], inplace = True,
axis = 1)
```

In [10]:

```
# format dates
```

```
Crime_Data.Date = pd.to_datetime(Crime_Data.Date, format = '%m/%d/%Y %I:%M:
%S %p')
Crime_Data.index = pd.DatetimeIndex(Crime_Data.Date)
```

In [11]:

```
# convert nominal features into categorical predictors

Crime_Data['Primary Type'] = pd.Categorical(Crime_Data['Primary Type'])
Crime_Data['Description'] = pd.Categorical(Crime_Data['Description'])
Crime_Data['Location Description'] = pd.Categorical(Crime_Data['Location De
scription'])
```

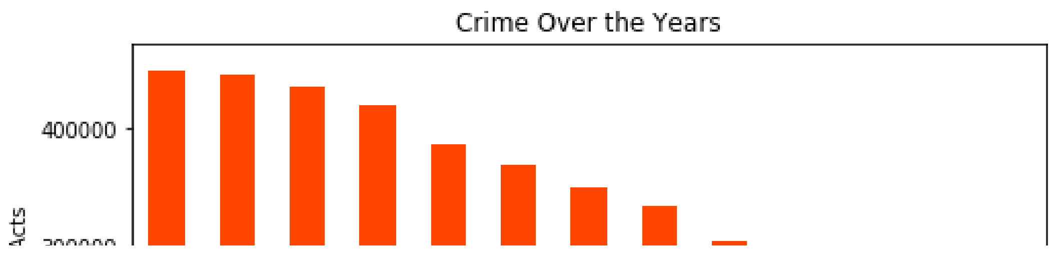
In [12]:

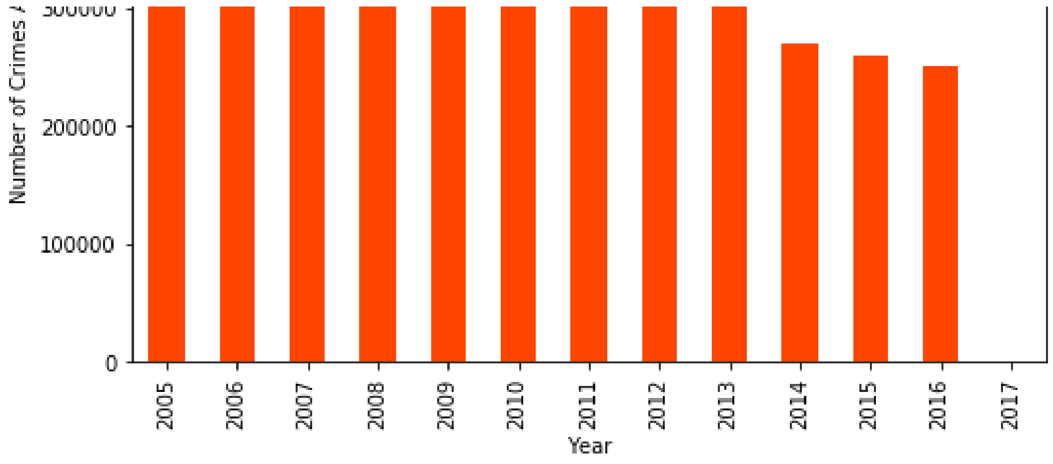
```
## EXPLORING DATA
print('Data Exploration')
```

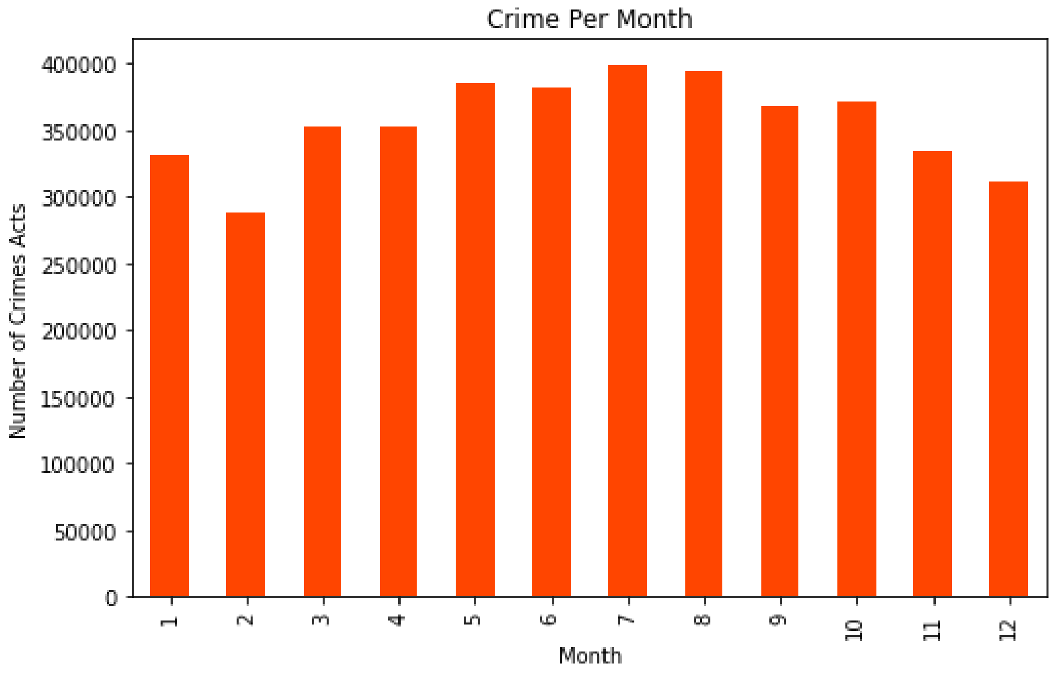
```
Data Exploration
```

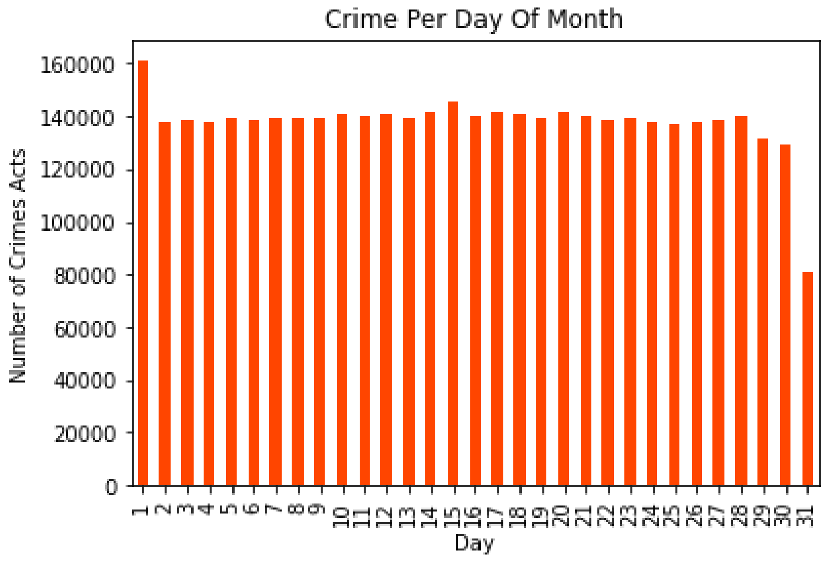
In [50]:

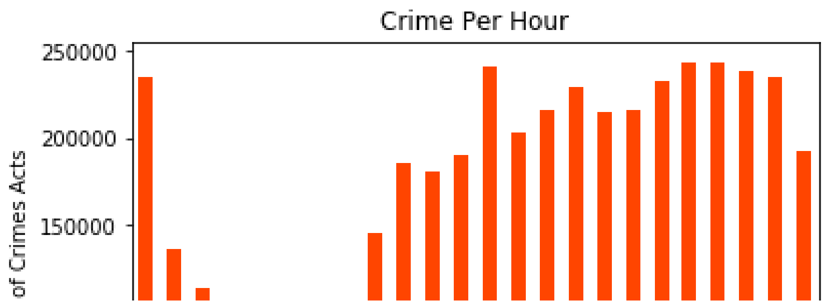
```
#make graph of crimes per year
plt.figure(figsize = (8,5))
Crime_Data.groupby([Crime_Data.index.year]).size().plot.bar(color = 'orange
red')
plt.title('Crime Over the Years')
plt.xlabel('Year')
plt.ylabel('Number of Crimes Acts')
plt.show()
plt.figure(figsize = (8,5))
#make graph of crimes per month
Crime_Data.groupby([Crime_Data.index.month]).size().plot.bar(color = 'orang
ered')
plt.title('Crime Per Month')
plt.xlabel('Month')
plt.ylabel('Number of Crimes Acts')
plt.show()
#graph of crimes per day
Crime_Data.groupby([Crime_Data.index.day]).size().plot.bar(color = 'oranger
ed')
plt.title('Crime Per Day Of Month')
plt.xlabel('Day')
plt.ylabel('Number of Crimes Acts')
plt.show()
#graph of crimes per hour
Crime_Data.groupby([Crime_Data.index.hour]).size().plot.bar(color = 'orange
red')
plt.title('Crime Per Hour')
plt.xlabel('Hour')
plt.ylabel('Number of Crimes Acts')
plt.show()
```

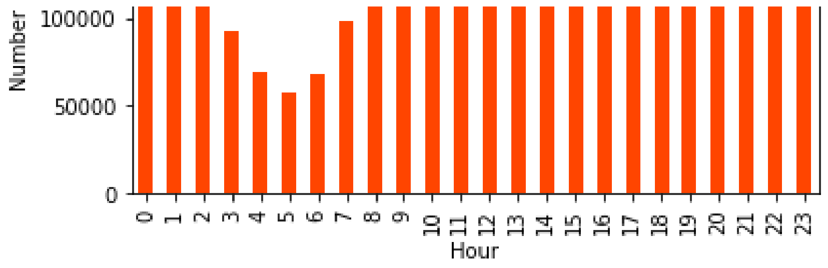



Number of Crimes A
300000
200000
100000
0
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
Year


Crime Per Month
Number of Crimes Acts
400000
350000
300000
250000
200000
150000
100000
50000
0
1
2
3
4
5
6
7
8
9
10
11
12
Month

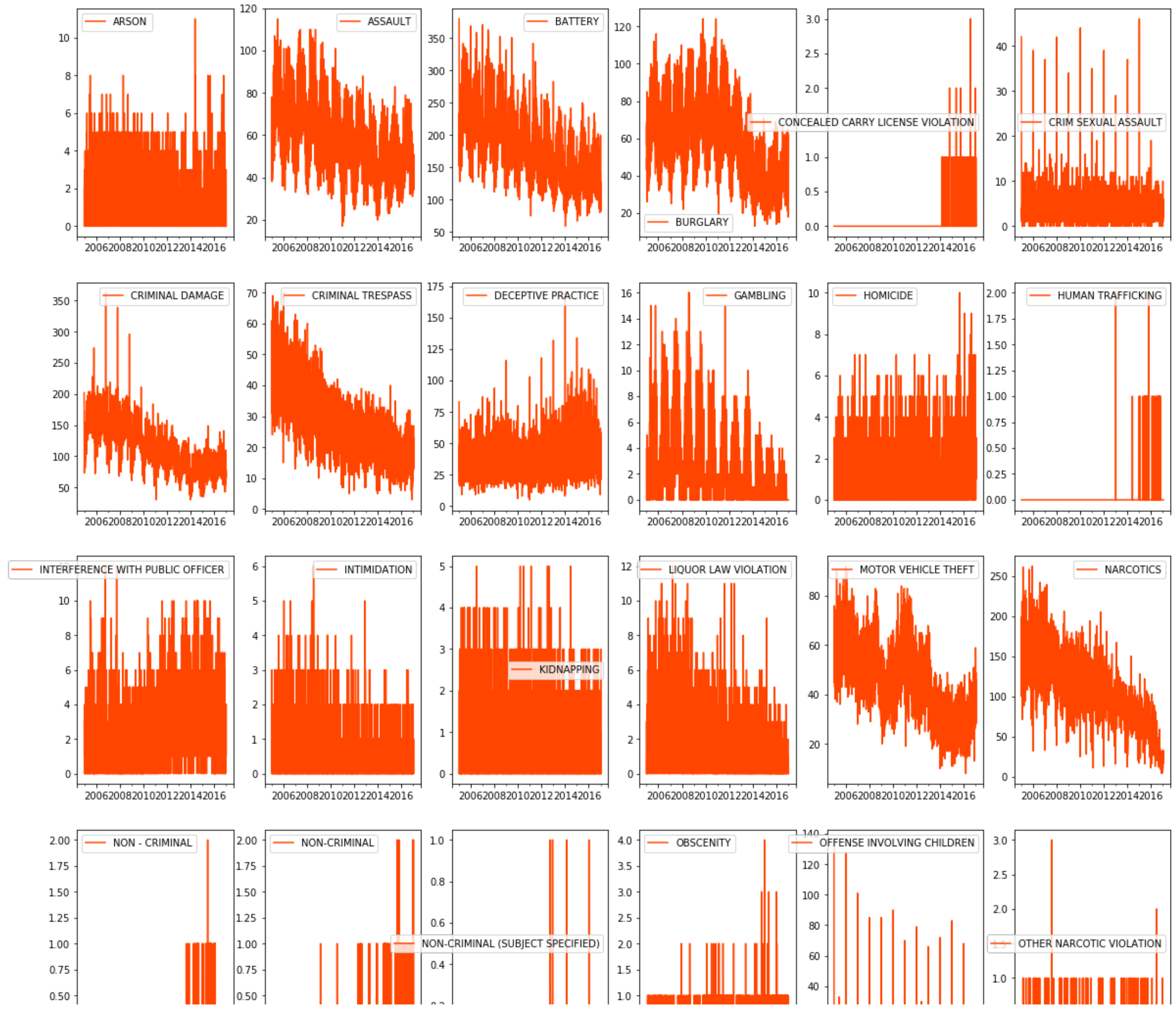

Crime Per Day Of Month
Number of Crimes Acts
160000
140000
120000
100000
80000
60000
40000
20000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Day

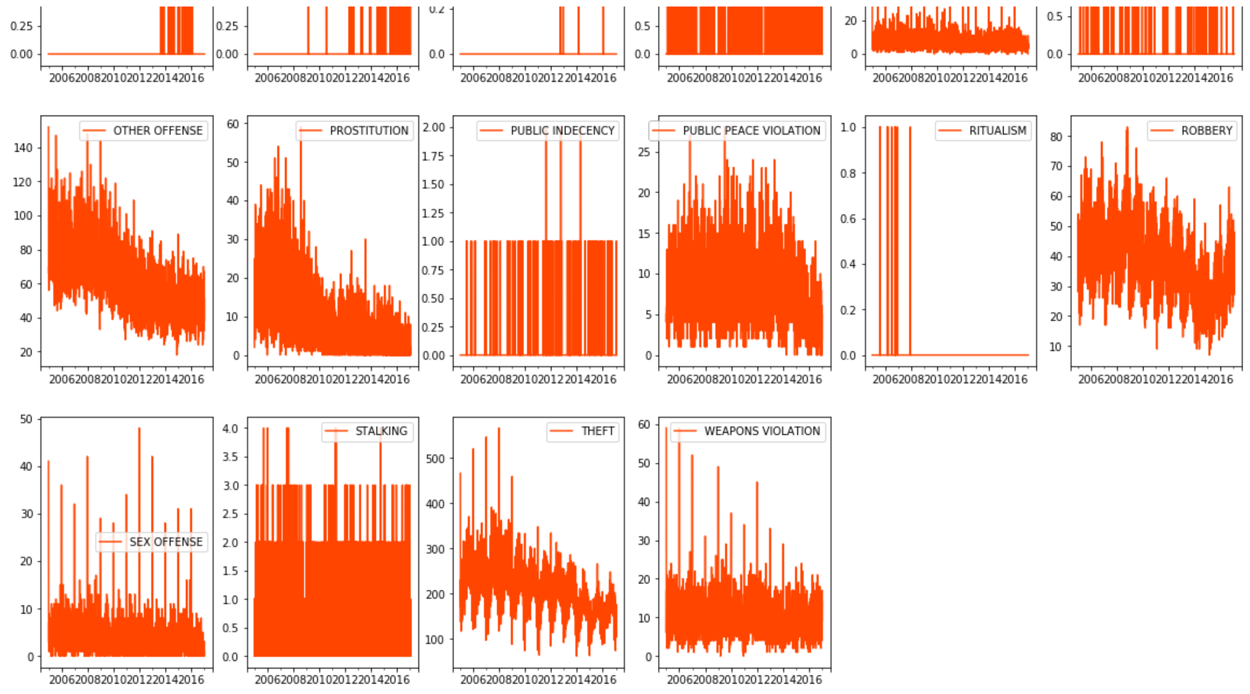

Crime Per Hour
of Crimes Acts
250000
200000
150000



In [14]:

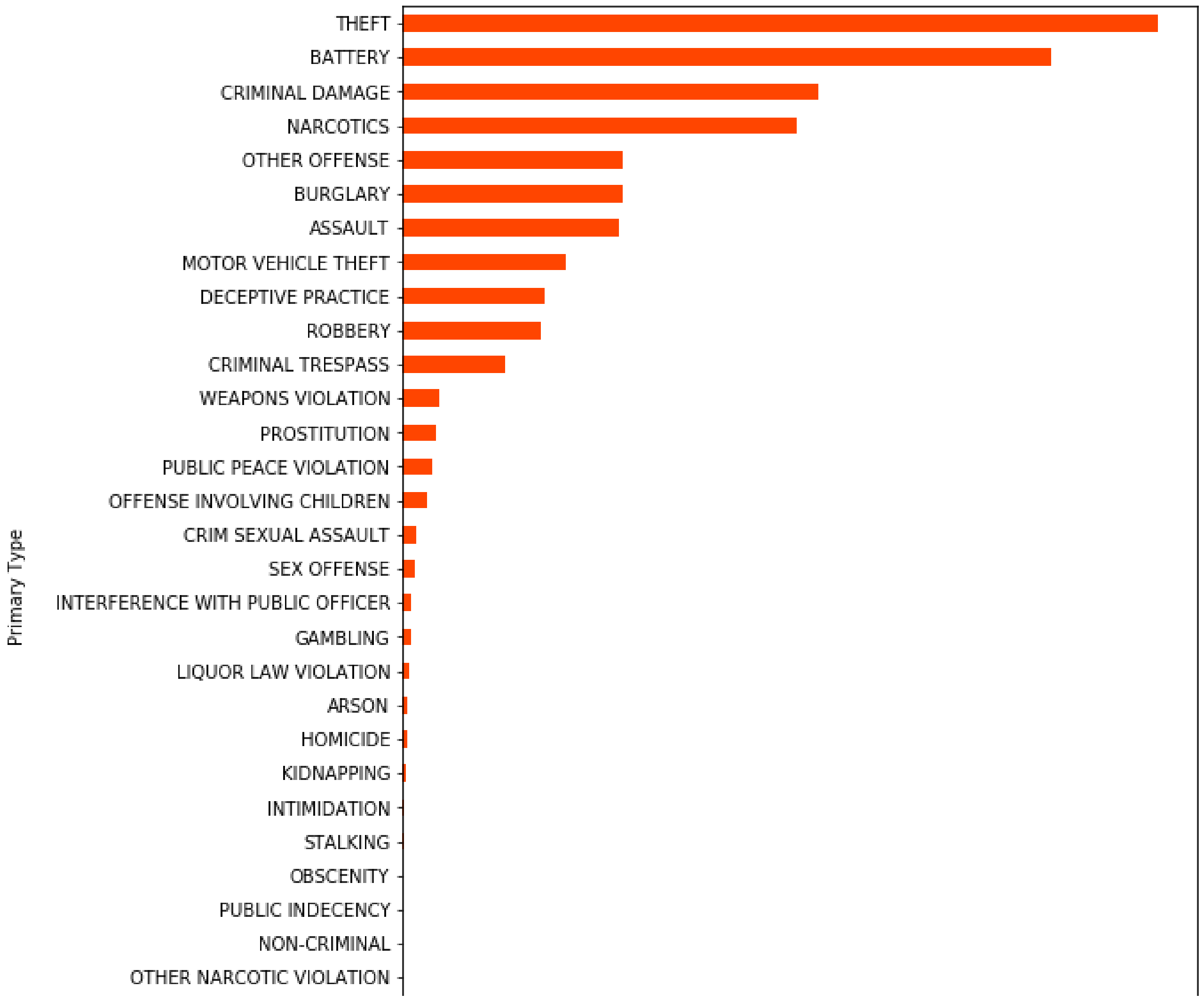
```
Crime_Data_date = Crime_Data.pivot_table('ID', aggfunc = np.size, columns =
'Primary Type',
                                         index = Crime_Data.index.date, fil
value = 0)
Crime_Data_date.index = pd.DatetimeIndex(Crime_Data_date.index)


# In[14]:


# visualize different types of crimes per month
Plot = Crime_Data_date.plot(figsize = (20,30), subplots = True, layout = (6
,6),
                                   sharex = False, sharey = False, color = 'c
angered')
plt.show(block=True)
```

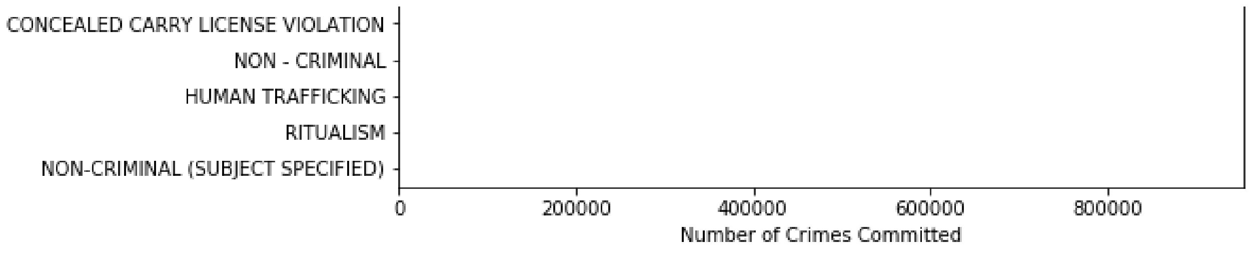




In [15]:

```
# visualize relative amounts of each type of crime
plt.figure(figsize = (8,12))
Crime_Data.groupby(Crime_Data['Primary Type']).size().sort_values(ascending
= True).plot(kind = 'barh', color = 'orangered')
plt.xlabel('Number of Crimes Committed')
plt.show(block=True)
```





In [16]:

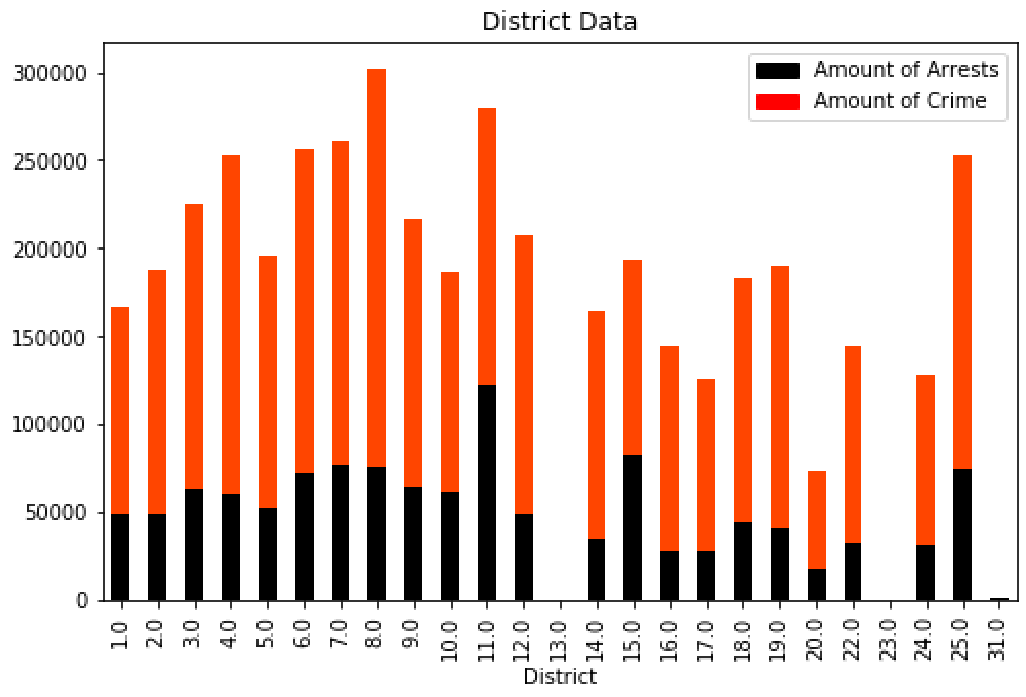
```
Crime_Data_district = Crime_Data.pivot_table('Arrest', aggfunc = np.sum, co
lumns = ['District'],
                                              index = Crime_Data.index.date, fil
value = 0)
Crime_Data_ward = Crime_Data.pivot_table('Arrest', aggfunc = np.sum,
columns = ['Ward'],
                                              index = Crime_Data.index.date, fil
value = 0)
Crime_Data_ca = Crime_Data.pivot_table('Arrest', aggfunc = np.sum, columns
= ['Community Area'],
                                              index = Crime_Data.index.date, fil
value = 0)
```

In [17]:

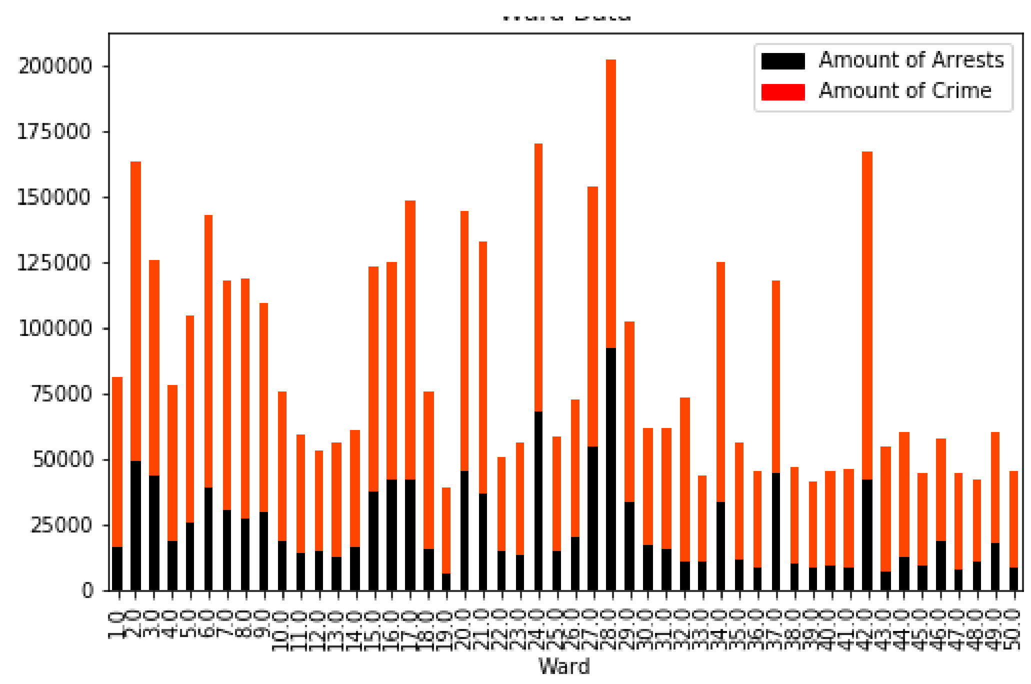
```
red = mpatches.Patch(color='black', label='Amount of Arrests')
blue = mpatches.Patch(color='red', label='Amount of Crime')

#crime per district
plt.figure(figsize = (8,5))
plt.hold(True)
Crime_Data.groupby(Crime_Data['District']).size().plot(kind = 'bar', color
= 'orangered')
Crime_Data_district.sum().plot(kind = 'bar', color = 'black')
plt.title('District Data')
plt.legend(handles=[red, blue])
plt.hold(False)
#crime per ward
plt.figure(figsize = (8,5))
plt.hold(True)
Crime_Data.groupby(Crime_Data['Ward']).size().plot(kind = 'bar', color = 'o
rangered')
Crime_Data_ward.sum().plot(kind = 'bar', color = 'black')
plt.title('Ward Data')
plt.legend(handles=[red, blue])
plt.hold(False)
#crime per Community Area
plt.figure(figsize = (12,5))
plt.hold(True)
Crime_Data.groupby(Crime_Data['Community Area']).size().plot(kind = 'bar',
color = 'orangered')
Crime_Data_ca.sum().plot(kind = 'bar', color = 'black')
plt.title('Community Area Data')
plt.legend(handles=[red, blue])
plt.hold(False)
plt.show(block=True)
```

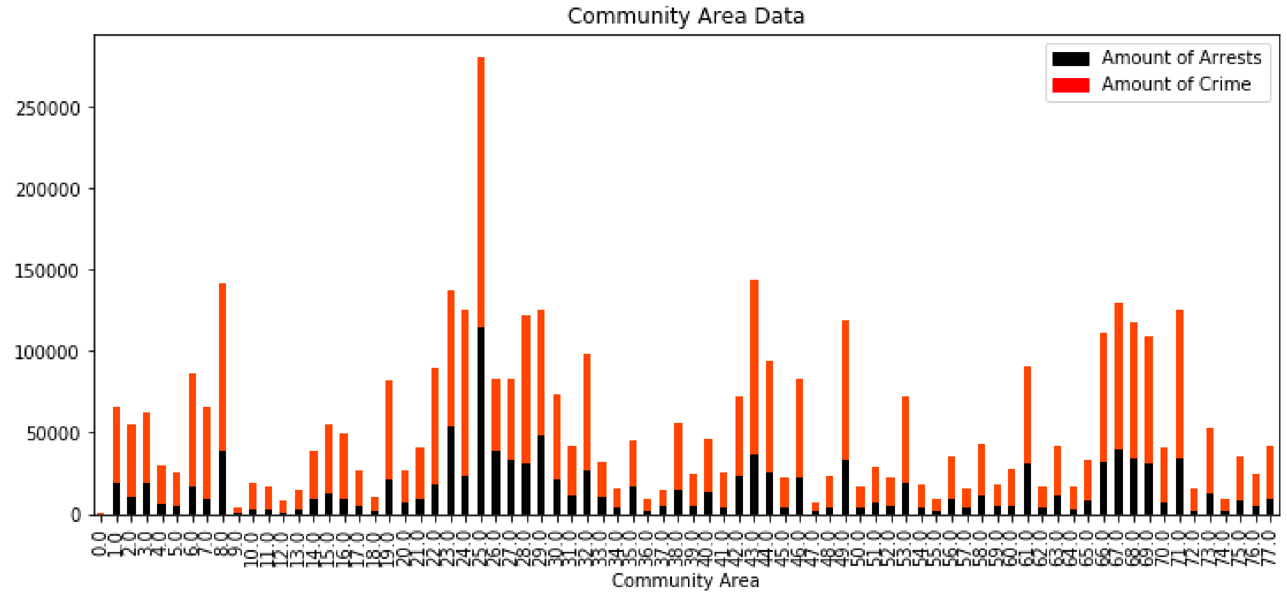
```
C:\Users\sanch\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: Matplot
libDeprecationWarning: pyplot.hold is deprecated.
```

```
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.

C:\Users\sanch\Anaconda3\lib\site-packages\matplotlib\__init__.py:911: Matp
lotlibDeprecationWarning: axes.hold is deprecated. Please remove it from yo
ur matplotlibrc and/or style files.
  mplDeprecation)
C:\Users\sanch\Anaconda3\lib\site-packages\matplotlib\rcsetup.py:156: Matpl
otlibDeprecationWarning: axes.hold is deprecated, will be removed in 3.0
  mplDeprecation)
C:\Users\sanch\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: Matplo
tlibDeprecationWarning: pyplot.hold is deprecated.
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.
  # This is added back by InteractiveShellApp.init_path()
C:\Users\sanch\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: Matplo
tlibDeprecationWarning: pyplot.hold is deprecated.
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.

C:\Users\sanch\Anaconda3\lib\site-packages\ipykernel_launcher.py:19: Matplo
tlibDeprecationWarning: pyplot.hold is deprecated.
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.
C:\Users\sanch\Anaconda3\lib\site-packages\ipykernel_launcher.py:22: Matplo
tlibDeprecationWarning: pyplot.hold is deprecated.
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.
C:\Users\sanch\Anaconda3\lib\site-packages\ipykernel_launcher.py:27: Matplo
tlibDeprecationWarning: pyplot.hold is deprecated.
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.
```



Ward Data

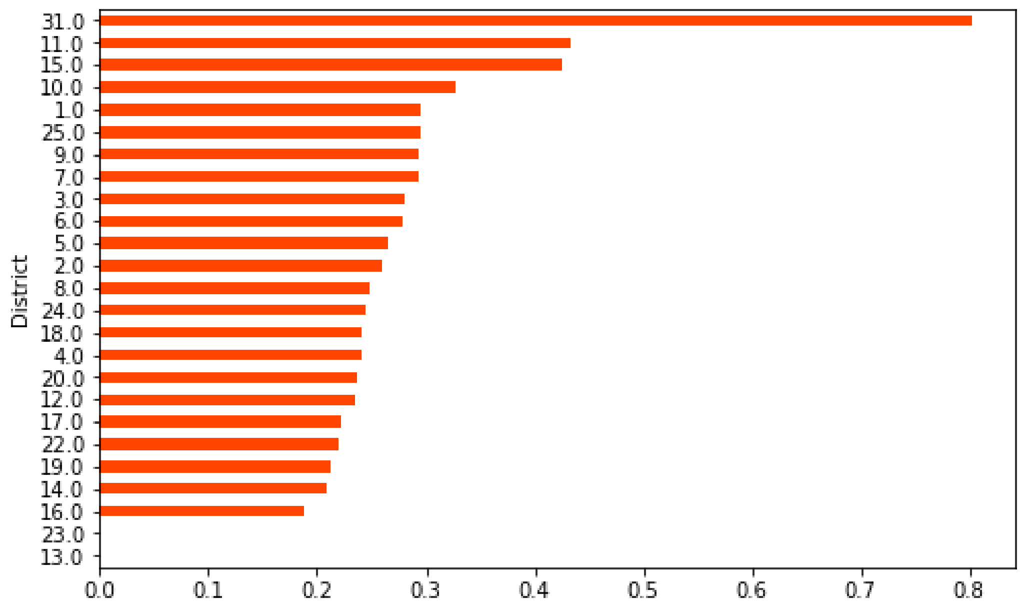




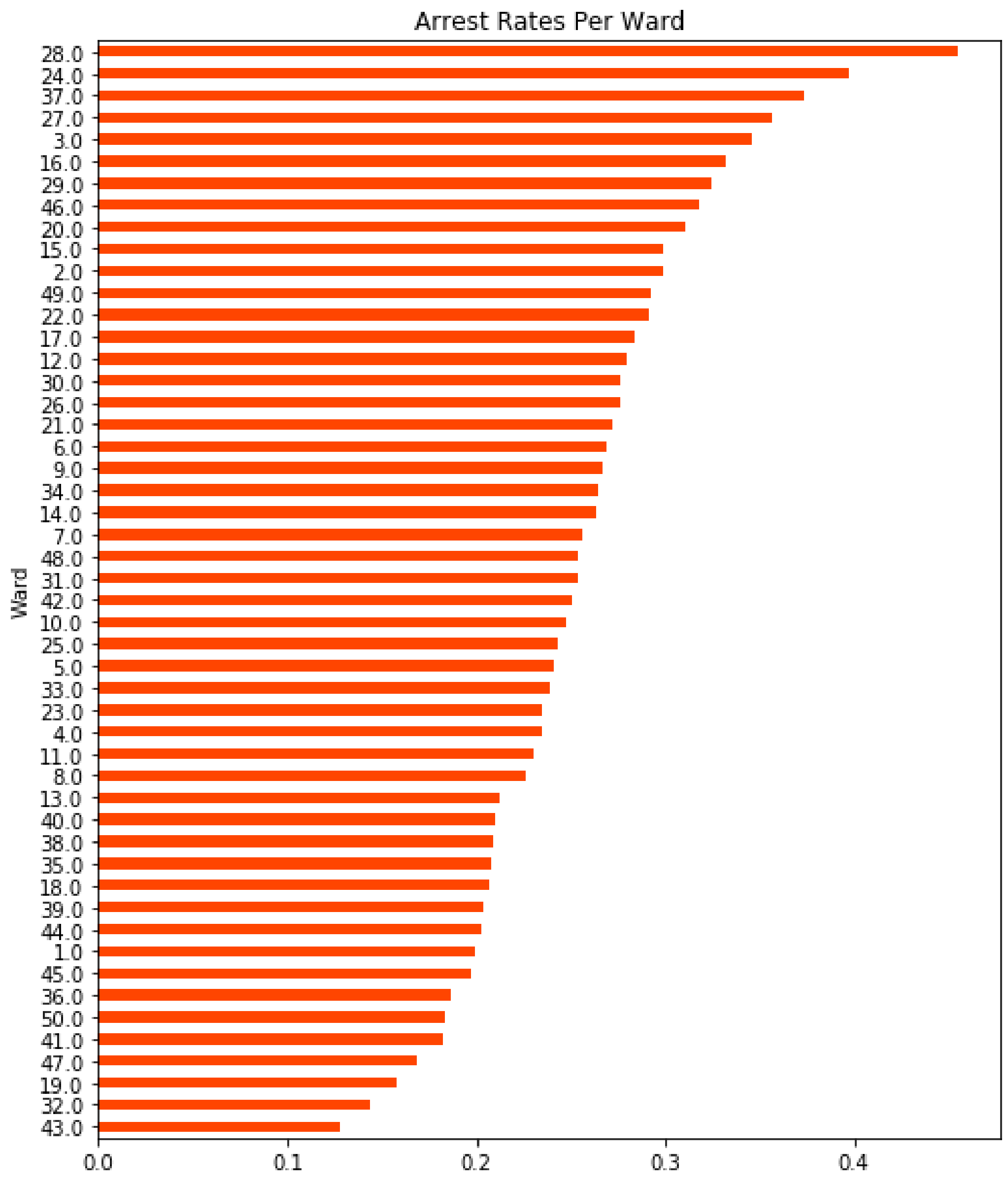
In [18]:

```
#Arrest rates per community area, ward, and district
A_R_D = Crime_Data_district.sum() /
Crime_Data.groupby(Crime_Data['District']).size()
A_R_W = Crime_Data_ward.sum() / Crime_Data.groupby(Crime_Data['Ward']).size
()
A_R_CA = Crime_Data_ca.sum() / Crime_Data.groupby(Crime_Data['Community Are
a']).size()

plt.figure(figsize = (8,5))
A_R_D.sort_values().plot(kind = 'barh', color = 'orangered')
plt.title('Arrest Rates Per District', color = 'k')
plt.figure(figsize = (8,10))
A_R_W.sort_values().plot(kind = 'barh', color = 'orangered')
plt.title('Arrest Rates Per Ward', color = 'k')
plt.figure(figsize=(8,12))
A_R_CA.sort_values().plot(kind = 'barh', color = 'orangered')
plt.title('Arrest Rates Per Community Area', color = 'k')
plt.show(block=True)
```

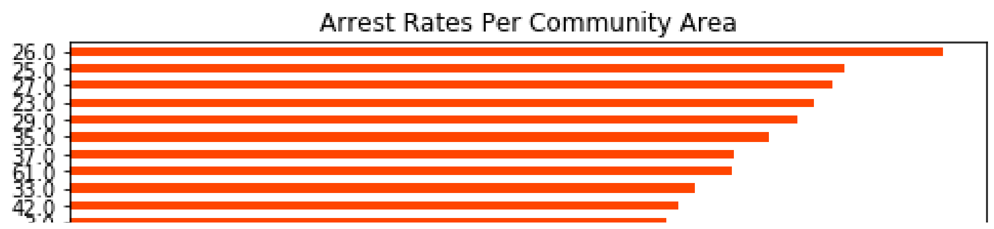
Arrest Rates Per District



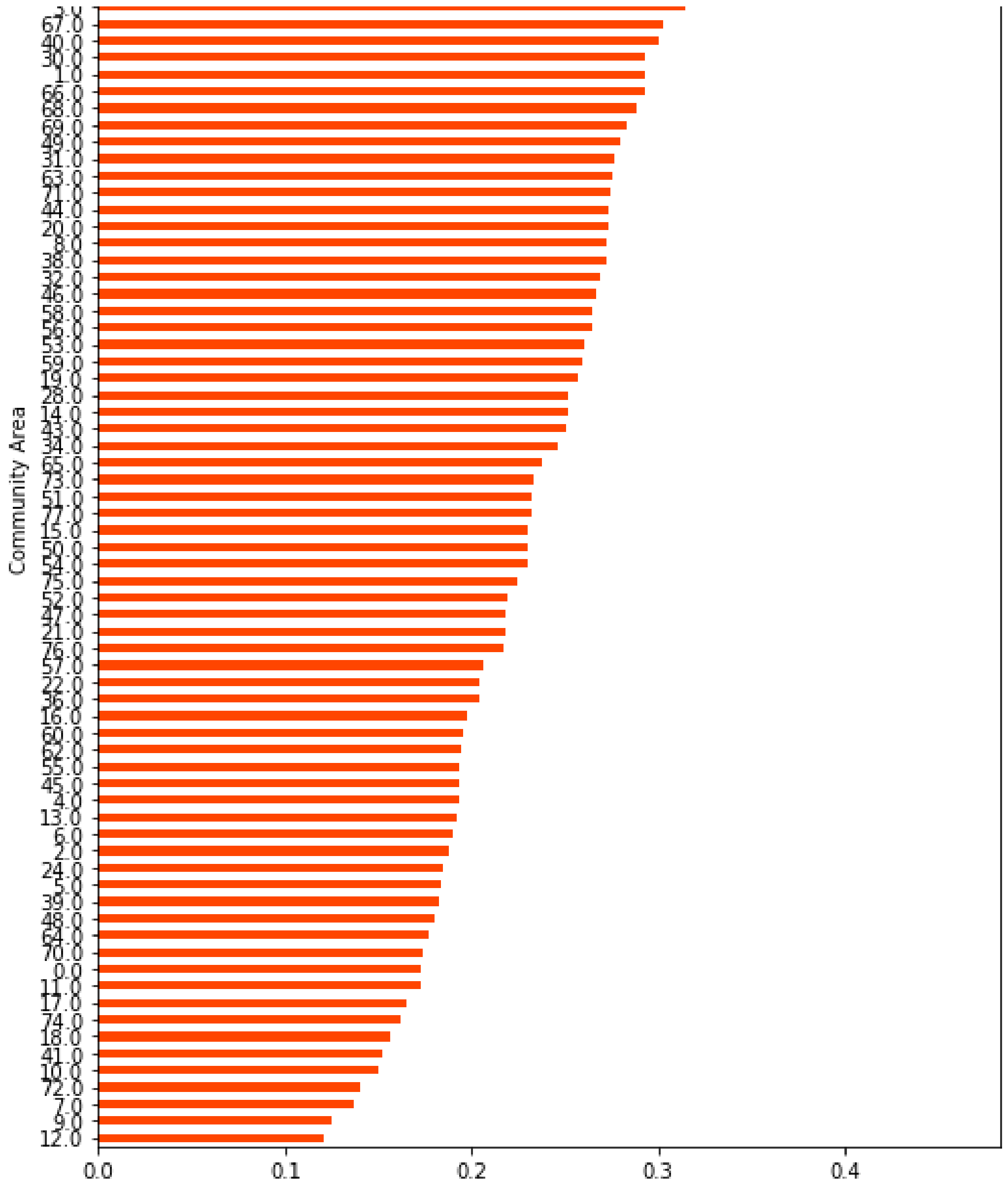
District
31.0
11.0
15.0
10.0
1.0
25.0
9.0
7.0
3.0
6.0
5.0
2.0
8.0
24.0
18.0
4.0
20.0
12.0
17.0
22.0
19.0
14.0
16.0
23.0
13.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8



Arrest Rates Per Ward
Ward
28.0
24.0
37.0
27.0
3.0
16.0
29.0
46.0
20.0
15.0
2.0
49.0
22.0
17.0
12.0
30.0
26.0
21.0
6.0
9.0
34.0
14.0
7.0
48.0
31.0
42.0
10.0
25.0
5.0
33.0
23.0
4.0
11.0
8.0
13.0
40.0
38.0
35.0
18.0
39.0
44.0
1.0
45.0
36.0
50.0
41.0
47.0
19.0
32.0
43.0
0.0
0.1
0.2
0.3
0.4



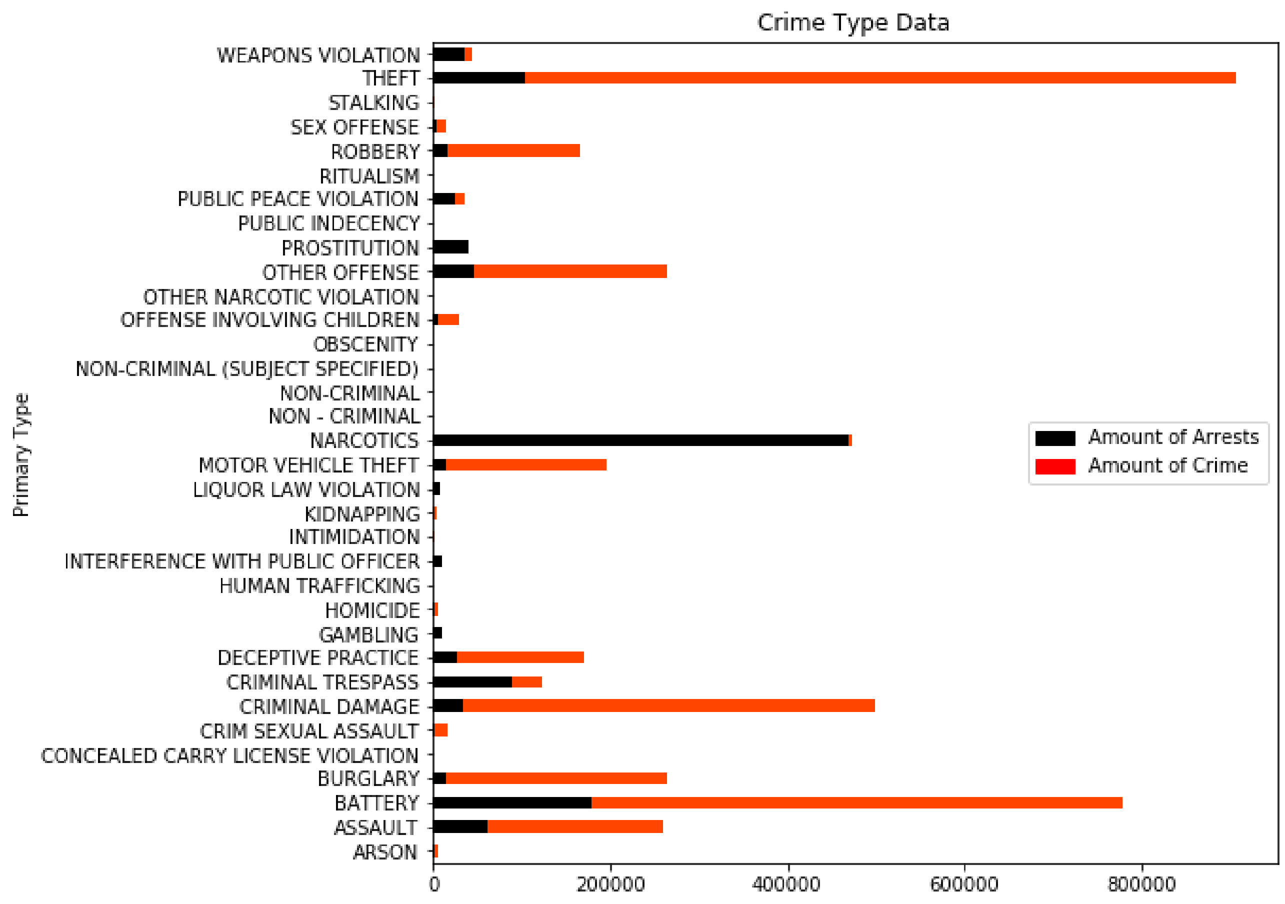
Arrest Rates Per Community Area
26.0
25.0
27.0
23.0
29.0
35.0
37.0
61.0
33.0
42.0

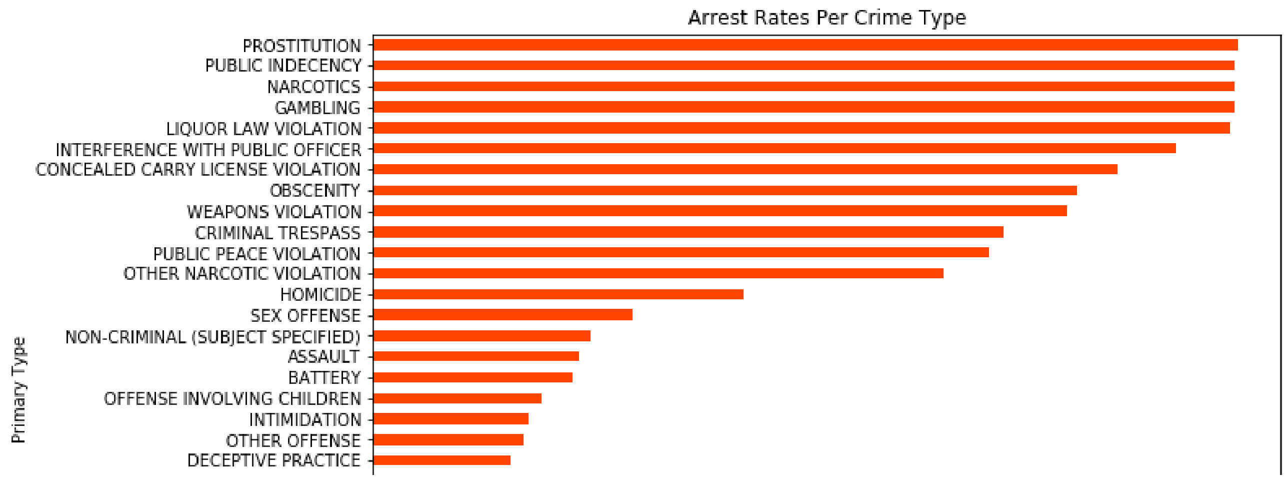


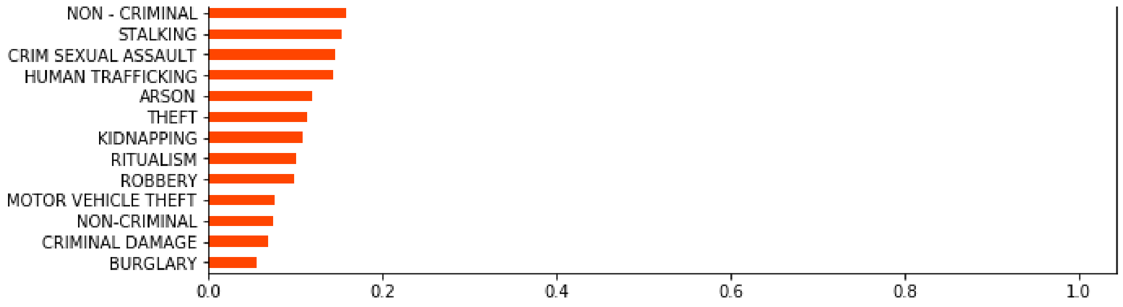
In [19]:

```
#Arrest rates per crime type
Crime_Data_Type = Crime_Data.pivot_table('Arrest', aggfunc = np.sum,
columns = ['Primary Type'],
                                        index = Crime_Data.index.date, fil
value = 0)
#crime per district
plt.figure(figsize = (8,8))
plt.hold(True)
Crime_Data.groupby(Crime_Data['Primary Type']).size().plot(kind = 'barh', c
olor='orangered')
Crime_Data_Type.sum().plot(kind = 'barh', color = 'k')
plt.title('Crime Type Data')
plt.legend(handles=[red, blue])
plt.hold(False)

plt.figure(figsize = (10,8))
A_R_PT = Crime_Data_Type.sum() / Crime_Data.groupby(Crime_Data['Primary Typ
e']).size()
A_R_PT.sort_values().plot(kind = 'barh', color = 'orangered')
plt.title('Arrest Rates Per Crime Type')
plt.show(block=True)
```

```
C:\Users\sanch\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: Matplot
libDeprecationWarning: pyplot.hold is deprecated.
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.

C:\Users\sanch\Anaconda3\lib\site-packages\matplotlib\__init__.py:911: Matp
lotlibDeprecationWarning: axes.hold is deprecated. Please remove it from yo
ur matplotlibrc and/or style files.
  mplDeprecation)
C:\Users\sanch\Anaconda3\lib\site-packages\matplotlib\rcsetup.py:156: Matpl
otlibDeprecationWarning: axes.hold is deprecated, will be removed in 3.0
  mplDeprecation)
C:\Users\sanch\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: Matplo
tlibDeprecationWarning: pyplot.hold is deprecated.
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.
  # This is added back by InteractiveShellApp.init_path()
```







In [20]:

```
print('--------- end of data exploration---------')
## PREPARING DATA
print('Data Preparing')
```

```
--------- end of data exploration---------
Data Preparing
```

In [21]:

```
#remove NAs from Longitude and Latitude data
Crime_Data = Crime_Data.dropna(axis = 0, how = 'any')

# drop bad data points

Crime_Data = Crime_Data[Crime_Data.Longitude != '-87.1:00:00 AM']

Arrest_Data = Crime_Data.drop('Arrest', axis = 1)
Arrest_Data = Arrest_Data.drop('Date', axis = 1)
Arrest_Data = Arrest_Data.drop('Block', axis = 1)
Arrest_Target = Crime_Data['Arrest']

Arrest_Data['Primary Type'] = (Arrest_Data['Primary Type']).cat.codes
Arrest_Data['Location Description'] = (Arrest_Data['Location Description'])
.cat.codes
Arrest_Data['Description'] = (Arrest_Data['Description']).cat.codes

names = []
names = list(Arrest_Data)
```

In [22]:

```
print(Arrest_Data.head())
print(Arrest_Target.head())
features = list(Arrest_Data.columns)
print(features)
```

```
                          ID  Primary Type  Description  Location
Description  \
Date

2006-04-02 13:00:00  4673626            24          173             1
2006-02-26 13:40:48  4673627            17          218             1
2006-01-08 23:16:00  4673628             1           40             1
2006-04-05 18:45:00  4673629             2          309             1
2006-02-17 21:03:14  4673630            17          267             
```

```
                         Domestic  Beat  District  Ward  Community Area  Year  `
Date
2006-04-02 13:00:00         False  1622      16.0  45.0            11.0  2006
2006-02-26 13:40:48         False   321       3.0  20.0            42.0  2006
2006-01-08 23:16:00         False   321       3.0   5.0            69.0  2006
2006-04-05 18:45:00         False  1633      16.0  38.0            17.0  2006
2006-02-17 21:03:14         False   822       8.0  13.0            65.0  2006

                          Latitude Longitude
Date
2006-04-02 13:00:00  41.981913   -87.772
2006-02-26 13:40:48  41.775733  -87.6119
2006-01-08 23:16:00  41.769897  -87.5937
2006-04-05 18:45:00  41.942984  -87.7801
2006-02-17 21:03:14  41.784211  -87.7167
Date
2006-04-02 13:00:00    False
2006-02-26 13:40:48     True
2006-01-08 23:16:00    False
2006-04-05 18:45:00    False
2006-02-17 21:03:14     True
Name: Arrest, dtype: bool
['ID', 'Primary Type', 'Description', 'Location Description', 'Domestic', '
Beat', 'District', 'Ward', 'Community Area', 'Year', 'Latitude', 'Longitude
']
```

In [23]:

```
## BUILDING MODELS
```

In [24]:

```
print ('Classification of Arrests')
```

```
Classification of Arrests
```

In [25]:

```
print ('-----------PCA--------------')
```

```
-----------PCA--------------
```

In [26]:

```
pca = PCA()
pca.fit(Arrest_Data[0:28])
PCAData = pca.fit_transform(Arrest_Data)
Arrest_Data = PCAData
print(Arrest_Data.shape)
print(pca.explained_variance_ratio_)
```

```
(4273758, 12)
[  9.99999876e-01   1.20182357e-07   3.01810372e-09   5.05481988e-10
   8.83901412e-11   3.31284311e-11   2.32182894e-11   1.08578709e-12
   1.06631638e-13   2.49923562e-14   8.04346946e-16   7.18341061e-16]
```

In [27]:

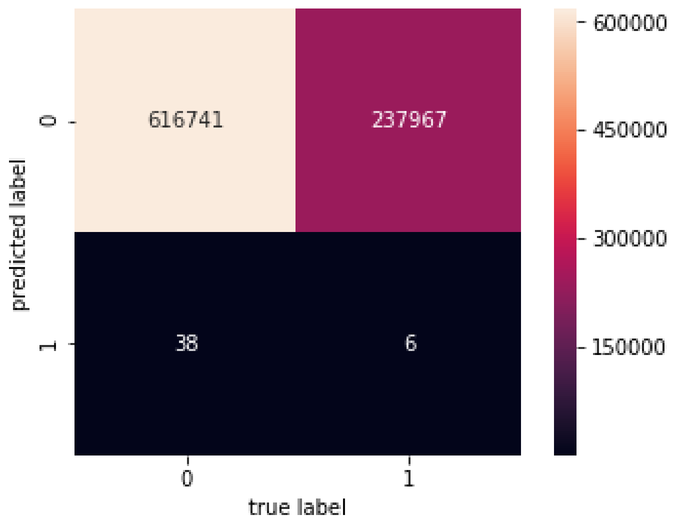
```
print ('-----------GLM Least Square Regression-----------------')
```

```
-----------GLM Least Square Regression-----------------
```

In [28]:

```
## GLM Least Squares

reg = linear_model.LinearRegression()
X_train, X_test, y_train, y_test = train_test_split(Arrest_Data,
Arrest_Target, test_size = 0.2)

reg.fit(X_train,y_train)
reg_pred = reg.predict(X_test)

# Converting probability of prediction to True or False
for i in range(len(reg_pred)):
    if (reg_pred[i] >= 0.5):
        reg_pred[i] = True
    else:
        reg_pred[i] = False

print(metrics.classification_report(reg_pred, y_test))

reg_mat = confusion_matrix(reg_pred, y_test)
sns.heatmap(reg_mat, square=True, annot=True, fmt='d', cbar=True)
plt.xlabel('true label')
plt.ylabel('predicted label')
print("Accuracy of Model is: %f"%accuracy_score(y_test, reg_pred))
```

```
             precision    recall  f1-score   support

        0.0       1.00      0.72      0.84    854708
        1.0       0.00      0.14      0.00        44

avg / total       1.00      0.72      0.84    854752

Accuracy of Model is: 0.721551
```



In [29]:

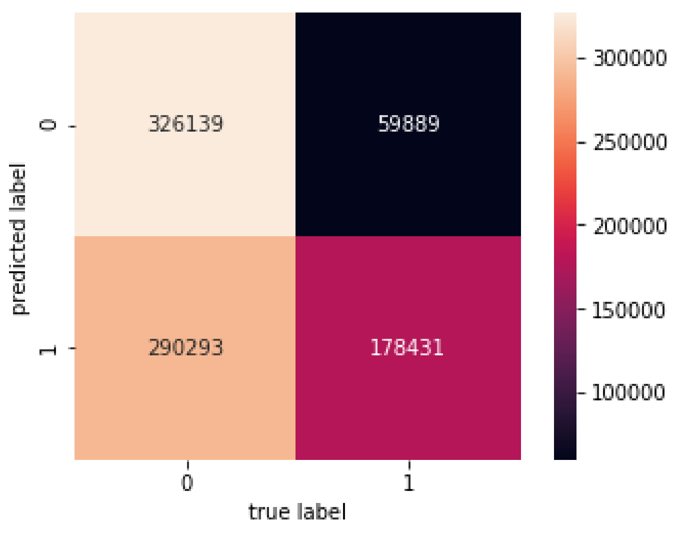
```
print ('-----------Logistic Regression-----------------')
```

```
-----------Logistic Regression-----------------
```

In [30]:

```
## Logistic Regression

logreg = LogisticRegression()
scores_logreg = cross_val_score(logreg, Arrest_Data, Arrest_Target)

print(scores_logreg)
print('Mean Cross Validation Accuracy for Logistic Regression: {}'.format(s
cores_logreg.mean()))

# split the dataset into train(80%) and test(20%)
X_train_log, X_test_log, y_train_log, y_test_log =
train_test_split(Arrest_Data, Arrest_Target, test_size = 0.2)
# model prediction and confusion matrix
logreg_fit = logreg.fit(X_train_log, y_train_log)
logreg_pred = logreg_fit.predict(X_test_log)
print(metrics.classification_report(logreg_pred, y_test_log))

logreg_mat = confusion_matrix(logreg_pred, y_test_log)
sns.heatmap(logreg_mat, square=True, annot=True, fmt='d', cbar=True)
plt.xlabel('true label')
plt.ylabel('predicted label')
print("Accuracy of Model is: %f"%accuracy_score(y_test, logreg_pred))
```

```
[ 0.28300202  0.59447727  0.33211146]
Mean Cross Validation Accuracy for Logistic Regression: 0.4031969168842588
             precision    recall  f1-score   support

      False       0.53      0.84      0.65    386028
       True       0.75      0.38      0.50    468724

avg / total       0.65      0.59      0.57    854752

Accuracy of Model is: 0.478898
```



In [31]:

```
print ('-----------Gaussian Naive Bayes -----------------')
```

```
-----------Gaussian Naive Bayes -----------------
```

In [32]:

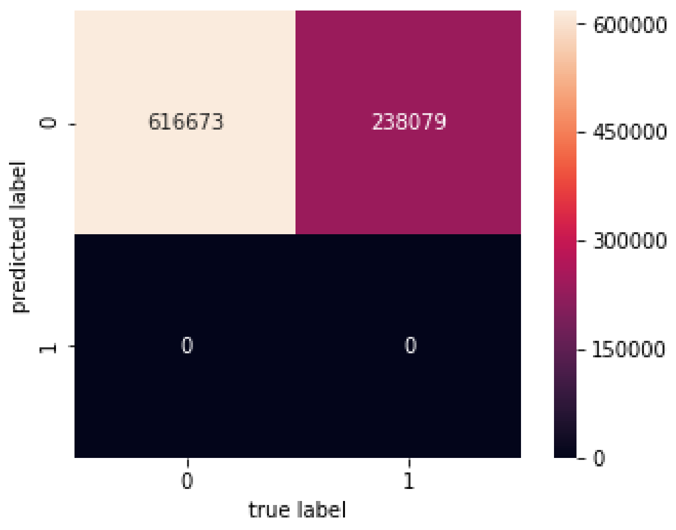
```
## Gaussian Naive Bayes
```

```
gnb = GaussianNB()
scores_gnb = cross_val_score(gnb, Arrest_Data, Arrest_Target)

print(scores_gnb)
print('Mean Cross Validation Accuracy for Gaussian Naive Bayes: {}'.format(
scores_gnb.mean()))

# split the dataset into train(80%) and test(20%)
X_train, X_test, y_train, y_test = train_test_split(Arrest_Data,
Arrest_Target, test_size = 0.2)
# model prediction and confusion matrix
gnb_fit = gnb.fit(X_train, y_train)
gnb_pred = gnb_fit.predict(X_test)
print(metrics.classification_report(y_test, gnb_pred))

gnb_mat = confusion_matrix(gnb_pred, y_test)
sns.heatmap(gnb_mat, square=True, annot=True, fmt='d', cbar=True)
plt.xlabel('true label')
plt.ylabel('predicted label')
print("Accuracy of Model is: %f"%accuracy_score(y_test, gnb_pred))
```

```
[ 0.61628528  0.72142573  0.72142624]
Mean Cross Validation Accuracy for Gaussian Naive Bayes: 0.6863790849039555
```

```
C:\Users\sanch\Anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1135: UndefinedMetricWarning: Pr
ecision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples.
  'precision', 'predicted', average, warn_for)
```

```
             precision    recall  f1-score   support

      False       0.72      1.00      0.84    616673
       True       0.00      0.00      0.00    238079

avg / total       0.52      0.72      0.60    854752

Accuracy of Model is: 0.721464
```



In [33]:

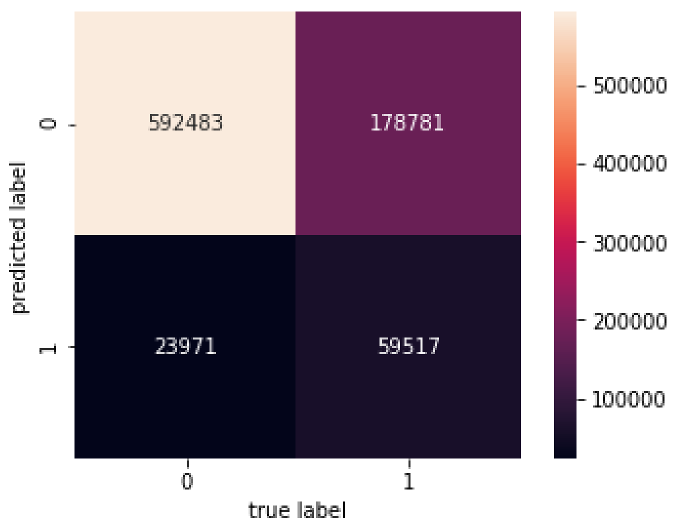
```
print ('-----------Bernoulli Naive Bayes -----------------')
```

```
-----------Bernoulli Naive Bayes -----------------
```

In [34]:

```
## Bernoulli Naive Bayes

bnb = BernoulliNB()
scores_bnb = cross_val_score(bnb, Arrest_Data, Arrest_Target)

print(scores_bnb)
print('Mean Cross Validation Accuracy for Bernoulli Naive Bayes: {}'.format
(scores_bnb.mean()))

# split the dataset into train(80%) and test(20%)
X_train, X_test, y_train, y_test = train_test_split(Arrest_Data,
Arrest_Target, test_size = 0.2)
# model prediction and confusion matrix
bnb_fit = bnb.fit(X_train, y_train)
bnb_pred = bnb_fit.predict(X_test)
print(metrics.classification_report(bnb_pred, y_test))

bnb_mat = confusion_matrix(bnb_pred, y_test)
sns.heatmap(bnb_mat, square=True, annot=True, fmt='d', cbar=True)
plt.xlabel('true label')
plt.ylabel('predicted label')
print("Accuracy of Model is: %f"%accuracy_score(y_test, bnb_pred))
```

```
[ 0.75910913  0.7614156   0.74496713]
Mean Cross Validation Accuracy for Bernoulli Naive Bayes:
0.7551639530965487
             precision    recall  f1-score   support

      False       0.96      0.77      0.85    771264
       True       0.25      0.71      0.37     83488

avg / total       0.89      0.76      0.81    854752

Accuracy of Model is: 0.762794
```



In [35]:

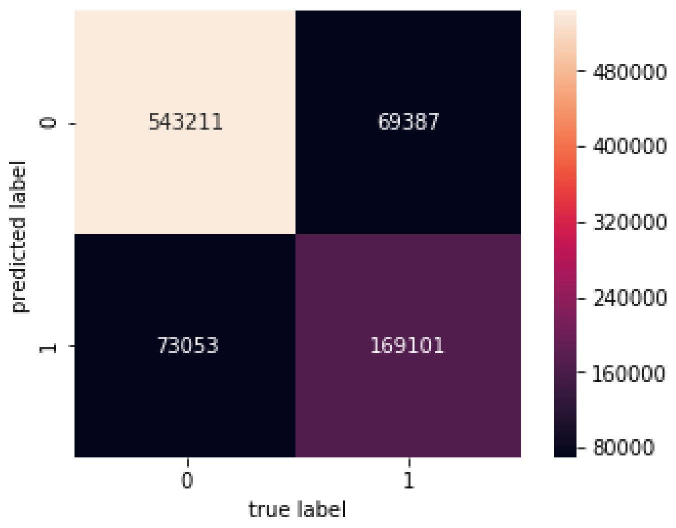
```
print ('-----------Decision Tree-----------------')
```

```
-----------Decision Tree-----------------
```

In [36]:

```
## Decision Tree

tree_entropy = DecisionTreeClassifier(criterion="entropy")
scores_dt = cross_val_score(tree_entropy, Arrest_Data, Arrest_Target)

print('Mean Cross Validation Accuracy for Decision Tree: {}'.format(scores_
dt.mean()))

# split the dataset into train(80%) and test(20%)
X_train, X_test, y_train, y_test = train_test_split(Arrest_Data,
Arrest_Target, test_size = 0.2)
# model prediction and confusion matrix
dt_fit = tree_entropy.fit(X_train, y_train)
dt_pred = dt_fit.predict(X_test)
print(metrics.classification_report(dt_pred, y_test))

bnb_mat = confusion_matrix(dt_pred, y_test)
sns.heatmap(bnb_mat, square=True, annot=True, fmt='d', cbar=True)
plt.xlabel('true label')
plt.ylabel('predicted label')
print("Accuracy of Model is: %f"%accuracy_score(y_test, dt_pred))
```

```
Mean Cross Validation Accuracy for Decision Tree: 0.5532466860100826
             precision    recall  f1-score   support

      False       0.88      0.89      0.88    612598
       True       0.71      0.70      0.70    242154

avg / total       0.83      0.83      0.83    854752

Accuracy of Model is: 0.833355
```



In [37]:

```
tree_entropy.get_params()

# # # Prune Decision Tree

# tree_entropy.set_params(max_depth=4)
```

Out[37]:

```
{'class_weight': None,
 'criterion': 'entropy',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'presort': False,
 'random_state': None,
 'splitter': 'best'}
```

In [38]:

```
# # Visualize model

# dot_data = export_graphviz(tree_entropy, feature_names = names, out_file=
None, filled=True, rounded=True)
# graph = pydotplus.graph_from_dot_data(dot_data)
```

In [39]:

```
print ('-----------Random Forest-----------------')
```

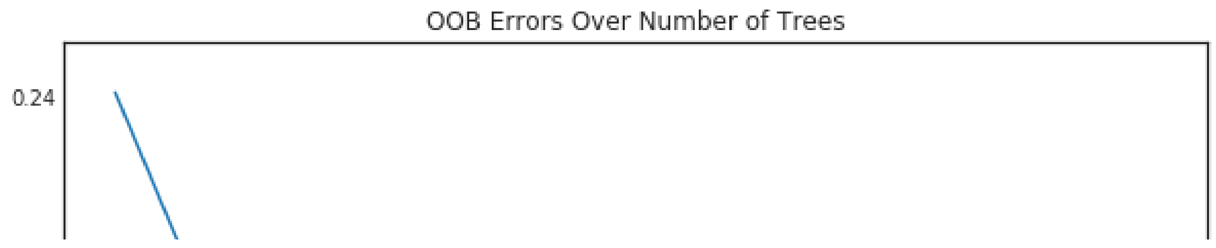
```
-----------Random Forest-----------------
```

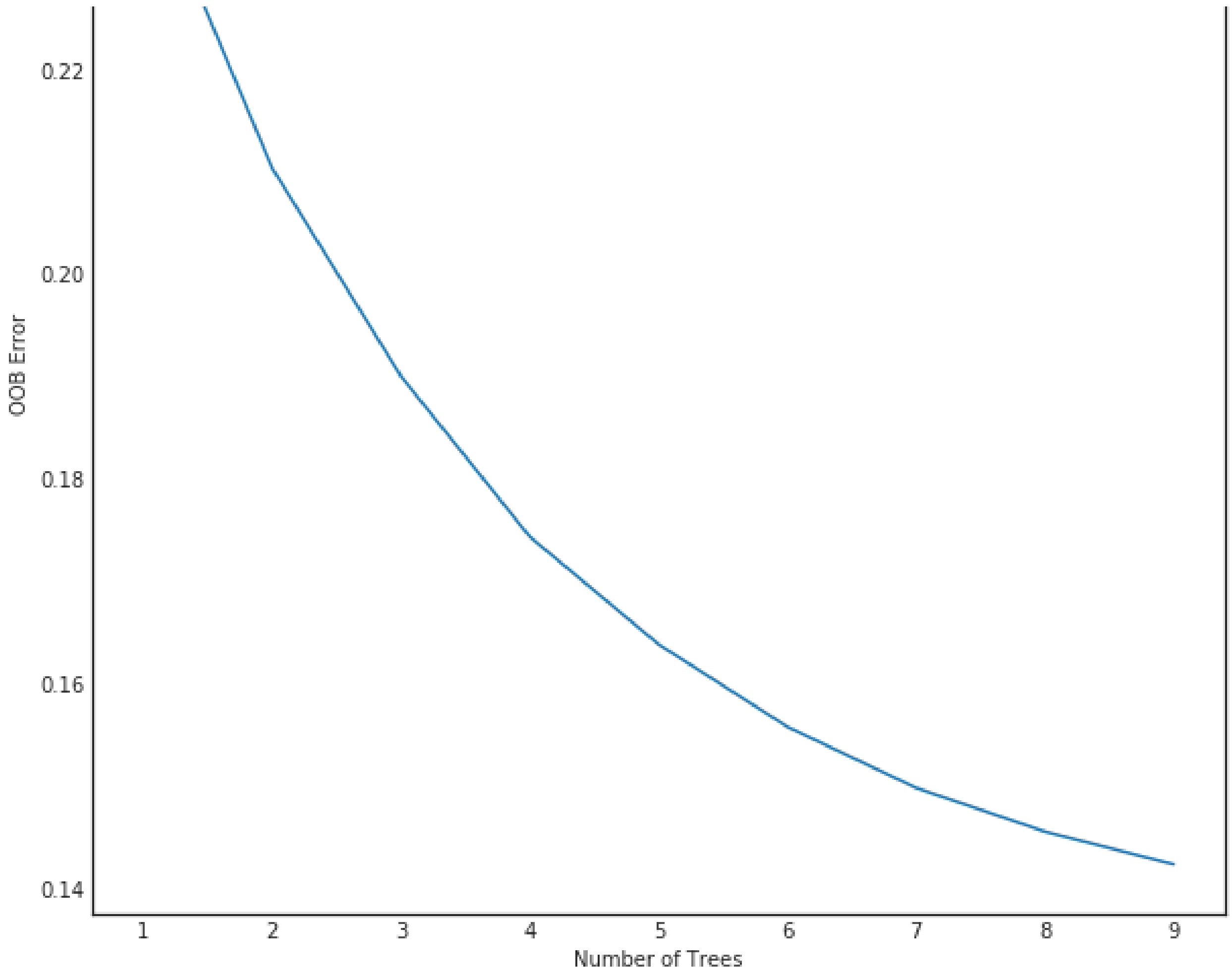
In [40]:

```
## Random Forest

# split the dataset into train(80%) and test(20%)
X_train, X_test, y_train, y_test = train_test_split(Arrest_Data,
Arrest_Target, test_size = 0.2)

OOB_Err = list(range(1,10))
for i in range(1,10):
    rfc = RandomForestClassifier(n_estimators = i, oob_score = True, n_jobs
= -1)
    rfc.fit(X_train,y_train)
    OOB_Err[i-1] = 1 - rfc.oob_score_

#plotting OOB Scores
plt.figure(figsize = (10,10))
with sns.axes_style("white"):
    plt.plot(list(range(1,10)), OOB_Err)
plt.title('OOB Errors Over Number of Trees')
plt.xlabel('Number of Trees')
plt.ylabel('OOB Error')
plt.show()
```

```
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:453:
UserWarning: Some inputs do not have OOB scores. This probably means too fe
w trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:458:
RuntimeWarning: invalid value encountered in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:453:
UserWarning: Some inputs do not have OOB scores. This probably means too fe
w trees were used to compute any reliable oob estimates.
```

```
  warn("Some inputs do not have OOB scores. "
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:458:
RuntimeWarning: invalid value encountered in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:453:
UserWarning: Some inputs do not have OOB scores. This probably means too fe
w trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:458:
RuntimeWarning: invalid value encountered in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:453:
UserWarning: Some inputs do not have OOB scores. This probably means too fe
w trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:458:
RuntimeWarning: invalid value encountered in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:453:
UserWarning: Some inputs do not have OOB scores. This probably means too fe
w trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:458:
RuntimeWarning: invalid value encountered in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:453:
UserWarning: Some inputs do not have OOB scores. This probably means too fe
w trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:458:
RuntimeWarning: invalid value encountered in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:453:
UserWarning: Some inputs do not have OOB scores. This probably means too fe
w trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:458:
RuntimeWarning: invalid value encountered in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:453:
UserWarning: Some inputs do not have OOB scores. This probably means too fe
w trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:458:
RuntimeWarning: invalid value encountered in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:453:
UserWarning: Some inputs do not have OOB scores. This probably means too fe
w trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\sanch\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:458:
RuntimeWarning: invalid value encountered in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
```

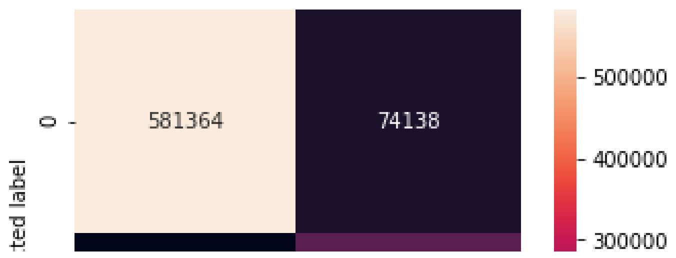


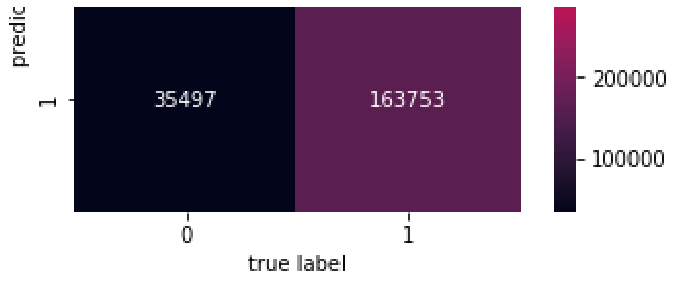


In [41]:

```
# Nine number of trees are found to be the lowest error rate
rforest = RandomForestClassifier(n_estimators = 9, n_jobs = -1)
rforest.fit(X_train, y_train)
rforest_pred = rforest.predict(X_test)

print(metrics.classification_report(rforest_pred, y_test))

rf_mat = confusion_matrix(rforest_pred, y_test)
sns.heatmap(rf_mat, square=True, annot=True, fmt='d', cbar=True)
plt.xlabel('true label')
plt.ylabel('predicted label')
print("Accuracy of Model is: %f"%accuracy_score(y_test, rforest_pred))
```

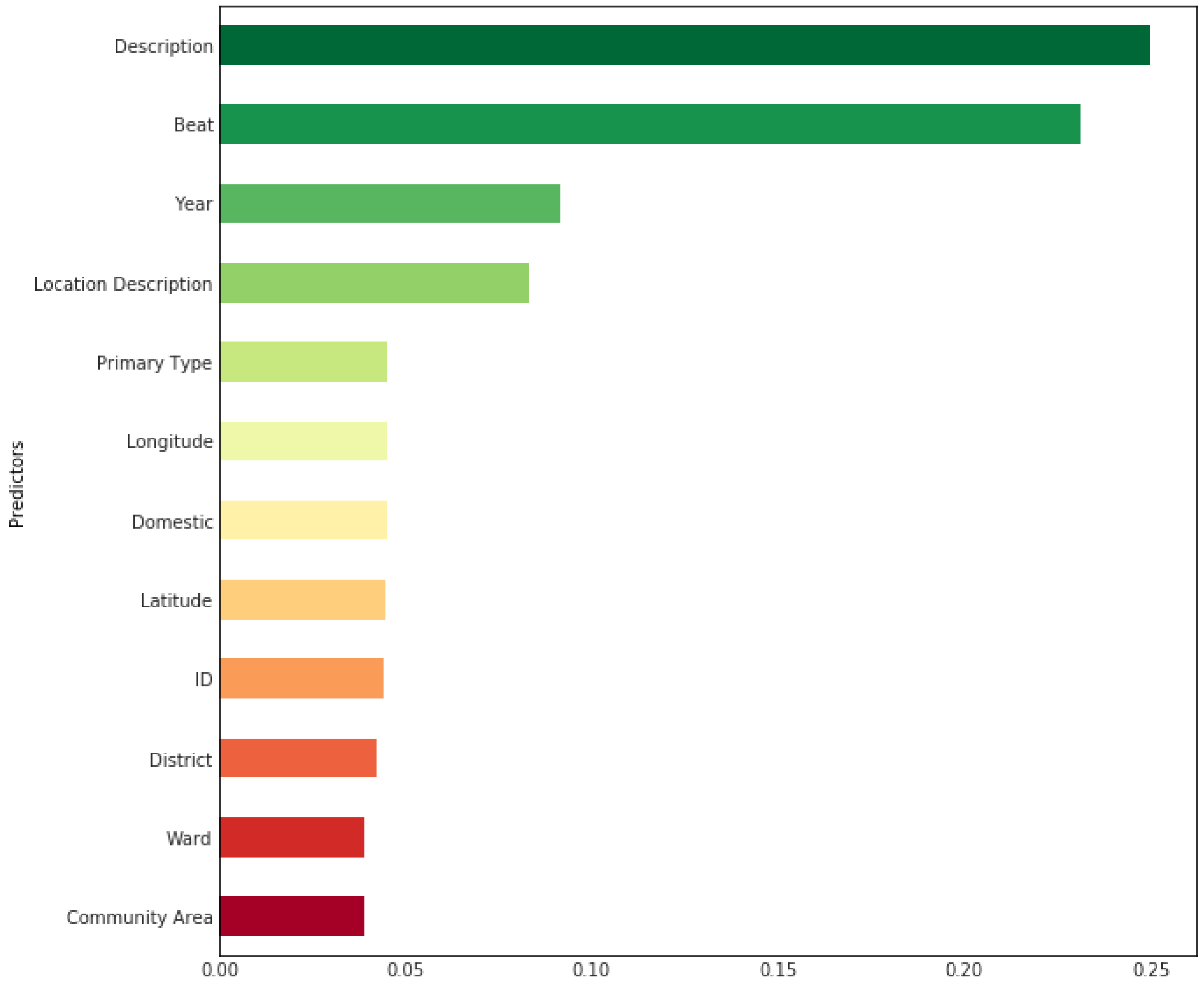
```
             precision    recall  f1-score   support

      False       0.94      0.89      0.91    655502
       True       0.69      0.82      0.75    199250

avg / total       0.88      0.87      0.88    854752

Accuracy of Model is: 0.871735
```





In [42]:

```
#Find most important variables in determining arrest rates using OOB

Col_Imp =[]
Col_Imp.append(features)
Col_Imp.append(list(rforest.feature_importances_))
Col_Imp = list(map(list, zip(*Col_Imp)))
Col_Imp = pd.DataFrame(Col_Imp, columns = ['Predictors','Feature
Importances'])

#plot feature importance
Col_Imp.index = Col_Imp['Predictors']
colors = plt.cm.RdYlGn(np.linspace(0,1,len(Col_Imp)))
plt.title('Feature Importances of Each Predictor')
plt.xlabel('Importance')
with sns.axes_style("white"):
    Col_Imp['Feature Importances'].sort_values().plot(figsize = (10,10), ki
nd = 'barh', color = colors)
plt.show()
```



In [43]:

```
# Unused Model Code for Future Analysis given more training time /
computing resources
```

In [44]:

```
# ## Optimize Dection Tree

# # tuning hyperparameters for decision tree using cross-validation
# best_score_tree = 0
# for i in range(1, 5):
#     for j in range(2, 10):
#         tree = DecisionTreeClassifier(criterion="entropy", max_depth = i,
max_leaf_nodes = j)
#         fold_accuracies_tree = cross_val_score(tree, X_train, y_train)
#         score_tree = fold_accuracies_tree.mean()
#         if score_tree > best_score_tree:
#             best_param_tree = {'max_depth' : i, 'max_leaf_nodes' : j}
#             best_score_tree = score_tree

# tree_opt = SVC(**best_param_tree)
# tree_opt.fit(X_trainval, y_trainval)
# test_score_tree = tree_opt.score(X_test, y_test)
# print("Best Score on validation set: {:.2f}".format(best_score_tree))
# print("Best parameters: {}".format(best_param_tree))
# print("Test set score: {:.2f}".format(test_score_tree))
```

In [45]:

```
# ## SVC

# #standardize data
# ss = StandardScaler()
# Arrest_Data_scaled = ss.fit_transform(Arrest_Data)

# svc = LinearSVC()
# scores_svc = cross_val_score(svc, Arrest_Data_scaled, Arrest_Target)

# print('Mean Cross Validation Accuracy for SVC:
{}'.format(scores_svc.mean()))

# X_trainval, X_test, y_trainval, y_test = train_test_split(Arrest_Data, Ar
rest_Target)

# # tuning hyperparameters for svc using cross-validation
# best_score_svc = 0
# for opt_c in [0.001, 0.01, 0.1, 1, 10, 100]:
#     svm = SVC(C = opt_c)
#     fold_accuracies_svc = cross_val_score(svm, X_trainval, y_trainval)
#     score_svc = fold_accuracies_svc.mean()
#     if score_svc > best_score_svc:
#         best_param_svc = {'C': opt_c}
#         best_score_svc = score_svc

# svm_opt = SVC(**best_param_svc)
# svm_opt.fit(X_trainval, y_trainval)
# test_score_svc = svm_opt.score(X_test, y_test)
# print("Best Score on validation set: {:.2f}".format(best_score_svc))
# print("Best parameters: {}".format(best_param_svc))
# print("Test set score: {:.2f}".format(test_score_svc))
```

In [46]:

```
# ## KNN

# #normalize data
# mms = MinMaxScaler()
# Arrest_Data_norm = mms.fit_transform(Arrest_Data)

# knn = KNeighborsClassifier(n_neighbors=1)
# scores_knn = cross_val_score(knn, Arrest_Data_norm, Arrest_Target)

# print('Mean Cross Validation Accuracy for KNN:
{}'.format(scores_knn.mean()))

# ##Optimizing KNN

# # split the dataset into train(70%) and test(30%)
# X_train, X_test, y_train, y_test = train_test_split(Arrest_Data,
Arrest_Target, test_size = 0.3)

# #normalize data
# mms = MinMaxScaler()
# X_train_norm = mms.fit_transform(X_train)
# X_test_norm = mms.fit_transform(X_test)

# # visualizing and optimizing hyperparameters for knn classifier using tra
in and test sample
# training_accuracy = []
# test_accuracy = []
# neighbors= [1,2,3]

# for opt_k in neighbors:
#     # Build the model
#     knn_clf = KNeighborsClassifier(n_neighbors = opt_k)
#     knn_clf.fit(X_train_norm, y_train)

#     # Record training set accuracy
#     trainAccuracy = knn_clf.score(X_train_norm, y_train)
#     training_accuracy.append(trainAccuracy)

#     # Record test set accuracy
#     testAccuracy = knn_clf.score(X_test_norm, y_test)
#     test_accuracy.append(testAccuracy)

# # Visualize train and test accuracy
# plt.plot(neighbors, training_accuracy, label = "Training Accuracy")
# plt.plot(neighbors, test_accuracy, label = "Test Accuracy")
# plt.ylabel("Accuracy")
# plt.xlabel("Number of neighbors")
# plt.legend()
```