Project Poirot: Machine Learning based Predictive Crime Analysis

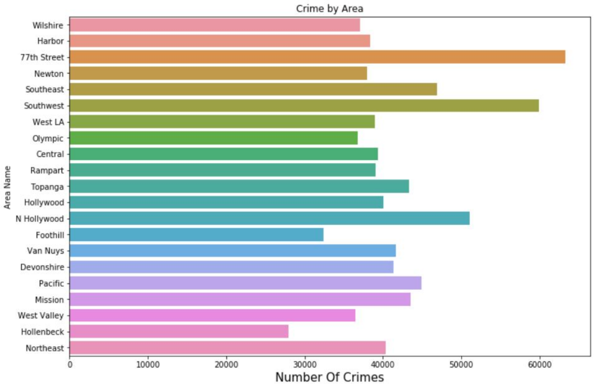
Mansi Ganatra[1], Naga Ritwik Indugu [2] and Uma Kanumuri [3]

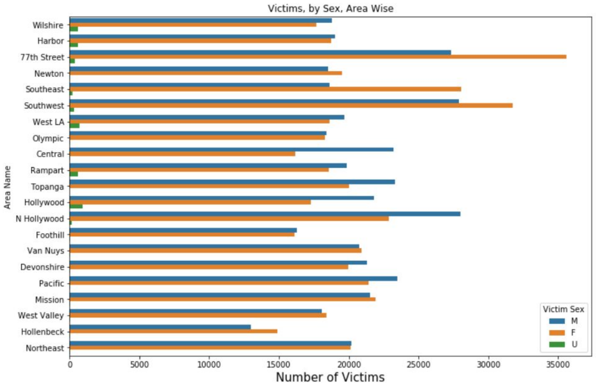
***Abstract*— LA is a very beautiful city, but one thing that scares a lot of people is the amount of crime that happens here. Our intention is to understand what makes this beautiful city to be infested with crime.To work upon this we have collected the dataset from LAPD official site.The dataset contains approximately 1.8 Million rows with around 28 attributes including DR Number, Date Reported, Date Occurred, Time Occurred, Area ID, Area Name Reporting District etc.Some of the questions which we are going to answer are Predicting the time of day, the days of a week, the days of a month, the days of a year in which different kinds of crimes are most likely to happen,to predict the hot spots for various crime categories, to predict the crime patterns, To predict the probability of an offender repeating the crime,Crimes correlation with age,Severity of crime by area (which crime is highest in each area) etc using the different machine learning algorithms like Decision Trees,Nave Bayes,Logistic Regression,K-Nearest Neighbours(K-NN),Support Vector Machines(SVM),Clustering Algorithms.The libraries which we are using include pandas, numpy, scikit-learn, pytorch, theano, etc.For the visualization of results the libraries being used are matplotlib, d3.js etc.**

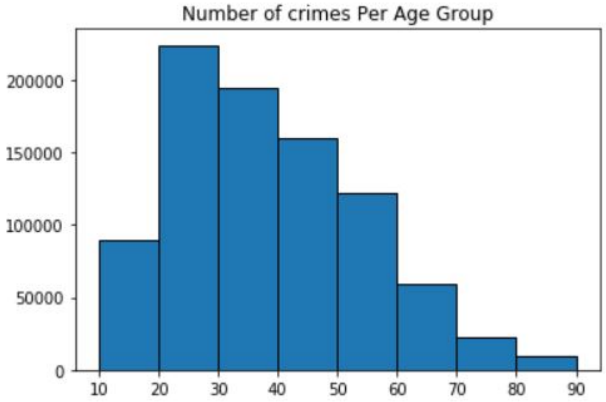
I. INTRODUCTION

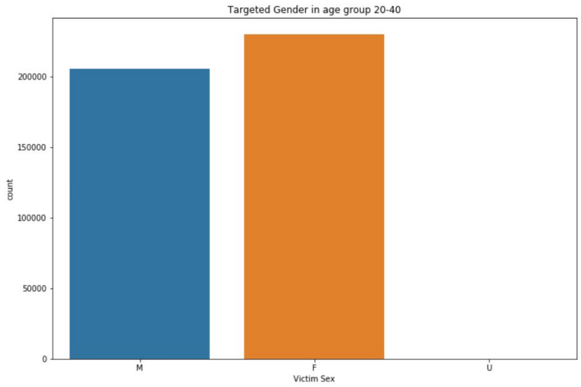
Beautiful and historic, Los Angeles is also infested with a high amount of crime. The total crimes reported to LAPD in 2017 were 129587(Violent: 29661, Property: 99926). The LAPD has made available the crime dataset from 2010-2018. In the Machine Learning experiments described below, we use the same crime dataset pre-processed from Kaggle containing records from 2010-2017. We performed some exploratory data analysis on the features and the results are as in the following section. In this project, we try to answer different questions based on the features. The questions and the algorithms used to evaluate answers to each question are described in the further section. We have used accuracy, F-score, mean squared error as our performance metrics. There have been multiple questions raised for the ethicality of such studies. The study in this project is only empirical and academic in nature. We do not intend to use the results and predictions on field or in conjunction with other predictive policing technologies in use today. Our aim is to apply various machine learning techniques to the selected features, keeping into account the literature survey, and evaluate answer to the selected questions. We intend to measure the accuracy of our results using the performance metric as described in the following section.

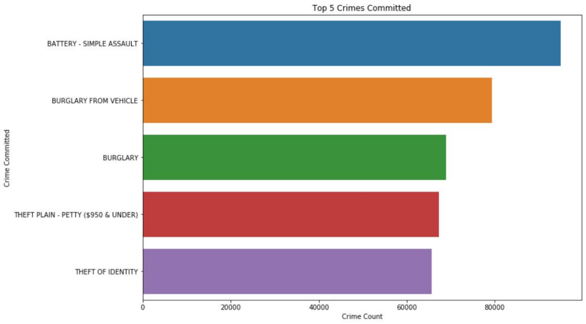
II. EXPLORATORY DATA ANALYSIS

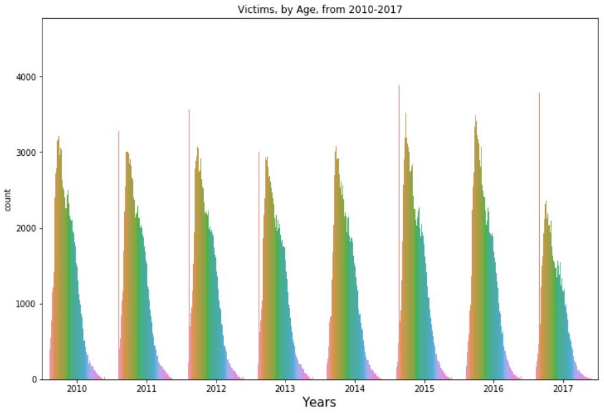


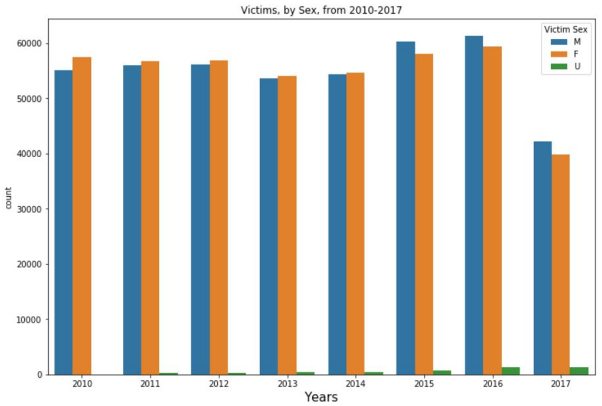












III. METHODOLOGY

A. Learning Algorithms

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations

- **SVM**: In this project we have used SVM with rbf kernel keeping the value of c to 1.0 and gamma to auto. Predict(X) method is used in predicting the crime code given victim age,victim sex.sklearn.svm has been used in importing the svm function.
- **Decision Trees**:Decision trees with a criteria of entropy and random state as 0 are taken.sklearn.tree.decision tree classifier is used to get the decision tree function.Predict(X) method has been used in predicting the crime code given victim age and victim sex
- **Logistic Regression**:Mutinomial logistic regression has been used in predicting the crime code given victim age and victim sex.Newton Solver has been used here. sklearn.linearmodel.logistic regression has been used for getting the regression function.Predict(X) method has been used for predicting the crime code.
- **Linear Regression**:Linear regression has been used in predicting the time occured given the weekdays.Here weekday is calculated from the date occurred using a python function.sklearn.linearmodel.linearregression has been used for getting the prediction function. predict(X) method has been used for predicting the linear regression.
- **Neural Networks**:Neural network with 3 layers and approxiamately 10000 weights have been used in predicting the time occurred given weekday and crimecode.Also neural networks have been used in predicting the crimecode given victim age and gender.

B. Performance Metrics

- **Accuracy score**:From sklearn.metrics import accuracy score has been used in getting the accuracy score function.It is used for calculating the accuracy's of svm, decision trees and logistic regression in this project.
- **Root mean square error**:RMSE has been used in linear regression for measuring the accuracy of predicting time of the day given day and time. from sklearn.metrics import mean_squared_error has been used for getting the mean_squared_error function.

IV. VISUALIZATION FUNCTIONS

Packages seaborn and matplotlib in python have been used in visualization of the results.

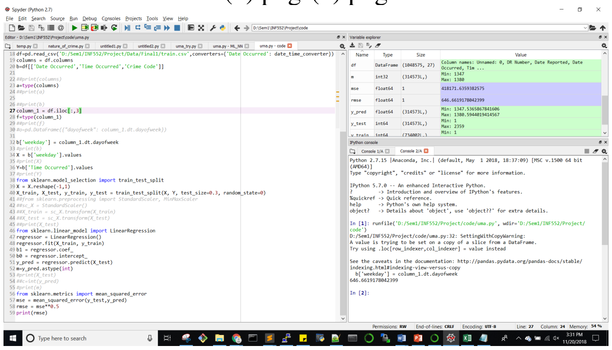
V. Current Results

Question	Logistic/Linear Regression	Decision Trees	SVM	KNN
Prediction of crime code given victim age and victim sex	With grouping: ~33.12% Without grouping: ~11.55%	With grouping: ~19.96% Without grouping: ~13.32%	H/W unable to handle	With grouping: ~21.61% Without grouping: ~16%
Prediction of time occurred given weekday	rmse: 646.66	-	-	-
Prediction of area code given date occurred, time occurred and crime type	~7%	~6.2%	H/W unable to handle	~6.7%

A. Prediction of time occurred given weekday and Crime Code

- **Preprocessing**:Weekday has been obtained from the date of the crime occurred with python function dt.dayofweek.
- **Linear regression**:Linear regression has been used for the prediction given only the weekday. Accuracy obtained was:

(1).png (1).png



Spyder (Python 2.7)

Editor - D:\Sem1\INF552\Project\code\uma.py

```python
df=pd.read_csv('D:/Sem1/INF552/Project/Data/Final1/train.csv',converters={'Date Occurred': date_time_converter})
columns = df.columns
b=df[['Date Occurred','Time Occurred','Crime Code']]

##print(columns)
a=type(columns)
##print(a)

##print(b)
column_1 = df.iloc[:,3]
f=type(column_1)
##print(f)
#o=pd.DataFrame({"dayofweek": column_1.dt.dayofweek})

b['weekday'] = column_1.dt.dayofweek
#print(b)
X = b['weekday'].values
#print(X)
Y=b['Time Occurred'].values
#print(Y)
from sklearn.model_selection import train_test_split
X = X.reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
##from sklearn.preprocessing import StandardScaler, MinMaxScaler
##sc_X = StandardScaler()
##X_train = sc_X.transform(X_train)
##X_test = sc_X.transform(X_test)
##print(X_test)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
b1 = regressor.coef_
b0 = regressor.intercept_
y_pred = regressor.predict(X_test)
m=y_pred.astype(int)
#print(X_test)
##c=int(y_pred)
#print(m)
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test,y_pred)
rmse = mse**0.5
print(rmse)
```

Variable explorer

Name	Type	Size	Value
df	DataFrame	(1048575, 27)	Column names: Unnamed: 0, DR Number, Date Reported, Date Occurred, Tim ...
m	int32	(314573L,)	Min: 1347 Max: 1380
mse	float64	1	418171.6359382575
rmse	float64	1	646.6619178042399
y_pred	float64	(314573L,)	Min: 1347.5365867841606 Max: 1380.5944019414567
y_test	int64	(314573L,)	Min: 1 Max: 2359
y_train	int64	(734002L,)	Min: 1

IPython console — Console 2/A

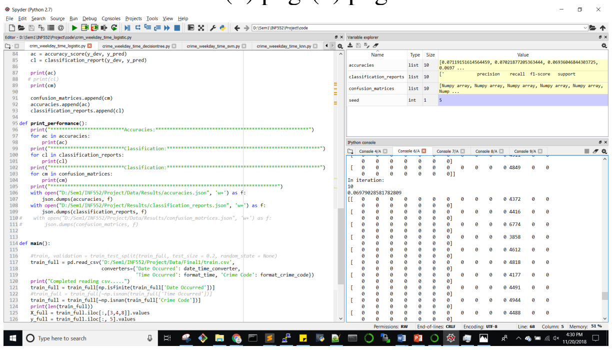
```
Python 2.7.15 |Anaconda, Inc.| (default, May  1 2018, 18:37:09) [MSC v.1500 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.7.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: runfile('D:/Sem1/INF552/Project/code/uma.py', wdir='D:/Sem1/INF552/Project/
code')
D:/Sem1/INF552/Project/code/uma.py:32: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
indexing.html#indexing-view-versus-copy
  b['weekday'] = column_1.dt.dayofweek
646.6619178042399

In [2]:
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 27 Column: 24 Memory: 54 %

3:31 PM 11/20/2018

- **Logistic Regression**:

(3).png (3).png



Spyder (Python 2.7)

Editor - D:\Sem1\INF552\Project\code\crim_weekday_time_logistic.py

```python
    ac = accuracy_score(y_dev, y_pred)
    cl = classification_report(y_dev, y_pred)

    print(ac)
   # print(cl)
    print(cm)

    confusion_matrices.append(cm)
    accuracies.append(ac)
    classification_reports.append(cl)

def print_performance():
    print("**************************Accuracies:*******************************************************")
    for ac in accuracies:
        print(ac)
    print("**************************Classification:***************************************************")
    for cl in classification_reports:
        print(cl)
    print("**************************Classification:***************************************************")
    for cm in confusion_matrices:
        print(cm)
    print("*******************************************************************************")
    with open("D:/Sem1/INF552/Project/Data/Results/accuracies.json", 'w+') as f:
        json.dumps(accuracies, f)
    with open("D:/Sem1/INF552/Project/Results/classification_reports.json", 'w+') as f:
        json.dumps(classification_reports, f)
#    with open("D:/Sem1/INF552/Project/Data/Results/confusion_matrices.json", 'w+') as f:
#        json.dumps(confusion_matrices, f)


def main():

    #train, validation = train_test_split(train_full, test_size = 0.2, random_state = None)
    train_full = pd.read_csv('D:/Sem1/INF552/Project/Data/Final1/train.csv',
                          converters={'Date Occurred': date_time_converter,
                                      'Time Occurred': format_time, 'Crime Code': format_crime_code})
    print("Completed reading csv.....")
    train_full = train_full[np.isfinite(train_full['Date Occurred'])]
    #train_full = train_full[~np.isnan(train_full['Time Occurred'])]
    train_full = train_full[~np.isnan(train_full['Crime Code'])]
    print(len(train_full))
    X_full = train_full.iloc[:,[3,4,8]].values
    y_full = train_full.iloc[:, 5].values
```

Variable explorer

Name	Type	Size	Value
accuracies	list	10	[0.07119151614564459, 0.07021877205363444, 0.06936046844303725, 0.0697 ...
classification_reports	list	10	[' precision recall f1-score support
confusion_matrices	list	10	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
seed	int	1	5

IPython console — Console 6/A

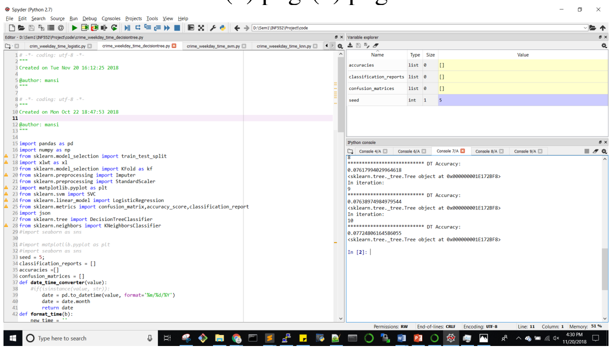
```
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0 4849    0    0
     0    0    0    0    0    0    0]]
In iteration:
10
0.06979028581782809
[[   0    0    0    0    0    0    0    0    0    0    0 4372    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0 4416    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0 6774    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0 3858    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0 4612    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0 4818    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0 4177    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0 4491    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0 4944    0    0
     0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0    0    0    0 4488    0    0
     0    0    0    0    0    0    0]
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 68 Column: 5 Memory: 51 %

4:30 PM 11/20/2018

- **Decision Trees**:

(4).png (4).png



Spyder (Python 2.7)

Editor - D:\Sem1\INF552\Project\code\crime_weekday_time_decisiontree.py

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Nov 20 16:12:25 2018

@author: mansi
"""

# -*- coding: utf-8 -*-
"""
Created on Mon Oct 22 18:47:53 2018

@author: mansi
"""

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import xlwt as xl
from sklearn.model_selection import KFold as kf
from sklearn.preprocessing import Imputer
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
import json
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
#import seaborn as sns

#import matplotlib.pyplot as plt
#import seaborn as sns
seed = 5;
classification_reports = []
accuracies =[]
confusion_matrices = []
def date_time_converter(value):
    #if(isinstance(value, str)):
        date = pd.to_datetime(value, format='%m/%d/%Y')
        date = date.month
        return date
def format_time(b):
    new_time = ''
```

Variable explorer

Name	Type	Size	Value
accuracies	list	0	[]
classification_reports	list	0	[]
confusion_matrices	list	0	[]
seed	int	1	5

IPython console — Console 7/A

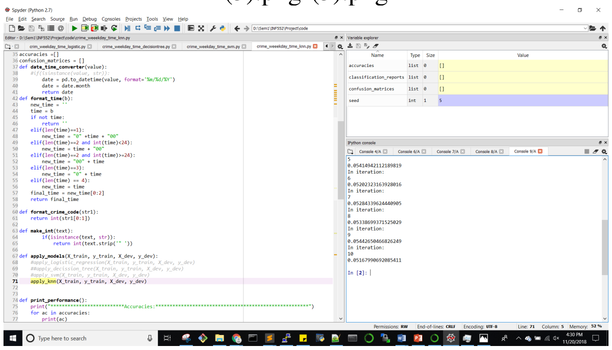
```
8
*************************** DT Accuracy:
0.07617994029964618
<sklearn.tree._tree.Tree object at 0x000000001E172BF8>
In iteration:
9
*************************** DT Accuracy:
0.07638974984979544
<sklearn.tree._tree.Tree object at 0x000000001E172BF8>
In iteration:
10
*************************** DT Accuracy:
0.07724806164586055
<sklearn.tree._tree.Tree object at 0x000000001E172BF8>

In [2]: |
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 11 Column: 1 Memory: 51 %

4:30 PM 11/20/2018

- **K-NN**:

(5).png (5).png



Spyder (Python 2.7)

Editor - D:\Sem1\INF552\Project\code\crime_weeekday_time_knn.py

```python
accuracies =[]
confusion_matrices = []
def date_time_converter(value):
    #if(isinstance(value, str)):
        date = pd.to_datetime(value, format='%m/%d/%Y')
        date = date.month
        return date
def format_time(b):
    new_time = ''
    time = b
    if not time:
        return ''
    elif(len(time)==1):
        new_time = "0" +time + "00"
    elif(len(time)==2 and int(time)<24):
        new_time = time + "00"
    elif(len(time)==2 and int(time)>=24):
        new_time = "00" + time
    elif(len(time)==3):
        new_time = "0" + time
    elif(len(time) == 4):
        new_time = time
    final_time = new_time[0:2]
    return final_time

def format_crime_code(str1):
    return int(str1[0:1])

def make_int(text):
        if(isinstance(text, str)):
            return int(text.strip('" '))

def apply_models(X_train, y_train, X_dev, y_dev):
    #apply_logistic_regression(X_train, y_train, X_dev, y_dev)
    ##apply_decission_tree(X_train, y_train, X_dev, y_dev)
    #apply_svm(X_train, y_train, X_dev, y_dev)
    apply_knn(X_train, y_train, X_dev, y_dev)


def print_performance():
    print("**************************Accuracies:*******************************************************")
    for ac in accuracies:
        print(ac)
```

Variable explorer

Name	Type	Size	Value
accuracies	list	0	[]
classification_reports	list	0	[]
confusion_matrices	list	0	[]
seed	int	1	5

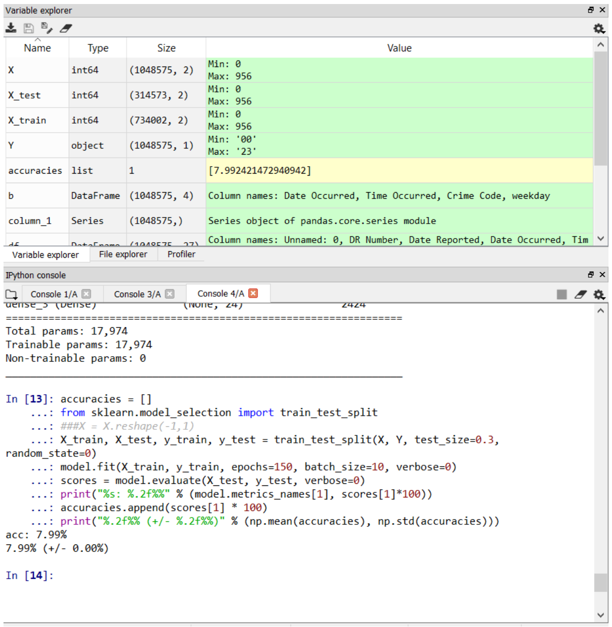
IPython console — Console 9/A

```
5
0.05414942112189819
In iteration:
6
0.05202323163928016
In iteration:
7
0.05284339624440905
In iteration:
8
0.05338699371525029
In iteration:
9
0.05442650466826249
In iteration:
10
0.05167990692085411

In [2]: |
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 71 Column: 5 Memory: 52 %

4:30 PM 11/20/2018

- **Neural Networks**: A neural network of sequential model with 3 layers and approximately 18000 weights have been built with relu,tanh and softmax functions. Accuracy obtained was:



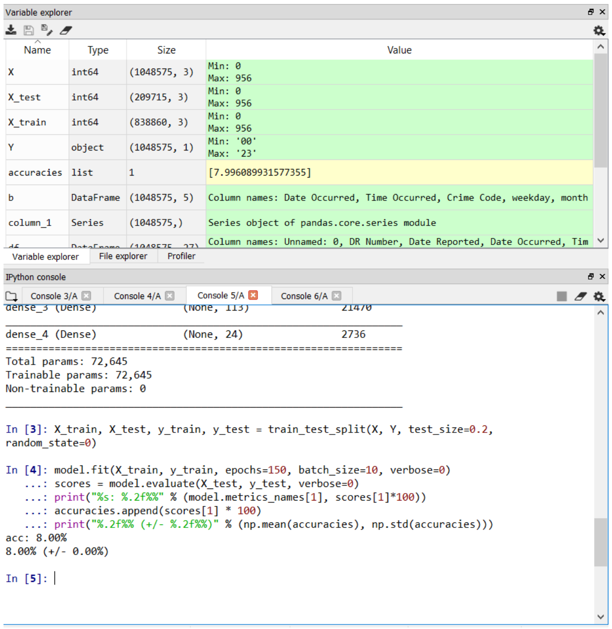
Variable explorer

Name	Type	Size	Value
X	int64	(1048575, 2)	Min: 0 Max: 956
X_test	int64	(314573, 2)	Min: 0 Max: 956
X_train	int64	(734002, 2)	Min: 0 Max: 956
Y	object	(1048575, 1)	Min: '00' Max: '23'
accuracies	list	1	[7.992421472940942]
b	DataFrame	(1048575, 4)	Column names: Date Occurred, Time Occurred, Crime Code, weekday
column_1	Series	(1048575,)	Series object of pandas.core.series module
df	DataFrame	(1048575, 27)	Column names: Unnamed: 0, DR Number, Date Reported, Date Occurred, Tim

IPython console — Console 4/A

```
dense_3 (Dense)              (None, 24)                2424
=================================================================
Total params: 17,974
Trainable params: 17,974
Non-trainable params: 0
_________________________________________________________________

In [13]: accuracies = []
    ...: from sklearn.model_selection import train_test_split
    ...: ###X = X.reshape(-1,1)
    ...: X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,
random_state=0)
    ...: model.fit(X_train, y_train, epochs=150, batch_size=10, verbose=0)
    ...: scores = model.evaluate(X_test, y_test, verbose=0)
    ...: print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    ...: accuracies.append(scores[1] * 100)
    ...: print("%.2f%% (+/- %.2f%%)" % (np.mean(accuracies), np.std(accuracies)))
acc: 7.99%
7.99% (+/- 0.00%)

In [14]:
```

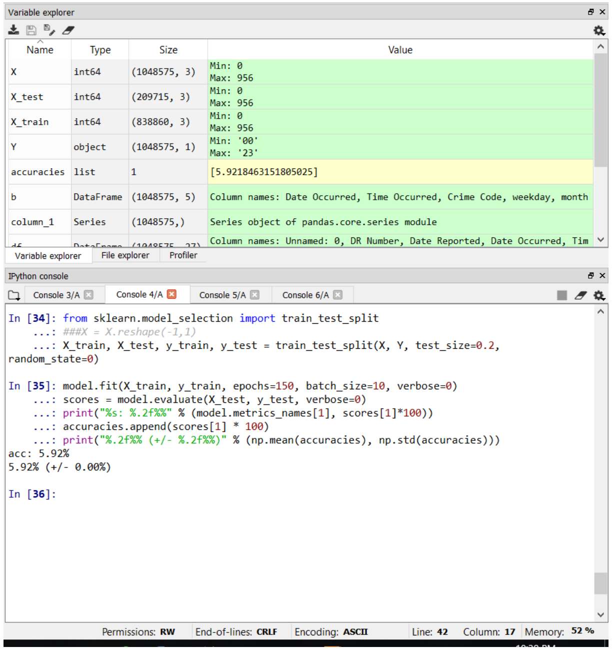
We later introduced a new input feature specifying whether the day was weekend or not. Accuracy obtained was:



Name	Type	Size	Value
X	int64	(1048575, 3)	Min: 0 Max: 956
X_test	int64	(209715, 3)	Min: 0 Max: 956
X_train	int64	(838860, 3)	Min: 0 Max: 956
Y	object	(1048575, 1)	Min: '00' Max: '23'
accuracies	list	1	[7.996089931577355]
b	DataFrame	(1048575, 5)	Column names: Date Occurred, Time Occurred, Crime Code, weekday, month
column_1	Series	(1048575,)	Series object of pandas.core.series module
df	DataFrame	(1048575, 27)	Column names: Unnamed: 0, DR Number, Date Reported, Date Occurred, Tim

```
dense_3 (Dense)              (None, 113)               21470
_________________________________________________________________
dense_4 (Dense)              (None, 24)                2736
=================================================================
Total params: 72,645
Trainable params: 72,645
Non-trainable params: 0
_________________________________________________________________

In [3]: X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=0)

In [4]: model.fit(X_train, y_train, epochs=150, batch_size=10, verbose=0)
   ...: scores = model.evaluate(X_test, y_test, verbose=0)
   ...: print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
   ...: accuracies.append(scores[1] * 100)
   ...: print("%.2f%% (+/- %.2f%%)" % (np.mean(accuracies), np.std(accuracies)))
acc: 8.00%
8.00% (+/- 0.00%)

In [5]: 
```

After this we introduced another input feature specifying the season corresponding to the day of crime. Accuracy obtained was:



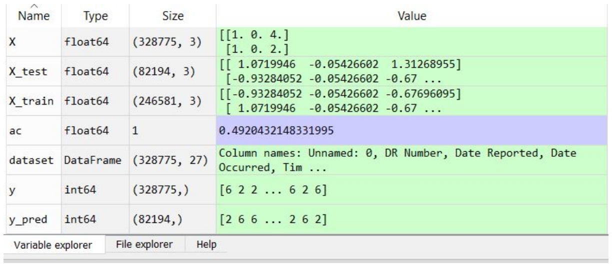
Name	Type	Size	Value
X	int64	(1048575, 3)	Min: 0 Max: 956
X_test	int64	(209715, 3)	Min: 0 Max: 956
X_train	int64	(838860, 3)	Min: 0 Max: 956
Y	object	(1048575, 1)	Min: '00' Max: '23'
accuracies	list	1	[5.9218463151805025]
b	DataFrame	(1048575, 5)	Column names: Date Occurred, Time Occurred, Crime Code, weekday, month
column_1	Series	(1048575,)	Series object of pandas.core.series module
df	DataFrame	(1048575, 27)	Column names: Unnamed: 0, DR Number, Date Reported, Date Occurred, Tim

```
In [34]: from sklearn.model_selection import train_test_split
    ...: ###X = X.reshape(-1,1)
    ...: X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=0)

In [35]: model.fit(X_train, y_train, epochs=150, batch_size=10, verbose=0)
    ...: scores = model.evaluate(X_test, y_test, verbose=0)
    ...: print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    ...: accuracies.append(scores[1] * 100)
    ...: print("%.2f%% (+/- %.2f%%)" % (np.mean(accuracies), np.std(accuracies)))
acc: 5.92%
5.92% (+/- 0.00%)

In [36]:
```

Permissions: RW | End-of-lines: CRLF | Encoding: ASCII | Line: 42 | Column: 17 | Memory: 52 %

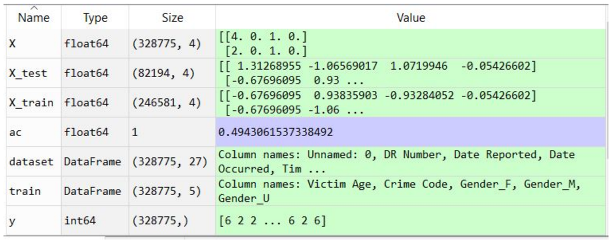
B. Prediction of Crime Code given victim information

- **Preprocessing**:The data had Victim Age range from 0-99. For better processing, we clustered the age into groups of 10.
- **Logistic Regression**:



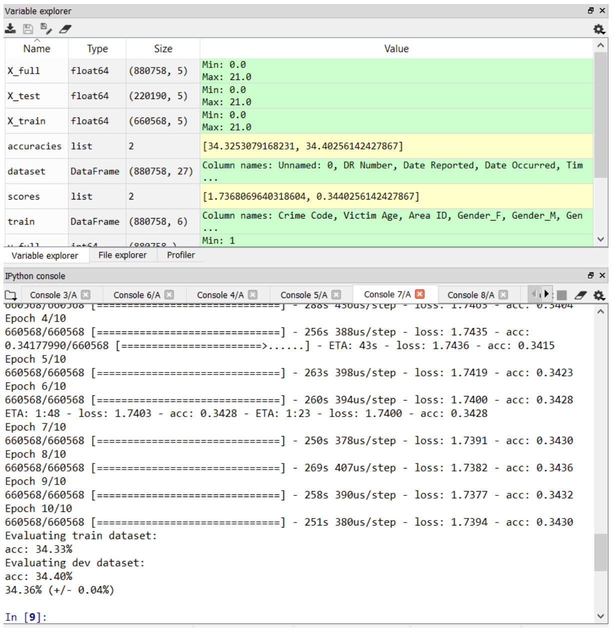
Name	Type	Size	Value
X	float64	(328775, 3)	[[1. 0. 4.] [1. 0. 2.]
X_test	float64	(82194, 3)	[[1.0719946 -0.05426602 1.31268955] [-0.93284052 -0.05426602 -0.67 ...
X_train	float64	(246581, 3)	[[-0.93284052 -0.05426602 -0.67696095] [1.0719946 -0.05426602 -0.67 ...
ac	float64	1	0.4920432148331995
dataset	DataFrame	(328775, 27)	Column names: Unnamed: 0, DR Number, Date Reported, Date Occurred, Tim ...
y	int64	(328775,)	[6 2 2 ... 6 2 6]
y_pred	int64	(82194,)	[2 6 6 ... 2 6 2]

- **Decision Trees**:



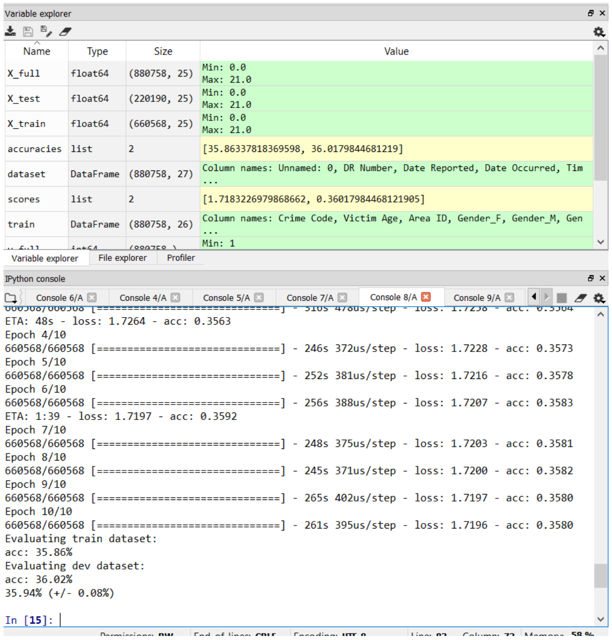
Name	Type	Size	Value
X	float64	(328775, 4)	[[4. 0. 1. 0.] [2. 0. 1. 0.]
X_test	float64	(82194, 4)	[[1.31268955 -1.06569017 1.0719946 -0.05426602] [-0.67696095 0.93 ...
X_train	float64	(246581, 4)	[[-0.67696095 0.93835903 -0.93284052 -0.05426602] [-0.67696095 -1.06 ...
ac	float64	1	0.4943061537338492
dataset	DataFrame	(328775, 27)	Column names: Unnamed: 0, DR Number, Date Reported, Date Occurred, Tim ...
train	DataFrame	(328775, 5)	Column names: Victim Age, Crime Code, Gender_F, Gender_M, Gender_U
y	int64	(328775,)	[6 2 2 ... 6 2 6]

- **Neural Networks**: A neural network of sequential model with 3 layers and approximately 18000 weights have been built with relu,tanh and softmax functions. Accuracy obtained was:



Name	Type	Size	Value
X_full	float64	(880758, 5)	Min: 0.0 Max: 21.0
X_test	float64	(220190, 5)	Min: 0.0 Max: 21.0
X_train	float64	(660568, 5)	Min: 0.0 Max: 21.0
accuracies	list	2	[34.3253079168231, 34.40256142427867]
dataset	DataFrame	(880758, 27)	Column names: Unnamed: 0, DR Number, Date Reported, Date Occurred, Tim ...
scores	list	2	[1.7368069640318604, 0.3440256142427867]
train	DataFrame	(880758, 6)	Column names: Crime Code, Victim Age, Area ID, Gender_F, Gender_M, Gen ...
y_full	int64	(880758,)	Min: 1

```
660568/660568 [==============================] - 288s 436us/step - loss: 1.7403 - acc: 0.3404
Epoch 4/10
660568/660568 [==============================] - 256s 388us/step - loss: 1.7435 - acc:
0.34177990/660568 [=======================>......] - ETA: 43s - loss: 1.7436 - acc: 0.3415
Epoch 5/10
660568/660568 [==============================] - 263s 398us/step - loss: 1.7419 - acc: 0.3423
Epoch 6/10
660568/660568 [==============================] - 260s 394us/step - loss: 1.7400 - acc: 0.3428
ETA: 1:48 - loss: 1.7403 - acc: 0.3428 - ETA: 1:23 - loss: 1.7400 - acc: 0.3428
Epoch 7/10
660568/660568 [==============================] - 250s 378us/step - loss: 1.7391 - acc: 0.3430
Epoch 8/10
660568/660568 [==============================] - 269s 407us/step - loss: 1.7382 - acc: 0.3436
Epoch 9/10
660568/660568 [==============================] - 258s 390us/step - loss: 1.7377 - acc: 0.3432
Epoch 10/10
660568/660568 [==============================] - 251s 380us/step - loss: 1.7394 - acc: 0.3430
Evaluating train dataset:
acc: 34.33%
Evaluating dev dataset:
acc: 34.40%
34.36% (+/- 0.04%)

In [9]:
```



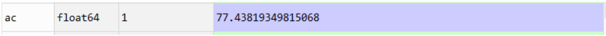
Name	Type	Size	Value
X_full	float64	(880758, 25)	Min: 0.0 Max: 21.0
X_test	float64	(220190, 25)	Min: 0.0 Max: 21.0
X_train	float64	(660568, 25)	Min: 0.0 Max: 21.0
accuracies	list	2	[35.86337818369598, 36.0179844681219]
dataset	DataFrame	(880758, 27)	Column names: Unnamed: 0, DR Number, Date Reported, Date Occurred, Tim ...
scores	list	2	[1.7183226979868662, 0.36017984468121905]
train	DataFrame	(880758, 26)	Column names: Crime Code, Victim Age, Area ID, Gender_F, Gender_M, Gen ...
y_full	int64	(880758,)	Min: 1

```
660568/660568 [==============================] - 310s 478us/step - loss: 1.7258 - acc: 0.3564
ETA: 48s - loss: 1.7264 - acc: 0.3563
Epoch 4/10
660568/660568 [==============================] - 246s 372us/step - loss: 1.7228 - acc: 0.3573
Epoch 5/10
660568/660568 [==============================] - 252s 381us/step - loss: 1.7216 - acc: 0.3578
Epoch 6/10
660568/660568 [==============================] - 256s 388us/step - loss: 1.7207 - acc: 0.3583
ETA: 1:39 - loss: 1.7197 - acc: 0.3592
Epoch 7/10
660568/660568 [==============================] - 248s 375us/step - loss: 1.7203 - acc: 0.3581
Epoch 8/10
660568/660568 [==============================] - 245s 371us/step - loss: 1.7200 - acc: 0.3582
Epoch 9/10
660568/660568 [==============================] - 265s 402us/step - loss: 1.7197 - acc: 0.3580
Epoch 10/10
660568/660568 [==============================] - 261s 395us/step - loss: 1.7196 - acc: 0.3580
Evaluating train dataset:
acc: 35.86%
Evaluating dev dataset:
acc: 36.02%
35.94% (+/- 0.08%)

In [15]: 
```

C. *Predicting Crime Categories*

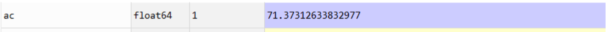
We categorized crimes into two major subtypes: violent and non-violent. We tried to predict the crime category using the victim information. This can provide useful insights into what type of crime each age group is targeted to. Below are the results:

- **Decision Trees**:



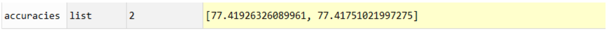
ac	float64	1	77.43819349815068

- **K-NN**:



ac	float64	1	71.37312633832977

- **Neural Networks**:

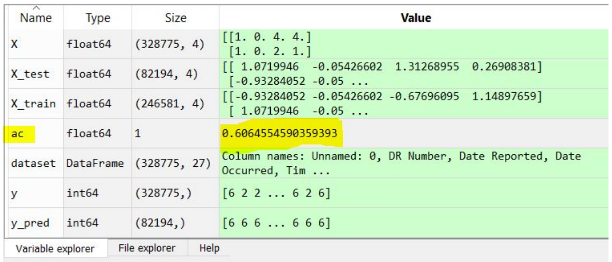


accuracies	list	2	[77.41926326089961, 77.41751021997275]

D. *Using the 'Weapon Used Code' feature*

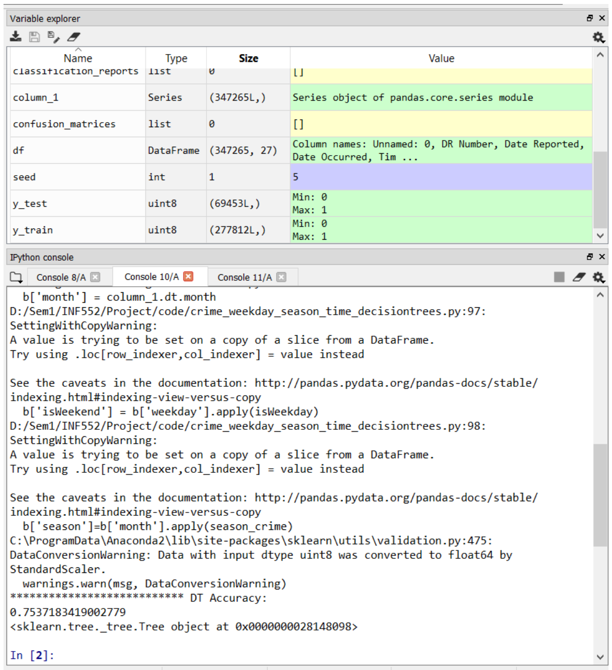
The data contains information regarding the weapon used for crime: Weapon Used Code, it's description. EDA indicated that Weapon Used Code is highly correlated to the type of crime committed. But, it is fairly easy to deduce the crime type if we know the type of weapon used in it. Below results prove that point, as accuracies to the previous questions drastically shoot up once this information is introduced in the input features.

- **Prediction of Crime Code given victim information**
  - **Logistic Regression**:



Name	Type	Size	Value
X	float64	(328775, 4)	[[1. 0. 4. 4.] [1. 0. 2. 1.]
X_test	float64	(82194, 4)	[[1.0719946 -0.05426602 1.31268955 0.26908381] [-0.93284052 -0.05 ...
X_train	float64	(246581, 4)	[[-0.93284052 -0.05426602 -0.67696095 1.14897659] [1.0719946 -0.05 ...
ac	float64	1	0.6064554590359393
dataset	DataFrame	(328775, 27)	Column names: Unnamed: 0, DR Number, Date Reported, Date Occurred, Tim ...
y	int64	(328775,)	[6 2 2 ... 6 2 6]
y_pred	int64	(82194,)	[6 6 6 ... 6 6 6]

- **Prediction of time occurred given weekday and Crime Code**
  - **Decision Trees**:



Name	Type	Size	Value
classification_reports	list	0	[]
column_1	Series	(347265L,)	Series object of pandas.core.series module
confusion_matrices	list	0	[]
df	DataFrame	(347265, 27)	Column names: Unnamed: 0, DR Number, Date Reported, Date Occurred, Tim ...
seed	int	1	5
y_test	uint8	(69453L,)	Min: 0 Max: 1
y_train	uint8	(277812L,)	Min: 0 Max: 1

```
  b['month'] = column_1.dt.month
D:/Sem1/INF552/Project/code/crime_weekday_season_time_decisiontrees.py:97:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
indexing.html#indexing-view-versus-copy
  b['isWeekend'] = b['weekday'].apply(isWeekday)
D:/Sem1/INF552/Project/code/crime_weekday_season_time_decisiontrees.py:98:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
indexing.html#indexing-view-versus-copy
  b['season']=b['month'].apply(season_crime)
C:\ProgramData\Anaconda2\lib\site-packages\sklearn\utils\validation.py:475:
DataConversionWarning: Data with input dtype uint8 was converted to float64 by
StandardScaler.
  warnings.warn(msg, DataConversionWarning)
*************************** DT Accuracy:
0.7537183419002779
<sklearn.tree._tree.Tree object at 0x0000000028148098>

In [2]:
```

VI. RELEVANT WORK

The pioneer work in this field has been done by the team behind what is today known as PredPol. The introductory paper "Randomized Controlled Field Trials of Predictive Policing"[1] was first published in 2015. PredPol uses Crime type, Crime location, Crime date and time features and proprietary machine learning algorithms for forecasting. Predictive Policing: The Role of Crime Forecasting in Law Enforcement Operations [2] details the predictive policing strategies and their use in Law Enforcement. Predictive Policing: A Review of the Literature [3] from Portland State University reviews all the literature on predictive policing. Recent advancements include Partially Generative Neural Networks for Gang Crime Classification with Partial Information [4].

VII. INDIVIDUAL CONTRIBUTIONS

- **Naga Ritwik Indugu**: Prediction of Crime Code given victim information
- **Uma Kanumuri**: Prediction of time occurred given weekday and Crime Code
- **Mansi Ganatra**: Predicting Crime Categories

ACKNOWLEDGMENT

We thank Prof. Ion Muslea for guiding this project and USC DPS Officer Wyman Thomas for helping us with the understanding of the data and domain.

REFERENCES

[1] G. O. Mohler, M. B. Short, Sean Malinowski, Mark Johnson, G. E. Tita, Andrea L. Bertozzi & P. J. Brantingham (2015) Randomized Controlled Field Trials of Predictive Policing, Journal of the American Statistical Association, 110:512, 1399-1411, DOI: 10.1080/01621459.2015.1077710

[2] Perry, Walter L., Brian McInnis, Carter C. Price, Susan Smith, and John S. Hollywood, Predictive Policing: The Role of Crime Forecasting in Law Enforcement Operations. Santa Monica, CA: RAND Corporation, 2013. https://www.rand.org/pubs/research_reports/RR233.html. Also available in print form.

[3] Portland State University. Criminology and Criminal Justice Senior Capstone, ”Predictive Policing: A Review of the Literature” (2012). Criminology and Criminal Justice Senior Capstone Project. 5.

[4] Seo, Sungyong, Hau Kong Chan, P. Jeffrey Brantingham, Jorja Leap, Phebe Vayanos, Milind Tambe and Yan Liu. Partially Generative Neural Networks for Gang Crime Classification with Partial Information. (2017).

[5] http://michael-harmon.com/blog/crimetime.html

[6] https://machinelearningmastery.com/