Crime data analysis and prediction in Los Angeles

CS-267 Topics in Databases

-Akshay Ravi (akshay.ravi@sjsu.edu)
- Brandon Palomino (brandon.palomino@sjsu.edu)
Team-2

https://colab.research.google.com/drive/1eQs50Kjuj34CJOyXOZ8_Eu8hhQblzXcq?usp=sharing

1. Project Intro & Goals

Introduction

Any city in any part of the world faces one common problem, i.e. crimes. Normal people are the most affected by this as people lose lives and livelihoods. In earlier days the police force resorted to take action on the criminals after a crime is reported. This proved to be help in bringing them to justice but did not help in reducing the crimes. To prevent crimes, the police needed to know where a crime is more likely to take place and position the police force accordingly.

With the advent of machine learning algorithms that help with classification and prediction, crimes could be predicted if we had sufficient data of past crimes. In the last decade, the collection of data has moved from papers to documents in the cloud. Every police department maintains a record of the crimes committed in the area, accused and victim information and event description. This data is available to everyone online and any individual can perform an analysis of it. With more and more data being available everyday, classification and prediction algorithms can improve their accuracies.

In this project, we aim to classify predict the crimes that happen in a particular city, Los Angeles, and try to offer insights into where a crime can take place, what the top crimes taking place are and classify crimes based on their type and based on whether they are violent or not. This information can play a huge role in letting the police assign officers during those times of the day and be on the lookout specifically for a crime. This has the potential to reduce a significant number of crimes.

Literature survey

For this project, we are focussing on papers that classify and predict crime using various machine learning methods.

In [1], the authors used two classification methods, namely Naive Bayesian and Decision Tree to predict crime in US states using WEKA, an open source tool in JAVA. They chose 12 out of a dataset of 28 attributes and added a new nominal attribute called ‘Crime Category’ with three values, ‘Low’, ‘Medium’, and ‘High’[1]. These values were decided using the percentage of violent crimes per population i.e. ‘Violent Crimes Per Pop’. For Decision Tree, the Accuracy, Precision and Recall are 83.9519%, 83.5% and 84%. Whereas the accuracy, precision and recall values for Naïve Bayesian are 70.8124%, 66.4% and 70.8%, respectively. Although the accuracy was surprisingly very high for decision tree, a con was that a small change in data would result in a big change in the structure. Potential extensions include plans to further apply other classification algorithms on the crime data set and evaluate their prediction performances.

The authors of [2] focussed on filling the missing the data first and then predicting crimes. Usually papers fill data manually however it takes a long time although it can be accurate. So they used 3 algorithms i.e. Maximum class filling algorithm, Roulette filling algorithm and GBWKNN filling algorithm to obtain the real crime dataset. For classification, they used C4.5 algorithm, Naive Bayesian algorithm and KNN algorithm and observed that the highest accuracy of 72.95% was achieved by combining GBWKNN filling algorithm (K=70) and KNN classification algorithm. However, the issue with this approach was that it was heavily reliant on a good fitting algorithm. So, if the data and features change a little bit, the fitting algorithm might not fill the data well and in turn, the prediction accuracy will be affected.

Literature survey

In [3], shojaee et al. [3] proposed a model where crimes were distinguished as critical and non-critical and the updated crime set was used to predict using multiple methods, out of which KNN rendered 87% accuracy. This method performed well for the given preprocessed data but a lot of data is lost while classifying crimes on a very general basis as done above.

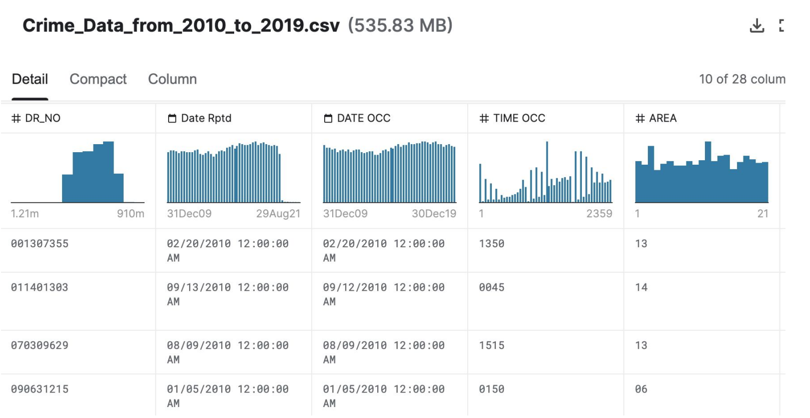
In [4], the authors used prediction several prediction models to calculate the future forecasting of potential crime happening. For this particular paper, 10 different models of analysis were used. Models like decision tree, KNN, Naive Bayes, Regression Model, SVM, and random forest regressor were studied. The accuracies are 59.15%, 66.69%, 87.0%, 42%, 84.37% and 97% respectively. Although the accuracy was surprisingly very high for random forest, a con was that newly added date could interfere with location of future crime incidents. After conducting the tests of these models, it was concluded that police should integrate newer technologies in order to improve the accuracy of these models. One suggested proposal was facial recognition since it could detect certain individuals who might commit a crime more. This proposed system will then be monitored more closely as more data is gathered to make a prediction on certain individuals. In other words, the flow chart of the proposed system first processes the data, determines a threat detection level, then classifies the threat, simulates a scenario, and then finally briefing authorities with a 60 word description.

Literature survey

The authors of [5] focussed on investigating the capability of Deep Learning methods to forecast hotspot areas in an urban environment, where crimes of certain types are more likely to occur in a defined future window. This goal is achieved when Deep Learning methods are fed with the minimum amount of data containing only spatial, temporal and crime type information. Across all methods, most accuracies averaged up between 88% to 100% accuracy. While it is true that most of the deep learning methods provided a high accuracy, a fallout from this is that this model might be bias due to tests happening within a small area radius. Models better understand the order of “hotness” with a dual output setting where the second output is the number of crimes that occurred in the same future window. In other words, the incorporation of temporal semantics is the main goal in order to predict crime fluncationations. Although preprocessing wasn’t discussed, the paper acknowledges that is is important to the overall crime classification process.

Crime Dataset Background

- 619.48 MB dataset.
- csv file format.
- https://www.kaggle.com/datasets/chaitanyakck/crime-data-from-2020-to-present?select=Crime_Data_from_2010_to_2019.csv
- Dataset reflects incidents of crime in Los Angeles from 2010 to present.
  - 1st Dataset: Crime data from 2010 to 2019
  - 2nd Dataset: Crime data from 2020 to present.
- Dataset transcribed from crime reports from paper.
  - Data cleaning is required.
  - i.e. missing location fields, etc.



Coding, Tools, and Imports

- Coding Language: Python
- Web IDE: Google Colaboratory
- Imported python libraries:
  - Pandas: python library for data manipulation and analysis
  - Matplotlib: python library for plotting data
  - Numpy: python library for multi-dimensional analysis
  - Scikit-learn: software machine learning library for python (ML algorithms)
  - Tensorflow: deep learning framework from Google; open source python library

2. Data Preprocessing

Data Dimensions

Checking current dimensionality of initial data set:

```
#Checking the number of rows and columns in the dataset
crime_data.shape
```

```
(2444416, 29)
```

Preview of columns and data types

```
Index(['DR_NO', 'Date Rptd', 'DATE OCC', 'TIME OCC', 'AREA ', 'AREA NAME',
       'Rpt Dist No', 'Part 1-2', 'Crm Cd', 'Crm Cd Desc', 'Mocodes',
       'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis Desc',
       'Weapon Used Cd', 'Weapon Desc', 'Status', 'Status Desc', 'Crm Cd 1',
       'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'LOCATION', 'Cross Street', 'LAT',
       'LON', 'AREA'],
      dtype='object')
```

Data cleaning-Ages

In the dataset, there was crime data with age values less than 0. These ages are considered to be noise, thus they were removed.

```
crime_data['Vict Age'].value_counts()
```

```
 0       449277
 25       54639
 26       54257
 27       53871
 28       53612
          ...
-10           2
-11           1
 114          1
 118          1
 120          1
Name: Vict Age, Length: 113, dtype: int64
```

```
#We could see that there were age values less than 0. So we removed them.
crime_data=crime_data[crime_data['Vict Age']>0]
```

Data Cleaning (cont.): LAT & LON

There are crime data observations that have latitudes and longitudes wrongly marked with the wrong numbers (i.e. LAT < 33, LAT > 35, LON < -119, LON > -117) as shown below:

```
#Checking how many latitudes and longitiudes are wrongly marked
crime_data[(crime_data['LAT']<33)|(crime_data['LAT']>35)|(crime_data['LON']<-119)|(crime_data['LON']>-117)]
```

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	...	Status Desc	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4	LOCATION	Cross Street	LAT	LON	AREA
49703	100618355	07/14/2010 12:00:00 AM	07/12/2010 12:00:00 AM	1900	6.0	Hollywood	665	1	330	BURGLARY FROM VEHICLE	...	Invest Cont	330.0	NaN	NaN	NaN	900 N CISTRUS AV	NaN	0.0	0.0	NaN
60870	100718479	11/29/2010 12:00:00 AM	11/29/2010 12:00:00 AM	1630	7.0	Wilshire	709	1	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	...	Invest Cont	230.0	998.0	NaN	NaN	HARBOR	CLINTON	0.0	0.0	NaN
85026	101016365	09/09/2010 12:00:00 AM	08/23/2010 12:00:00 AM	1500	10.0	West Valley	1000	2	626	INTIMATE PARTNER - SIMPLE ASSAULT	...	Invest Cont	626.0	NaN	NaN	NaN	CITY OF WINNETKA	CITY OF WINNETKA	0.0	0.0	NaN
205973	120215454	07/31/2012 12:00:00 AM	01/01/2010 12:00:00 AM	1400	2.0	Rampart	289	2	812	CRM AGNST CHLD (13 OR UNDER) (14-15 & SUSP 10 ...	...	Invest Cont	812.0	NaN	NaN	NaN	1100 S UNION AV	NaN	0.0	0.0	NaN
206921	122113857	07/30/2012 12:00:00 AM	03/18/2010 12:00:00 AM	1300	21.0	Topanga	2197	1	341	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LI...	...	Invest Cont	341.0	NaN	NaN	NaN	4800 QUEEN VICTORIA RD	NaN	0.0	0.0	NaN

Because of these inconsistencies, they are removed from the dataset.

```
#We noticed that there were 2573 latitudes and longitudes values that were not mapped to LA. So we dropped them.
crime_data=crime_data[(crime_data['LAT']>33)|(crime_data['LAT']<35)|(crime_data['LON']>-119)|(crime_data['LON']<-117)]
```

Data Cleaning-Area & Sex

Imported dataset had two columns for “AREA”, thus one of them was removed:

```
#Next, we noticed two repetitive columns for "AREA". So we removed the extra one.
crime_data=crime_data.drop(columns=['AREA'])
```

Under victim’s sex, there were several other sexes other than male or female. Thus they were replaced with “O” to aid in classification.

```
#We replaced sexes otheer than "M' and 'F" with 'O'
crime_data['Vict Sex']=crime_data['Vict Sex'].replace(['X','H','-','N'],'O')
```

Data Cleaning-Victim Descent

There were NULL values listed under Victim Descent column. We filled these values with ‘-’.

```
crime_data['Vict Descent'].unique()
```

```
array(['H', 'W', 'B', 'A', 'O', 'K', 'I', 'X', 'J', 'F', 'C', 'P', 'V',
       nan, 'U', 'G', 'D', 'S', 'Z', 'L', '-'], dtype=object)
```

```
crime_data["Vict Descent"].fillna("-",inplace=True)
crime_data["Mocodes"].fillna("0",inplace=True)
crime_data["Crm Cd 1"].fillna(0,inplace=True)
crime_data["Crm Cd 2"].fillna(0,inplace=True)
crime_data["Crm Cd 3"].fillna(0,inplace=True)
crime_data["Crm Cd 4"].fillna(0,inplace=True)
crime_data["AREA "].fillna(0,inplace=True)
crime_data["Cross Street"].fillna("N/A",inplace=True)
crime_data["Premis Cd"].fillna(0,inplace=True)
crime_data["Premis Desc"].fillna("Unknown",inplace=True)
crime_data["Weapon Used Cd"].fillna(0,inplace=True)
crime_data["Weapon Desc"].fillna("NO WEAPON",inplace=True)
crime_data["Status"].fillna(0,inplace=True)
```

Data Cleaning-Duplicates

Duplicates were removed from the dataset:

```
crime_data.drop_duplicates(inplace=True)
```

```
crime_data.shape
```

```
(1994488, 28)
```

The final dimensionality of the cleaned data set after cleaning is-

Number of rows=1,994,488, Number of columns=28

Data preprocessing-Mapping to numerical values

We mapped the Victim Sex and Victim Descent columns to numbers to help in further parts of the project.

```
crime_data['Vict Sex']=crime_data['Vict Sex'].replace(['M'],1)
crime_data['Vict Sex']=crime_data['Vict Sex'].replace(['F'],2)
crime_data['Vict Sex']=crime_data['Vict Sex'].replace(['O'],3)
crime_data['Vict Sex'].unique()
```

```
[56] crime_data['Vict Descent']=crime_data['Vict Descent'].replace(['H', 'W', 'B'
     crime_data['Vict Descent'].unique()

array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 20, 14, 15, 16,
        17, 18, 19])
```

3. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA)

EDA refers to the process of performing an initial study of the data to discover details, patterns, differences, anomalies, etc. It is also used to get summary statistics and graphical representations of the data.

We performed the following analysis on the data-

1. Vizualizing the number of crimes for different ages
2. Vizualizing the number of crimes for different victim sexes
3. Vizualizing the number of crimes for different victim descents
4. Vizualizing the number of crimes in different areas
5. Vizualizing the number of crimes in different years,months and days
6. Word cloud for premis desc
7. Word cloud for crime desc
8. Vizualizing the frequencies of different types of crimes in different

Exploratory Data Analysis: Ages

```
#Vizualizing the number of crimes for different ages
crime_age_num=crime_data.groupby(['Vict Age'])['Vict Age'].count()
crime_age_num.sort_values(ascending=False,inplace=True)
```

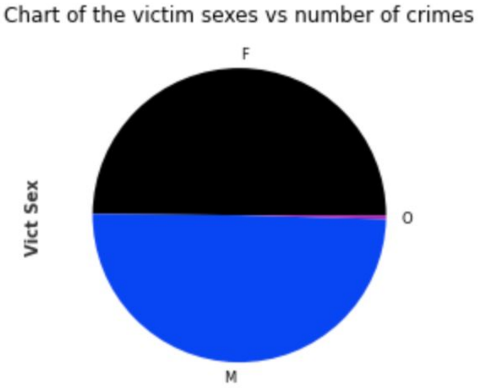
- Victim’s age to whom crimes most occur (age: 25)
- Victim’s age to whom crimes least occur (age: 120)

```
The top 10 victim ages to whom crimes most occur on are the following
 Vict Age
25     54639
26     54257
27     53871
28     53612
29     53146
24     52715
30     52703
23     50776
31     50152
32     48635
```

```
The top 10 victim ages to whom crimes least occur on are the following
 Vict Age
92      554
93      492
94      367
95      283
96      219
97      193
98      144
114       1
118       1
120       1
```

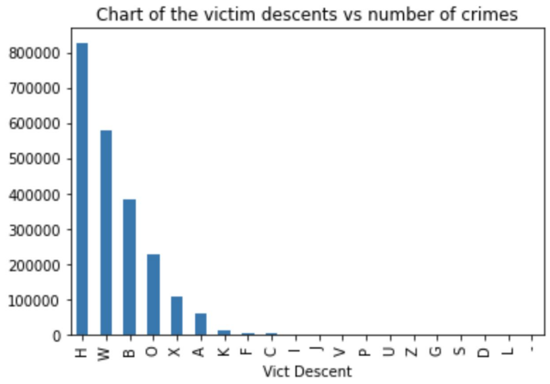
Exploratory Data Analysis cont.

- Victim's Sex vs Number of Crimes



Indicates a 50/50 split chart between male and female. There are slightly more males involved in crimes than females.

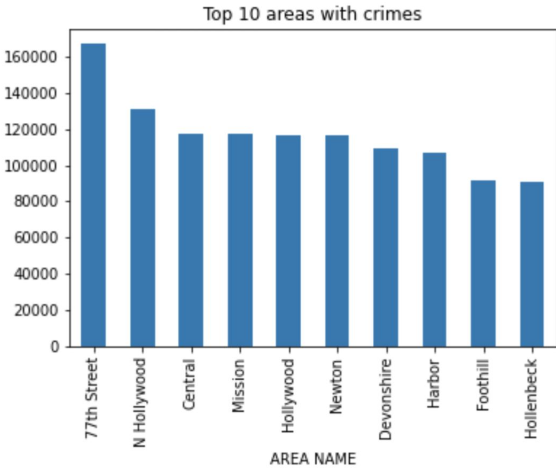
- Victim Descents vs Number of Crimes



Highest victime descent is H, W, B and O.

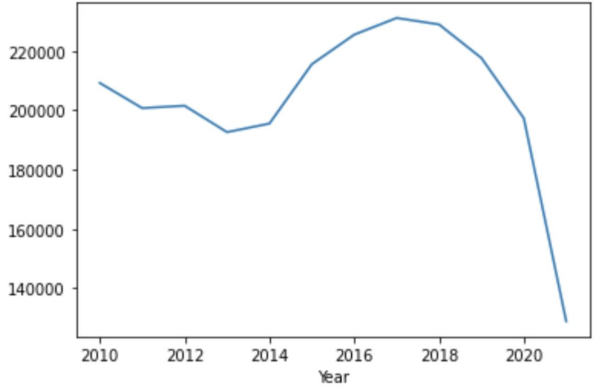
Exploratory Data Analysis cont.

- Top 10 areas with crimes occurring most



Most crime has been occuring in the Hollywood area and downtown Los Angeles.

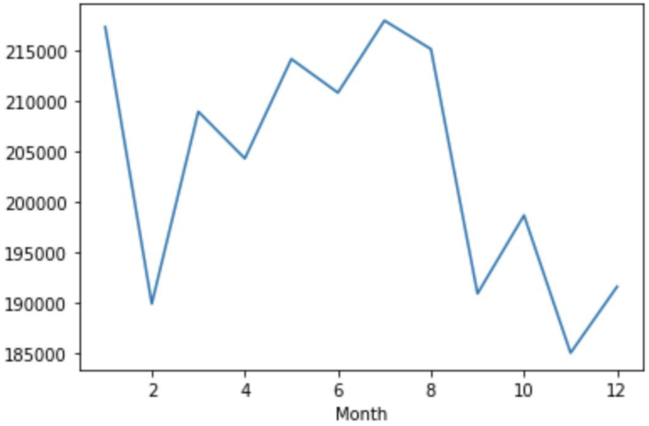
- Frequency of crime based by year



Crime frequency was on a downward rate before 2014, but picked up until 2018. Less crimes recorded in 2020 data since dataset was conducted during that year.

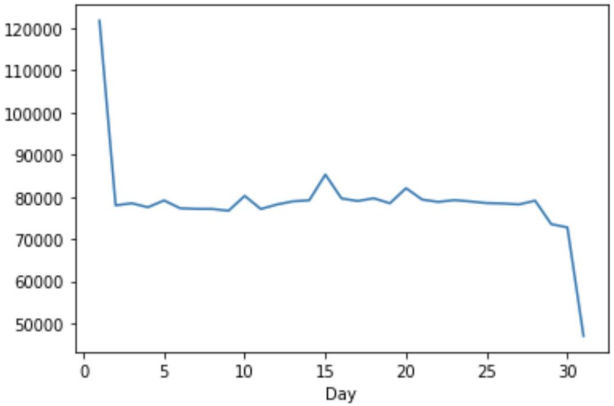
Exploratory Data Analysis cont.

- Frequency of crimes based on month



Crime more likely occurred during the summer months (6, 7, and 8) while it was much lower in the fall and winter months.

- Frequency of crimes based on days of occurrence



Crime numbers are high towards the beginning of the month, but then decrease overtime.

Exploratory Data Analysis cont.

- Common Premise Description

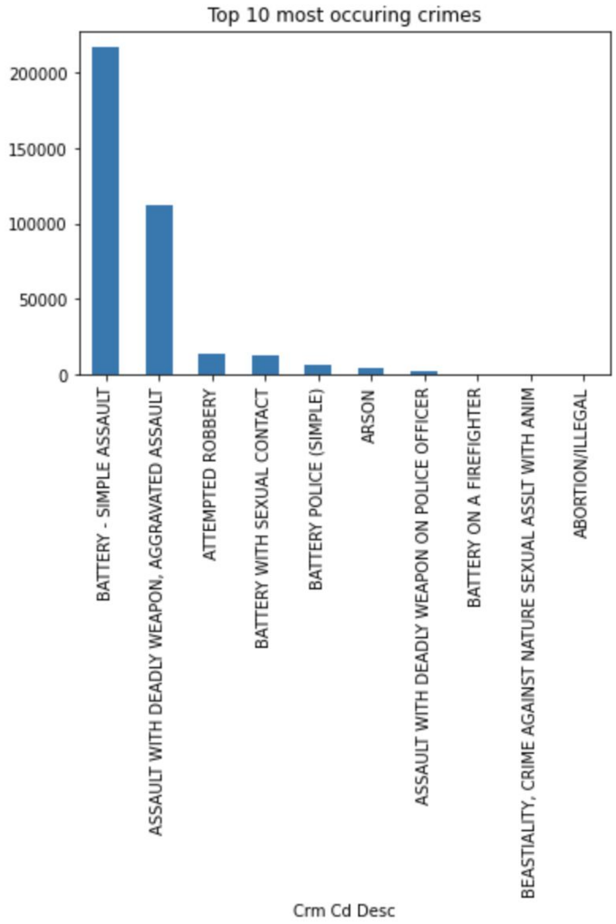
Crimes are often described to be taken place in an individual's home or outside in public. Involves cars and buildings.

- Common Crime Type

Most common crime types are assault, theft, burglary, and vandalism.

Top 10 most occurring crimes

- Battery Simple Assault
  - > 200,000 cases
- Assault with Deadly Weapon/Aggravated Assault
  - est. 100,000 cases
- Attempted robbery, Arson, Abortion, etc.
  - *remainder of cases



4. Data Modeling & Time Series Forecasting

Data preparation for forecasting

For time series analysis, we need the date column to be along with another feature. We chose to project the number of crimes happening per day and predict it using various techniques.

We chose the ‘DATE OCC’ column and extracted the number of unique dates from it. We then calculated the number of crimes using value_counts().

```
[130] #date vs number of crimes on that date
      time_df=pd.DataFrame(columns=['ds','y'])
      time_df['ds']=crime_data['DATE OCC'].unique()
      time_df['y']=crime_data['DATE OCC'].value_counts().values
```

```
time_df.head()
```

	ds	y
0	2010-02-20	2154
1	2010-01-05	2090
2	2010-01-02	1721
3	2010-01-04	1528
4	2010-01-07	1446

Prophet

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It handles outliers well and is known for taking care of missing values as well. It is also known for seasonality based prediction as well.

We trained the prophet model with the the time dataframe.

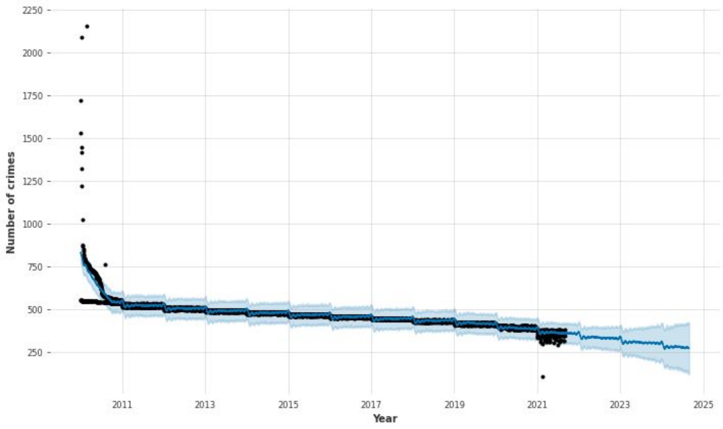
```
[133] #FB Prophet model
      fbp=Prophet()
      fbp.fit(time_df)

      2022-05-05 21:21:06 fbprophet INFO: Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
      <fbprophet.forecaster.Prophet at 0x7ff383ba5e90>
```

Prophet

We then predicted the number of crimes over the next 3 years. The graph shown a downwards trend in the number of crimes in LA.

```
# creating a df to predict crimes 3 years ahead of the current date
future_crimes=fbp.make_future_dataframe(periods=365*3)
crime_pred=fbp.predict(future_crimes)
crime_pred.head()
```

```
fbp.plot(crime_pred)
plt.xlabel("Year")
plt.ylabel("Number of crimes")
```



5. Dataset Classification and Prediction using Machine Learning

Classification

There are two main types of classification problems. They are the following-

1. Binary class classification
2. Multi class classification

Binary class refers to classification problems that have two class labels. Common problems that use binary class classification are marking emails as spam or not spam.

Multi class classification refers to those classification problems that have more than two class labels. Their examples include a range of possible values to predict.

Common algorithms

Popular algorithms that can be used for both classification include:

- Logistic Regression
- Naive Bayes
- k-Nearest Neighbors.
- Decision Trees.
- Support Vector Machine

This involves using a strategy of fitting multiple binary classification models for each class vs. all other classes (called one-vs-rest) or one model for each pair of classes (called one-vs-one).

- **One-vs-Rest**: Fit one binary classification model for each class vs. all other classes.
- **One-vs-One**: Fit one binary classification model for each pair of classes.

5.1. Binary Class Classification

Data preparation-Categorization

For binary class classification, we need to split the predicted value ‘y’ to binary values. Hence, we categorized the crimes in the dataset into two different groups, violent or not violent.

This was done based on whether a weapon was used in the crime or not. If it was then it is considered a violent crime and if not, it is non-violent.

```
[78] crime_data.loc[crime_data['Weapon Desc']=='NO WEAPON','Violent']=0

[79] crime_data["Violent"].fillna(1,inplace=True)

crime_data['Violent'].value_counts()

0.0    1207385
1.0     787103
Name: Violent, dtype: int64
```

Data preparation-Feature selection

In order to classify the data well, we need to choose significant features from the dataset. Moreover, we need to make sure that we don’t use repeated features like multiple columns of the same category. After some speculation, we decided to go with the following features-[‘Rpt Dist No','Crm Cd','Vict Age','Vict Sex','Vict Descent','Premis Cd','LAT','LON','Violent']

```
crime_data_binary.head()
```

	Rpt Dist No	Crm Cd	Vict Age	Vict Sex	Vict Descent	Premis Cd	LAT	LON	Violent
0	1385	900	48	1	1	501.0	33.9825	-118.2695	0.0
3	646	900	47	2	2	101.0	34.1016	-118.3295	1.0
4	176	122	47	2	1	103.0	34.0387	-118.2488	1.0
5	162	442	23	1	3	404.0	34.0480	-118.2577	0.0
6	182	330	46	1	1	101.0	34.0389	-118.2643	0.0

Data preparation-Test train split

We set our X and y to the following-

X=crime_data_binary.drop(columns=['Violent'])

y=crime_data_binary['Violent']

We used sklearn's train_test_split to make our test dataset and train dataset and the standard scaler to scale our values to smaller values.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

ML algorithms used

1. Logistic Regression
2. Gaussian Naive Bayes Classifier
3. Bernoulli Naive Bayes Classifier
4. Random Forest Classifier
5. K-Nearest Neighbours
6. Neural networks

Logistic Regression

Def: A supervised learning algorithm used for classification problems. Converts prediction to a probability of an observation. Performs classification based on probability. Usually used for binary classification.

```
lr=LogisticRegression(max_iter=5000)
lr.fit(X_train,y_train)
y_pred_lr=lr.predict(X_test)
cm=confusion_matrix(y_test,y_pred_lr)
print("Logistic Regression accuracy:",accuracy_score(y_test,y_pred_lr))
```

Accuracy: 0.6290983028536505

Gaussian Naive Bayes Classifier

Def: A classification machine learning algorithm. A generalization of the Gaussian probability distribution for classification or regression. i.e. a normal distribution.

```
from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
gnb.fit(X_train,y_train)
y_pred_gnb=gnb.predict(X_test)
print("GNB accuracy:",accuracy_score(y_test,y_pred_gnb))
#gnb.score(y_test,y_pred_gnb)
```

GNB accuracy: 0.6234776810111858

Bernoulli Naive Bayes Classifier

Def: A classification algorithm based on Bayes theorem which gives the likelihood of occurrence of event or data. It is a probabilistic classifier for all different classes.

```
from sklearn.naive_bayes import BernoulliNB
#X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)
BNB=BernoulliNB(binarize=0.0)
BNB.fit(X_train,y_train)
y_pred_bnb=BNB.predict(X_test)
print("BNB accuracy:",accuracy_score(y_test,y_pred_bnb))
#BNB.score(y_test,y_pred_bnb)
```

BNB accuracy: 0.6711941398553014

Neural Networks

Def: Neural networks is modelled inherently to work with both, binary and multi-class classification problems. Consist of perceptrons. Mimic brain neurons.

Neural Networks

```
from sklearn.neural_network import MLPClassifier
NN=MLPClassifier(solver='lbfgs',alpha=1e-5,hidden_layer_sizes=(5,2),random_state=1,max_iter=5000)
NN.fit(X_train,y_train)
y_pred_nn=NN.predict(X_test)
print("Neural networks accuracy:",accuracy_score(y_test,y_pred_nn))
```

Neural networks accuracy: 0.848179233789089

K Nearest neighbours

Def: Machine learning algorithm that is a non-parametric supervised learning method. Used for both classification and regression problems. Input consists of k closest training examples in a data set.

```
knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,y_train)
y_pred_knn=knn.predict(X_test)
print("KNN accuracy:",accuracy_score(y_test,y_pred_knn))
```

KNN accuracy: 0.923519295659542

Random Forest Classifier

Def: A classifier that contains a number of decision trees on various subsets of a given dataset. Takes the average to improve the predictive accuracy of dataset.

```
#n_estimators is the number of trees you want to build before taking the maximum voting or averages of predictions.
#Higher number of trees give you better performance but makes your code slower.
rf=RandomForestClassifier(n_estimators=150)
rf.fit(X_train,y_train)
y_pred_rf=rf.predict(X_test)
print("Random Forest Accuracy:",accuracy_score(y_test,y_pred_rf))
```

Random Forest Accuracy: 0.94885158611976

5.2. Multi-class Classification

Data preparation-Categorization

In order to classify crimes based on multiple labels, we used the LAPD’s crime coding document to map the ranges of crime types.

```
def crime_mapping(i):
 if i in range(207,211): return 0 #kidnap
 if i in range(220,222): return 1 #intent to murder or commit felony
 elif i in range(230,231): return 2 #crime against employers
 elif i in range(236,238): return 3 #Human trafficking
 elif i in range(302,311): return 4 #crime against religion and offence against good morals
 elif i in range(346,368): return 5 #injuries
 elif i in range(403,423): return 6 #crimes against public peace
 Else: return 7 #others
```

Data preparation-Feature selection

In order to classify the data well, we need to choose significant features from the dataset. Moreover, we need to make sure that we don't use repeated features like multiple columns of the same category. After some speculation, we decided to go with the following features-['Rpt Dist No','CrimeType,'Vict Age','Vict Sex','Vict Descent','Premis Cd','LAT','LON']. These were the same features used for binary as well, expect for the added crime type mapping.

	Rpt Dist No	Vict Age	Vict Sex	Vict Descent	Premis Cd	LAT	LON	Crime Type
0	1385	48	1	1	501.0	33.9825	-118.2695	7
3	646	47	2	2	101.0	34.1016	-118.3295	7
4	176	47	2	1	103.0	34.0387	-118.2488	7
5	162	23	1	3	404.0	34.0480	-118.2577	7
6	182	46	1	1	101.0	34.0389	-118.2643	7
...	...	...	...	...	...	...	...	...
326206	2143	44	1	2	101.0	34.1855	-118.6296	7
326207	1524	38	2	1	108.0	34.1867	-118.3965	7
326209	564	41	2	3	502.0	33.7424	-118.2814	7
326210	1798	40	1	1	501.0	34.2302	-118.4775	7
326211	363	15	2	1	101.0	34.0088	-118.3351	5

1994488 rows × 8 columns

Data preparation-Test train split

We set our X and y to the following-

X=crime_data_multi.drop(columns=['Crime Type'])

y=crime_data_multi['Crime Type']

We used sklearn's train_test_split to make our test dataset and train dataset and the standard scaler to scale our values to smaller values.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

One vs Rest Classifier

The One-vs-Rest strategy splits a multi-class classification into one binary classification problem per class.

```
from sklearn.linear_model import Perceptron
from sklearn.multiclass import OneVsRestClassifier
ovr=OneVsRestClassifier(estimator=Perceptron())
_ = ovr.fit(X_train,y_train)
y_pred_ovr=ovr.predict(X_test)
print("Multi class One vs rest estimator accuracy:",accuracy_score(y_test,y_pred_ovr))
#print(len(ovr.estimators_))
```

Multi class One vs rest estimator accuracy: 0.7083038771816355

Logistic Regression

Logistic Regression can be modelled for multi-class classification by setting the multi-class parameter to "multinomial".

```
lr=LogisticRegression(random_state=0,solver='lbfgs',multi_class='multinomial',max_iter=5000)
lr.fit(X_train,y_train)
y_pred_lr=lr.predict(X_test)
#cm=confusion_matrix(y_test,y_pred_lr)
print("Multi class Logistic Regression accuracy:",accuracy_score(y_test,y_pred_lr))
```

Multi class Logistic Regression accuracy: 0.7939949561040667

Bernoulli Naive Bayes Classifier

NBC works inherently with multiclass classification without any changes to its function call.

```
from sklearn.naive_bayes import BernoulliNB
BNB=BernoulliNB(binarize=0.0)
BNB.fit(X_train,y_train)
y_pred_bnb=BNB.predict(X_test)
print("Multi class BNB accuracy:",accuracy_score(y_test,y_pred_bnb))
```

Multi class BNB accuracy: 0.6998255193056871

Neural Networks

Neural networks is modelled inherently to work with both, binary and multi-class classification problems. We set the number of hidden layers to (150,10) with a learning rate of 1e-5 to train our model.

```
from sklearn.neural_network import MLPClassifier
NN=MLPClassifier(solver='lbfgs',alpha=1e-5,hidden_layer_sizes=(150,10),random_state=1,max_iter=5000)
NN.fit(X_train,y_train)
y_pred_nn=NN.predict(X_test)
print("Multi class classification Neural networks accuracy:",accuracy_score(y_test,y_pred_nn))
```

Multi class classification Neural networks accuracy: 0.9285782330320032

K Nearest neighbours

Def: Machine learning algorithm that is a non-parametric supervised learning method. Used for both classification and regression problems. Input consists of k closest training examples in a data set.

```
knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,y_train)
y_pred_knn=knn.predict(X_test)
print("Multi class KNN accuracy:",accuracy_score(y_test,y_pred_knn))
```

Multi class KNN accuracy: 0.9402604174500749

Random Forest Classifier

Random forest classifier has to be set with the max_depth and random state.

```
rf=RandomForestClassifier(n_estimators=100,max_depth=10,random_state=0)
rf.fit(X_train,y_train)
y_pred_rf=rf.predict(X_test)
print("Multi class classification Random Forest Accuracy:",accuracy_score(y_test,y_pred_rf))
```

Multi class classification Random Forest Accuracy: 0.9746576819136722

Gaussian Naive Bayes Classifier

NBC works inherently with multiclass classification without any changes to its function call.

```
from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
gnb.fit(X_train,y_train)
y_pred_gnb=gnb.predict(X_test)
print("Multi class GNB accuracy:",accuracy_score(y_test,y_pred_gnb))
```

(variable) y_test: Ar

Multi class GNB accuracy: 0.9884556954409398

6. Results,Comparative analysis and conclusion

Comparative Analysis: Accuracy Comparison

ML Algorithm	Binary-class classification	Multi-class classification
Logistic Regression	62.90%	79.40%
Gaussian Naive Bayes Classifier	62.35%	98.84%
Bernoulli Naive Bayes Classifier	67.12%	69.98%
Random Forest Classifier	94.89%	97.47%
K-Nearest Neighbours	92.35%	94.03%
Neural Networks	84.82%	92.86%

Comparative Analysis: Results

- Logistic Regression: Multi-class accuracy is about 16 % higher than binary-class.
- Gaussian Classifier: Multi-class accuracy is about 37 % higher than binary-class.
  - Largest percentage difference across all ML Algorithms.
- Bernoulli Classifier: Multi-class accuracy is about 2 % higher than binary-class.
- Random Forest Classifier: Multi-class accuracy is about 3 % higher than binary-class.
- K-Nearest Neighbours: Multi-class accuracy is about 2 % higher than binary-class.
- Neural Networks: Multi-class accuracy is about 8 % higher than binary-class.
- Overall, multi-class classification algorithms were tested to have a higher relative accuracy in comparison to its binary-class classification counterpart.

Conclusion

- In summary, this project aimed to showcase analysis and breakdown of a large-scaled dataset such as the crime data set provided from Los Angeles crime records.
- The project experimented with a variety of preprocessing/cleaning techniques to make working with the data more sufficient.
- It is evident that Random forest and KNN performed consistently well over both the classification sets.
- Random forest was better with an overall average accuracy of 96%.
- Contributions on working with the crime dataset will be a good reference for future researchers who would want to work with similar datasets from different city counties.

References

1. Iqbal, Murad, M. A. A., Mustapha, A., Panahy, P. H. S., & Khanahmadliravi, N. (2013). An experimental study of classification algorithms for crime prediction. *Indian Journal of Science and Technology*, *6*(3), 4219–4225.
2. Sun, Yao, C.-L., Li, X., & Lee, K. (2014). Detecting crime types using classification algorithms. Journal of Digital Information Management, 12(5), 321–327.
3. Shojaee, Somayeh & Mustapha, Aida & Sidi, Fatimah & A. Jabar, Marzanah. (2013). A Study on Classification Learning Algorithms to Predict Crime Status. International Journal of Digital Content Technology and its Applications. 7. 361-369. 10.4156/jdcta.vol7.issue9.43.
4. Shah, Neil, Bhagat, Nandish, Shah, Manan. (2021). Crime forecasting: a machine learning and computer vision approach to crime prediction and prevention. Shat et al. Visual Computing for Industry, Biomedicine, and Art, 16(5), 421–492
5. Stalidis, Panagiotis, Semertzidis, Theodoros, Daras, Petros. (2019). Examining Deep Learning Architectures for Crime Classification and Prediction. Forecasting 2021, 3, 741-762

Thank You!