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.getcwd())));
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', 'Chicago_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
```

```
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 E 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'])
```

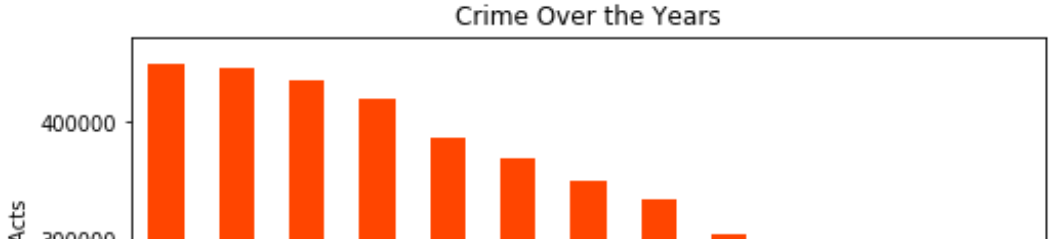
In [12]:

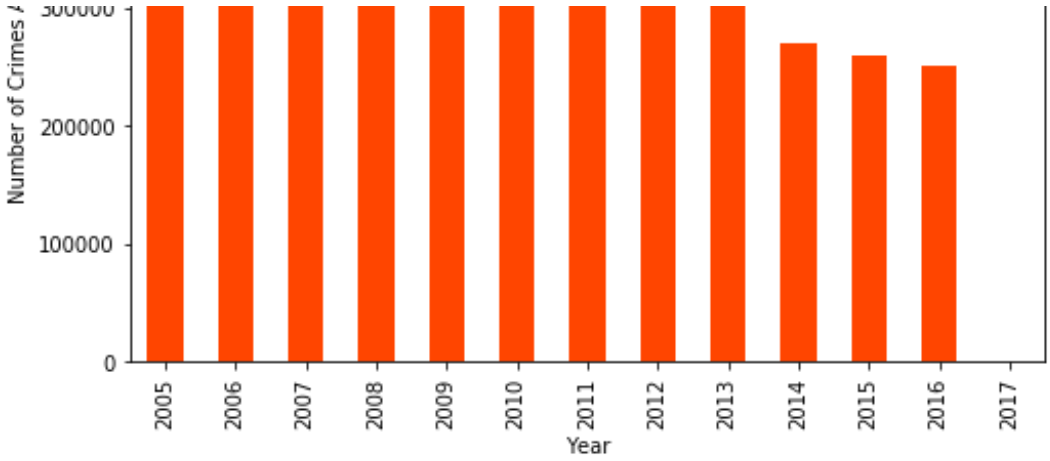
```
## EXPLORING DATA
print('Data Exploration')
```

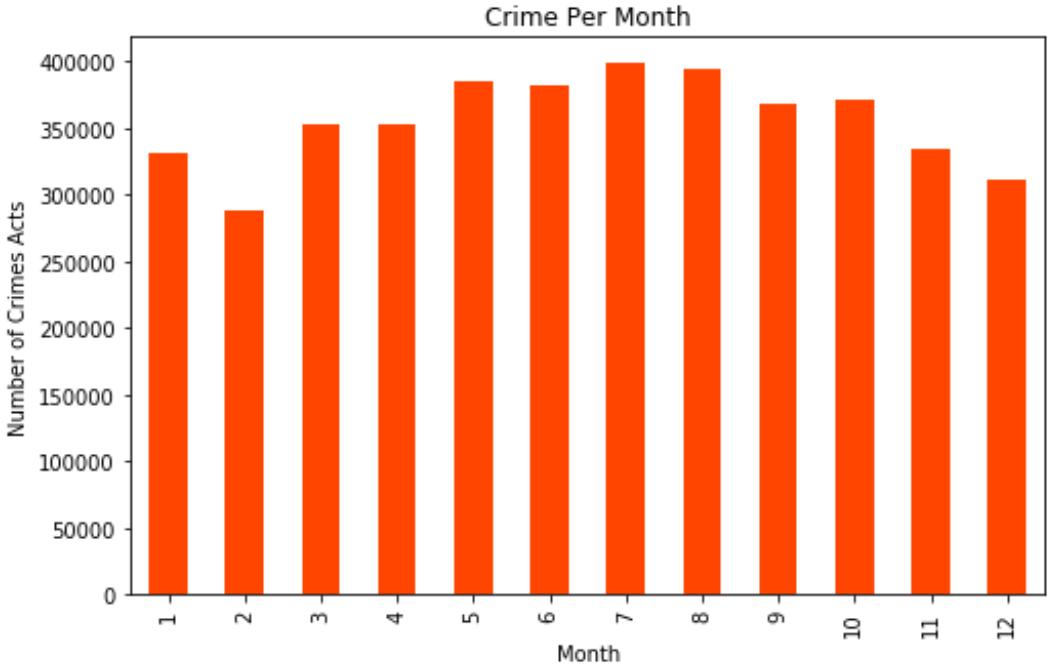
```
Data Exploration
```

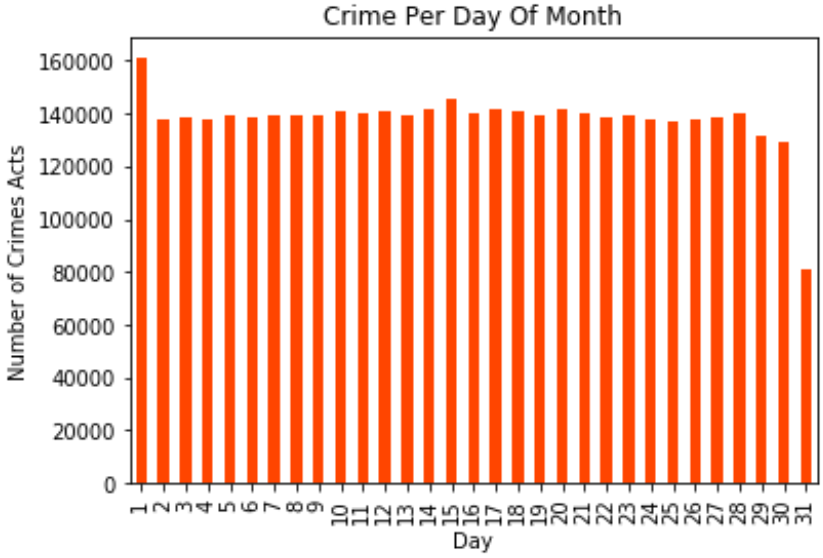
In [50]:

```
#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()
```

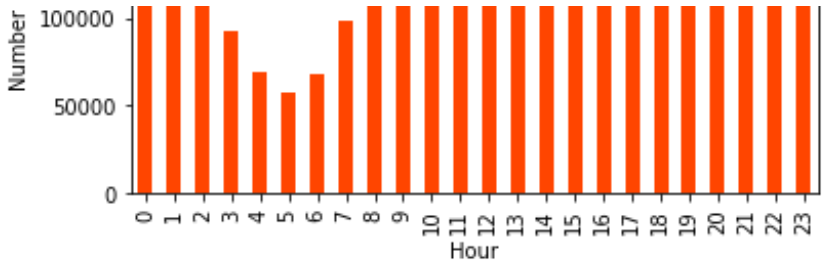



Number of Crimes A
300000
200000
100000
0
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
Year


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

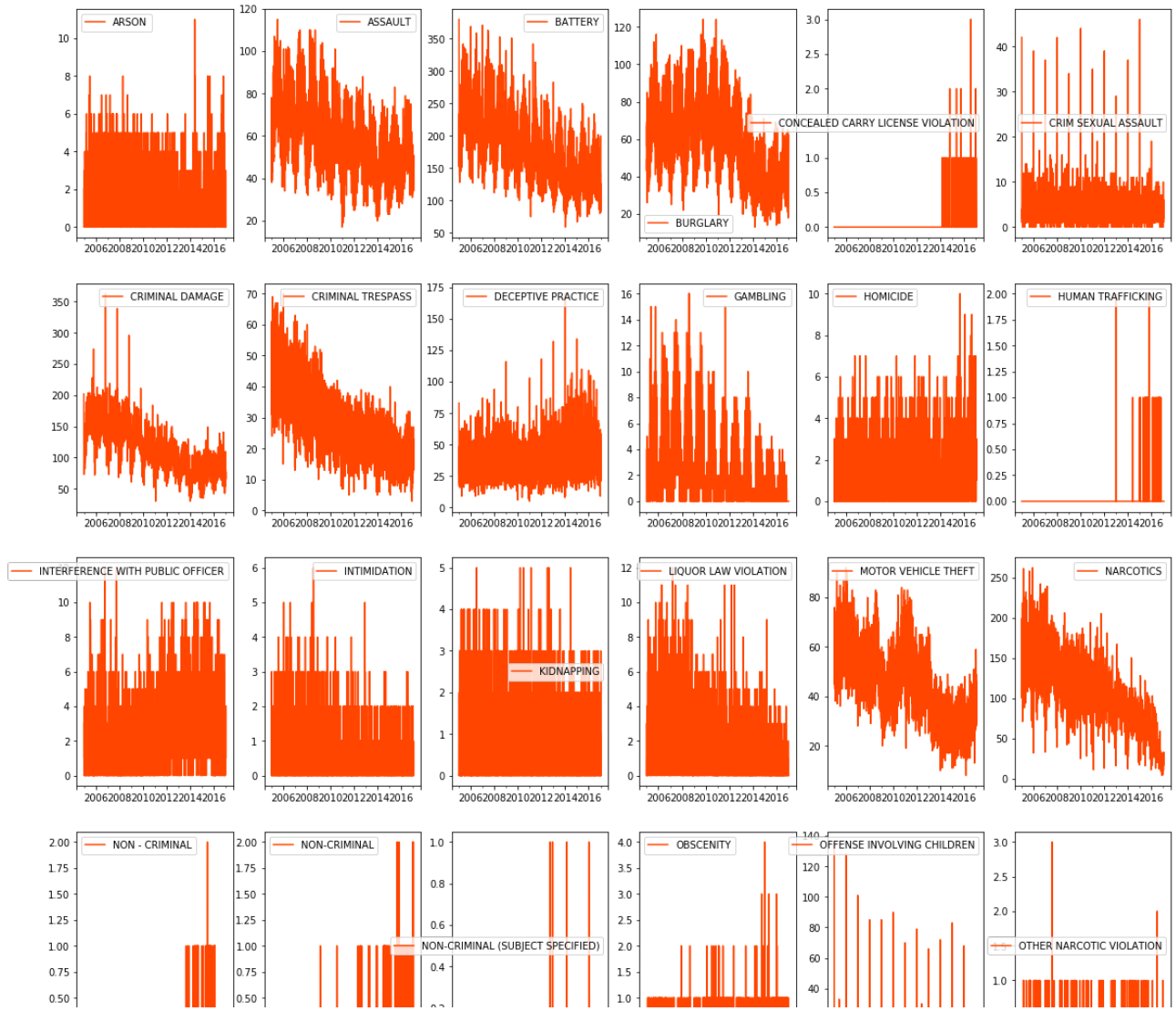

Crime Per Day Of Month
Number of Crimes Acts
160000
140000
120000
100000
80000
60000
40000
20000
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
Day

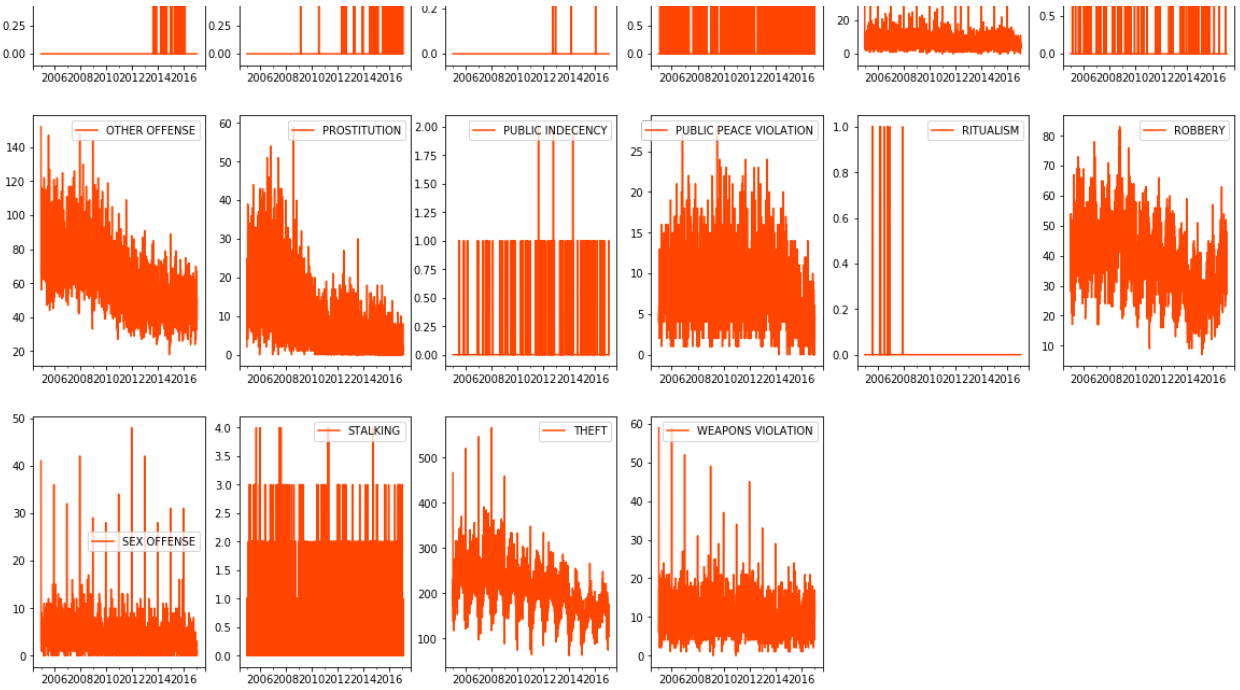

Crime Per Hour
250000
200000
150000



In [14]:

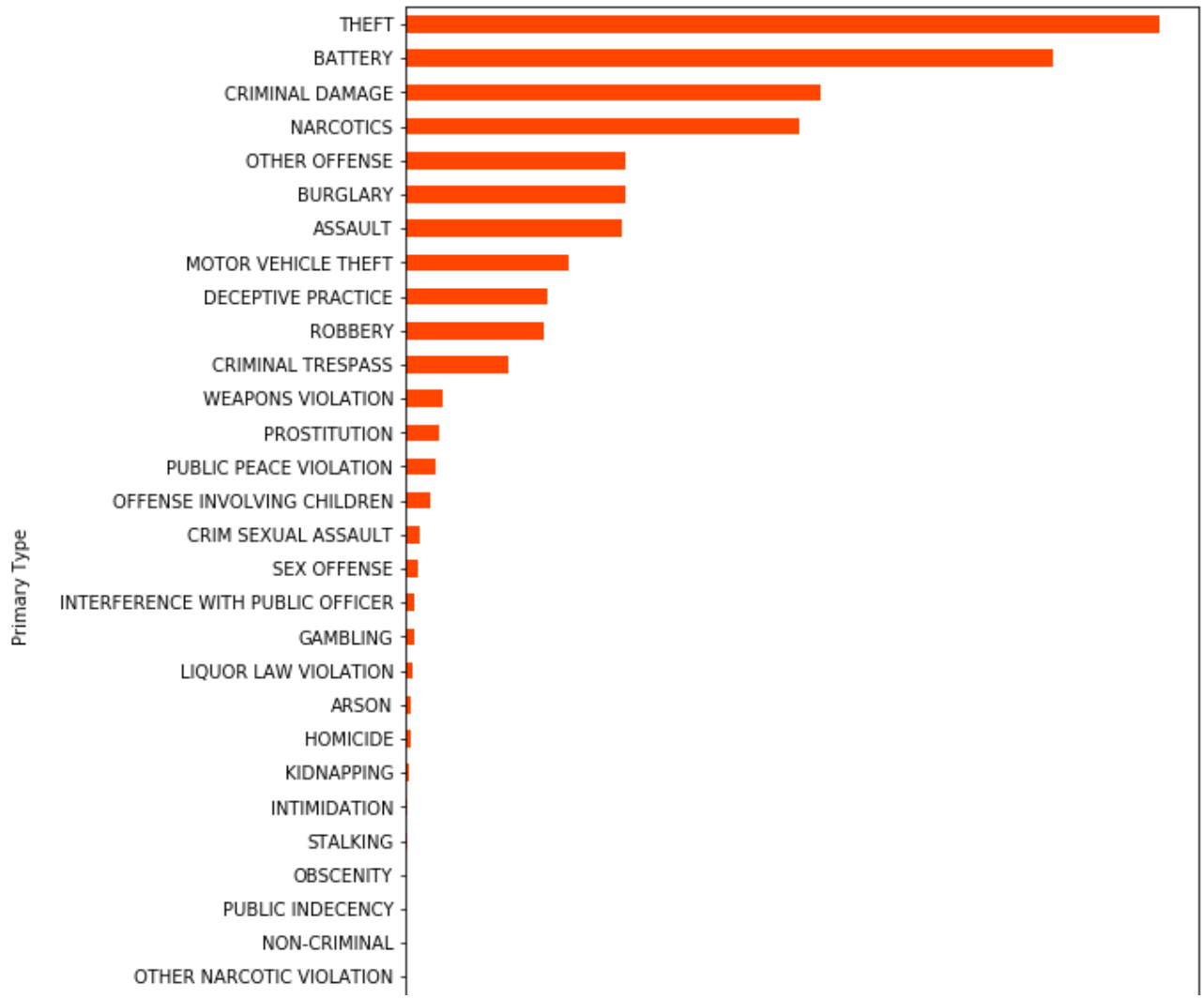
```
Crime_Data_date = Crime_Data.pivot_table('ID', aggfunc = np.size, columns =
'Primary Type',
                                         index = Crime_Data.index.date, fil
value = 0)
Crime_Data_date.index = pd.DatetimeIndex(Crime_Data_date.index)


# 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)
```

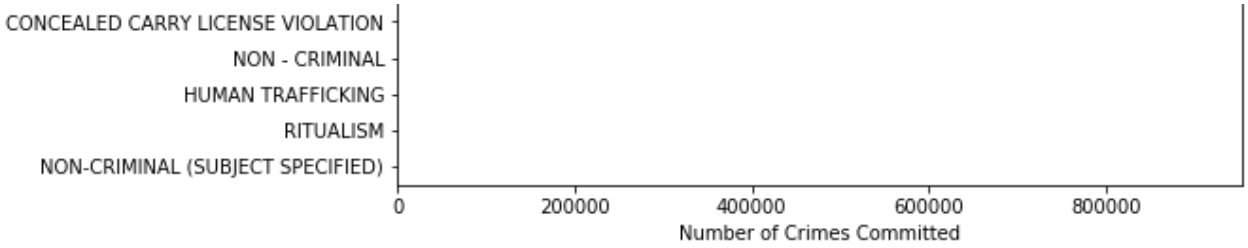




In [15]:

```
# 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]:

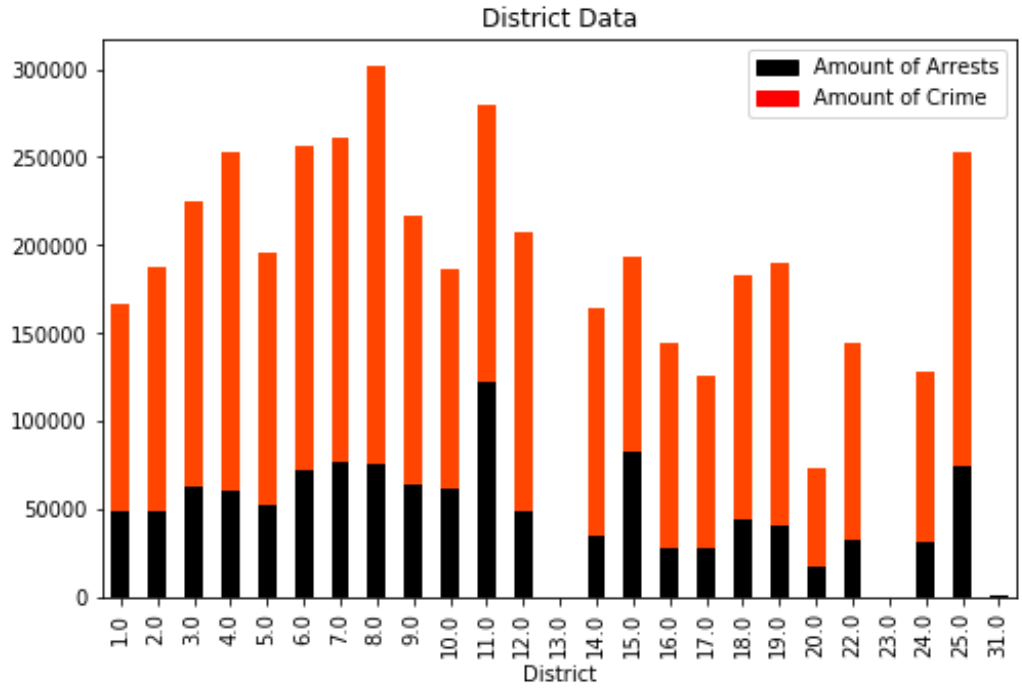
```
Crime_Data_district = Crime_Data.pivot_table('Arrest', aggfunc = np.sum, co
lumns = ['District'],
                                             index = Crime_Data.index.date, fil
value = 0)
Crime_Data_ward = Crime_Data.pivot_table('Arrest', aggfunc = np.sum,
columns = ['Ward'],
                  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

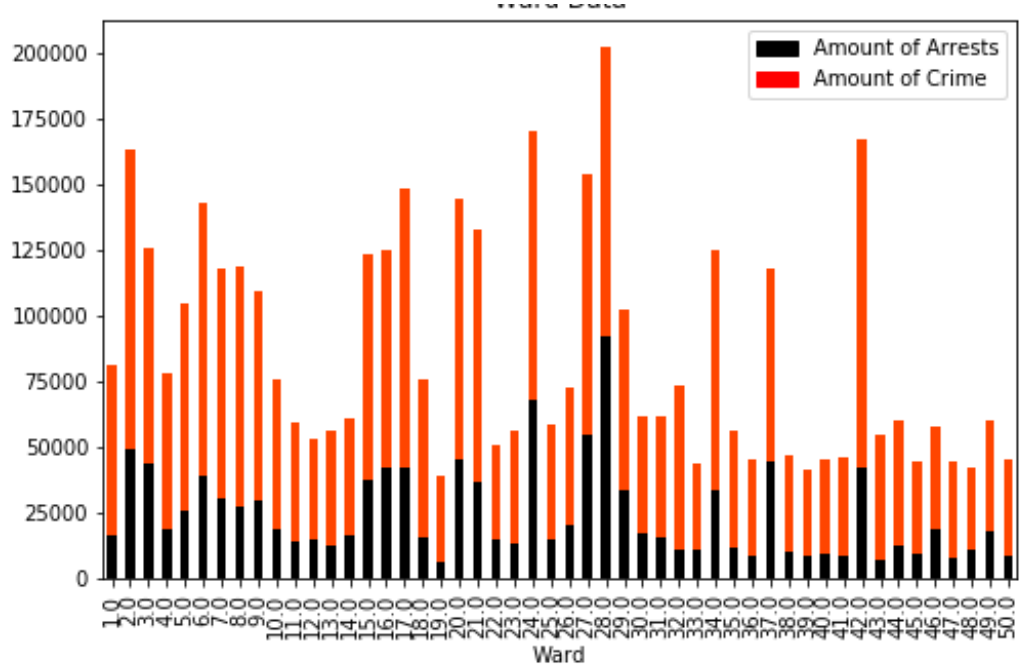
```
red = mpatches.Patch(color='black', label='Amount of Arrests')
blue = mpatches.Patch(color='red', label='Amount of Crime')

#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)
```

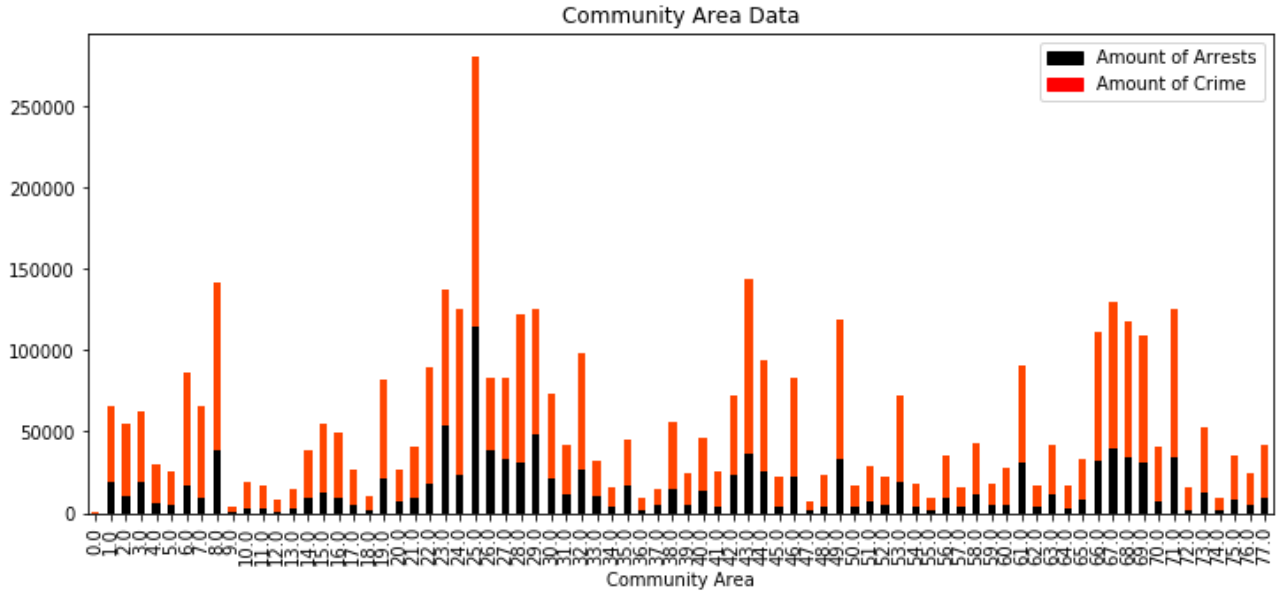
```
C:\Users\sanch\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: Matplot
libDeprecationWarning: pyplot.hold is deprecated.
```

```
    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

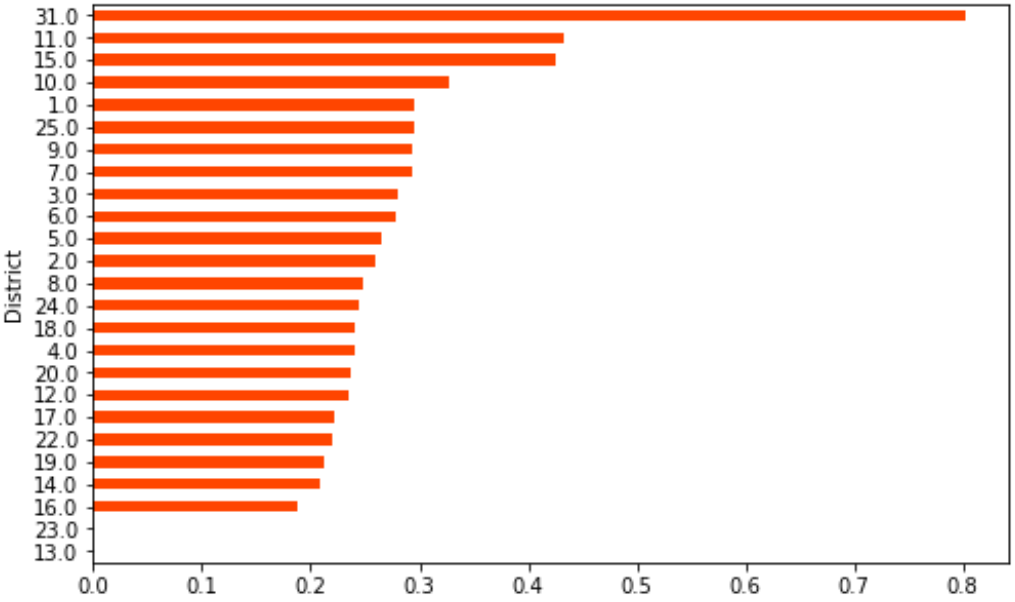




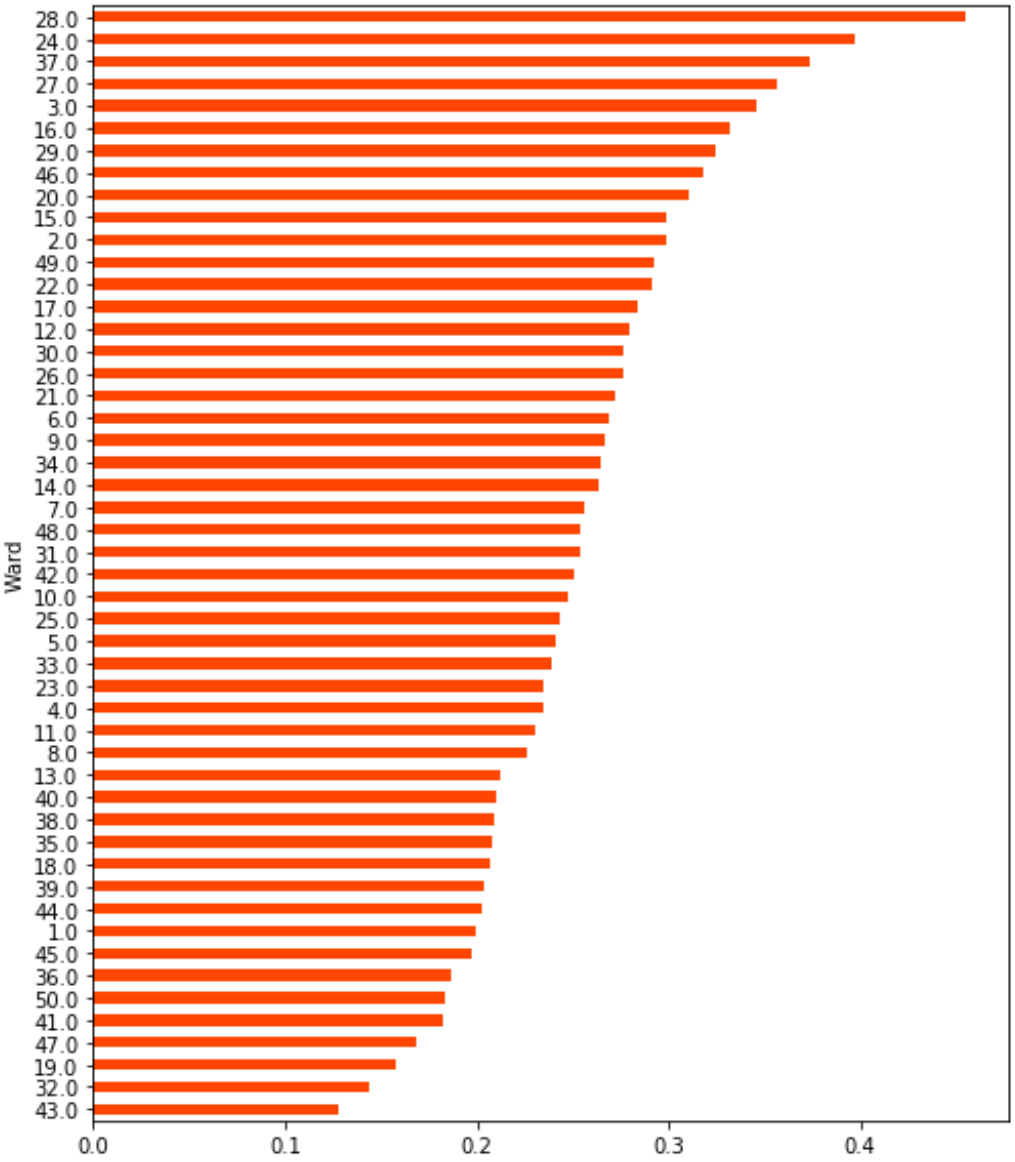
In [18]:

```
#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)
```

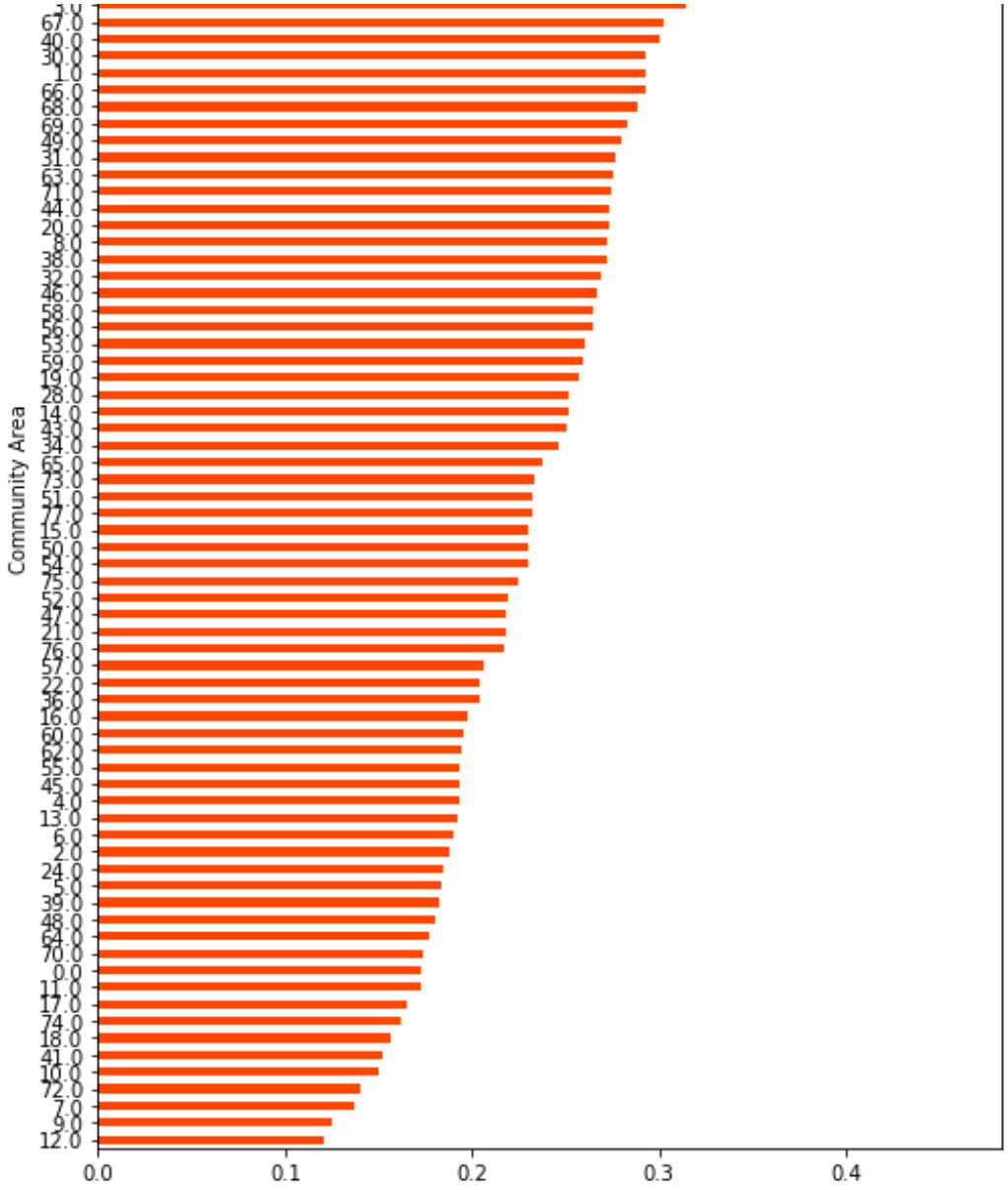
Arrest Rates Per District



Arrest Rates Per Ward



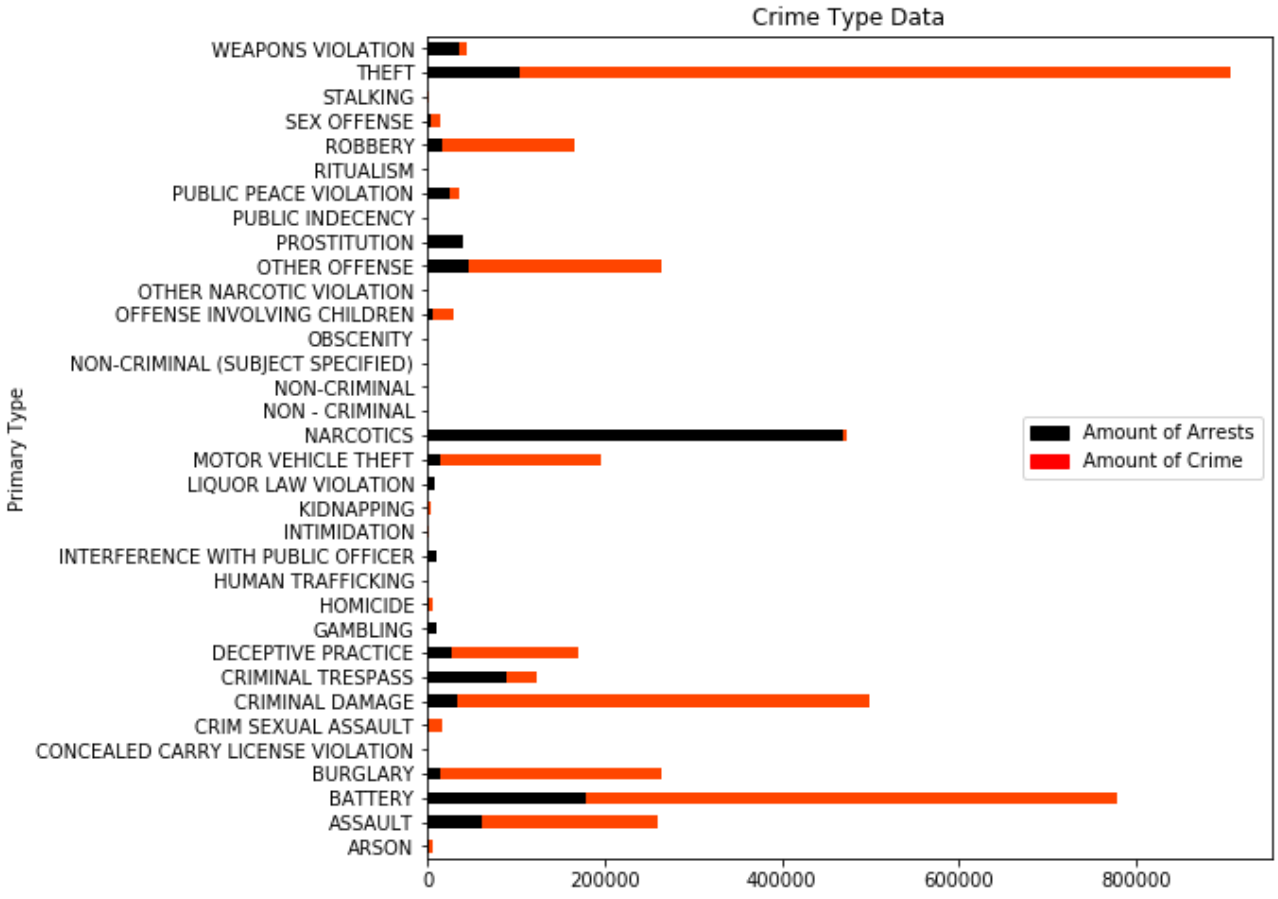
Arrest Rates Per Community Area

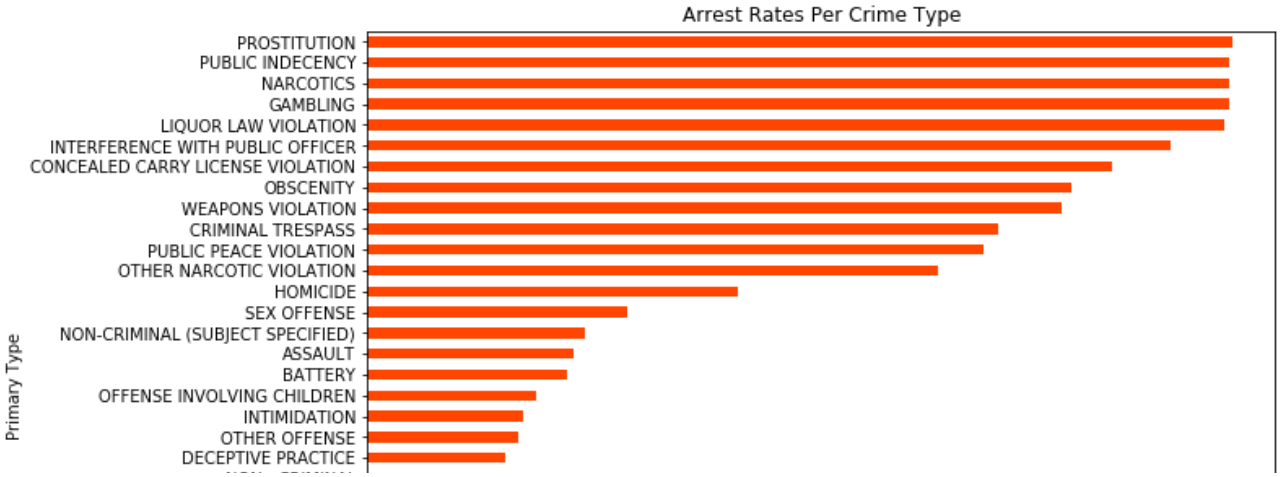


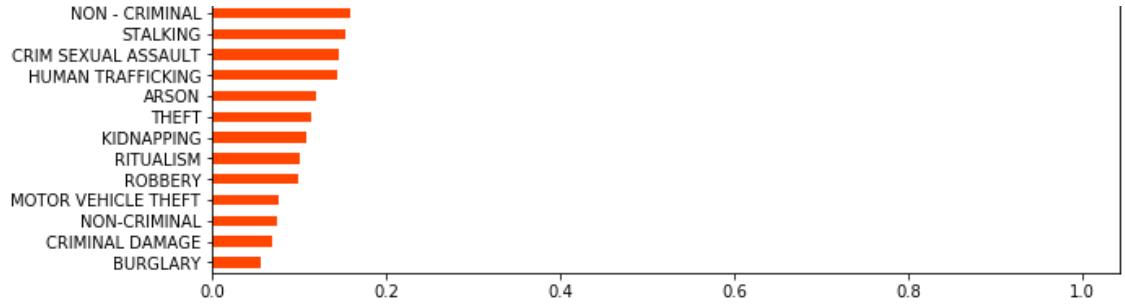
In [19]:

```
#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                    12

2006-02-26 13:40:48  4673627            17          218                    11

2006-01-08 23:16:00  4673628             1           40                    11

2006-04-05 18:45:00  4673629             2          309                    12

2006-02-17 21:03:14  4673630            17          267                     1
```

```
                     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]:

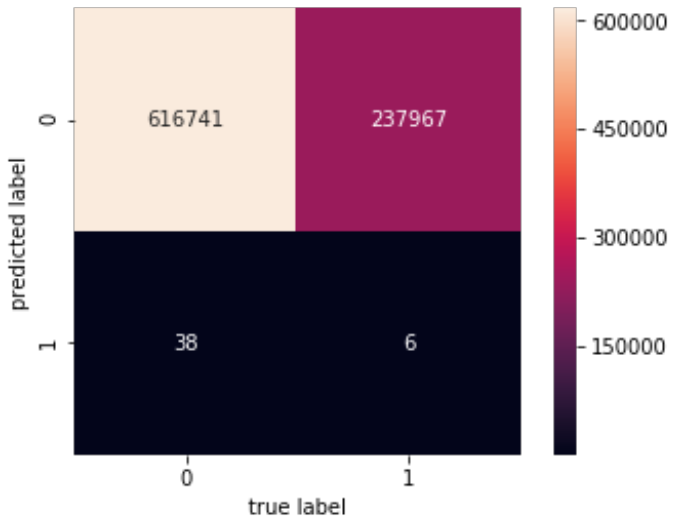
```
print ('-----------GLM Least Square Regression-----------------')
```

```
-----------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]:

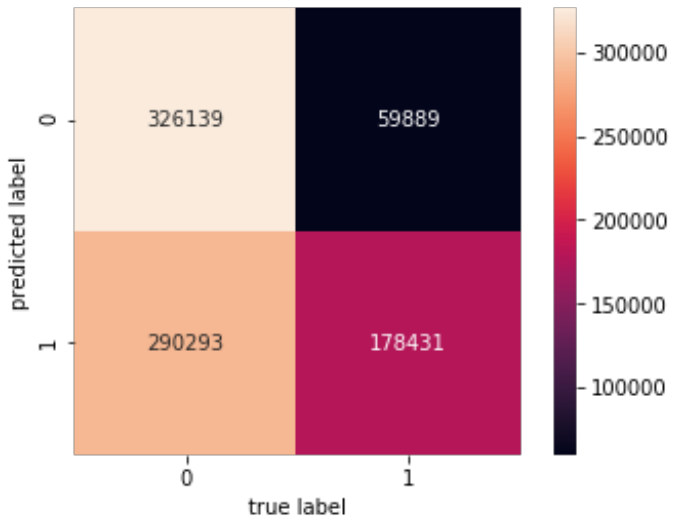
```
print ('-----------Logistic Regression-----------------')
```

```
-----------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]:

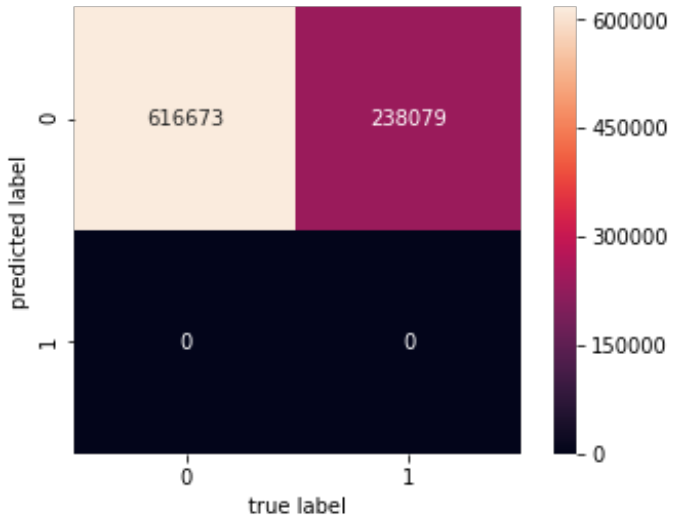
```
## Gaussian Naive Bayes
```

```
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]:

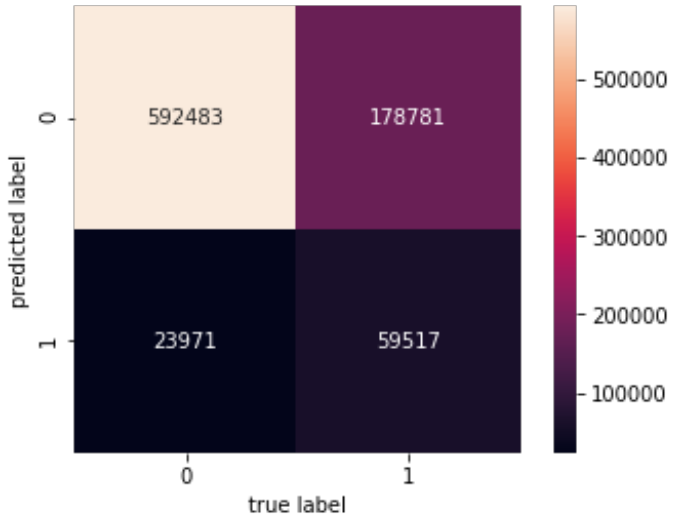
```
print ('----------Bernoulli Naive Bayes ----------------')
```

```
----------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-----------------')
```

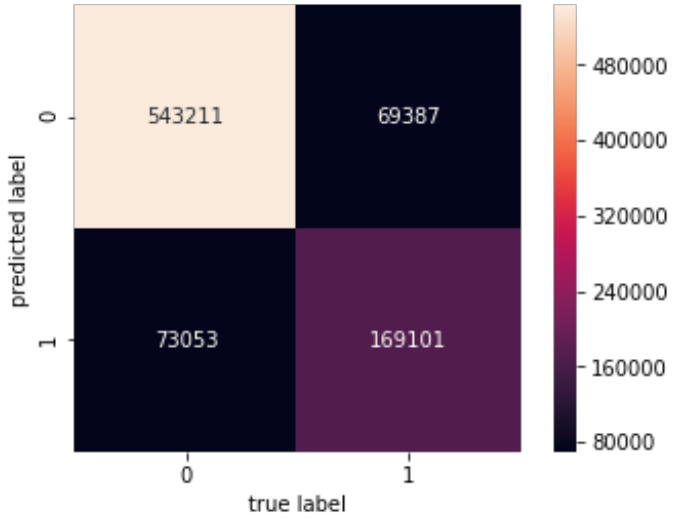
```
-----------Decision Tree-----------------
```

In [36]:

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-----------------')
```

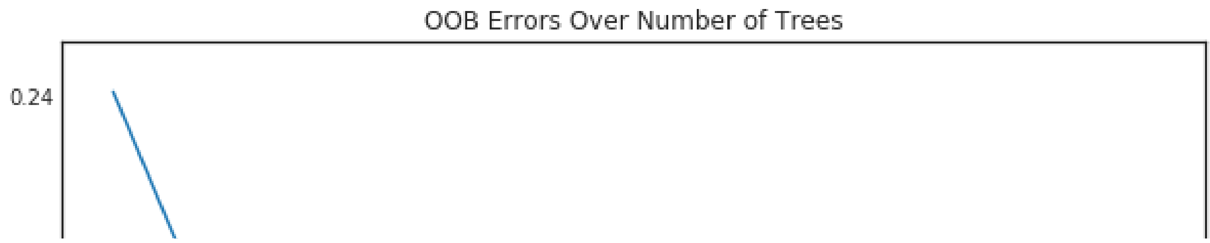
```
-----------Random Forest-----------------
```

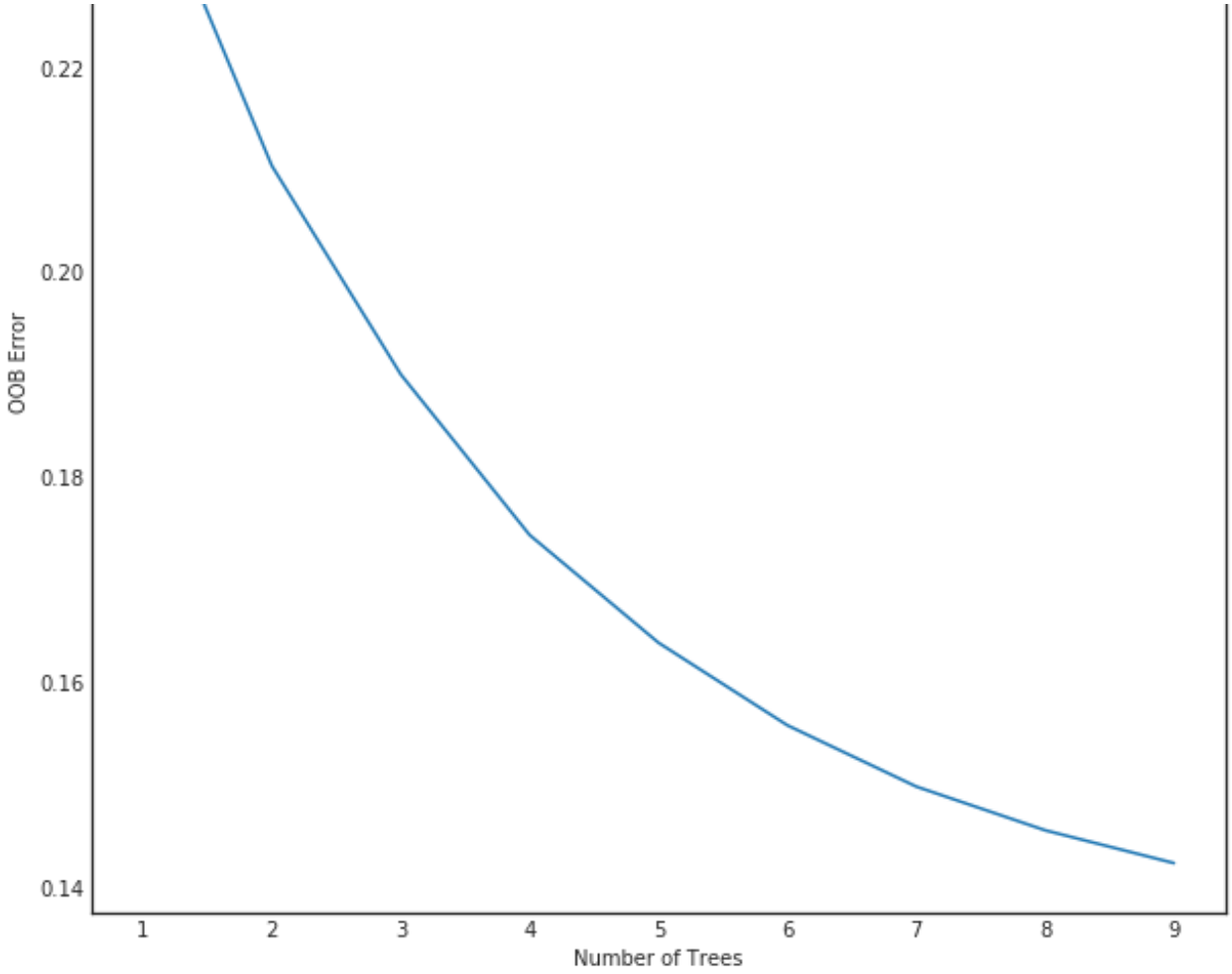
In [40]:

```
## 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. "
```

```
  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])
```

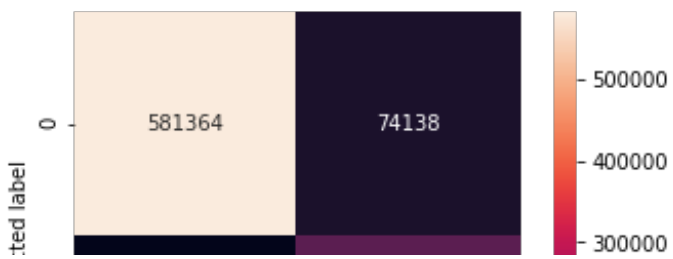


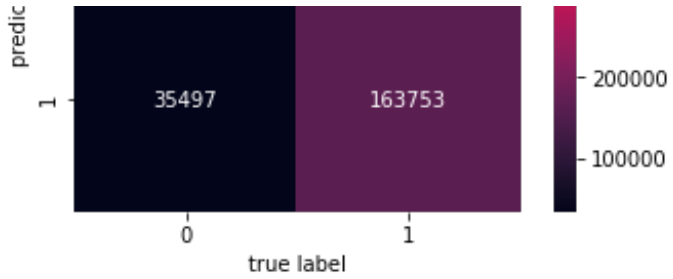


In [41]:

```
# 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))
```

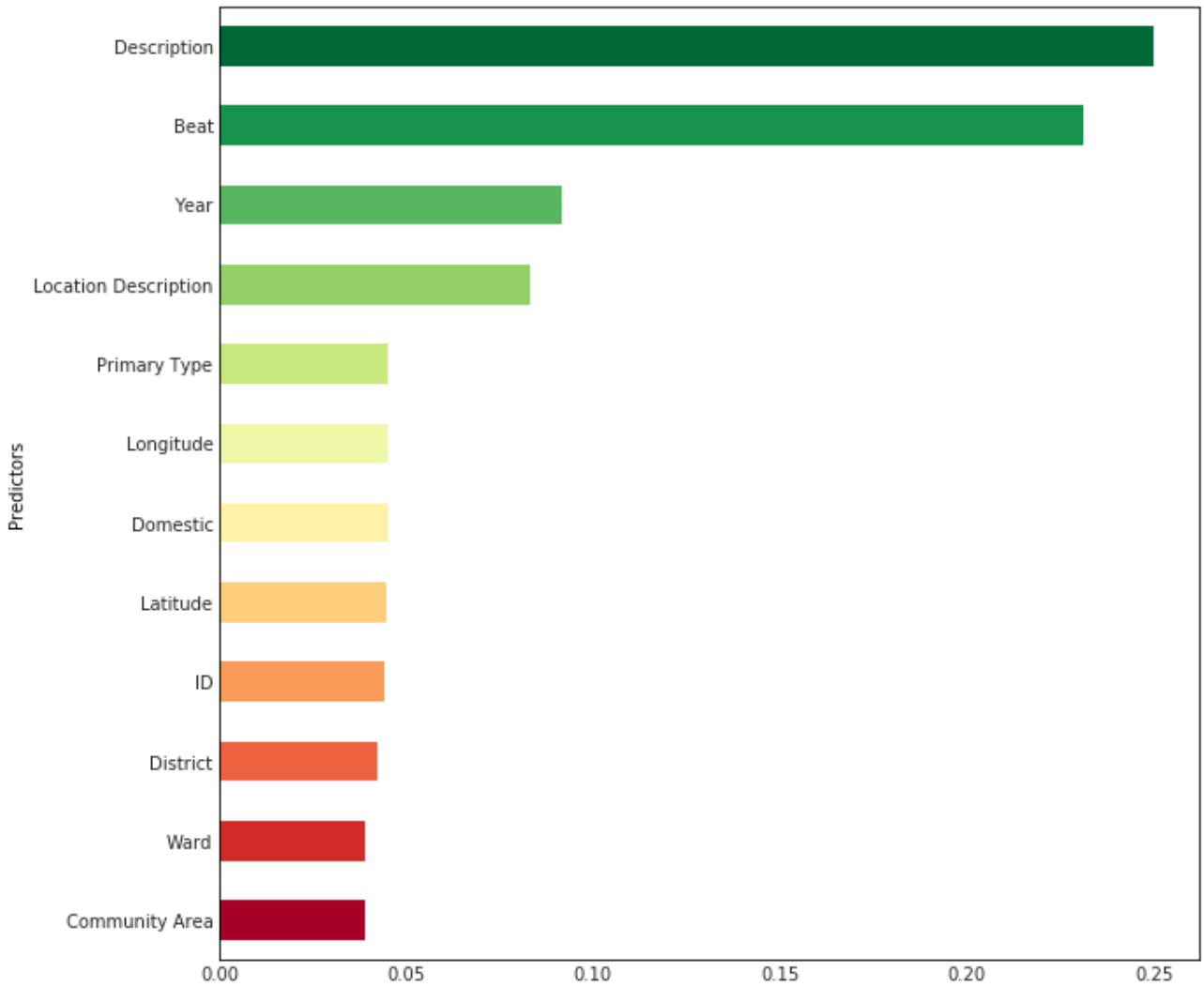
```
             precision    recall  f1-score   support

      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()
```