LA Crime Analysis Report

Table of Contents

1. **Introduction**
    - Background of the Dataset
    - Motivation Behind the Study
2. **Dataset Overview**
    - Dataset Description
    - Attribute Explanation
3. **Algorithms and Techniques**
    - Description of Algorithms
    - Implementation Approach
4. **Evaluation of Algorithm Results**
    - Performance Metrics for Each Algorithm
5. **Data Analysis and Visualization**
    - Data Exploration
    - Data Preprocessing
    - Key Visualizations
6. **Solution Implementation**
    - K-Means Clustering
    - Decision Tree Classifier
    - DBSCAN Clustering
    - Logistic Regression
7. **Key Findings and Comparisons**
    - Python vs. R Studio Implementation
8. **Summary and Conclusion**

1. INTRODUCTION

1.1 Background of the Dataset

The United States has long been known as a land of opportunity, where individuals strive to achieve success and security. However, crime remains a significant challenge in many urban areas, particularly in **Los Angeles (LA)**. Understanding crime trends can help in making data-driven decisions for safety and urban planning. This study aims to analyse crime patterns in LA to determine the safest and most dangerous areas in 2023.

1.2 Motivation behind this Problem Statement

LA is where Hollywood lies. Hollywood is where movies are made. As avid movie watcher, the motivation behind choosing this dataset is to study more about LA and find its safest areas. The aim behind choosing this problem statement is to figure out the following:

- Analyse the safest area where the number of crimes is the least.
- Identifying the most dangerous place with the highest number of crimes in the year 2023.
- Figuring out the most common crime committed in LA in the year 2023.
- To figure out the month which witnessed the greatest number of crimes.
- To understand at what time of the day the highest number of crimes are perpetrated.
- To analyze and present a comprehensive victim profile that will highlight the commonality between the victims.
- To find the percentage of criminals who committed additional crimes along with the primary crime.

2. DATASET

2.1 Dataset Description

The raw dataset for this assignment is taken from Data LA (*LINK*). This website has uploaded data from the Los Angeles Police Department (LAPD). This dataset is a real-world dataset that is also available on the official US website of Data.gov. The raw dataset contains data on crimes committed in LA from 2020 to 2023 and it has 28 attributes (columns) and 788,768 instances (rows). The plan is to investigate the crimes committed in 2023 due to which I will extract data that is relevant for the year 2023.

2.2 Attribute Description

Following are the attributes and their explanations:

- DR_NO: Division of Records Number: Official file number made up of a 2-digit year, area ID, and 5 digits.
- Date Rptd: Date at which the crime was reported. This value is in MM/DD/YYYY format.
- DATE OCC: Date at which the crime occurred. This value is in MM/DD/YYYY format.
- TIME OCC: It stands for the time at which the crime occurred. This value is in 24-hour military time.
- AREA: The LAPD has 21 Community Police Stations referred to as Geographic Areas within the department. These Geographic Areas are sequentially numbered from 1-21.
- AREA NAME: The 21 Geographic Areas or Patrol Divisions are also given a name designation that references a landmark or the surrounding community that it is responsible for. For example, the 77th Street Division is located at the intersection of South Broadway and 77th Street, serving neighborhoods in South Los Angeles.
- Rpt Dist No: A four-digit code that represents a sub-area within a Geographic Area. All crime records reference the "RD" that it occurred in for statistical comparisons. Find LAPD Reporting Districts on the LA City GeoHub.
- Part 1-2
- Crm Cd: It indicates the crime committed. It is also referred as Crime Code 1
- Crm Cd Desc: It stands for the description of the crime code (Crm Cd).
- Mocodes: It stands for the Modus Operandi Codes. It refers to Activities associated with the suspect in commission of the crime.See attached PDF for list of MO Codes in numerical order.
- Vict Age: It stands for age of the victim.

- Vict Descent: It stands for the race of the victim. It is referred to by Descent Code. Descent Codes are as follows:
  - A - Other Asian
  - B - Black
  - C - Chinese
  - D - Cambodian
  - F – Filipino
  - G - Guamanian
  - H - Hispanic/Latin/Mexican
  - I - American Indian/Alaskan Native
  - J - Japanese
  - K - Korean
  - L - Laotian
  - O - Other
  - P - Pacific Islander
  - S - Samoan
  - U - Hawaiian
  - V - Vietnamese
  - W - White
  - X - Unknown
  - Z - Asian Indian
- Vict Sex: It stands for the sex of the victim.
  - F - Female
  - M - Male
  - X – Unknown
- Premis Cd: It stands for Premise code.
- Premis Desc: It defines the Premise description.
- Weapon Used Cd: It refers to the type of weapon used.
- Weapon Desc: It describes the weapon code defined.
- Status: It refers to the status of the case. It has values IC, AA, AO, CC, JA, JO.

- Status Desc: It describes the status mentioned in the Status column. IC stands for Investigation Cont, AA stands for Adult Arrest, Adult Other, Juv(enile) Arrest, Juv(enile) Other, and UNK.
- Crm cd 1: It indicates the crime committed. Crime Code 1 is the primary and most serious one. Crime Codes 2, 3, and 4 are respectively less serious offenses. Lower crime class numbers are more serious.
- Crm cd 2: May contain a code for an additional crime, less serious than Crime Code 1.
- Crm cd 3: May contain a code for an additional crime, less serious than Crime Code 1.
- Crm cd 4: May contain a code for an additional crime, less serious than Crime Code 1.
- LOCATION: The street address of the crime incident was rounded to the nearest hundred block to maintain anonymity.
- Cross Street: It stands for Cross Street of rounded Address.
- LAT: Latitude
- LONG: Longitude

3. ALGORITHMS/TECHNIQUES

3.1 Description of Algorithms

I will be using multiple algorithms to address all the problems highlighted above. Following are the algorithms along with the problem it will address:

- ECLAT Algorithm:

  It is a data mining algorithm that is used to find frequent items in a vertical data format. This algorithm will be used to find the most frequently occurring value in the 'mocode' column. It will be used to figure out the most common crime committed in LA in 2023.

- K-means:

  K-means clustering is an unsupervised learning algorithm that groups items into different clusters. Each cluster has data points with some commonalities. Each cluster is away from the other cluster. By using K-means we will cluster the crime based on the LAPD Community Police station number. In total, there are 21 community Police stations in LA. With the help of K-means, we will be able to analyze the area with the maximum and the minimum number of crimes.

- DBSCAN:

  DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. It will be used to put items with similar data points grouped together. It will be used to find out the month that witnessed the greatest number of crimes and the time of the day most crimes happen.

- Decision Tree:

  A Decision Tree is a tree-like structure that represents a series of decisions and their possible outcomes. It will help us determine what features like age, race, and gender might be the chance of that person to be a victim. It will help us in creating a victim profile.

- Z-Score:

  Z-Score is a statistics-based method to detect outliers. The Z-Score measures how many standard deviations a data point is away from the mean of the dataset.

4. Evaluation of Algorithm Results

The choice of evaluation metrics depends on the problem, the nature of the data, and the goal of the algorithm. It is crucial to choose the right metric to evaluate an algorithm's performance. It is also preferable to choose multiple evaluation metrics to get a comprehensive understanding of the algorithm and gauge its capabilities.

So, for this project, I have chosen appropriate metrics for each algorithm's performance.

❖ ECLAT Algorithm

Sr. No	Metric Name	Description
1	Accuracy	Since this algorithm is focused on the quality and relevance of patterns, applying accuracy metrics won't be suitable for it.
2	Precision	It can be used to assess the relevance of patterns found.
3	Recall	It can also be used to assess the relevance of patterns found.
4	F1 Score	The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall and can be used to evaluate the overall accuracy of patterns discovered by ECLAT.
5	Confusion Matrix	Since there are no predictions, it is not suitable to evaluate this algorithm.

❖ **K-means Clustering**

Since this algorithm is unsupervised, a different group of metrics is used to evaluate this algorithm.

Sr. No	Metric Name	Description
1	Inertia	It computes the sum of squared distances inside a cluster. Lower inertia suggests more compact clusters.
2	Silhouette Score	It compares the similarity of data points to their allocated clusters to other clusters. A higher silhouette score indicates that the clusters are better.
3	Visual Inspection	To measure cluster quality visually, use scatter plots, cluster profiles, or other visualization tools to visualize the clustering findings.

❖ DBSCAN

It is also a clustering algorithm so it will need different metrics.

Sr. No	Metric	Description
1	Silhouette Score	It compares the similarity of data points to their allocated clusters to other clusters. A higher silhouette score indicates that the clusters are better.
2	Visual Inspection	To measure cluster quality visually, use scatter plots, cluster profiles, or other visualization tools to visualize the clustering findings.
3	Cluster Size	Examine the cluster size distribution. Make sure the clusters are neither too tiny or too large, since this may have an influence on the practical usability of the clustering results.

❖ Decision Trees

Sr. No	Metric Name	Description
1	Accuracy	It can be used to measure the model's predictive accuracy on a holdout test dataset.
2	Precision	It can be used to assess the performance of Decision Trees.
3	Recall	It can also be used to assess the performance of Decision Trees.
4	F1 Score	The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall and can be used to evaluate the overall accuracy of patterns discovered by Decision Trees.
5	Confusion Matrix	Create a confusion matrix using the model's predictions and the validation set's true labels. True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) are the four values in the confusion matrix.

❖ Z – Score:

Sr. No	Metric Name	Description
1	Visual Inspection	To measure cluster quality visually, use scatter plots, cluster profiles, or other visualization tools to visualize the clustering findings.
2	Confusion Matrix	Create a confusion matrix using the model's predictions and the validation set's true labels. True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) are the four values in the confusion matrix.
3	Receiver Operating Characteristic (ROC) Curve	We can compute the ROC curve and the area under the ROC curve (AUC) if you have access to labeled data (ground truth). The effectiveness of the algorithm in identifying between outliers and non-outliers at various threshold values may be evaluated using ROC analysis.

5. PRELIMINARY DATA ANALYSIS & VISUALIZATION

5.1 Data Exploration

The data set contains 788,768 instances and 28 attributes.

Following is the step-by-step execution undertaken for data exploration:

- Firstly, importing all the necessary libraries as well as our raw data set.

```
Installing libraries and loading up data

In [61]: import pandas as pd
         import numpy as np
         import matplotlib as plt

         df = pd.read_csv('C:/Users/61431/Downloads/Crime_Data_from_2020_to_Present.csv', sep = ',')
         df
```

Out[61]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	...	Status	Status Desc	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4	LOCATION
0	10304468	01/08/2020 12:00:00 AM	01/08/2020 12:00:00 AM	2230	3	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	...	AO	Adult Other	624.0	NaN	NaN	NaN	1100 W 39TH PL
1	190101086	01/02/2020 12:00:00 AM	01/01/2020 12:00:00 AM	330	1	Central	163	2	624	BATTERY - SIMPLE ASSAULT	...	IC	Invest Cont	624.0	NaN	NaN	NaN	700 S HILL ST
2	200110444	04/14/2020 12:00:00 AM	02/13/2020 12:00:00 AM	1200	1	Central	155	2	845	SEX OFFENDER REGISTRANT OUT OF COMPLIANCE	...	AA	Adult Arrest	845.0	NaN	NaN	NaN	200 E 6TH ST
3	191501505	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	1730	15	N Hollywood	1543	2	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	...	IC	Invest Cont	745.0	998.0	NaN	NaN	5400 CORTEEN PL
4	191921269	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	415	19	Mission	1998	2	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	...	IC	Invest Cont	740.0	NaN	NaN	NaN	14400 TITUS ST

- Extracted information about the attributes.

```
This was the unprocessed raw Data. Now, getting information about the dataset.

In [62]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 788767 entries, 0 to 788766
Data columns (total 28 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DR_NO           788767 non-null  int64
 1   Date Rptd       788767 non-null  object
 2   DATE OCC        788767 non-null  object
 3   TIME OCC        788767 non-null  int64
 4   AREA            788767 non-null  int64
 5   AREA NAME       788767 non-null  object
 6   Rpt Dist No     788767 non-null  int64
 7   Part 1-2        788767 non-null  int64
 8   Crm Cd          788767 non-null  int64
 9   Crm Cd Desc     788767 non-null  object
 10  Mocodes         680162 non-null  object
 11  Vict Age        788767 non-null  int64
 12  Vict Sex        685415 non-null  object
 13  Vict Descent    685407 non-null  object
 14  Premis Cd       788758 non-null  float64
 15  Premis Desc     788300 non-null  object
 16  Weapon Used Cd  274517 non-null  float64
 17  Weapon Desc     274517 non-null  object
 18  Status          788767 non-null  object
 19  Status Desc     788767 non-null  object
 20  Crm Cd 1        788757 non-null  float64
 21  Crm Cd 2        58171 non-null   float64
 22  Crm Cd 3        1936 non-null    float64
 23  Crm Cd 4        57 non-null      float64
 24  LOCATION        788767 non-null  object
 25  Cross Street    126409 non-null  object
 26  LAT             788767 non-null  float64
 27  LON             788767 non-null  float64
dtypes: float64(8), int64(7), object(13)
memory usage: 168.5+ MB
```

❖ Dropping the irrelevant columns from the raw data set.

```
There are many columns here that are not necessary for our analysis like Latitude, Longitude, Cross Street, Parts 1-2, DR_NO
and even the Date on which crime is reported. We will now remove them.
```

```
In [63]: df.drop(df.columns[[0, 7, 16, 17, 18, 19, 24, 25, 26, 27]], axis = 1, inplace = True)
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 788767 entries, 0 to 788766
Data columns (total 18 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Date Rptd     788767 non-null  object
 1   DATE OCC      788767 non-null  object
 2   TIME OCC      788767 non-null  int64
 3   AREA          788767 non-null  int64
 4   AREA NAME     788767 non-null  object
 5   Rpt Dist No   788767 non-null  int64
 6   Crm Cd        788767 non-null  int64
 7   Crm Cd Desc   788767 non-null  object
 8   Mocodes       680162 non-null  object
 9   Vict Age      788767 non-null  int64
 10  Vict Sex      685415 non-null  object
 11  Vict Descent  685407 non-null  object
 12  Premis Cd     788758 non-null  float64
 13  Premis Desc   788300 non-null  object
 14  Crm Cd 1      788757 non-null  float64
 15  Crm Cd 2      58171 non-null   float64
 16  Crm Cd 3      1936 non-null    float64
 17  Crm Cd 4      57 non-null      float64
dtypes: float64(5), int64(5), object(8)
memory usage: 108.3+ MB
```

❖ Date extraction for each crime.

```
Extracting the date on which the crime occured.
```

In [64]:
```
df['DATE OCC'] = df['DATE OCC'].str.extract(r'(\d{2}/\d{2}/\d{4})')
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'], format='%m/%d/%Y')
df['Day'] = df['DATE OCC'].dt.day_name()
df['Month'] = df['DATE OCC'].dt.month_name()
df['Year'] = df['DATE OCC'].dt.year
```

```
The time of crime in TIME OCC is in integer form. Converting it to prpoer 24-hour time format. For this, we are going tocreate
a function which will convert 4 digit integer to a time value.
```

In [65]:
```
from datetime import time


def convert_to_time(value):
    hours = value // 100
    minutes = value % 100
    return time(hours, minutes)


df['TIME OCC'] = df['TIME OCC'].apply(convert_to_time)
```

In [66]:
```
df['TIME OCC'] = df['TIME OCC'].astype(str).str.zfill(4)
df
```

Out[66]:

	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Crm Cd	Crm Cd Desc	Mocodes	Vict Age	...	Vict Descent	Premis Cd	Premis Desc	Crm Cd 1	Crm Cd 2	C
0	01/08/2020 12:00:00 AM	2020-01-08	22:30:00	3	Southwest	377	624	BATTERY - SIMPLE ASSAULT	0444 0913	36	...	B	501.0	SINGLE FAMILY DWELLING	624.0	NaN	N
1	01/02/2020 12:00:00 AM	2020-01-01	03:30:00	1	Central	163	624	BATTERY - SIMPLE ASSAULT	0416 1822 1414	25	...	H	102.0	SIDEWALK	624.0	NaN	N
2	04/14/2020 12:00:00 AM	2020-02-13	12:00:00	1	Central	155	845	SEX OFFENDER REGISTRANT OUT OF COMPLIANCE	1501	0	...	X	726.0	POLICE FACILITY	845.0	NaN	N
3	01/01/2020 12:00:00 AM	2020-01-01	17:30:00	15	N Hollywood	1543	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	0329 1402	76	...	W	502.0	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	745.0	998.0	N
4	01/01/2020 12:00:00 AM	2020-01-01	04:15:00	19	Mission	1998	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	0329	31	...	X	409.0	BEAUTY SUPPLY STORE	740.0	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
788762	01/27/2023 12:00:00 AM	2023-01-26	18:00:00	16	Foothill	1663	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	1300 0329	23	...	H	122.0	VEHICLE, PASSENGER/TRUCK	740.0	NaN	N
788763	03/22/2023 12:00:00 AM	2023-03-22	10:00:00	16	Foothill	1602	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	0416 0411 1822	25	...	H	102.0	SIDEWALK	230.0	NaN	N
788764	04/12/2023 12:00:00 AM	2023-04-12	16:30:00	12	77th Street	1239	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	0601 0445 0416 0359	29	...	B	222.0	LAUNDROMAT	230.0	NaN	N
788765	07/02/2023 12:00:00 AM	2023-07-01	00:01:00	1	Central	154	352	PICKPOCKET	1822 0344	24	...	H	735.0	NIGHT CLUB (OPEN EVENINGS ONLY)	352.0	NaN	N
								VANDALISM -						MULTI-UNIT			

- Removing Date Occ column as we have the complete date divided into Day, Month, and Year.

```
Since we have divided the DATE OCC table into Day, Month and Year, we can now remove DATE OCC and rerrange the data frame
columns.

In [69]: df.drop(df.columns[[0, 1]], axis = 1, inplace = True)
         df.columns

Out[69]: Index(['TIME OCC', 'AREA', 'AREA NAME', 'Rpt Dist No', 'Crm Cd', 'Crm Cd Desc',
                'Mocodes', 'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd',
                'Premis Desc', 'Crm Cd 1', 'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'Day',
                'Month', 'Year'],
               dtype='object')

In [70]: df = df[['Day',
                  'Month', 'Year', 'TIME OCC', 'AREA', 'AREA NAME', 'Rpt Dist No', 'Crm Cd', 'Crm Cd Desc',
                  'Mocodes', 'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd',
                  'Premis Desc', 'Crm Cd 1', 'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4']]

In [75]: df = df[df['Year'] > 2022]
         df
```

Out[75]:

	Day	Month	Year	TIME OCC	AREA	AREA NAME	Rpt Dist No	Crm Cd	Crm Cd Desc	Mocodes	Vict Age	Vict Sex	Vict Descent	Premis Cd	Premis Desc	Crm Cd
337797	Friday	January	2023	12:50:00	11	Northeast	1151	510	VEHICLE - STOLEN	NaN	0	NaN	NaN	101.0	STREET	510.
408555	Tuesday	August	2023	03:45:00	15	N Hollywood	1524	310	BURGLARY	1609	0	M	O	203.0	OTHER BUSINESS	310.
408583	Saturday	March	2023	09:30:00	17	Devonshire	1792	510	VEHICLE - STOLEN	NaN	0	NaN	NaN	101.0	STREET	510.
408612	Thursday	July	2023	18:00:00	21	Topanga	2156	442	SHOPLIFTING - PETTY THEFT ($950 & UNDER)	2024 0325 0352	35	M	X	404.0	DEPARTMENT STORE	442.
408669	Thursday	January	2023	16:30:00	19	Mission	1916	850	INDECENT EXPOSURE	1822 0529 1259	16	F	H	102.0	SIDEWALK	850.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
788762	Thursday	January	2023	18:00:00	16	Foothill	1663	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	1300 0329	23	M	H	122.0	VEHICLE, PASSENGER/TRUCK	740.
788763	Wednesday	March	2023	10:00:00	16	Foothill	1602	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	0416 0411 1822	25	F	H	102.0	SIDEWALK	230.
788764	Wednesday	April	2023	16:30:00	12	77th Street	1239	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	0601 0445 0416 0359	29	M	B	222.0	LAUNDROMAT	230.
788765	Saturday	July	2023	00:01:00	1	Central	154	352	PICKPOCKET	1822 0344	24	F	H	735.0	NIGHT CLUB (OPEN EVENINGS ONLY)	352.
788766	Sunday	March	2023	09:00:00	9	Van Nuys	914	745	VANDALISM - MISDEAMEANOR ($399 OR	0329 1822	53	F	H	502.0	MULTI-UNIT DWELLING (APARTMENT,	745.

5.2 Data Preprocessing

After extracting the relevant set of data from the raw data set, the next step is to perform basic data processing so that the resultant data set can answer all the problem statements highlighted in the document above.

The following preprocessing steps were taken:

- Removing the noise from the data.

```
In [76]: df.describe()
```

Out[76]:

	Year	AREA	Rpt Dist No	Crm Cd	Vict Age	Premis Cd	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4
count	146156.0	146156.000000	146156.000000	146156.000000	146156.000000	146155.000000	146153.000000	9787.000000	301.000000	9.000000
mean	2023.0	10.654356	1112.164694	497.653384	29.103157	312.048969	497.422865	966.678860	987.259136	992.333333
std	0.0	6.117829	611.725150	206.145303	22.114912	217.993100	205.947316	98.559066	34.906436	17.378147
min	2023.0	1.000000	101.000000	110.000000	0.000000	101.000000	110.000000	220.000000	761.000000	946.000000
25%	2023.0	5.000000	585.000000	331.000000	0.000000	101.000000	331.000000	998.000000	998.000000	998.000000
50%	2023.0	11.000000	1136.000000	442.000000	30.000000	210.000000	442.000000	998.000000	998.000000	998.000000
75%	2023.0	16.000000	1608.000000	626.000000	44.000000	501.000000	626.000000	998.000000	998.000000	998.000000
max	2023.0	21.000000	2198.000000	956.000000	99.000000	974.000000	956.000000	999.000000	999.000000	999.000000

- Find the null values.

```
In [77]: df.isnull().any()
```

Out[77]:

```
Day             False
Month           False
Year            False
TIME OCC        False
AREA            False
AREA NAME       False
Rpt Dist No     False
Crm Cd          False
Crm Cd Desc     False
Mocodes          True
Vict Age        False
Vict Sex         True
Vict Descent     True
Premis Cd        True
Premis Desc      True
Crm Cd 1         True
Crm Cd 2         True
Crm Cd 3         True
Crm Cd 4         True
dtype: bool
```

Here, 'Mocodes', 'Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis Desc', 'Crm Cd 1', Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4' have some null values.

- Filling the Crm Cd 2 to 4 crime code by zero where no value is present as they represent the additional crimes committed with the major crime. If no value is present, it reflects that there were no additional crimes.

Crm Cd 2 to 4 are crime codes in case a culprit committed additional crimes while comitting primary crime Crm cd 1. So filling the null values with 0.

```
In [78]: values1 = {'Crm Cd 2': 0, 'Crm Cd 3': 0, 'Crm Cd 4': 0}
clean_df = df.fillna(value = values1)
clean_df.head(10)
```

Out[78]:

	Day	Month	Year	TIME OCC	AREA	AREA NAME	Rpt Dist No	Crm Cd	Crm Cd Desc	Mocodes	Vict Age	Vict Sex	Vict Descent	Premis Cd	Premis Desc	Crm Cd 1	Crm Cd 2
337797	Friday	January	2023	12:50:00	11	Northeast	1151	510	VEHICLE - STOLEN	NaN	0	NaN	NaN	101.0	STREET	510.0	0.0
408555	Tuesday	August	2023	03:45:00	15	N Hollywood	1524	310	BURGLARY	1609	0	M	O	203.0	OTHER BUSINESS	310.0	0.0
408583	Saturday	March	2023	09:30:00	17	Devonshire	1792	510	VEHICLE - STOLEN	NaN	0	NaN	NaN	101.0	STREET	510.0	0.0
408612	Thursday	July	2023	18:00:00	21	Topanga	2156	442	SHOPLIFTING - PETTY THEFT ($950 & UNDER)	2024 0325 0352	35	M	X	404.0	DEPARTMENT STORE	442.0	0.0
408669	Thursday	January	2023	16:30:00	19	Mission	1916	850	INDECENT EXPOSURE	1822 0529 1259	16	F	H	102.0	SIDEWALK	850.0	0.0
408863	Wednesday	May	2023	11:00:00	18	Southeast	1803	440	THEFT PLAIN - PETTY ($950 & UNDER)	0344	50	M	B	102.0	SIDEWALK	440.0	0.0
409227	Tuesday	January	2023	00:30:00	17	Devonshire	1761	354	THEFT OF IDENTITY	0930	36	M	W	501.0	SINGLE FAMILY DWELLING	354.0	0.0

- Null values dropped from “Crm Cd 1” and “Mocodes” as this indicates that no crime was committed.

For 'Crm Cd 1' and 'Mocodes', We will drop the row where this is Null because it stands for primary crime and the Modus Operandi of the culprit.

In [79]:
```
clean_df.dropna(subset = ['Crm Cd 1', 'Mocodes'], inplace = True)
clean_df
```

Out[79]:

	Day	Month	Year	TIME OCC	AREA	AREA NAME	Rpt Dist No	Crm Cd	Crm Cd Desc	Mocodes	Vict Age	Vict Sex	Vict Descent	Premis Cd	Premis Desc	Crm Cd 1
408555	Tuesday	August	2023	03:45:00	15	N Hollywood	1524	310	BURGLARY	1609	0	M	O	203.0	OTHER BUSINESS	310.0
408612	Thursday	July	2023	18:00:00	21	Topanga	2156	442	SHOPLIFTING - PETTY THEFT ($950 & UNDER)	2024 0325 0352	35	M	X	404.0	DEPARTMENT STORE	442.0
408669	Thursday	January	2023	16:30:00	19	Mission	1916	850	INDECENT EXPOSURE	1822 0529 1259	16	F	H	102.0	SIDEWALK	850.0
408863	Wednesday	May	2023	11:00:00	18	Southeast	1803	440	THEFT PLAIN - PETTY ($950 & UNDER)	0344	50	M	B	102.0	SIDEWALK	440.0
409227	Tuesday	January	2023	00:30:00	17	Devonshire	1761	354	THEFT OF IDENTITY	0930	36	M	W	501.0	SINGLE FAMILY DWELLING	354.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
788762	Thursday	January	2023	18:00:00	16	Foothill	1663	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	1300 0329	23	M	H	122.0	VEHICLE, PASSENGER/TRUCK	740.0
788763	Wednesday	March	2023	10:00:00	16	Foothill	1602	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	0416 0411 1822	25	F	H	102.0	SIDEWALK	230.0
788764	Wednesday	April	2023	16:30:00	12	77th Street	1239	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	0601 0445 0416 0359	29	M	B	222.0	LAUNDROMAT	230.0
788765	Saturday	July	2023	00:01:00	1	Central	154	352	PICKPOCKET	1822 0344	24	F	H	735.0	NIGHT CLUB (OPEN EVENINGS ONLY)	352.0
788766	Sunday	March	2023	09:00:00	9	Van Nuys	914	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	0329 1822	53	F	H	502.0	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	745.0

125755 rows × 19 columns

- Write X for unknown values “Vict Sex” and “Vict Descent” so that they can be easily eliminated.

For Unknown values in 'Vict Sex' and 'Vict Descent', we are going to put value as X. It is also to be noted that there's a value written as H in Vict Sex but there is no definiton for it. So, considering it as a mistake in data we will convert that to X.

In [80]:
```
values2 = {'Vict Sex': 'X', 'Vict Descent': 'X'}
clean_df2 = clean_df.fillna(value = values2)
clean_df2.replace(['H'], 'X', inplace = True)
clean_df2
```

Out[80]:

	Day	Month	Year	TIME OCC	AREA	AREA NAME	Rpt Dist No	Crm Cd	Crm Cd Desc	Mocodes	Vict Age	Vict Sex	Vict Descent	Premis Cd	Premis Desc	Crm Cd 1
408555	Tuesday	August	2023	03:45:00	15	N Hollywood	1524	310	BURGLARY	1609	0	M	O	203.0	OTHER BUSINESS	310.(
408612	Thursday	July	2023	18:00:00	21	Topanga	2156	442	SHOPLIFTING - PETTY THEFT ($950 & UNDER)	2024 0325 0352	35	M	X	404.0	DEPARTMENT STORE	442.(
408669	Thursday	January	2023	16:30:00	19	Mission	1916	850	INDECENT EXPOSURE	1822 0529 1259	16	F	X	102.0	SIDEWALK	850.(
408863	Wednesday	May	2023	11:00:00	18	Southeast	1803	440	THEFT PLAIN - PETTY ($950 & UNDER)	0344	50	M	B	102.0	SIDEWALK	440.(
409227	Tuesday	January	2023	00:30:00	17	Devonshire	1761	354	THEFT OF IDENTITY	0930	36	M	W	501.0	SINGLE FAMILY DWELLING	354.(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
788762	Thursday	January	2023	18:00:00	16	Foothill	1663	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	1300 0329	23	M	X	122.0	VEHICLE, PASSENGER/TRUCK	740.(
788763	Wednesday	March	2023	10:00:00	16	Foothill	1602	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	0416 0411 1822	25	F	X	102.0	SIDEWALK	230.(

- Remove null value rows with null values of Premis Cd and Premis Desc.

'Premis Cd' and 'Premis Desc' are description of the premise where the crime occurs. We will also remove the rows with them as Null.

In [81]:
```
clean_df2.dropna(subset = ['Premis Cd', 'Premis Desc'], inplace = True)
clean_df2
```

Out[81]:

	Day	Month	Year	TIME OCC	AREA	AREA NAME	Rpt Dist No	Crm Cd	Crm Cd Desc	Mocodes	Vict Age	Vict Sex	Vict Descent	Premis Cd	Premis Desc	Crm Cd 1
408555	Tuesday	August	2023	03:45:00	15	N Hollywood	1524	310	BURGLARY	1609	0	M	O	203.0	OTHER BUSINESS	310.(
408612	Thursday	July	2023	18:00:00	21	Topanga	2156	442	SHOPLIFTING - PETTY THEFT ($950 & UNDER)	2024 0325 0352	35	M	X	404.0	DEPARTMENT STORE	442.(
408669	Thursday	January	2023	16:30:00	19	Mission	1916	850	INDECENT EXPOSURE	1822 0529 1259	16	F	X	102.0	SIDEWALK	850.(
408863	Wednesday	May	2023	11:00:00	18	Southeast	1803	440	THEFT PLAIN - PETTY ($950 & UNDER)	0344	50	M	B	102.0	SIDEWALK	440.(
409227	Tuesday	January	2023	00:30:00	17	Devonshire	1761	354	THEFT OF IDENTITY	0930	36	M	W	501.0	SINGLE FAMILY DWELLING	354.(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
788762	Thursday	January	2023	18:00:00	16	Foothill	1663	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	1300 0329	23	M	X	122.0	VEHICLE, PASSENGER/TRUCK	740.(
788763	Wednesday	March	2023	10:00:00	16	Foothill	1602	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	0416 0411 1822	25	F	X	102.0	SIDEWALK	230.(
788764	Wednesday	April	2023	16:30:00	12	77th Street	1239	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	0601 0445 0416 0359	29	M	B	222.0	LAUNDROMAT	230.(
788765	Saturday	July	2023	00:01:00	1	Central	154	352	PICKPOCKET	1822 0344	24	F	X	735.0	NIGHT CLUB (OPEN EVENINGS ONLY)	352.(

5.3 Data Visualization

The following execution was done for the visualization of our data set:

- Find the five-summary.

```
It should be emphasized that while many columns actually have a numerical value, that value refers to a specific crime or the
perpetrator's method of operation. Age is the sole attribute that takes the number at face value.

In [83]: clean_df2['Vict Age'].describe()

Out[83]: count    125596.000000
         mean         33.562757
         std          20.350727
         min           0.000000
         25%          22.000000
         50%          33.000000
         75%          47.000000
         max          99.000000
         Name: Vict Age, dtype: float64
```

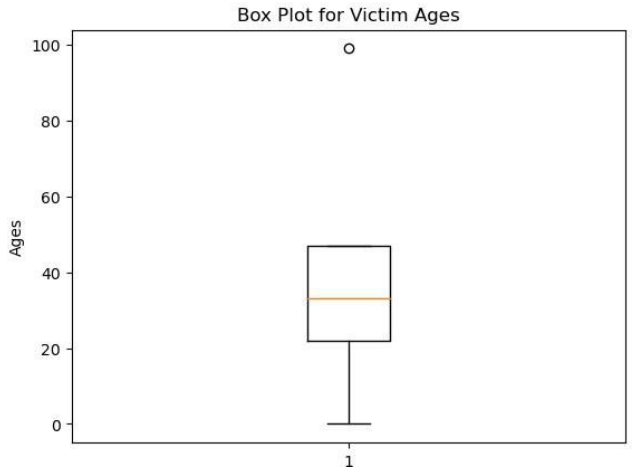
- Checking the data set again.

```
Checking the dataset again

In [82]: clean_df2.isnull().any()

Out[82]: Day             False
         Month           False
         Year            False
         TIME OCC        False
         AREA            False
         AREA NAME       False
         Rpt Dist No     False
         Crm Cd          False
         Crm Cd Desc     False
         Mocodes         False
         Vict Age        False
         Vict Sex        False
         Vict Descent    False
         Premis Cd       False
         Premis Desc     False
         Crm Cd 1        False
         Crm Cd 2        False
         Crm Cd 3        False
         Crm Cd 4        False
         dtype: bool

Thus, we no longer have null values in our data.
```

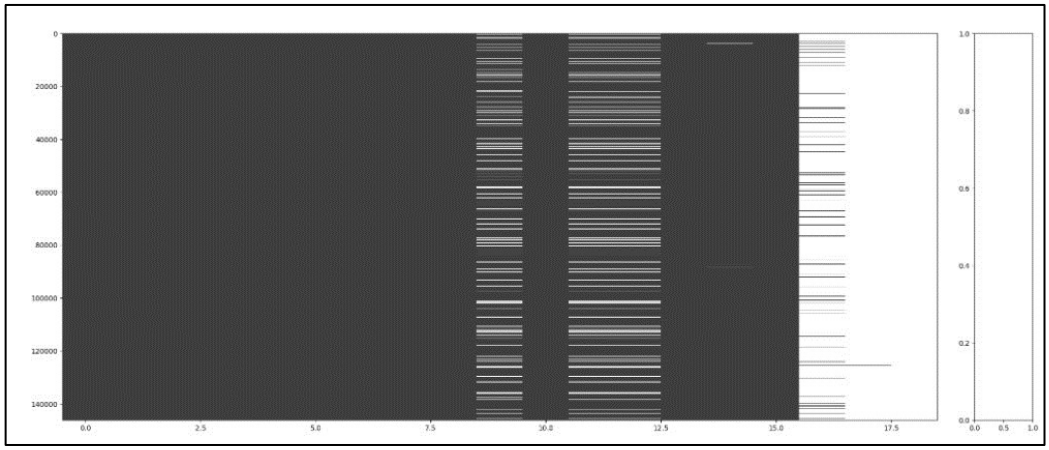
❖ Making a box plot.

Making a box plot for the data above

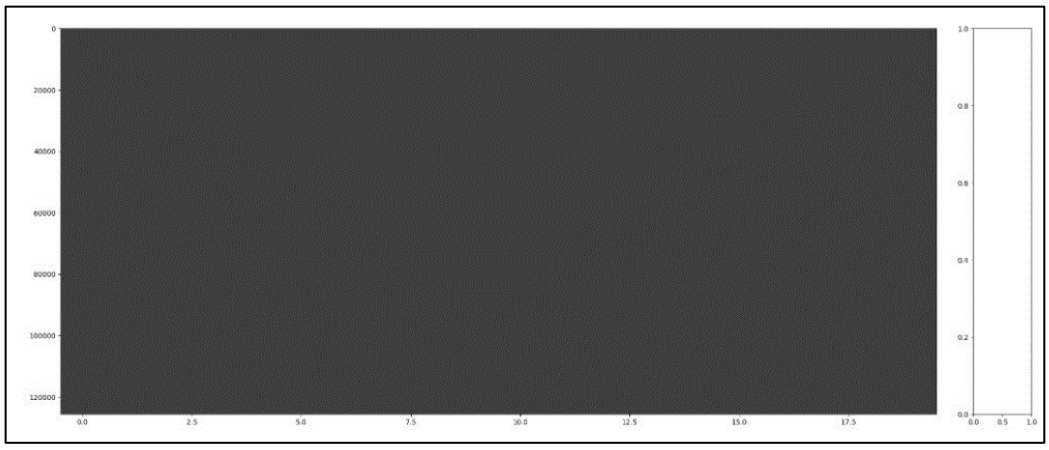
In [85]:
```
data = [0.0, 22.0, 33.0, 47.0, 99.0]
import matplotlib.pyplot as plt


plt.boxplot(data)
plt.title('Box Plot for Victim Ages')
plt.ylabel('Ages')

plt.show()
```



❖ Graph before cleaning the data.

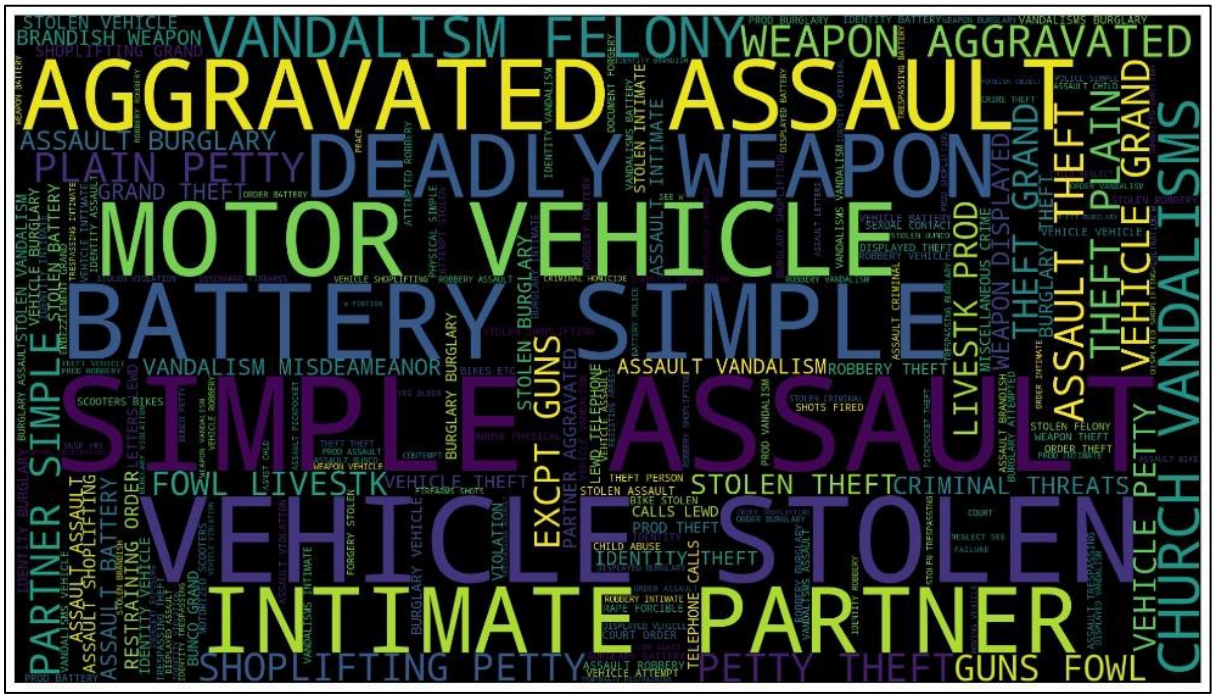


❖ Graph after cleaning the data.



❖ Creating a word cloud using the terms mentioned in crime code description (Crm Cd Description).

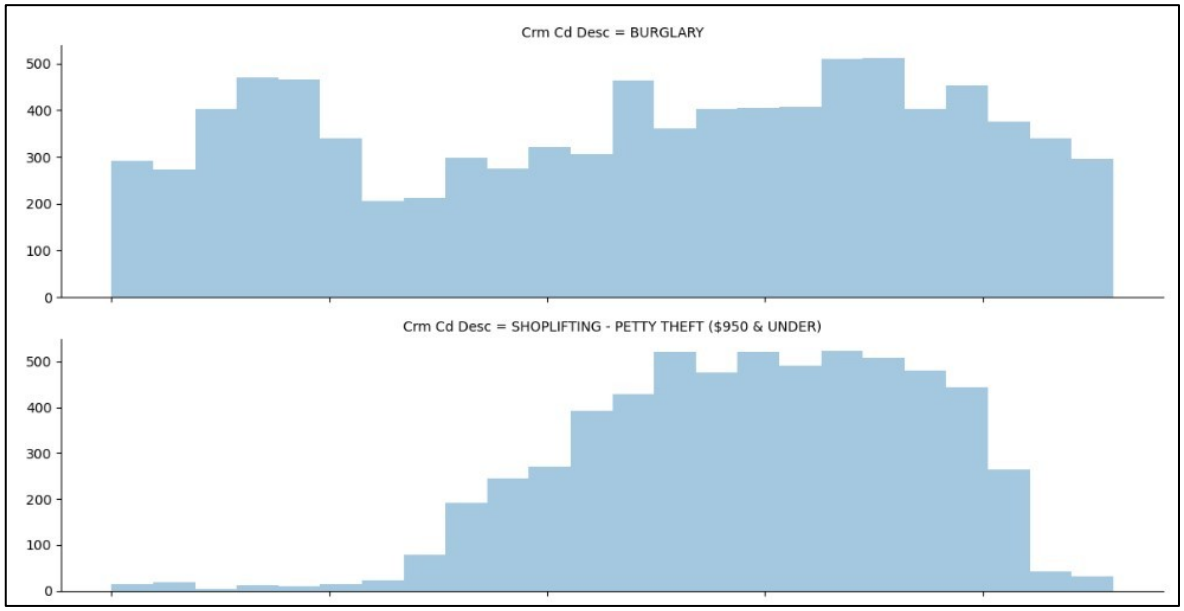
```
In [94]: from wordcloud import WordCloud,STOPWORDS

         plt.figure(figsize=(25,15))
         wordcloud = WordCloud(
                                  background_color='black',
                                  width=1920,
                                  height=1080
                                 ).generate(" ".join(df['Crm Cd Desc']))
         plt.imshow(wordcloud)
         plt.axis('off')
         plt.savefig('graph.png')
         plt.show()
```

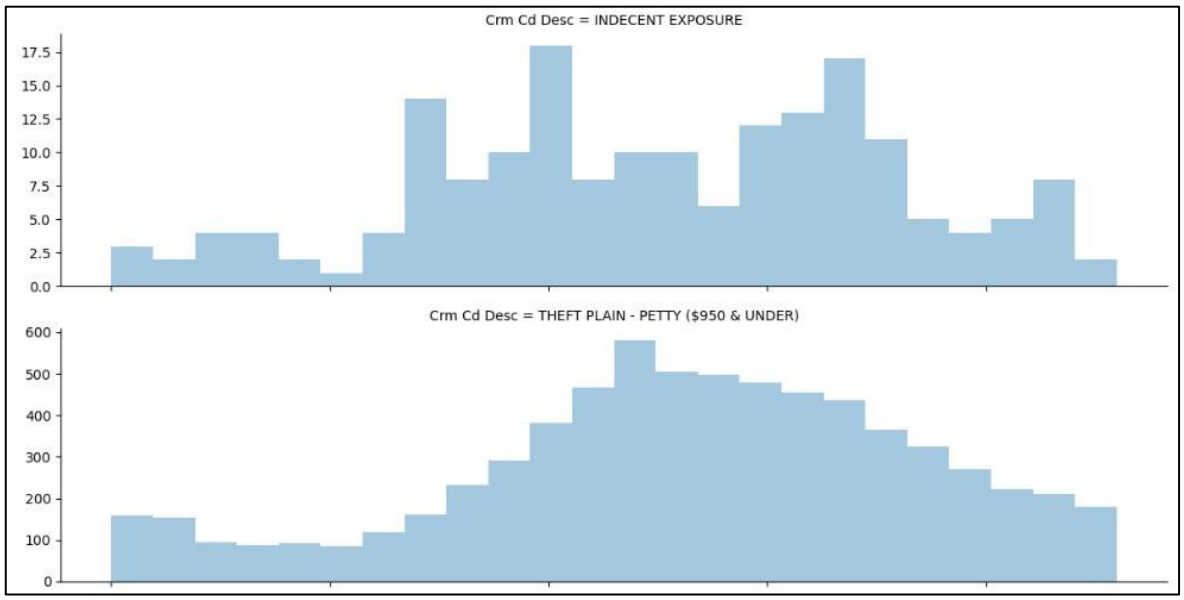


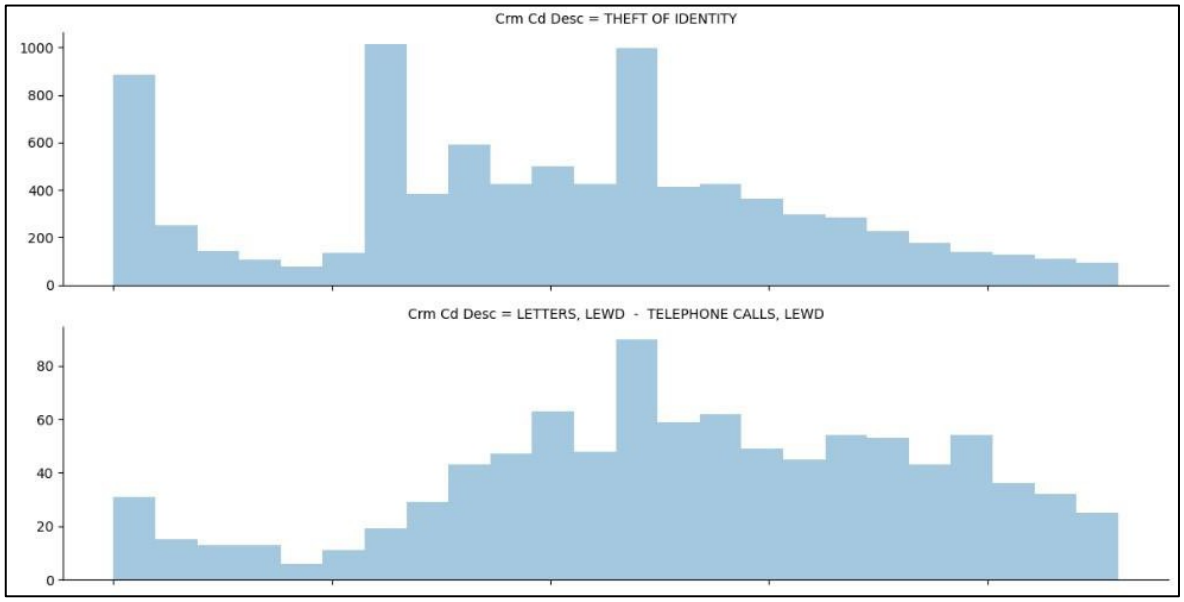
❖ Extracting HH from the HH:MM time format.

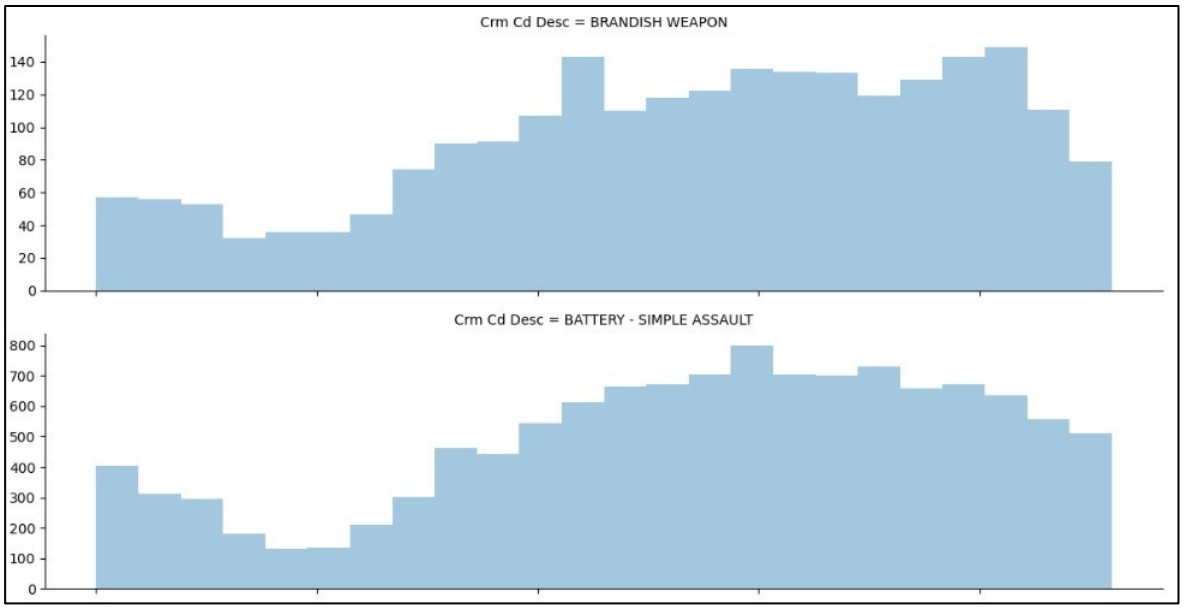
```
In [107]: clean_df2['TIME OCC'] = clean_df2['TIME OCC'].astype(str).str.zfill(4)
          clean_df2['HOUR OCC'] = clean_df2['TIME OCC'].apply(lambda t: int(t[:2]))

          f = sns.FacetGrid(clean_df2,row="Crm Cd Desc", aspect=4,sharex=True,sharey=False)
          f.map(sns.distplot, "HOUR OCC", bins=24, kde=False, rug=False)
```

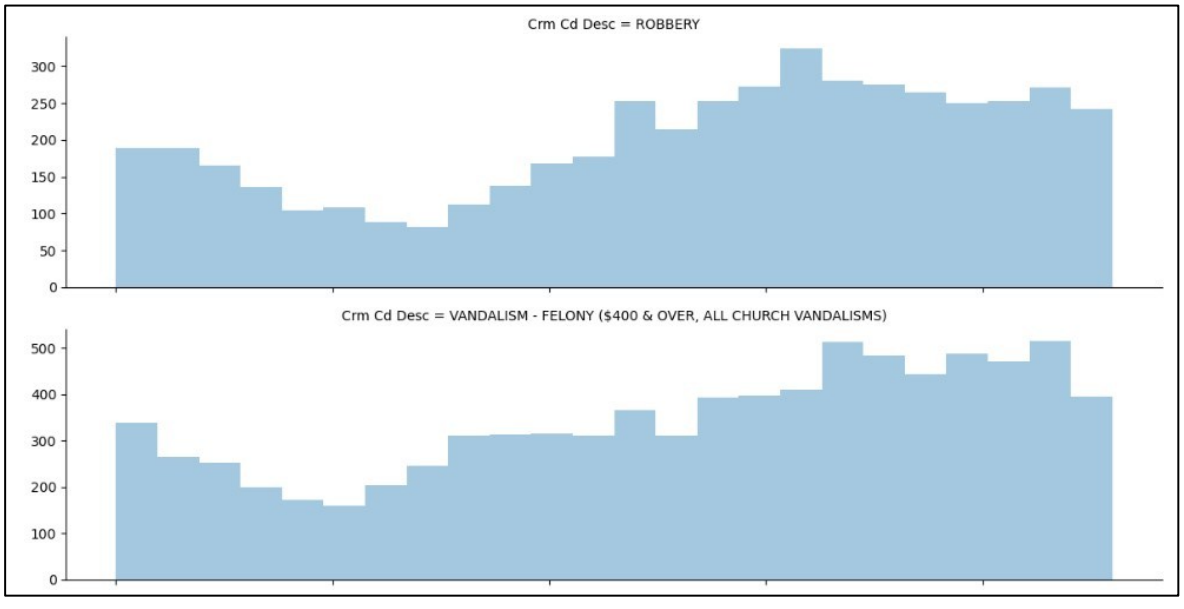
❖ Creating visualization graphs.

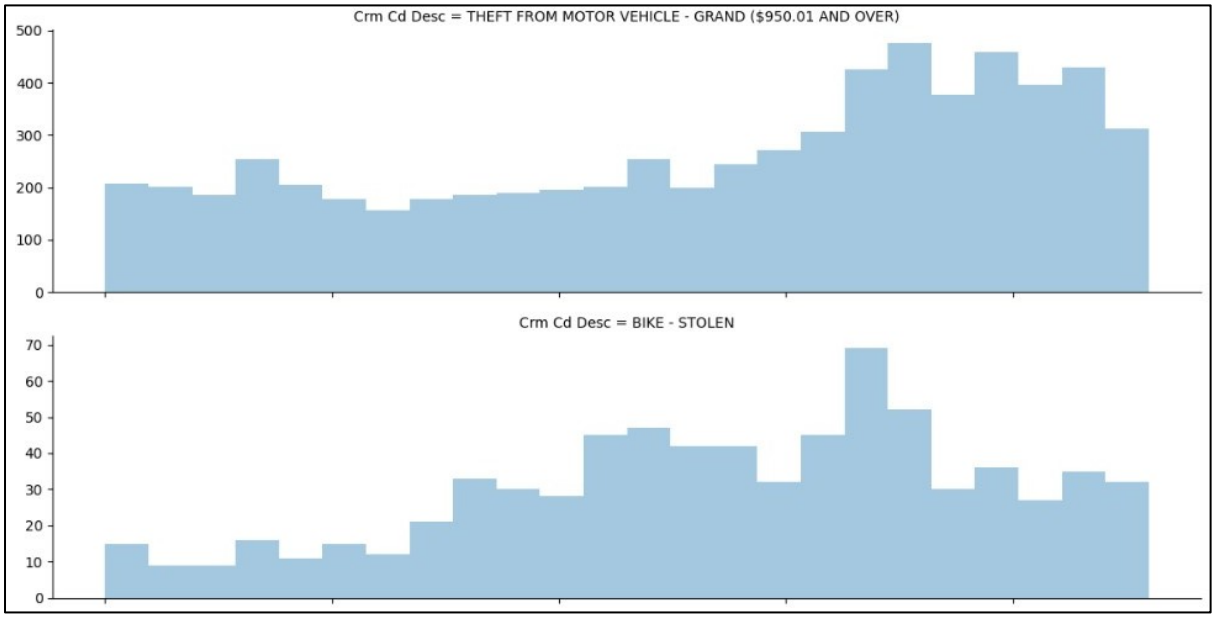


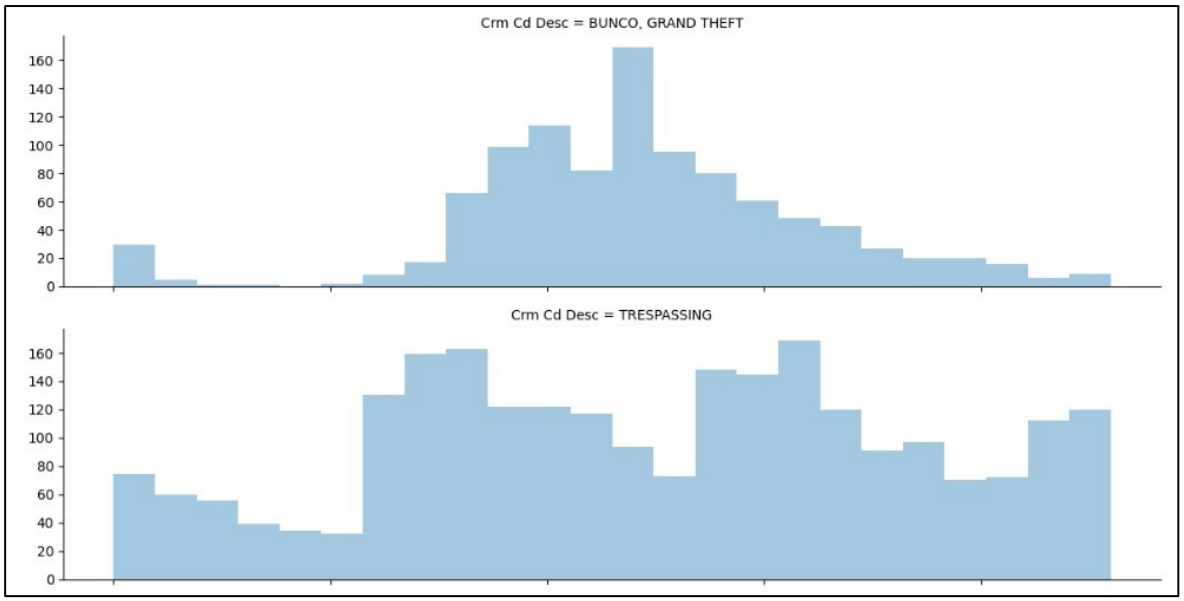


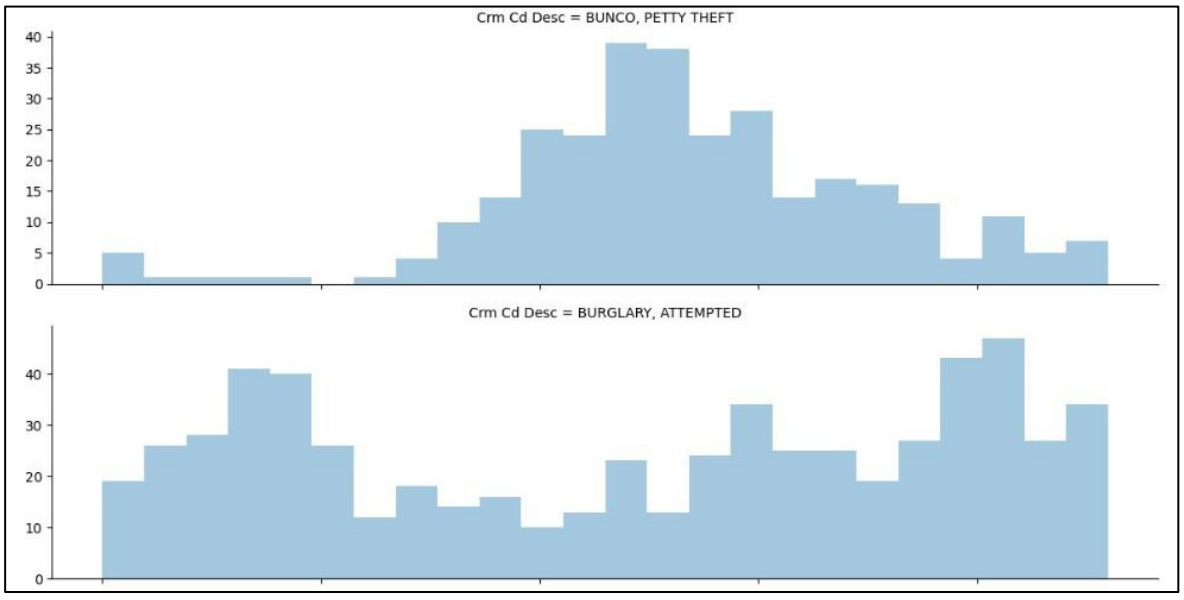

Crm Cd Desc = THEFT OF IDENTITY
1000
800
600
400
200
0
Crm Cd Desc = LETTERS, LEWD - TELEPHONE CALLS, LEWD
80
60
40
20
0

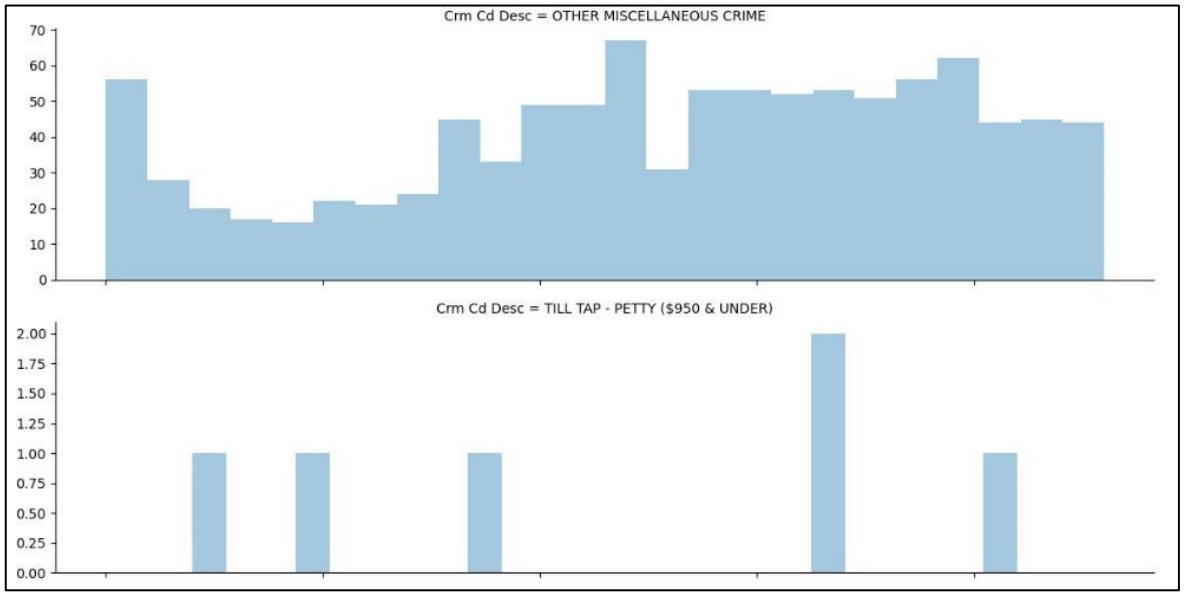

Crm Cd Desc = BRANDISH WEAPON
140
120
100
80
60
40
20
0
Crm Cd Desc = BATTERY - SIMPLE ASSAULT
800
700
600
500
400
300
200
100
0

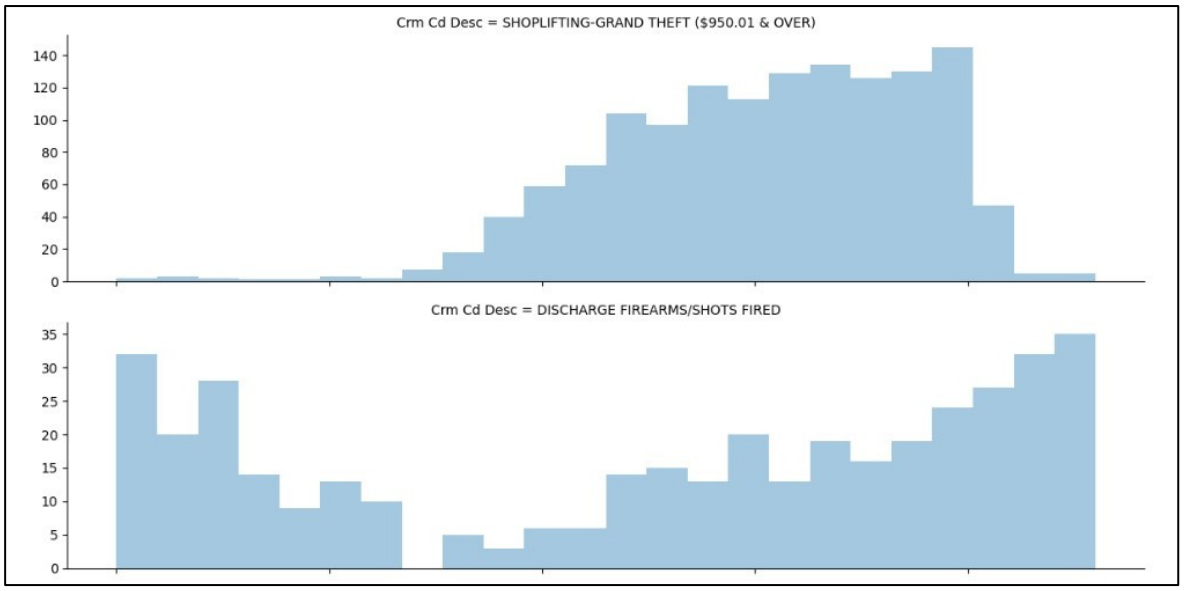

Crm Cd Desc = ROBBERY
300
250
200
150
100
50
0
Crm Cd Desc = VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)
500
400
300
200
100
0

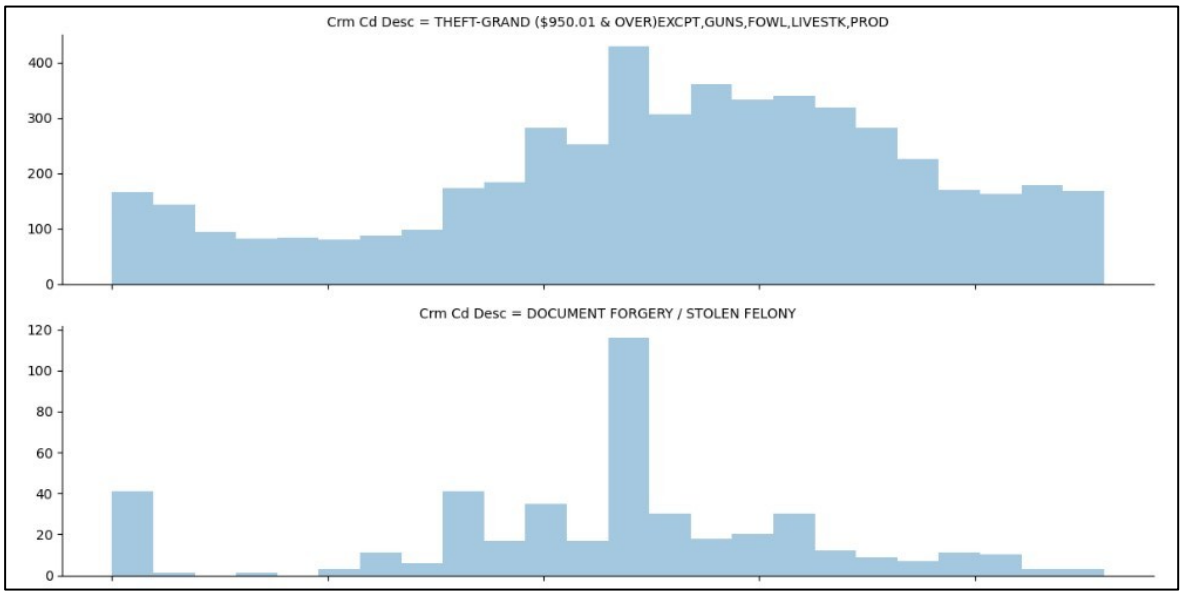

Crm Cd Desc = THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)
500
400
300
200
100
0
Crm Cd Desc = BIKE - STOLEN
70
60
50
40
30
20
10
0

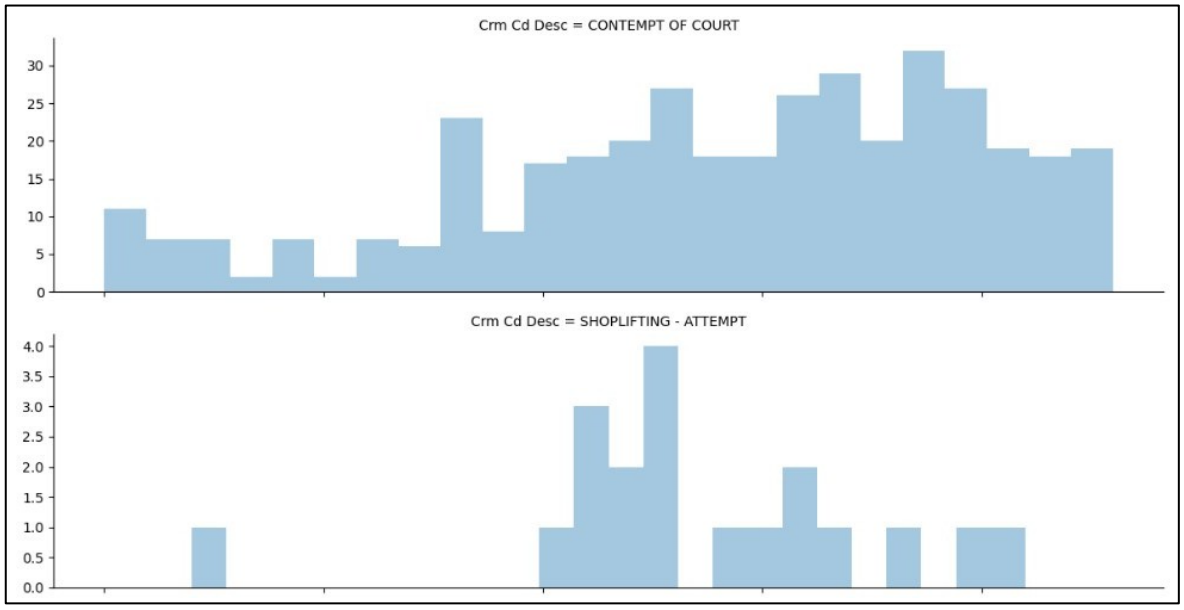

Crm Cd Desc = BUNCO, GRAND THEFT
160
140
120
100
80
60
40
20
0
Crm Cd Desc = TRESPASSING
160
140
120
100
80
60
40
20
0

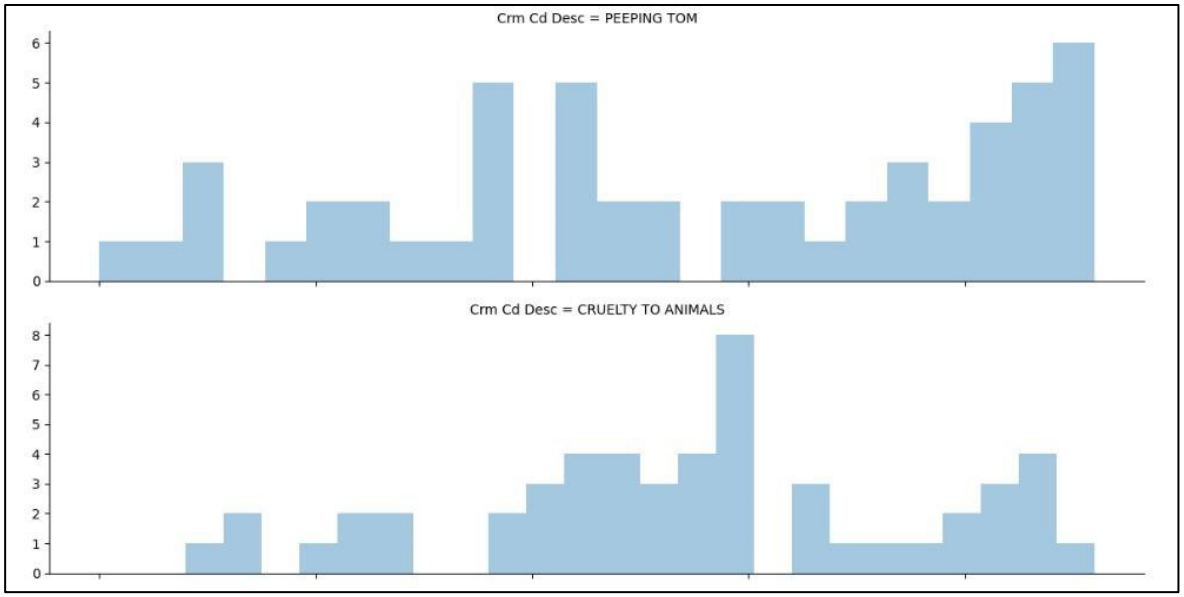

Crm Cd Desc = BUNCO, PETTY THEFT
40
35
30
25
20
15
10
5
0
Crm Cd Desc = BURGLARY, ATTEMPTED
40
30
20
10
0

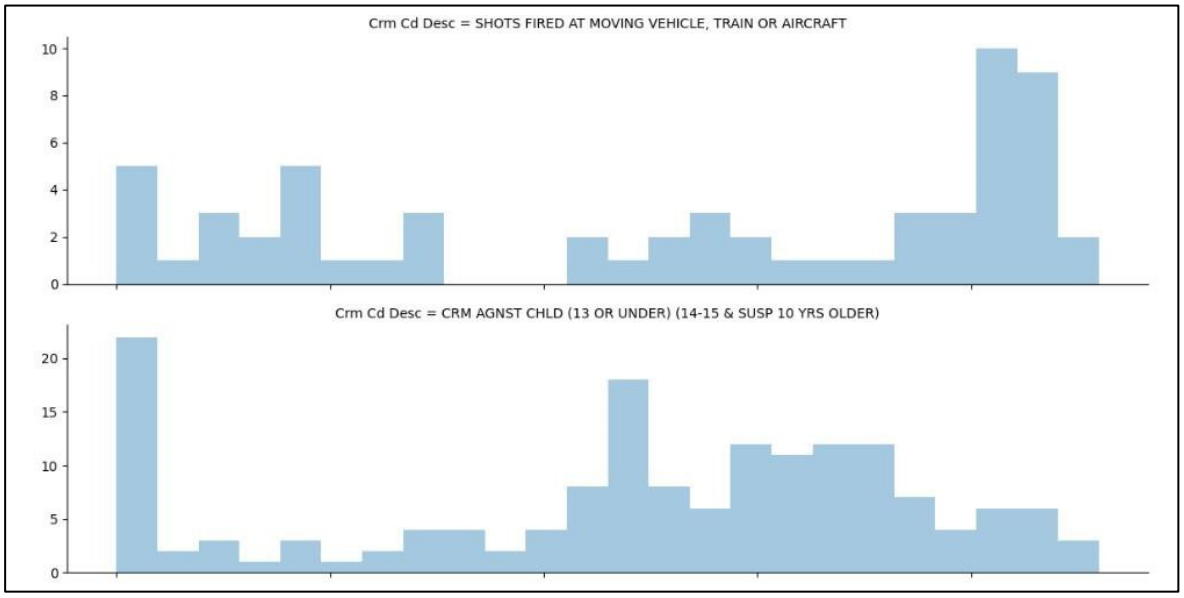

Crm Cd Desc = OTHER MISCELLANEOUS CRIME
70
60
50
40
30
20
10
0
Crm Cd Desc = TILL TAP - PETTY ($950 & UNDER)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

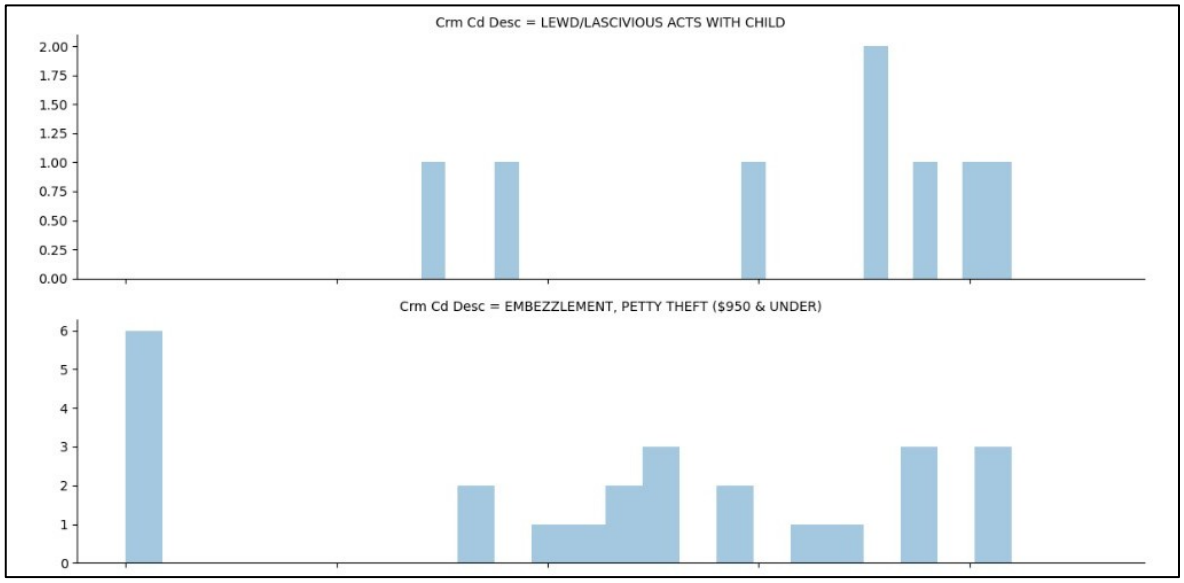

Crm Cd Desc = SHOPLIFTING-GRAND THEFT ($950.01 & OVER)
140
120
100
80
60
40
20
0
Crm Cd Desc = DISCHARGE FIREARMS/SHOTS FIRED
35
30
25
20
15
10
5
0

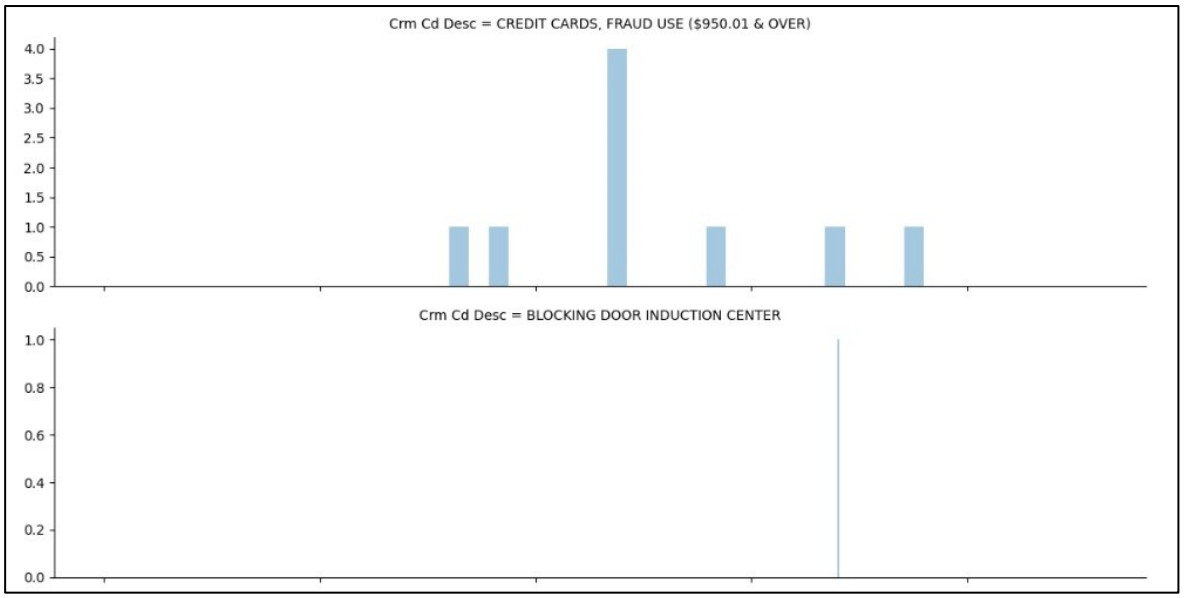

Crm Cd Desc = THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVESTK,PROD
400
300
200
100
0
Crm Cd Desc = DOCUMENT FORGERY / STOLEN FELONY
120
100
80
60
40
20
0


Crm Cd Desc = CONTEMPT OF COURT
Crm Cd Desc = SHOPLIFTING - ATTEMPT


Crm Cd Desc = PEEPING TOM
Crm Cd Desc = CRUELTY TO ANIMALS


Crm Cd Desc = SHOTS FIRED AT MOVING VEHICLE, TRAIN OR AIRCRAFT
Crm Cd Desc = CRM AGNST CHLD (13 OR UNDER) (14-15 & SUSP 10 YRS OLDER)


Crm Cd Desc = LEWD/LASCIVIOUS ACTS WITH CHILD
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Crm Cd Desc = EMBEZZLEMENT, PETTY THEFT ($950 & UNDER)
6
5
4
3
2
1
0


Crm Cd Desc = CREDIT CARDS, FRAUD USE ($950.01 & OVER)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Crm Cd Desc = BLOCKING DOOR INDUCTION CENTER
1.0
0.8
0.6
0.4
0.2
0.0

❖ Pie chart on Victim Gender.

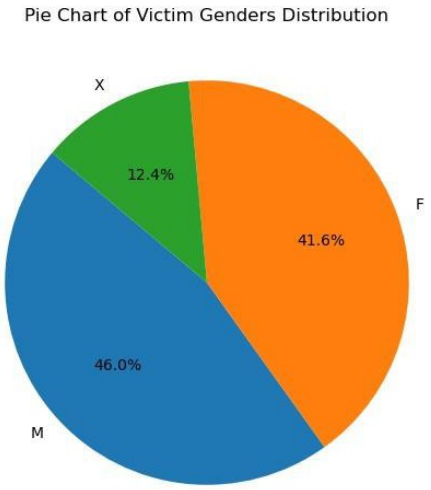
Making a Piechart on Victim Genders

In [115]:
```
data2 = pd.DataFrame(clean_df2['Vict Sex'])
to_count = ['M', 'F', 'X']
counts = data2['Vict Sex'].value_counts().reindex(to_count, fill_value = 0)
for term, count in counts.items():
    print(f"{term}: {count}")
```

```
M: 57778
F: 52196
X: 15621
```

Now that we know the number of Victims of each gender, we can plot the piechart

In [119]:
```
plt.figure(figsize = (6, 6))
plt.pie(counts, labels = counts.index, autopct = "%1.1f%%", startangle = 140)
plt.title('Pie Chart of Victim Genders Distribution')
plt.show()
```

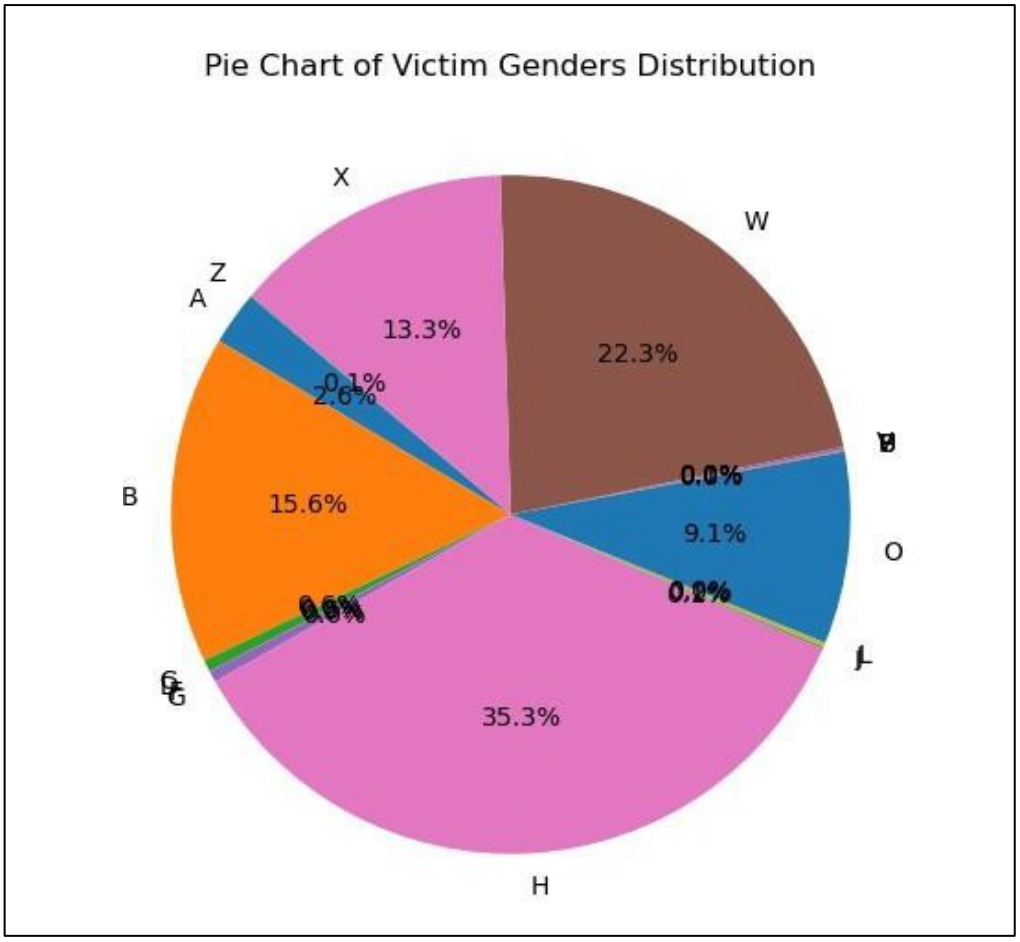


❖ Pie chart for Victim Descent.

Now, doing the same for Victim Descent

In [161]:
```
data3 = pd.DataFrame(exp_df4['Vict Descent'])
to_count = ['A', 'B', 'C', 'D', 'F', 'G', 'H', 'I', 'J', 'L', 'O', 'P', 'S', 'U', 'V', 'W', 'X', 'Z']
counts = data3['Vict Descent'].value_counts().reindex(to_count, fill_value = 0)
for term, count in counts.items():
    print(f"{term}: {count}")

plt.figure(figsize = (6, 6))
plt.pie(counts, labels = counts.index, autopct = "%1.1f%%", startangle = 140)
plt.title('Pie Chart of Victim Genders Distribution')
plt.show()
```

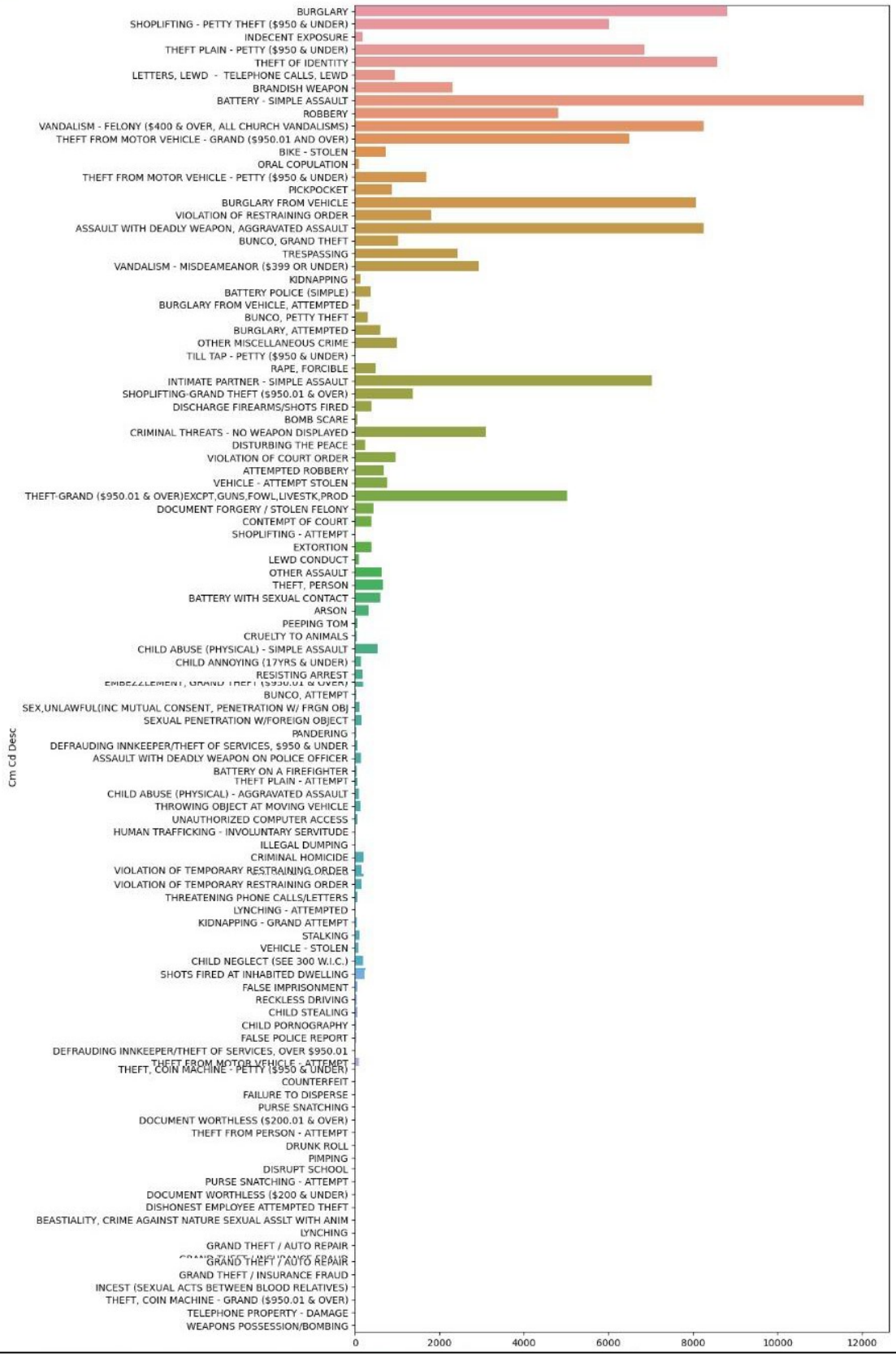
```
A: 3222
B: 19669
C: 704
D: 11
F: 683
G: 7
H: 44424
I: 161
J: 245
L: 12
O: 11503
P: 43
S: 9
U: 41
V: 183
W: 28044
X: 16768
Z: 90
```



❖ Crime frequency graph.

```
Plotting a Crime Frequency Graph

In [121]: plt.figure(figsize=(10,30))
          count = sns.countplot(
              data = clean_df2,
              y = 'Crm Cd Desc',
          )
          plt.show()
```

In [121]:
```
plt.figure(figsize=(10,30))
count = sns.countplot(
    data = clean_df2,
    y = 'Crm Cd Desc',
)
plt.show()
```



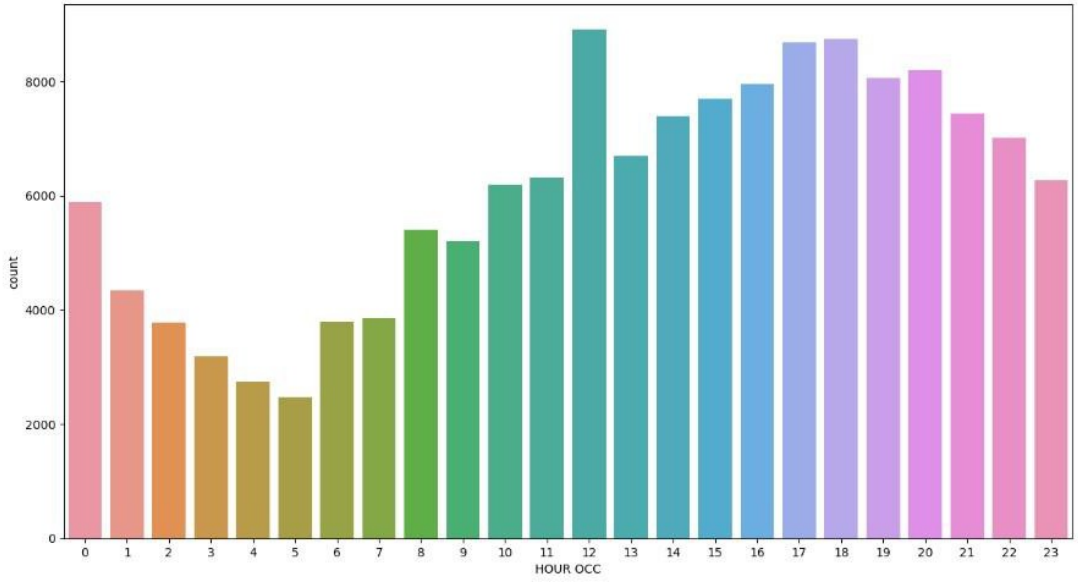
❖ Visualizing for peak crime time.

Visualizing for peak crime time

In [147]:
```
exp_df4 = pd.read_csv('C:/Users/61431/Downloads/Crime_Data_from_2020_to_Present.csv', sep = ',')

exp_df4['DATE OCC'] = exp_df4['DATE OCC'].str.extract(r'(\d{2}/\d{2}/\d{4})')
exp_df4['DATE OCC'] = pd.to_datetime(exp_df4['DATE OCC'], format='%m/%d/%Y')
exp_df4['Day'] = exp_df4['DATE OCC'].dt.day_name()
exp_df4['Month'] = exp_df4['DATE OCC'].dt.month_name()
exp_df4['Year'] = exp_df4['DATE OCC'].dt.year
exp_df4['TIME OCC'] = exp_df4['TIME OCC'].astype(str).str.zfill(4)
exp_df4['HOUR OCC'] = exp_df4['TIME OCC'].apply(lambda t: int(t[:2]))
```

In [148]:
```
exp_df4 = exp_df4[exp_df4['Year'] > 2022]
```

In [159]:
```
plt.figure(figsize=(15,8))
ax = sns.countplot(x="HOUR OCC", data=exp_df4)
```



6. SOLUTION

For this, I have used R on R Studio and Python on Google Collab as both have the necessary libraries to process and execute the algorithms.

6.1 K-Means

AIM: The aim is to determine areas names with different crime codes.

INPUT: The input file is a .csv file that has the following attributes:

- AREA NAME: The 21 Geographic Areas or Patrol Divisions are also given a name designation that references a landmark or the surrounding community that it is responsible for. For example, the 77th Street Division is located at the intersection of South Broadway and 77th Street, serving neighborhoods in South Los Angeles.
- Crm Cd: It indicates the crime committed. It is also referred as Crime Code 1

Pre-Processing Data:

For the execution of K-Means, I have pre-processed the input Excel file before loading it into Python and RStudio.

Following is the snapshot of the Excel file after dropping the unwanted columns:

AREA NAME	Crm Cd
Southwest	624
Central	624
Central	845
N Hollywood	745
Mission	740
Central	121
Central	442
Central	946
Central	341
Devonshire	341
Central	330
Central	930
Central	341
Central	648
Central	442
Mission	626
Northeast	626
Harbor	440
Van Nuys	354

After dropping the columns, I applied the transpose function on the second column that represents the crime conducted by the criminal to make it the header of the table.

I have also removed the area name duplicates.

Following is the snapshot of the table after applying the transpose function:

	624	845	745	740	121	442	946	341	330	930	648
Southwest											
Central											
N Hollywood											
Mission											
Devonshire											
Northeast											
Harbor											
Van Nuys											
West Valley											
West LA											
Wilshire											
Pacific											
Rampart											
77th Street											
Hollenbeck											
Southeast											
Hollywood											
Newton											
Topanga											
Foothill											
Olympic											

After setting up the input table, I pulled the values for each cell. These values represent the number of incidents of each crime for their respective area location.

Following is the snapshot of the table after applying the index match function:

	624	845	745	740	121	442	946	341	330	930	648	626	440	354	210
Southwest	3743	44	1325	2578	202	812	283	1102	1629	1130	101	2439	2202	3142	1656
Central	5787	139	1288	3727	309	1519	402	2173	7351	1000	181	1931	3195	1580	2263
N Hollywood	2791	21	1007	2609	157	986	341	1531	2856	532	117	1887	2145	2663	915
Mission	2111	21	762	1801	138	767	239	720	1451	550	85	2152	1158	2506	849
Devonshire	2185	11	632	1564	95	1188	245	1135	1965	583	46	1636	1528	2683	703
Northeast	2227	13	981	2306	88	537	175	1101	2467	748	106	1227	2122	1937	860
Harbor	2613	63	1120	2153	119	501	216	812	1331	825	76	2085	1308	1608	849
Van Nuys	2388	16	878	2167	67	1109	293	1014	2217	577	83	1766	1586	2639	725
West Valley	2259	35	770	1966	127	549	262	1083	2021	834	67	1793	1447	2629	774
West LA	2452	1	870	2302	110	1579	282	1700	2747	659	65	919	2467	2724	551
Wilshire	2638	143	884	2335	115	2001	417	1580	2500	788	111	1232	2469	2064	1304
Pacific	2750	5	1056	2647	143	656	422	2514	3050	691	113	1352	3548	2229	927
Rampart	3516	16	1067	2244	206	768	206	817	2071	804	115	2102	1706	2191	1715
77th Street	3819	65	1539	2720	266	287	386	876	1461	1339	170	3429	1552	3673	3211
Hollenbeck	2675	4	1032	1960	93	222	162	739	1070	617	86	1472	1261	1751	967
Southeast	3225	103	1515	2114	163	212	240	695	1236	1230	146	2831	1231	3270	2152
Hollywood	3784	12	977	2865	192	1039	352	1900	3398	925	108	1798	2997	2123	1774
Newton	3596	66	1110	2371	157	315	157	1052	2074	678	162	2275	1326	2197	1916
Topanga	2289	7	627	1676	110	1875	203	980	1831	578	42	1608	1630	2675	783
Foothill	2009	11	760	1609	100	442	211	769	1215	501	79	1552	1039	2141	589
Olympic	3774	9	1112	2485	173	465	230	1221	2538	767	127	2298	2312	1979	1550

For example, 3743 is the incident of crime code 624 that occurred in the area named Southwest.

RSTUDIO

Loading Necessary Libraries

```
# Suppressing warnings
options(warn=-1)

# Loading required libraries
library(dplyr)
```

Setting up the Multiple Histograms:

```
# Set up the layout for multiple histograms
par(mfrow=c(12, 12), mar=c(3, 3, 1, 1), oma=c(1, 1, 1, 1))

# Create histograms for each column
for (col in names(df)) {
  hist(df[[col]], main=col, xlab=col, ylab="Frequency", col="skyblue", border="black", breaks=100)
}

k_fit <- kmeans(df, centers = 19)

print(k_fit)
```

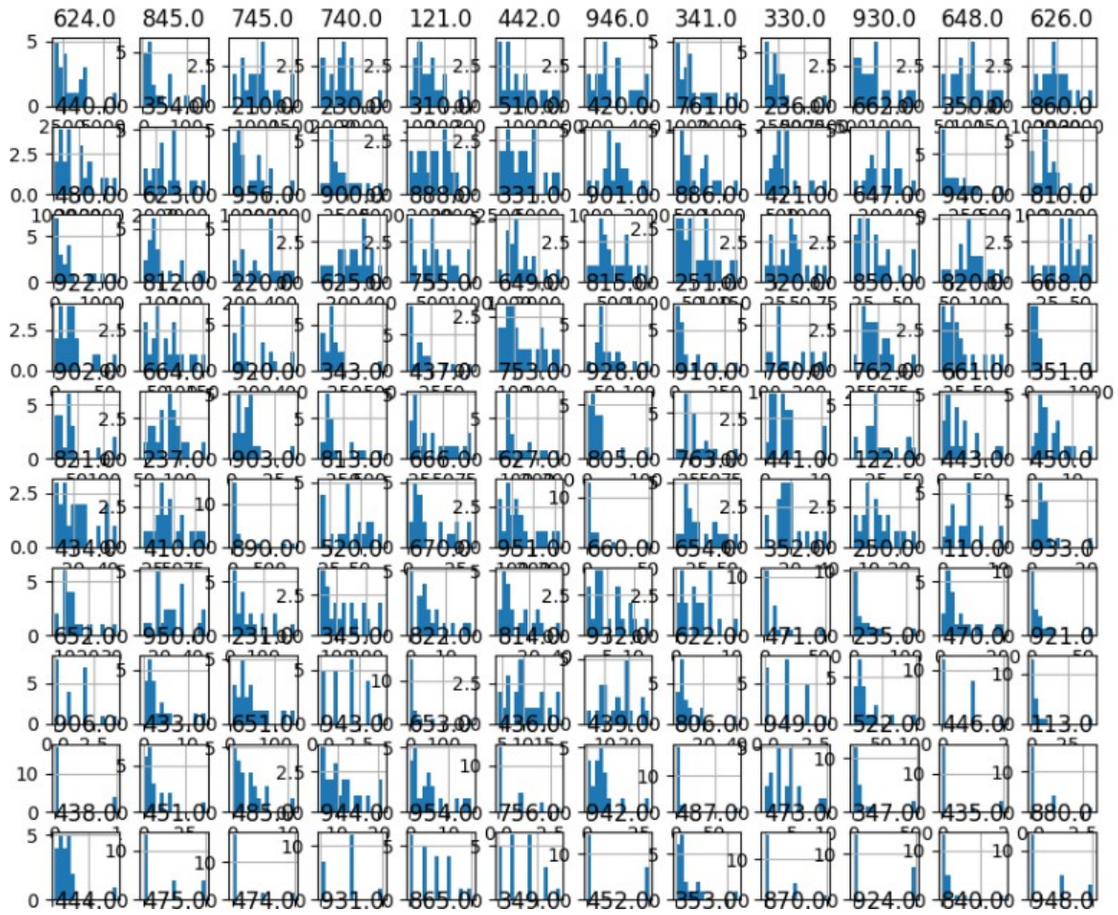
PYTHON

Loading Necessary Libraries

```
from warnings import filterwarnings
filterwarnings("ignore")
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy as sp
from sklearn.cluster import KMeans
```

Setting up the Multiple Histograms:

```
df.hist(bins=15,figsize=(10,10))
```

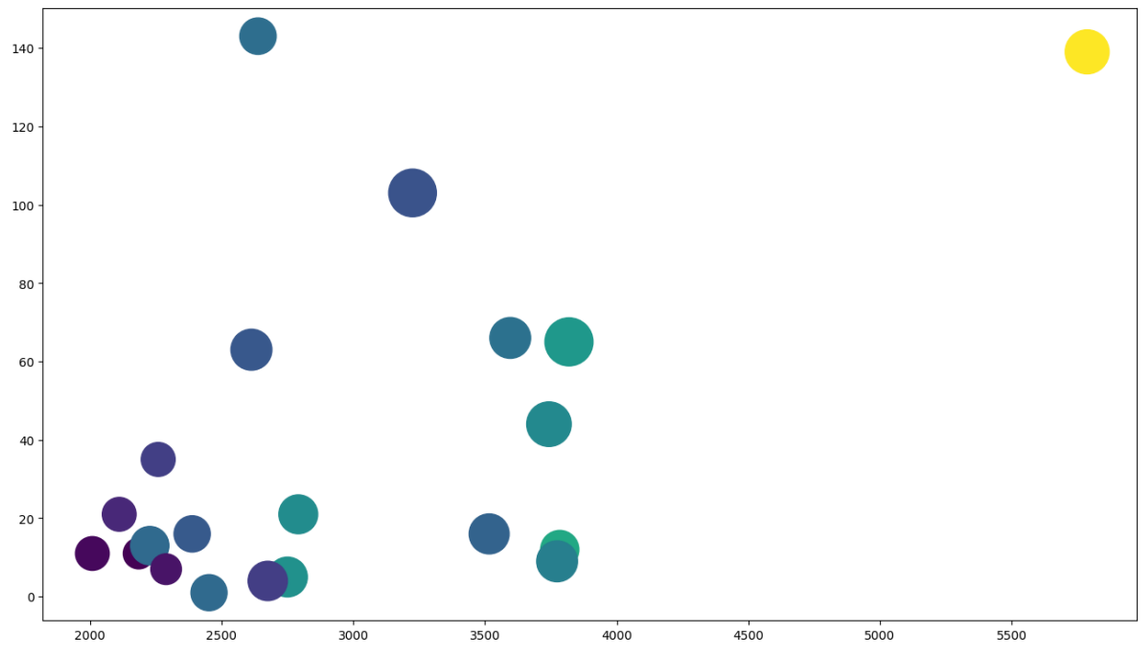


Forming clusters and representing them in scatter plot

```
kmeans= KMeans(n_clusters=21)
k_fit = kmeans.fit(df)
```

```
clusters = k_fit.labels_
```

```
plt.scatter(df.iloc[:,0],df.iloc[:,1],df.iloc[:,2],df.iloc[:,3],)
```



Silhouette Score:

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import numpy as np
```

```
labels = kmeans.labels_
```

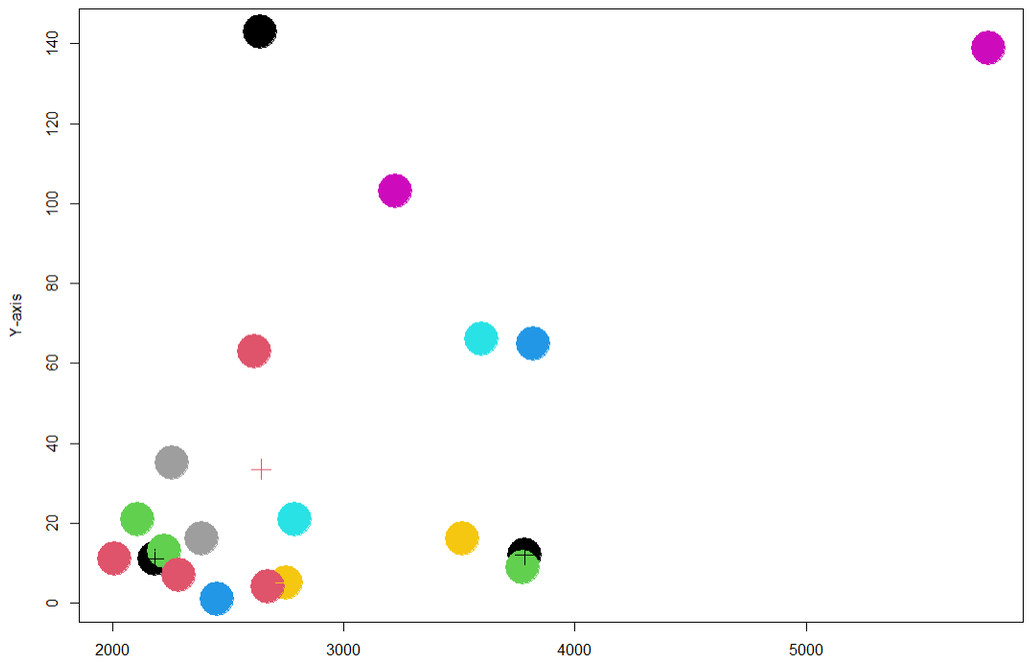
```
silhouette_avg = silhouette_score(df, labels)
```

```
print(f'Silhouette Score: {silhouette_avg}')
```

```
Silhouette Score: 0.008153126674973575
```

Forming clusters and representing them in scatter plot

```
# Scatter Plot
plot(df[, 1], df[, 2], col = k_fit$cluster, pch = 19, cex = 0.2,
main = "K-Means Clustering", xlab = "X-axis", ylab = "Y-axis")
```



Silhouette Score:

```
# Install and load the required libraries
install.packages(c("fpc", "cluster"))
library(fpc)
library(cluster)

# Assuming 'df' is the name of your data frame
# Fit K-means clustering
k_fit <- kmeans(df, centers = 19)

# Calculate dissimilarity matrix
diss_matrix <- daisy(df)

# Calculate silhouette score
silhouette_score <- cluster.stats(diss_matrix, k_fit$cluster)$silinfo$avg.width

# Print the silhouette score
cat("Silhouette Score:", silhouette_score, "\n")
```

silhouette_score	0.0091

Cluster Size

```r
# Obtain cluster sizes using table function
cluster_sizes <- table(k_fit$cluster)

# Display the cluster sizes
print("Cluster Sizes:")
print(cluster_sizes)
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  1  1  1  1
```

Optimization of Cluster Numbers - Finding Minimum Error - Model Tuning

```r
# Install and load the required packages
install.packages("factoextra")
library(factoextra)
library(cluster)

# Assuming df is your DataFrame
# Convert all column names to strings
colnames(df) <- as.character(colnames(df))

# Measure the time taken for fitting the k-means model
fit_time <- system.time({
  # Create a k-means model
  kmeans_model <- kmeans(df, centers = 8)
})[3]

# Use the fviz_nbclust function for the elbow method
elbow_visualization <- fviz_nbclust(df, kmeans, method = "wss")

# Display the visualization
print(elbow_visualization)
```

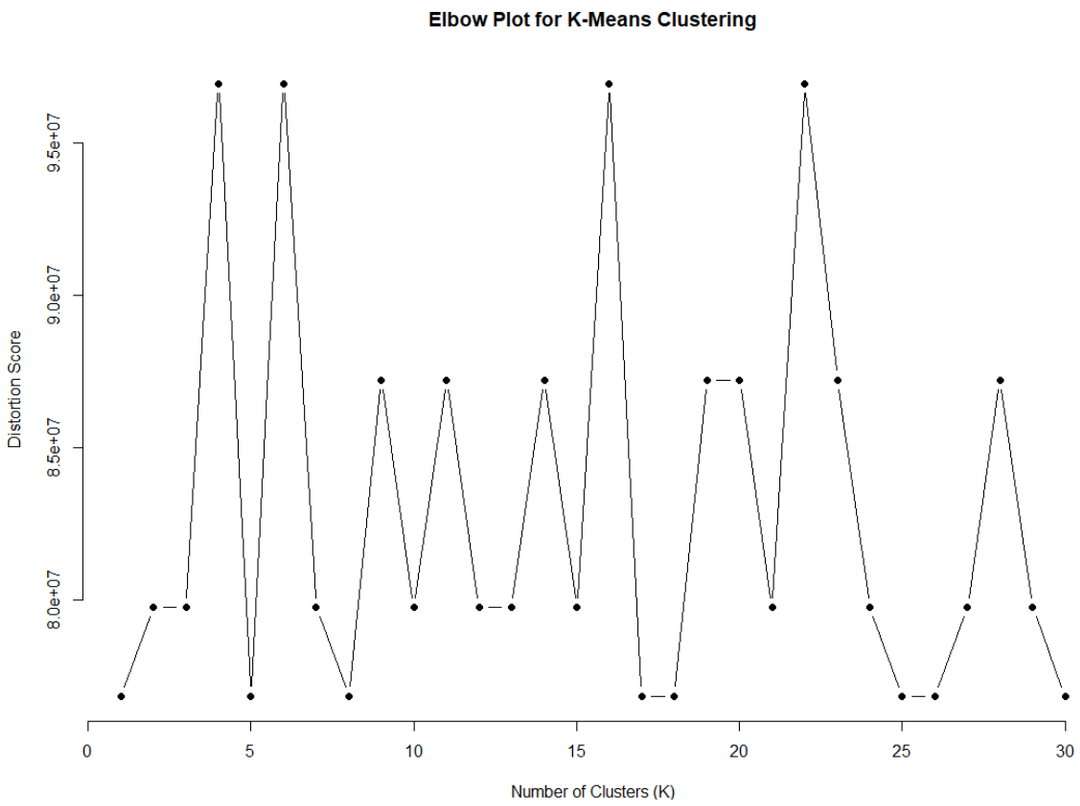
Cluster Size

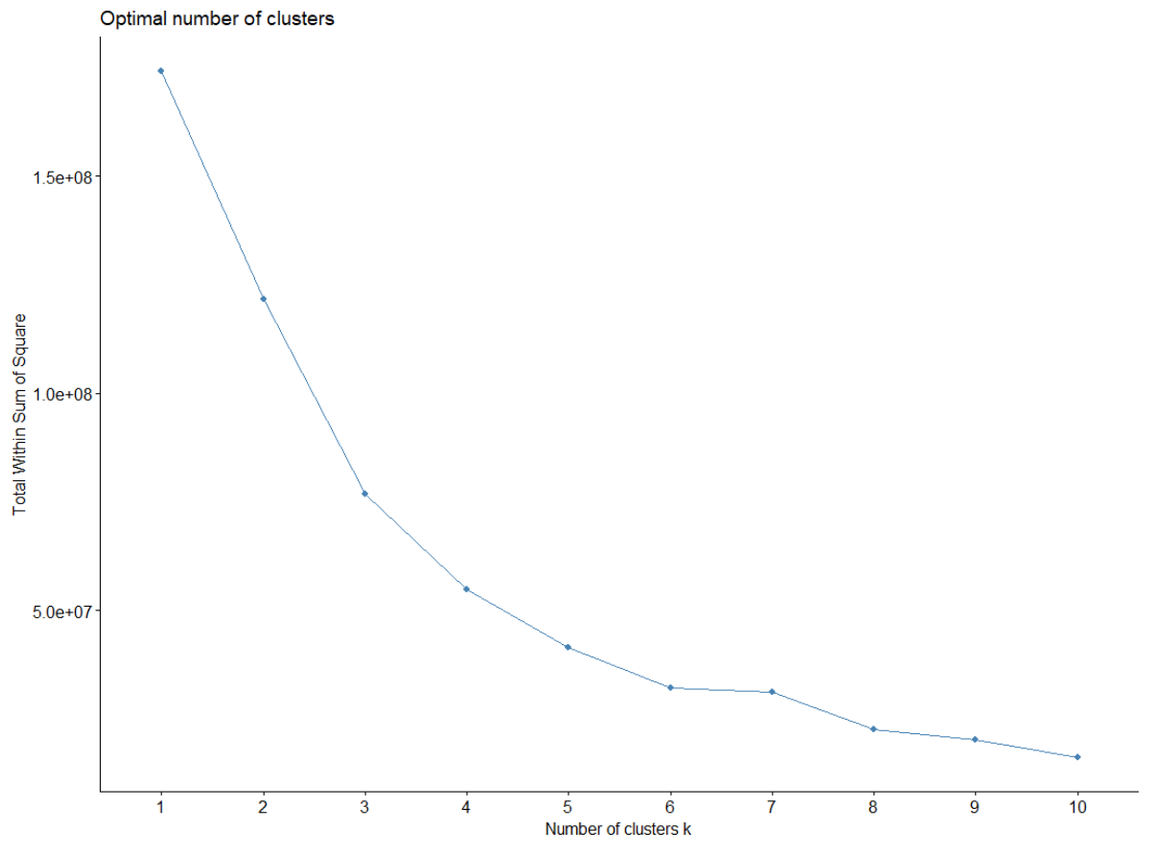
```python
for label, size in zip(unique_labels, cluster_sizes):
    print(f'Cluster {label + 1} size: {size}')
```

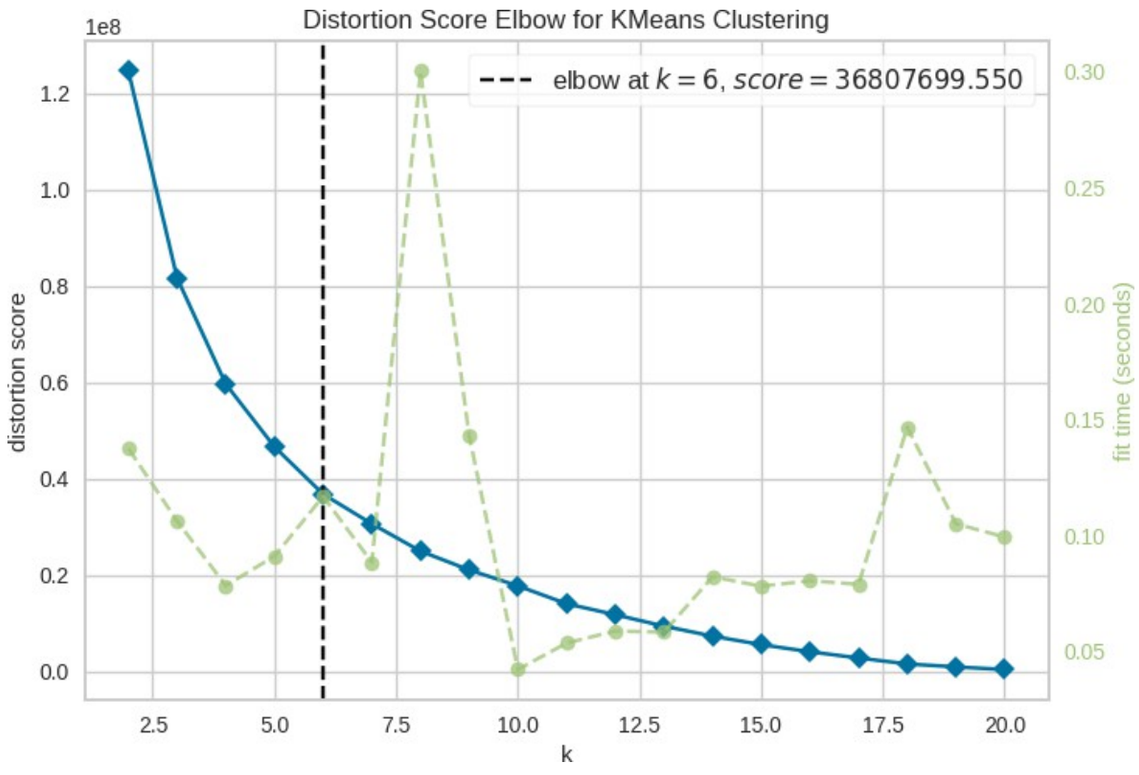
```
Cluster 1 size: 1
Cluster 2 size: 1
Cluster 3 size: 1
Cluster 4 size: 1
Cluster 5 size: 1
Cluster 6 size: 1
Cluster 7 size: 1
Cluster 8 size: 1
Cluster 9 size: 1
Cluster 10 size: 1
Cluster 11 size: 1
Cluster 12 size: 1
Cluster 13 size: 1
Cluster 14 size: 1
Cluster 15 size: 1
Cluster 16 size: 1
Cluster 17 size: 1
Cluster 18 size: 1
Cluster 19 size: 1
```

Optimization of Cluster Numbers - Finding Minimum Error - Model Tuning

```python
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans

# Assuming df is your DataFrame

# Convert all feature names to strings
df.columns = df.columns.astype(str)

# Create a KMeans model
kmeans = KMeans()

visualizer = KElbowVisualizer(kmeans, k=(2, min(50, len(df))))

# Fit the visualizer on your data
visualizer.fit(df)

# Display the visualization
visualizer.poof()
```


Elbow Plot for K-Means Clustering
Distortion Score
8.0e+07
8.5e+07
9.0e+07
9.5e+07
0
5
10
15
20
25
30
Number of Clusters (K)


Optimal number of clusters
Total Within Sum of Square
5.0e+07
1.0e+08
1.5e+08
1
2
3
4
5
6
7
8
9
10
Number of clusters k


Distortion Score Elbow for KMeans Clustering
1e8
distortion score
fit time (seconds)
elbow at k = 6, score = 36807699.550
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.05
0.10
0.15
0.20
0.25
0.30
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
k

6.2 DECISION TREE

AIM: To map all the columns with reference to the CRM_DESC. The purpose is to create a target for supervised machine learning.

INPUT: The input file is a .csv file that has the following attributes:

- DR_NO: Division of Records Number: Official file number made up of a 2-digit year, area ID, and 5 digits.
- Date Rptd: Date at which the crime was reported. This value is in MM/DD/YYYY format.
- DATE OCC: Date at which the crime occurred. This value is in MM/DD/YYYY format.
- TIME OCC: It stands for the time at which the crime occurred. This value is in 24-hour military time.
- AREA: The LAPD has 21 Community Police Stations referred to as Geographic Areas within the department. These Geographic Areas are sequentially numbered from 1-21.
- AREA NAME: The 21 Geographic Areas or Patrol Divisions are also given a name designation that references a landmark or the surrounding community that it is responsible for. For example, the 77th Street Division is located at the intersection of South Broadway and 77th Street, serving neighborhoods in South Los Angeles.
- Rpt Dist No: A four-digit code that represents a sub-area within a Geographic Area. All crime records reference the "RD" that it occurred in for statistical comparisons. Find LAPD Reporting Districts on the LA City GeoHub.
  - Part 1-2
  - Crm Cd: It indicates the crime committed. It is also referred as Crime Code 1
  - Crm Cd Desc: It stands for the description of the crime code (Crm Cd).
- Mocodes: It stands for the Modus Operandi Codes. It refers to Activities associated with the suspect in commission of the crime.See attached PDF for list of MO Codes in numerical order.
  - Vict Age: It stands for age of the victim.
- Vict Descent: It stands for the race of the victim. It is referred to by Descent Code. Descent Codes are as follows:
  - Vict Sex: It stands for the sex of the victim.
  - Premis Cd: It stands for Premise code.
  - Premis Desc: It defines the Premise description.

- Weapon Used Cd: It refers to the type of weapon used
- Weapon Desc: It describes the weapon code defined.

- Status: It refers to the status of the case. It has values IC, AA, AO, CC, JA, JO.
- Status Desc: It describes the status mentioned in the Status column. IC stands for Investigation Cont, AA stands for Adult Arrest, Adult Other, Juv(enile) Arrest, Juv(enile) Other, and UNK.
- Crm cd 1: It indicates the crime committed. Crime Code 1 is the primary and most serious one. Crime Codes 2, 3, and 4 are respectively less serious offenses. Lower crime class numbers are more serious.
- Crm cd 2: May contain a code for an additional crime, less serious than Crime Code 1.
- Crm cd 3: May contain a code for an additional crime, less serious than Crime Code 1.
- Crm cd 4: May contain a code for an additional crime, less serious than Crime Code 1.
- LOCATION: The street address of the crime incident was rounded to the nearest hundred blocks to maintain anonymity.
  - Cross Street: It stands for Cross Street of rounded Address.
  - LAT: Latitude
  - LONG: Longitude

RSTUDIO

Loading Necessary Libraries

```r
library("ggplot2")
library("dplyr")
library(lubridate)
library(tidyr)
library(caret)
library(randomForest)
library(adabag)
library(e1071)
library(party)
library(rpart)
library(pROC)
library(corrplot)
library(rattle)
library(randomForest)
library(class)
library(kernlab)
library(rpart)
```

Pre-Processing Data

```r
df <- read.csv("Crime_Data_from_2020_to_Present.csv")

df <- df[, !names(df) %in% c("AREA", "Date.Rptd", "Premis.Cd", "Weapon.Used.Cd", "Status")]

#Predicitng crime type based on other details

library(lubridate)

# Conve# Convert DATE_OCC column to a datetime object
df$DATE.OCC <- strptime(df$DATE.OCC, format = "%d/%m/%Y")

# Filter rows based on date conditions
df$DATE.OCC <- as.Date(df$DATE.OCC)

# Filter rows based on date conditions
df2 <- df %>%
  filter(format(DATE.OCC, "%Y-%m") == '2023-01' | format(DATE.OCC, "%Y-%m") == '2023-08')

summary(df2)

# Define values1 for filling missing values
values1 <- list('Crm Cd 2' = 0, 'Crm Cd 3' = 0, 'Crm Cd 4' = 0)

# Fill missing values using values1
clean_df <- df2
clean_df[is.na(clean_df$`Crm Cd 2`), 'Crm Cd 2'] <- values1$`Crm Cd 2`
clean_df[is.na(clean_df$`Crm Cd 3`), 'Crm Cd 3'] <- values1$`Crm Cd 3`
clean_df[is.na(clean_df$`Crm Cd 4`), 'Crm Cd 4'] <- values1$`Crm Cd 4`

# Drop rows with missing values in specified columns
clean_df <- clean_df[!is.na(clean_df$Crm.Cd.1), ]
clean_df <- clean_df[!is.na(clean_df$Mocodes), ]

# Define values2 for filling missing values
values2 <- list('Vict Sex' = 'X', 'Vict Descent' = 'X')

# Fill missing values using values2
clean_df2 <- clean_df
clean_df2$Vict.Sex[is.na(clean_df2$Vict.Sex)] <- values2$Vict.Sex
clean_df2$Vict.Descent[is.na(clean_df2$Vict.Descent)] <- values2$Vict.Descent

# Replace 'H' with 'X' in 'Vict Descent' column
clean_df2$Vict.Descent[clean_df2$Vict.Descent == 'H'] <- 'X'

#Remove unnecessary columns
clean_df2 <- clean_df2 %>%
  select(-c(24, 25, 26))
```

PYTHON

Loading Necessary Libraries

```python
import sklearn as sk
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
```

Pre-Processing Data

```python
df = pd.read_csv('Crime_Data_from_2020_to_Present.csv')

df.head(5)
```

```python
df = df.drop(['AREA','Date Rptd', 'Premis Cd', 'Weapon Used Cd', 'Status'], axis = 1)
df
```

```python
df.rename(columns={'DATE OCC':'DATE_OCC','Status Desc':'STATUS',
                   'AREA NAME':'AREA_NAME','Weapon Desc':'WEAPON_DESC',
                   'Crm Cd Desc':'CRM_DESC','Vict Age':'VICT_AGE', 'Vict Sex':'VICT_SEX',
                   'Vict Descent':'VICT_DESC','Crm Cd 2':'CRM_CD2','Premis Desc':'PREMIS_DESC',
                   'Crm Cd 3':'CRM_CD3', 'Crm Cd 4':'CRM_CD4', 'Rpt Dist No':'DISTR_NO','Part 1-2':'PART1_2',
                   'Crm Cd':'CRM_CD'},
                   inplace = True)
```

Now, predicitng crime type based on other details

```python
df['DATE_OCC'] = pd.to_datetime(df['DATE_OCC']) #transforming column to datetime object

df2 = df.loc[(df['DATE_OCC'].dt.to_period('M') == '2023-01')|
                            (df['DATE_OCC'].dt.to_period('M') == '2023-08')].copy()
df2
```

```python
values1 = {'Crm Cd 2': 0, 'Crm Cd 3': 0, 'Crm Cd 4': 0}
clean_df = df2.fillna(value = values1)
clean_df.head(10)

clean_df.dropna(subset = ['Crm Cd 1', 'Mocodes'], inplace = True)
clean_df

values2 = {'Vict Sex': 'X', 'Vict Descent': 'X'}
clean_df2 = clean_df.fillna(value = values2)
clean_df2.replace(['H'], 'X', inplace = True)
clean_df2
```

```r
# Function to convert four-digit integer to time
convert_to_time <- function(value) {
  # Convert to character and pad with leading zeros if needed
  value_str <- sprintf("%04d", value)

  # Parse the time as HHMM
  time <- strptime(value_str, format = "%H%M")

  # Format the time as HH:MM
  time_formatted <- format(time, format = "%H:%M")

  return(time_formatted)
}



# Apply the conversion function to TIME OCC column
clean_df3$TIME.OCC <- sapply(clean_df3$TIME.OCC, convert_to_time)


# Convert DATE OCC and TIME OCC columns to appropriate formats if needed
clean_df3$DATE.OCC <- as.Date(clean_df3$DATE.OCC, format = "%Y-%m-%d")
clean_df3$TIME.OCC <- as.character(clean_df3$TIME.OCC)  # Convert to character



# Merge DATE OCC and TIME OCC columns into a single datetime column
clean_df3$Datetime <- as.POSIXct(paste(clean_df3$DATE.OCC, clean_df3$TIME.OCC), format = "%Y-%m-%d %H:%M")

# Display the updated DataFrame
print(clean_df3)
```

```r
# Drop columns at index positions 1, 2, 3, and 20
clean_df3 <- clean_df3 %>%
  select(-c(1, 2, 3, 20))

# Add a new 'WEEKDAY' column
clean_df3$WEEKDAY <- as.character(weekdays(clean_df3$Datetime))

# Add a new 'HOUR' column
clean_df3$HOUR <- substr(as.character(clean_df3$Datetime), 12, 13)

# Display the updated DataFrame
print(clean_df3)

# Drop the 'Datetime' column
clean_df3 <- clean_df3 %>%
  select(-Datetime)

# Create a new data frame with selected columns
clean_df3 <- clean_df3 %>%
  select(
    LAT, LON, Crm.Cd, Crm.Cd.2, Crm.Cd.3, Crm.Cd.4, Vict.Age,
    Mocodes, WEEKDAY, AREA.NAME, Rpt.Dist.No, Crm.Cd.Desc, Premis.Desc, HOUR,
    Weapon.Desc, Vict.Descent, Vict.Sex, Part.1.2
  )

# Split the 'Mocodes' column by spaces and create a list of values
mocodes_list <- strsplit(clean_df3$Mocodes, " ")



# Create a data frame of dummy variables
```

```r
# Create a data frame of dummy variables
mocodes_dummies <- as.data.frame(matrix(0, nrow = nrow(clean_df3), ncol = length(unique(unlist(mocodes_list)))))
colnames(mocodes_dummies) <- unique(unlist(mocodes_list))



# Populate the dummy variables
for (i in 1:nrow(mocodes_dummies)) {
  mocodes_dummies[i, unlist(mocodes_list[[i]])] <- 1
}



# Display the created dummy variables
print(mocodes_dummies)

# Merge the data frames based on the row index
df_final <- merge(clean_df3, mocodes_dummies, by = 0)

# Rename the 'Row.names' column (which corresponds to the row index) to 'index'
colnames(df_final)[1] <- 'index'

# Drop the 'index' column
df_final <- df_final %>%
  select(-index)

# Display the final data frame
print(df_final)

# Combine CRM_CD2, CRM_CD3, and CRM_CD4 columns into a single column
df_final$Crm.Cd.2 <- paste(df_final$Crm.Cd.2, df_final$Crm.Cd.3, df_final$Crm.Cd.4, sep = " ")
```

```python
from datetime import time


def convert_to_time(value):
    hours = value // 100
    minutes = value % 100
    return time(hours, minutes)


clean_df2['TIME OCC'] = clean_df2['TIME OCC'].apply(convert_to_time)

clean_df2['TIME OCC'] = clean_df2['TIME OCC'].astype(str).str.zfill(4)

clean_df2
```

```python
import pandas as pd

# Assuming you have a DataFrame 'clean_df2' with 'DATE_OCC' and 'TIME OCC' columns
# Convert 'DATE_OCC' and 'TIME OCC' columns to datetime separately
clean_df2['DATE_OCC'] = pd.to_datetime(clean_df2['DATE_OCC'])
clean_df2['TIME OCC'] = pd.to_datetime(clean_df2['TIME OCC'])

# Merge date and time columns into a single datetime column
clean_df2['Datetime'] = clean_df2['DATE_OCC'] + pd.to_timedelta(clean_df2['TIME OCC'].dt.strftime('%H:%M:%S'))

# Display the updated DataFrame
print(clean_df2)
```

```python
clean_df2.drop(clean_df2.columns[[0, 1, 2, 20]], axis = 1, inplace = True)
```

```python
clean_df2['WEEKDAY'] = clean_df2['Datetime'].dt.dayofweek.astype(str).copy()
clean_df2['HOUR'] = clean_df2['Datetime'].astype(str).str[11:13].copy()

clean_df2
```

```python
clean_df2.drop('Datetime', axis=1, inplace=True)
```

```python
clean_df2.info()
```

```
clean_df2 = clean_df2[['LAT','LON','LOCATION', 'CRM_CD', 'CRM_CD2','CRM_CD3','CRM_CD4','VICT_AGE','Mocodes','WEEKDAY',
                       'AREA_NAME','DISTR_NO','CRM_DESC','PREMIS_DESC','HOUR',
                       'WEAPON_DESC','VICT_DESC','VICT_SEX','PART1_2']].copy()
clean_df2
```

```
mocodes_dummies = clean_df2['Mocodes'].str.join(sep='').str.get_dummies(sep=' ')
mocodes_dummies
```

```
df_final = pd.merge(clean_df2,mocodes_dummies, left_on=clean_df2.index, right_on=mocodes_dummies.index )
df_final = df_final.drop(['key_0'], axis=1) # dropping the new column of the index(keys)
df_final
```

```
# Drop CRM_CD3 and CRM_CD4 columns
df_final <- df_final %>%
  select(-Crm.Cd.3, -Crm.Cd.4)

# Display the updated data frame
print(df_final)

library(tidyr)

# Create dummy variables from the 'CRM_CD2' column
crime_codes_dummies <- df_final %>%
  separate_rows(Crm.Cd.2, sep = " ") %>%
  pivot_wider(names_from = Crm.Cd.2, values_from = Crm.Cd.2, values_fn = length, values_fill = 0)

# Drop columns containing "nan" or "NA" in the name
crime_codes_dummies <- crime_codes_dummies %>%
  select(-matches("nan|NA", ignore.case = TRUE))

# Create a temporary index column for both data frames
df_final$index <- seq_len(nrow(df_final))
crime_codes_dummies$index <- seq_len(nrow(crime_codes_dummies))

# Merge the data frames based on the 'index' column
df_final2 <- merge(df_final, crime_codes_dummies, by = "index")

# Remove the extra 'index' column
df_final2 <- df_final2 %>%
  select(-index)
```

```
# Load required packages
library(caret)
library(dplyr)

# Columns to create dummy variables for
cols_to_dummify <- c('Vict_Age', 'AREA_NAME', 'Rpt.Dist.No', 'Premis_Desc',
                     'Weapon_Desc', 'Vict_Descent', 'Vict_Sex', 'Part.1.2', 'WEEKDAY', 'HOUR')

# Create a new data frame for dummy variables
df_final3 <- df_final2
colnames(df_final2)

# Select only the columns to keep
df_final2 = df_final2[columns_to_keep]
# Loop through each column and create dummy variables
for (col in cols_to_dummify) {
  dummy_col <- as.data.frame(model.matrix(~ . - 1, data = df_final2[col]))
  colnames(dummy_col) <- paste(col, colnames(dummy_col), sep = "_")
  df_final3 <- cbind(df_final3, dummy_col)
}

# Drop specified columns
cols_to_drop <- c('Mocodes', 'CRM_CD2')
df_final3 <- df_final3 %>%
  select(-one_of(cols_to_drop))
```

Shuffling the Data

```
#Shuffling the data
# Load libraries
library(caret)
library(randomForest)
library(adabag)
library(e1071)
library(party)
library(rpart)
library(pROC)
library(corrplot)
library(rattle)
library(randomForest)
library(class)
library(kernlab)

# Set a random seed for reproducibility
set.seed(9)

# Shuffle the data (equivalent to random_state=9 in Python)
df_final3 <- df_final3[sample(nrow(df_final3)), ]

# Separate the target column
target <- df_final3$Crm.Cd.Desc.x
df_final3 <- df_final3[, !(names(df_final3) %in% c("Crm.Cd.x"))]

# Split the dataset into training and testing sets
set.seed(7)  # Equivalent to random_state=7 in Python
splitIndex <- createDataPartition(target, p = 0.8, list = FALSE)
```

```
df_final['CRM_CD2'] = df_final['CRM_CD2'].astype(str)+ ' ' +df_final['CRM_CD3'].astype(str) +' '+\
                          +df_final['CRM_CD4'].astype(str)
df_final = df_final.drop(['CRM_CD3','CRM_CD4'], axis=1)
df_final
```

```
crime_codes_dummies= df_final['CRM_CD2'].str.join(sep='').str.get_dummies(sep=' ')
crime_codes_dummies = crime_codes_dummies.drop(['nan'], axis=1) # dropping -1 column as it concerns incidents with no extra crime type
crime_codes_dummies
```

```
df_final = pd.merge(df_final, crime_codes_dummies, left_on=df_final.index,
                            right_on=crime_codes_dummies.index )
df_final2 = df_final.drop(['key_0'], axis=1) # dropping the new column of the index(keys)
df_final2
```

```
df_final3 = pd.get_dummies(df_final2, columns=['LOCATION','VICT_AGE',
                                               'AREA_NAME','DISTR_NO', 'PREMIS_DESC', 'WEAPON_DESC',
                                               'VICT_DESC', 'VICT_SEX', 'PART1_2','WEEKDAY','HOUR'])
df_final3 = df_final3.drop(['Mocodes','CRM_CD2'], axis=1)
df_final3
```

Shuffling the Data

```
import sklearn as sk
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
import random
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import classification_report
```

```
#Training
X_train <- df_final3[splitIndex, ]
X_test <- df_final3[-splitIndex, ]
y_train <- target[splitIndex]
y_test <- target[-splitIndex]
```

Executing the Decision Tree Algorithm

```
X_test <- read.csv("X_test.csv")
X_train <- read.csv("X_train.csv")
y_train <- read.csv("y_train.csv")
y_test <- read.csv("y_test.csv")

X_test2 <- X_test[1:1500, ]
X_train2 <- X_train[1:3000, ]
y_test2 <- y_test[1:1500, ]
y_train2 <- y_train[1:3000, ]

y_test3 <- y_test[1:79, ]

train_data <- data.frame(y_train2, X_train2)

# Create the Decision Tree classifier
clf_gini <- rpart(y_train2 ~ ., data = train_data, method = "class", parms = list(split = "gini"))

# Make predictions on the test data
gini_predict <- predict(clf_gini, X_test2, type = "class")


# Calculate accuracy
accuracy <- sum(y_test2 == gini_predict) / length(y_test2)

# Print the accuracy
cat("Accuracy (gini):", accuracy, "\n")
```

```
# Load the required packages
library(caret)
library(e1071)

#testing
unique_levels_gini <- levels(gini_predict)
unique_levels_y_test <- levels(y_test2)

extra_levels_in_gini <- setdiff(unique_levels_gini, unique_levels_y_test)
gini_predict <- factor(gini_predict, levels = unique_levels_y_test)

# Ensure that gini_predict and y_test2 are factors with the same levels
gini_predict <- factor(gini_predict, levels = levels(y_test2))
# Create a confusion matrix
conf_matrix <- confusionMatrix(gini_predict, y_test3)

# Generate the classification report
classification_report <- data.frame(
  Precision = conf_matrix$byClass["Pos Pred Value"],
  Recall = conf_matrix$byClass["Sensitivity"],
  F1_Score = conf_matrix$byClass["F1"],
  Support = conf_matrix$byClass["Detection Rate"]
)

# Print the classification report
print(classification_report)


temp <- names(X_test2)
```

```
df_final3 = df_final3.sample(frac = 1, random_state=9) #shuffling data
target = df_final3['CRM_DESC'].copy() # target column separated
df_final3 = df_final3.drop(['CRM_DESC'], axis=1) #deleting target column from data
X_train, X_test, y_train, y_test = train_test_split(df_final3, target, test_size=0.2,  random_state=7)
```

```
y_test
```

```
5699     VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...
18723                                BURGLARY FROM VEHICLE
2946                                        OTHER ASSAULT
20583    THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND ...
21832    VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...
                               ...
23865    VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...
3507                   INTIMATE PARTNER - SIMPLE ASSAULT
4279                               BATTERY - SIMPLE ASSAULT
4691     THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND ...
4471                               VIOLATION OF COURT ORDER
Name: CRM_DESC, Length: 5919, dtype: object
```

```
# Save X_train, X_test, y_train, and y_test as CSV files
X_train.to_csv('X_train.csv', index=False)
X_test.to_csv('X_test.csv', index=False)
y_train.to_csv('y_train.csv', index=False, header=['y_train'])  # Adding a header for y_train
y_test.to_csv('y_test.csv', index=False, header=['y_test'])  # Adding a header for y_test
```

```
clf_gini = tree.DecisionTreeClassifier(criterion='gini')
clf_gini = clf_gini.fit(X_train, y_train)
gini_predict = clf_gini.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
```

```
accuracy = accuracy_score(y_test, gini_predict)

# Print the accuracy
print("Accuracy (gini):", accuracy)
```

```
Accuracy (gini): 0.9993242101706369
```

```
clf_ent = tree.DecisionTreeClassifier(criterion='entropy')
clf_ent = clf_ent.fit(X_train, y_train)
ent_predict = clf_ent.predict(X_test)
```

```
accuracy2 = accuracy_score(y_test, ent_predict)

# Print the accuracy
print("Accuracy (ent):", accuracy2)
```

```
Accuracy (ent): 0.9996621050853185
```

```
print(classification_report(y_test, gini_predict, zero_division=1))
```

DECISION TREE FOR R STUDIO

```
LOCATION_20600....COHASSET.....ST = FALSE, LOCATION_20600....DEER.GRASS.....CT = FALSE, LOCATION_20600....HARTLAND.....ST = FALSE, LOCATION_20600....LASSEN.....ST = FAL
SE, LOCATION_20600....LONDELIUS.....ST = FALSE, LOCATION_20600....NORDHOFF.....ST = FALSE, LOCATION_20600....PLUMMER.....ST = FALSE, LOCATION_20600....SHERMAN.....WY =
FALSE, LOCATION_20600....SKOURAS.....DR = FALSE,
LOCATION_20600....TRIBUNE.....ST = FALSE, LOCATION_20600....TULSA.....ST = FALSE, LOCATION_20600....VANOWEN.....ST = FALSE, LOCATION_20600....VENTURA.....BL = FAL
SE, LOCATION_20700....ARMINTA.....ST = FALSE, LOCATION_20700....BERMUDA.....ST = FALSE, LOCATION_20700....BURBANK.....BL = FALSE, LOCATION_20700....DEVONSHIRE.....ST =
FALSE, LOCATION_20700....KITTRIDGE.....ST = FALSE,
LOCATION_20700....LEMARSH.....ST = FALSE, LOCATION_20700....NORDHOFF.....ST = FALSE, LOCATION_20700....VANOWEN.....ST = FALSE, LOCATION_20700....WALNUT.CIR = FALSE, LOCATION_2080
0....BRYANT.....ST = FALSE, LOCATION_20800....LASSEN.....ST = FALSE, LOCATION_20800....MARTHA.....ST = FALSE, LOCATION_20800....SHERMAN.....WY = FALSE, LOCATION_
20800....VENTURA.....BL = FALSE,
LOCATION_20800....VINTAGE.....ST = FALSE, LOCATION_20900....BLYTHE.....ST = FALSE, LOCATION_20900....COMMUNITY.....ST = FALSE, LOCATION_20900....DE.SOTO.....ST = FAL
SE, LOCATION_20900....GRESHAM.....ST = FALSE, LOCATION_20900....HALLDALE.....AV = FALSE, LOCATION_20900....HEMMINGWAY.....ST = FALSE, LOCATION_20900....INGOMAR.....ST =
FALSE, LOCATION_20900....LASSEN.....ST = FALSE,
LOCATION_20900....PARTHENIA.....ST = FALSE, LOCATION_20900....SATICOY.....ST = FALSE, LOCATION_20900....SHERMAN.....WY = FALSE, LOCATION_20900....VANOWEN.....ST = FAL
SE, LOCATION_20900....VOSE.....ST = FALSE, LOCATION_20900....WYANDOTTE.....ST = FALSE, LOCATION_20TH = FALSE, LOCATION_2100....6TH.....AV = FALSE, LOCATION_2100....99T
H.....PL = FALSE, LOCATION_2100....AARON.....ST = FALSE,
LOCATION_2100....ALTA.....ST = FALSE, LOCATION_2100....BELLEVUE.....AV = FALSE, LOCATION_2100....BELOIT.....AV = FALSE, LOCATION_2100....BEVERLY.....BL = FALSE,
LOCATION_2100....BONSALLO.....AV = FALSE, LOCATION_2100....BUTLER.....AV = FALSE, LOCATION_2100....CAMORILLA.....DR = FALSE, LOCATION_2100....CENTURY.PARK.....LN = FALSE,
LOCATION_2100....COLBY.....AV = FALSE,
LOCATION_2100....COLORADO.....BL = FALSE, LOCATION_2100....COTNER.....AV = FALSE, LOCATION_2100....COVE.....AV = FALSE, LOCATION_2100....DUANE.....ST = FALSE,
LOCATION_2100....DUXBURY.CIR = FALSE, LOCATION_2100....ELSINORE.....ST = FALSE, LOCATION_2100....FAIR.PARK.....AV = FALSE, LOCATION_2100....GATES.....ST = FALSE, LOCATION_2100....GLE
NDON.....AV = FALSE,
LOCATION_2100....HILLHURST.....AV = FALSE, LOCATION_2100....JAMES.M.WOOD.....BL = FALSE, LOCATION_2100....JOHN.S.GIBSON.....BL = FALSE, LOCATION_2100....KELTON.....AV = FALSE,
LOCATION_2100....KENILWORTH.....AV = FALSE, LOCATION_2100....KERWOOD.....AV = FALSE, LOCATION_2100....LINCOLN.PARK.....AV = FALSE, LOCATION_2100....MANITOU.....AV = FALSE,
LOCATION_2100....MANNING.....AV = FALSE,
LOCATION_2100....MONTANA.....ST = FALSE, LOCATION_2100....PARKSIDE.....AV = FALSE, LOCATION_2100....RIDGE.....DR = FALSE, LOCATION_2100....SANTA.ANA.....BL = FALSE,
LOCATION_2100....VENICE.....BL = FALSE, LOCATION_2100....VINE.....ST = FALSE, LOCATION_2100....VIOLET.....ST = FALSE, LOCATION_2100....WEST.....BL = FALSE,
LOCATION_2100....YOSEMITE.....DR = FALSE,
LOCATION_2100.E..101ST.....ST = FALSE, LOCATION_2100.E..103RD.....ST = FALSE, LOCATION_2100.E..111TH.....ST = FALSE, LOCATION_2100.E..114TH.....ST = FALSE,
LOCATION_2100.E..115TH.....ST = FALSE, LOCATION_2100.E..14TH.....ST = FALSE, LOCATION_2100.E..1ST.....ST = FALSE, LOCATION_2100.E..7TH.....PL = FALSE,
LOCATION_2100.E..97TH.....ST = FALSE,
LOCATION_2100.E..CESAR.E.CHAVEZ.....AV = FALSE, LOCATION_2100.N..BEACHWOOD.....DR = FALSE, LOCATION_2100.N..BROADWAY = FALSE, LOCATION_2100.N..CAHUENGA.....BL = FALSE, LOCATION_2100.N..GAFFE
Y.....ST = FALSE, LOCATION_2100.N..HICKS.....AV = FALSE, LOCATION_2100.N..HIGHLAND.....AV = FALSE, LOCATION_2100.N..MARIANNA.....AV = FALSE, LOCATION_2100.S..ALM
A.....ST = FALSE, LOCATION_2100.S..BEVERLY.GLEN.....BL = FALSE,
LOCATION_2100.S..BROADWAY = FALSE, LOCATION_2100.S..CENTRAL.....AV = FALSE, LOCATION_2100.S..DUNSMUIR.....AV = FALSE, LOCATION_2100.S..GAFFEY.....ST = FALSE, LOCATION_2100.S..GRAN
D.....AV = FALSE, LOCATION_2100.S..ORANGE.....DR = FALSE, LOCATION_2100.S..PACIFIC.....AV = FALSE, LOCATION_2100.W..25TH.....ST = FALSE, LOCATION_2100.W..26T
H.....PL = FALSE, LOCATION_2100.W..27TH.....ST = FALSE,
LOCATION_2100.W..30TH.....ST = FALSE, LOCATION_2100.W..31ST.....ST = FALSE, LOCATION_2100.W..3RD.....ST = FALSE, LOCATION_2100.W..74TH.....ST = FALSE,
LOCATION_2100.W..77TH.....ST = FALSE, LOCATION_2100.W..78TH.....ST = FALSE, LOCATION_2100.W..7TH.....ST = FALSE, LOCATION_2100.W..80TH.....ST = FALSE,
LOCATION_2100.W..83RD.....ST = FALSE,
LOCATION_2100.W..8TH.....ST = FALSE, LOCATION_2100.W..FLORENCE.....AV = FALSE, LOCATION_2100.W..MANCHESTER.....AV = FALSE, LOCATION_2100.W..PASEO.DEL.MAR = FALSE, LOCATION_2100.W..S
UNSET.....BL = FALSE, LOCATION_21000....BRYANT.....ST = FALSE, LOCATION_21000....DEVONSHIRE.....ST = FALSE, LOCATION_21000....ERWIN.....ST = FALSE, LOCATION_2100
0....GAULT.....ST = FALSE,
LOCATION_21000....KITTRIDGE.....ST = FALSE, LOCATION_21000....LASSEN.....ST = FALSE, LOCATION_21000....LEMARSH.....ST = FALSE, LOCATION_21000....OSBORNE.....ST = FAL
SE, LOCATION_21000....PARTHENIA.....ST = FALSE, LOCATION_21000....PLUMMER.....ST = FALSE, LOCATION_21000....RODAX.....ST = FALSE, LOCATION_21000....SATICOY.....ST =
FALSE, LOCATION_21000....SHERMAN.....WY = FALSE,
LOCATION_21000....VANOWEN.....ST = FALSE, LOCATION_21000....VENTURA.....BL = FALSE, LOCATION_21000....VICTORY.....BL = FALSE, LOCATION_21100....CHASE.....ST = FAL
SE, LOCATION_21100....DENKER.....AV = FALSE, LOCATION_21100....DEVONSHIRE.....ST = FALSE, LOCATION_21100....LASSEN.....ST = FALSE, LOCATION_21100....NASHVILLE.....ST =
FALSE, LOCATION_21100....SATICOY.....ST = FALSE,
LOCATION_21100....SHERMAN.....WY = FALSE, LOCATION_21100....VENTURA.....BL = FALSE, LOCATION_21200....DEERING.....CT = FALSE, LOCATION_21200....DEVONSHIRE.....ST = FAL
SE, LOCATION_21200....KITTRIDGE.....ST = FALSE, LOCATION_21200....LASSEN.....ST = FALSE, LOCATION_21200....LULL.....ST = FALSE, LOCATION_21200....MULHOLLAND.....DR =
FALSE, LOCATION_21200....OXNARD.....ST = FALSE,
LOCATION_21200....ROSCOE.....BL = FALSE, LOCATION_21200....SATICOY.....ST = FALSE, LOCATION_21200....SHERMAN.....WY = FALSE, LOCATION_21200....VENTURA.....BL = FAL
SE, LOCATION_21200.S..WESTERN.....AV = FALSE, LOCATION_21300....CHASE.....ST = FALSE, LOCATION_21300....COMMUNITY.....ST = FALSE, LOCATION_21300....COSTANSO.....ST =
FALSE, LOCATION_21300....ERWIN.....ST = FALSE,
LOCATION_21300....LEMARSH.....ST = FALSE, LOCATION_21300....NORDHOFF.....ST = FALSE, LOCATION_21300....PARTHENIA.....ST = FALSE, LOCATION_21300....ROSCOE.....BL = FAL
SE, LOCATION_21300....SATICOY.....ST = FALSE, LOCATION_21300....SHERMAN.....WY = FALSE, LOCATION_21300....SUPERIOR.....ST = FALSE, LOCATION_21300....VANOWEN.....ST =
FALSE, LOCATION_21300....VENTURA.....BL = FALSE,
LOCATION_21300....VICTORY.....BL = FALSE, LOCATION_21400....CHASE.....ST = FALSE, LOCATION_21400....DEVONSHIRE.....ST = FALSE, LOCATION_21400....GLEDHILL.....ST = FAL
```

TEXT REPRESENTATION OF DECISION TREE FOR PYTHON

```
|--- CRM_CD <= 330.50
|    |--- CRM_CD <= 325.00
|    |    |--- CRM_CD <= 280.50
|    |    |    |--- CRM_CD <= 215.00
|    |    |    |    |--- CRM_CD <= 166.00
|    |    |    |    |    |--- CRM_CD <= 115.50
|    |    |    |    |    |    |--- class: CRIMINAL HOMICIDE
|    |    |    |    |    |--- CRM_CD >  115.50
|    |    |    |    |    |    |--- CRM_CD <= 121.50
|    |    |    |    |    |    |    |--- class: RAPE, FORCIBLE
|    |    |    |    |    |    |--- CRM_CD >  121.50
|    |    |    |    |    |    |    |--- class: RAPE,  ATTEMPTED
|    |    |    |    |--- CRM_CD >  166.00
|    |    |    |    |    |--- class: ROBBERY
|    |    |    |--- CRM_CD >  215.00
|    |    |    |    |--- CRM_CD <= 230.50
|    |    |    |    |    |--- CRM_CD <= 225.00
|    |    |    |    |    |    |--- class: ATTEMPTED ROBBERY
|    |    |    |    |    |--- CRM_CD >  225.00
|    |    |    |    |    |    |--- class: ASSAULT WITH DEADLY WEAPON,          AGGRAVATED
                                                                               ASSAULT
|    |    |    |    |--- CRM_CD >  230.50
|    |    |    |    |    |--- 2000 <= 0.50
|    |    |    |    |    |    |--- CRM_CD <= 250.50
|    |    |    |    |    |    |    |--- CRM_CD <= 233.00
|    |    |    |    |    |    |    |    |--- class: ASSAULT WITH DEADLY WEAPON ON POLICE OFFICER
|    |    |    |    |    |    |    |--- CRM_CD >  233.00
|    |    |    |    |    |    |    |    |--- PART1_2_2 <= 0.50
|    |    |    |    |    |    |    |    |    |--- CRM_CD <= 235.50
|    |    |    |    |    |    |    |    |    |    |--- class: CHILD ABUSE (PHYSICAL) - AGGRAVATED ASSAULT
|    |    |    |    |    |    |    |    |    |--- CRM_CD >  235.50
|    |    |    |    |    |    |    |    |    |    |--- CRM_CD <= 243.00
|    |    |    |    |    |    |    |    |    |    |    |--- class: INTIMATE PARTNER - AGGRAVATED ASSAULT
|    |    |    |    |    |    |    |    |    |    |--- CRM_CD >  243.00
|    |    |    |    |    |    |    |    |    |    |    |--- class: SHOTS FIRED AT MOVING VEHICLE, TRAIN OR          AIRCRAFT
|    |    |    |    |    |    |    |    |--- PART1_2_2 >  0.50
|    |    |    |    |    |    |    |    |    |--- class: CHILD NEGLECT (SEE 300 W.I.C.)
|    |    |    |    |    |    |    |--- CRM_CD >  250.50
|    |    |    |    |    |    |    |    |--- class: SHOTS FIRED AT INHABITED DWELLING
|    |    |    |    |    |    |--- 2000 >  0.50
|    |    |    |    |    |    |    |--- class: INTIMATE PARTNER - AGGRAVATED ASSAULT
|    |    |    |--- CRM_CD >  280.50
|    |    |    |    |--- CRM_CD <= 315.00
|    |    |    |    |    |--- class: BURGLARY
|    |    |    |--- CRM_CD >  315.00
|    |    |    |    |--- class: BURGLARY, ATTEMPTED
|    |--- CRM_CD >  325.00
|    |    |--- class: BURGLARY FROM VEHICLE
|--- CRM_CD >  330.50
|    |--- CRM_CD <= 336.00
|    |    |--- class: THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)
|    |--- CRM_CD >  336.00
|    |    |--- CRM_CD <= 382.00
|    |    |    |--- CRM_CD <= 353.00
|    |    |    |    |--- CRM_CD <= 342.00
|    |    |    |    |    |--- class: THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVESTK,PROD
|    |    |    |    |--- CRM_CD >  342.00
|    |    |    |    |    |--- CRM_CD <= 351.50
|    |    |    |    |    |    |--- CRM_CD <= 344.00
|    |    |    |    |    |    |    |--- class: SHOPLIFTING-GRAND THEFT ($950.01 & OVER)
```

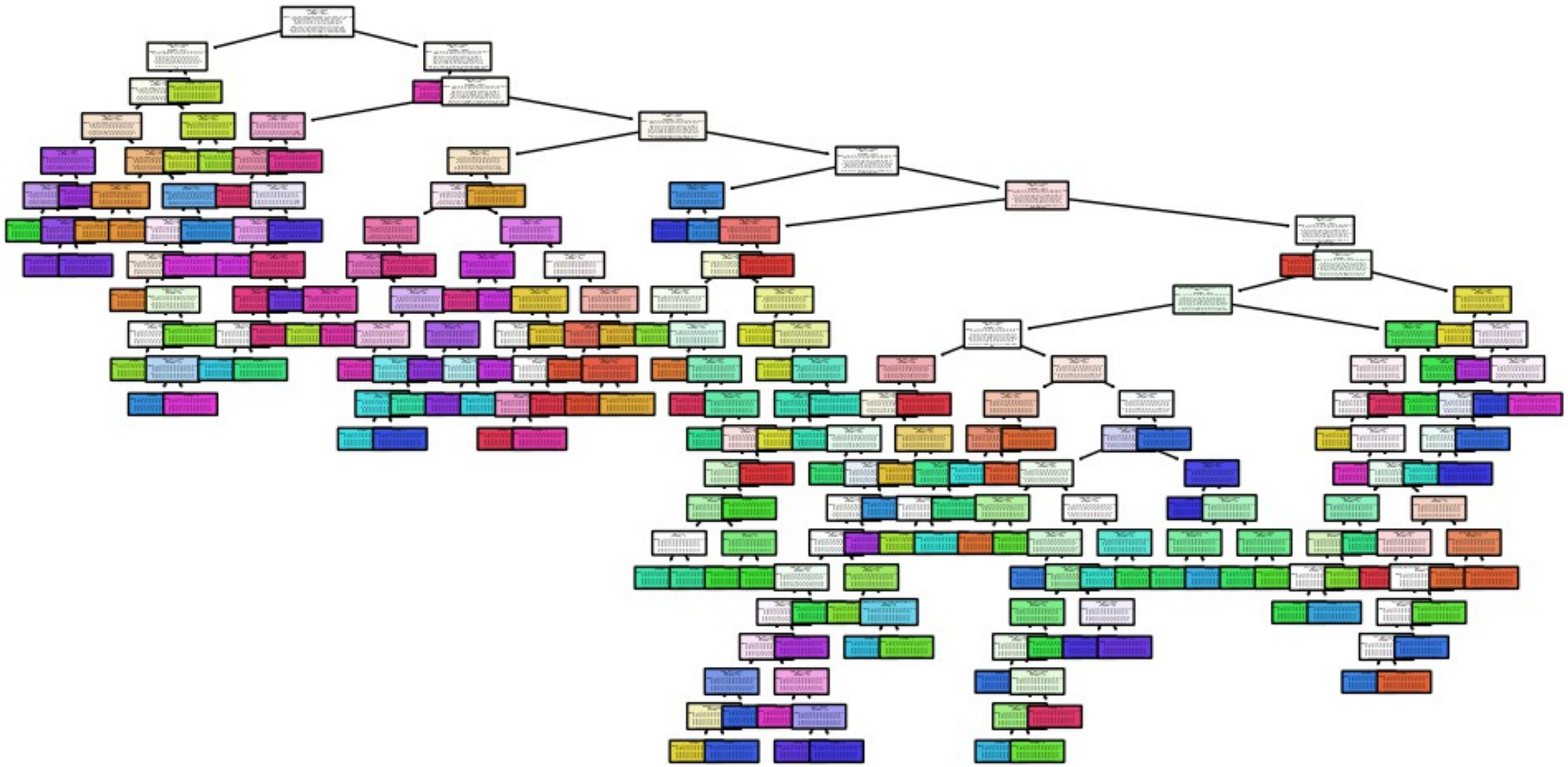
```
|   |   |   |   |   |   |--- CRM_CD >  344.00
```

```
|   |   |   |   |   |   |   |   |--- CRM_CD <= 350.50
|   |   |   |   |   |   |   |   |   |--- CRM_CD <= 348.50
|   |   |   |   |   |   |   |   |   |   |--- HOUR_17 <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: GRAND THEFT / INSURANCE FRAUD
|   |   |   |   |   |   |   |   |   |   |   |--- HOUR_17 >  0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: DISHONEST EMPLOYEE - GRAND THEFT
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  348.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: THEFT, PERSON
|   |   |   |   |   |   |   |   |   |--- CRM_CD >  350.50
|   |   |   |   |   |   |   |   |   |   |--- class: PURSE SNATCHING
|   |   |   |   |   |   |   |   |--- CRM_CD >  351.50
|   |   |   |   |   |   |   |   |   |--- class: PICKPOCKET
|   |   |   |   |   |   |   |--- CRM_CD >  353.00
|   |   |   |   |   |   |   |   |--- class: THEFT OF IDENTITY
|   |   |   |   |--- CRM_CD >  382.00
|   |   |   |   |--- CRM_CD <= 624.50
|   |   |   |   |   |--- CRM_CD <= 623.50
|   |   |   |   |   |   |--- CRM_CD <= 440.50
|   |   |   |   |   |   |   |--- CRM_CD <= 439.50
|   |   |   |   |   |   |   |   |--- CRM_CD <= 420.50
|   |   |   |   |   |   |   |   |   |--- CRM_CD <= 415.00
|   |   |   |   |   |   |   |   |   |   |--- class: BURGLARY FROM VEHICLE, ATTEMPTED
|   |   |   |   |   |   |   |   |   |--- CRM_CD >  415.00
|   |   |   |   |   |   |   |   |   |   |--- class: THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)
|   |   |   |   |   |   |   |   |--- CRM_CD >  420.50
|   |   |   |   |   |   |   |   |   |--- CRM_CD <= 436.00
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 427.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: THEFT FROM MOTOR VEHICLE - ATTEMPT
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  427.00
|   |   |   |   |   |   |   |   |   |   |   |--- PART1_2_1 <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |   |--- PART1_2_1 >  0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: DRIVING WITHOUT OWNER CONSENT (DWOC)
|   |   |   |   |   |   |   |   |   |--- CRM_CD >  436.00
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 437.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: RESISTING ARREST
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  437.50
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 438.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: RECKLESS DRIVING
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  438.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: FALSE POLICE REPORT
|   |   |   |   |   |   |   |   |--- CRM_CD >  439.50
|   |   |   |   |   |   |   |   |   |--- class: THEFT PLAIN - PETTY ($950 & UNDER)
|   |   |   |   |   |   |   |--- CRM_CD >  440.50
|   |   |   |   |   |   |   |   |--- CRM_CD <= 442.50
|   |   |   |   |   |   |   |   |   |--- CRM_CD <= 441.50
|   |   |   |   |   |   |   |   |   |   |--- class: THEFT PLAIN - ATTEMPT
|   |   |   |   |   |   |   |   |   |--- CRM_CD >  441.50
|   |   |   |   |   |   |   |   |   |   |--- class: SHOPLIFTING - PETTY THEFT ($950 &                UNDER)
|   |   |   |   |   |   |   |   |--- CRM_CD >  442.50
|   |   |   |   |   |   |   |   |   |--- CRM_CD <= 495.00
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 475.50
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 446.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: SHOPLIFTING - ATTEMPT
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  446.50
|   |   |   |   |   |   |   |   |   |   |   |--- VICT_SEX_M <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |   |--- VICT_SEX_M >  0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: TILL TAP - PETTY ($950 & UNDER)
|   |   |   |   |   |   |   |   |   |--- CRM_CD >  475.50
|   |   |   |   |   |   |   |   |   |   |--- class: BIKE - STOLEN
|   |   |   |   |   |   |   |   |--- CRM_CD >  495.00
|   |   |   |   |   |   |   |   |   |--- CRM_CD <= 622.50
```

```
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 515.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: VEHICLE - STOLEN
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  515.00
|   |   |   |   |   |   |   |   |   |   |   |--- PART1_2_2 <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: VEHICLE - ATTEMPT STOLEN
|   |   |   |   |   |   |   |   |   |   |   |--- PART1_2_2 >  0.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: BATTERY ON A FIREFIGHTER
|   |   |   |   |   |   |   |   |   |--- CRM_CD >  622.50
|   |   |   |   |   |   |   |   |   |   |--- class: BATTERY POLICE (SIMPLE)
|   |   |   |   |   |--- CRM_CD >  623.50
|   |   |   |   |   |   |--- class: BATTERY - SIMPLE ASSAULT
|   |   |   |   |--- CRM_CD >  624.50
|   |   |   |   |   |--- CRM_CD <= 626.50
|   |   |   |   |   |   |--- CRM_CD <= 625.50
|   |   |   |   |   |   |   |--- class: OTHER ASSAULT
|   |   |   |   |   |   |--- CRM_CD >  625.50
|   |   |   |   |   |   |   |--- class: INTIMATE PARTNER - SIMPLE ASSAULT
|   |   |   |   |   |--- CRM_CD >  626.50
|   |   |   |   |   |   |--- CRM_CD <= 742.50
|   |   |   |   |   |   |   |--- CRM_CD <= 705.00
|   |   |   |   |   |   |   |   |--- CRM_CD <= 661.50
|   |   |   |   |   |   |   |   |   |--- CRM_CD <= 637.00
|   |   |   |   |   |   |   |   |   |   |--- class: CHILD ABUSE (PHYSICAL) - SIMPLE ASSAULT
|   |   |   |   |   |   |   |   |   |--- CRM_CD >  637.00
|   |   |   |   |   |   |   |   |   |   |--- PART1_2_2 <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: ARSON
|   |   |   |   |   |   |   |   |   |   |--- PART1_2_2 >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 648.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: THROWING OBJECT AT          MOVING VEHICLE
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  648.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 6
|   |   |   |   |   |   |   |   |   |--- CRM_CD >  661.50
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 663.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: BUNCO, GRAND THEFT
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  663.00
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 665.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: BUNCO, PETTY THEFT
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  665.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 669.00
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  669.00
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- class: EMBEZZLEMENT, PETTY THEFT ($950 &          UNDER)
|   |   |   |   |   |   |   |   |--- CRM_CD >  705.00
|   |   |   |   |   |   |   |   |   |--- class: VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)
|   |   |   |   |   |   |   |--- CRM_CD >  742.50
|   |   |   |   |   |   |   |   |--- CRM_CD <= 749.00
|   |   |   |   |   |   |   |   |   |--- class: VANDALISM - MISDEAMEANOR ($399 OR UNDER)
|   |   |   |   |   |   |   |   |--- CRM_CD >  749.00
|   |   |   |   |   |   |   |   |   |--- PART1_2_1 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- WEAPON_DESC_VERBAL THREAT <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 889.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 887.00
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of              depth     11
|   |   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  887.00
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- class: TRESPASSING
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  889.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 901.50
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch          of    depth     3
|   |   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  901.50
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch          of    depth     11
|   |   |   |   |   |   |   |   |   |   |--- WEAPON_DESC_VERBAL THREAT >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 929.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 924.00
```

```
|   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 9
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  924.00

|   |   |   |   |   |   |   |   |   |   |   |   |--- class: THREATENING PHONE CALLS/LETTERS
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  929.00
|   |   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 935.00
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- class: CRIMINAL THREATS - NO WEAPON DISPLAYED
|   |   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  935.00
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |--- PART1_2_1 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 788.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: BRANDISH WEAPON
|   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  788.00
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 817.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: SEXUAL PENETRATION W/FOREIGN OBJECT
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  817.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD <= 820.50
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- class: ORAL COPULATION
|   |   |   |   |   |   |   |   |   |   |   |--- CRM_CD >  820.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: SODOMY/SEXUAL CONTACT B/W PENIS OF ONE PERS TO ANUS OTH
```

VISUALIZATION OF DECISION TRESS



6.3 LINEAR & LOGISTIC REGRESSION

AIM: To establish the relationship between Area Name and Time of crime occurrence.

INPUT: The input file is a .csv file that has the following attributes:

- DR_NO: Division of Records Number: Official file number made up of a 2-digit year, area ID, and 5 digits.
- Date Rptd: Date at which the crime was reported. This value is in MM/DD/YYYY format.
- DATE OCC: Date at which the crime occurred. This value is in MM/DD/YYYY format.
- TIME OCC: It stands for the time at which the crime occurred. This value is in 24-hour military time.
- AREA: The LAPD has 21 Community Police Stations referred to as Geographic Areas within the department. These Geographic Areas are sequentially numbered from 1-21.
- AREA NAME: The 21 Geographic Areas or Patrol Divisions are also given a name designation that references a landmark or the surrounding community that it is responsible for. For example, the 77th Street Division is located at the intersection of South Broadway and 77th Street, serving neighborhoods in South Los Angeles.
- Rpt Dist No: A four-digit code that represents a sub-area within a Geographic Area. All crime records reference the "RD" that it occurred in for statistical comparisons. Find LAPD Reporting Districts on the LA City GeoHub.
  - Part 1-2
  - Crm Cd: It indicates the crime committed. It is also referred as Crime Code 1
  - Crm Cd Desc: It stands for the description of the crime code (Crm Cd).
- Mocodes: It stands for the Modus Operandi Codes. It refers to Activities associated with the suspect in commission of the crime.See attached PDF for list of MO Codes in numerical order.
  - Vict Age: It stands for age of the victim.
- Vict Descent: It stands for the race of the victim. It is referred to by Descent Code. Descent Codes are as follows:
  - Vict Sex: It stands for the sex of the victim.
  - Premis Cd: It stands for Premise code.
  - Premis Desc: It defines the Premise description.

- Weapon Used Cd: It refers to the type of weapon used.

- Weapon Desc: It describes the weapon code defined.
- Status: It refers to the status of the case. It has values IC, AA, AO, CC, JA, JO.
- Status Desc: It describes the status mentioned in the Status column. IC stands for Investigation Cont, AA stands for Adult Arrest, Adult Other, Juv(enile) Arrest, Juv(enile) Other, and UNK.
- Crm cd 1: It indicates the crime committed. Crime Code 1 is the primary and most serious one. Crime Codes 2, 3, and 4 are respectively less serious offenses. Lower crime class numbers are more serious.
- Crm cd 2: May contain a code for an additional crime, less serious than Crime Code 1.
- Crm cd 3: May contain a code for an additional crime, less serious than Crime Code 1.
- Crm cd 4: May contain a code for an additional crime, less serious than Crime Code 1.
- LOCATION: The street address of the crime incident was rounded to the nearest hundred blocks to maintain anonymity.
- Cross Street: It stands for Cross Street of rounded Address.
- LAT: Latitude
- LONG: Longitude

RSTUDIO

Loading Necessary Libraries

```
library("ggplot2")
library("dplyr")
```

Pre-Processing the Data

```
df <- read.csv("Crime_Data_from_2020_to_Present.csv")
df <- df %>%
  select(-c(1, 7, 16, 17, 18, 19, 24, 25))

#Predicitng crime type based on other details

library(lubridate)

# Conve# Convert DATE_OCC column to a datetime object
df$DATE.OCC <- strptime(df$DATE.OCC, format = "%d/%m/%Y")

# Filter rows based on date conditions
df$DATE.OCC <- as.Date(df$DATE.OCC)

# Filter rows based on date conditions
df2 <- df %>%
  filter(format(DATE.OCC, "%Y-%m") == '2023-01' | format(DATE.OCC, "%Y-%m") == '2023-08')

# Convert 'DATE OCC' column to Date format
df$DATE.OCC <- as.Date(df$DATE.OCC, format = "%m/%d/%Y")

# Create 'Day', 'Month', and 'Year' columns
df$Day <- format(df$DATE.OCC, "%A")
df$Month <- format(df$DATE.OCC, "%B")
df$Year <- format(df$DATE.OCC, "%Y")
```

```
# Define a function to convert 'TIME OCC' to time format
convert_to_time <- function(value) {
  hours <- as.integer(value %/% 100)
  minutes <- as.integer(value %% 100)
  return(paste0(formatC(hours, width = 2, format = "d", flag = "0"), ":",
                formatC(minutes, width = 2, format = "d", flag = "0")))
}

# Apply the 'convert_to_time' function to 'TIME OCC' column
df$TIME.OCC <- lapply(df$TIME.OCC, convert_to_time)



# Convert 'TIME OCC' to character and pad with leading zeros
df$TIME.OCC <- format(df$TIME.OCC, format = "%H%M")

# Drop columns by specifying the column indices to keep
df <- df[, -c(1, 2)]

# Filter for rows where 'Year' is greater than 2022
df <- df[df$Year > 2022, ]

# Fill missing values using values1
clean_df <- df2
clean_df[is.na(clean_df$`Crm Cd 2`), 'Crm Cd 2'] <- values1$`Crm Cd 2`
clean_df[is.na(clean_df$`Crm Cd 3`), 'Crm Cd 3'] <- values1$`Crm Cd 3`
clean_df[is.na(clean_df$`Crm Cd 4`), 'Crm Cd 4'] <- values1$`Crm Cd 4`
```

PYTHON

Loading Necessary Libraries

```
import pandas as pd
import numpy as np
import matplotlib as plt
```

Pre-Processing the Data

```
df = pd.read_csv('Crime_Data_from_2020_to_Present.csv')

df.head(5)
```

```
df.drop(df.columns[[0, 7, 16, 17, 18, 19, 24, 25]], axis = 1, inplace = True)
```

```python
df['DATE OCC'] = df['DATE OCC'].str.extract(r'(\d{2}/\d{2}/\d{4})')
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'], format='%m/%d/%Y')
df['Day'] = df['DATE OCC'].dt.day_name()
df['Month'] = df['DATE OCC'].dt.month_name()
df['Year'] = df['DATE OCC'].dt.year

from datetime import time


def convert_to_time(value):
    hours = value // 100
    minutes = value % 100
    return time(hours, minutes)


df['TIME OCC'] = df['TIME OCC'].apply(convert_to_time)

df['TIME OCC'] = df['TIME OCC'].astype(str).str.zfill(4)

df.drop(df.columns[[0, 1]], axis = 1, inplace = True)
df = df[['Day',
       'Month', 'Year', 'TIME OCC', 'AREA', 'AREA NAME', 'Rpt Dist No', 'Crm Cd', 'Crm Cd Desc',
       'Mocodes', 'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd',
       'Premis Desc', 'Crm Cd 1', 'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'LAT', 'LON']]
```

```python
df = df[df['Year'] > 2022]
```

```python
values1 = {'Crm Cd 2': 0, 'Crm Cd 3': 0, 'Crm Cd 4': 0}
clean_df = df.fillna(value = values1)
clean_df.head(10)

clean_df.dropna(subset = ['Crm Cd 1', 'Mocodes'], inplace = True)
clean_df

values2 = {'Vict Sex': 'X', 'Vict Descent': 'X'}
clean_df2 = clean_df.fillna(value = values2)
clean_df2.replace(['H'], 'X', inplace = True)
clean_df2

clean_df2.dropna(subset = ['Premis Cd', 'Premis Desc'], inplace = True)
clean_df2
```

```r
# Drop rows with missing values in specified columns
clean_df <- clean_df[!is.na(clean_df$Crm.Cd.1), ]
clean_df <- clean_df[!is.na(clean_df$Mocodes), ]

# Define values2 for filling missing values
values2 <- list('Vict Sex' = 'X', 'Vict Descent' = 'X')

# Fill missing values using values2
clean_df2 <- clean_df
clean_df2$Vict.Sex[is.na(clean_df2$Vict.Sex)] <- values2$Vict.Sex
clean_df2$Vict.Descent[is.na(clean_df2$Vict.Descent)] <- values2$Vict.Descent

# Replace 'H' with 'X' in 'Vict Descent' column
clean_df2$Vict.Descent[clean_df2$Vict.Descent == 'H'] <- 'X'

area_name_counts <- table(clean_df2$AREA.NAME)
print(area_name_counts)

set.seed(1)  # Set the random seed for reproducibility
clean_df2_shuffled <- clean_df2[sample(nrow(clean_df2)), ]
clean_df2_shuffled

dummy_vars <- model.matrix(~ clean_df2_shuffled$AREA.NAME - 1)  # Remove intercept term (-1)
dummy_df <- as.data.frame(dummy_vars)

# Display the first few rows of the dummy variables
head(dummy_df)

clean_df2_shuffled <- clean_df2_shuffled[, !(names(clean_df2_shuffled) == "AREA.NAME")]
```

```r
clean_df2_final <- cbind(clean_df2_shuffled[, !(names(clean_df2_shuffled) == "AREA.NAME")], dummy_df)

# Define the column indices you want to drop
columns_to_drop <- c(1, 2, 3, 4, 8, 10, 11, 13, 18, 19)
# Drop the specified columns
clean_df2_final <- clean_df2_final[, -columns_to_drop]

# Drop the column at index 4
clean_df2_final <- clean_df2_final[, -4]

# Remove rows with missing values (NA)
clean_df2_final <- na.omit(clean_df2_final)

# Get the length of the resulting data frame
n_rows <- nrow(clean_df2_final)
n_rows

# Split the data frame
train_pd <- clean_df2_final[1:10, ]
test_pd <- clean_df2_final[11:17, ]
val_pd <- clean_df2_final[18:nrow(clean_df2_final), ]

# Get the lengths of the resulting sets
n_train <- nrow(train_pd)
n_test <- nrow(test_pd)
n_val <- nrow(val_pd)
```

```
clean_df2_shuffled = clean_df2.sample(n=len(clean_df2), random_state=1)
clean_df2_shuffled
```

```
pd.get_dummies(clean_df2_shuffled['AREA NAME']).head()
```

```
clean_df2_final = pd.concat([clean_df2_shuffled.drop('AREA NAME', axis = 1), pd.get_dummies(clean_df2_shuffled['AREA NAME'])], axis = 1)
clean_df2_final
```

```
clean_df2_final = clean_df2_final[['Day', 'Month',  'Year', 'TIME OCC', 'AREA', 'Rpt Dist No',  'Crm Cd', 'Crm Cd Desc',
                                   'Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis Desc', 'Crm Cd 1', 'Crm Cd 2', 'Crm
                                   'LAT', 'LON', 'Southwest', 'Central', 'N Hollywood', 'Mission', 'Devonshire', 'Northe

clean_df2_final
```

```
clean_df2_final.drop(clean_df2_final.columns[[0, 1, 2, 3, 7, 9, 10, 12, 17, 18]], axis = 1, inplace = True)
```

```
clean_df2_final.drop(clean_df2_final.columns[[3]], axis = 1, inplace = True)
```

```
<ipython-input-16-75cb8fc8cdb9>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  clean_df2_final.drop(clean_df2_final.columns[[3]], axis = 1, inplace = True)
```

```
clean_df2_final = clean_df2_final.dropna()
len(clean_df2_final)
```

```
125596
```

```
train_pd, test_pd, val_pd = clean_df2_final[:100000], clean_df2_final[100000:112798], clean_df2_final[112798:]
len(train_pd), len(test_pd), len(val_pd)
```

```
(100000, 12798, 12798)
```

```
X_train, y_train = train_pd.to_numpy()[:, :-1], train_pd.to_numpy()[:, -1]
X_val, y_val = val_pd.to_numpy()[:, :-1], val_pd.to_numpy()[:, -1]
X_test, y_test = test_pd.to_numpy()[:, :-1], test_pd.to_numpy()[:, -1]

X_train.shape, y_train.shape, X_val.shape, y_val.shape, X_test.shape, y_test.shape
```

```
((100000, 29), (100000,), (12798, 29), (12798,), (12798, 29), (12798,))
```

```
column_index = clean_df2_final.columns.get_loc('Southwest')
print(column_index)
```

```
8
```

```
from sklearn.preprocessing import StandardScaler
import numpy as np

scaler = StandardScaler().fit(X_train[:, :8])

def preprocessor(X):
  A = np.copy(X)
  A[:, :8] = scaler.transform(A[:, :8])
  return A

X_train, X_val, X_test = preprocessor(X_train), preprocessor(X_val), preprocessor(X_test)
```

```
X_train_preprocessed = preprocessor(X_train)
X_train_preprocessed
```

```
# Print the lengths
n_train
n_test
n_val

# Extract X_train and y_train
X_train <- train_pd[, -ncol(train_pd)]
y_train <- train_pd[, ncol(train_pd)]

# Extract X_val and y_val
X_val <- val_pd[, -ncol(val_pd)]
y_val <- val_pd[, ncol(val_pd)]

# Extract X_test and y_test
X_test <- test_pd[, -ncol(test_pd)]
y_test <- test_pd[, ncol(test_pd)]

# Get the dimensions (shapes) of the resulting matrices and vectors
dim(X_train)
length(y_train)
dim(X_val)
length(y_val)
dim(X_test)
length(y_test)

# Find the index of the 'Southwest' column by name
column_name <- 'Southwest'
column_index <- which(names(clean_df2_final) == column_name)
```

```
# Print the index
print(column_index)


# Fit the scaler on X_train
scaler <- scale(X_train[, ])

# Define a preprocessor function
preprocessor <- function(X, scaler) {
  A <- scale(X[, 1:8], center = scaler$center, scale = scaler$scale)
  X[, 1:8] <- A
  return(X)
}

# Apply the preprocessor function to X_train, X_val, and X_test
X_train <- preprocessor(X_train, scaler)
X_val <- preprocessor(X_val, scaler)
X_test <- preprocessor(X_test, scaler)
```

```
pd.DataFrame(X_train_preprocessed).hist()
```

```
array([[<Axes: title={'center': '0'}>, <Axes: title={'center': '1'}>,
        <Axes: title={'center': '2'}>, <Axes: title={'center': '3'}>,
        <Axes: title={'center': '4'}>],
       [<Axes: title={'center': '5'}>, <Axes: title={'center': '6'}>,
        <Axes: title={'center': '7'}>, <Axes: title={'center': '8'}>,
        <Axes: title={'center': '9'}>],
       [<Axes: title={'center': '10'}>, <Axes: title={'center': '11'}>,
        <Axes: title={'center': '12'}>, <Axes: title={'center': '13'}>,
        <Axes: title={'center': '14'}>],
       [<Axes: title={'center': '15'}>, <Axes: title={'center': '16'}>,
        <Axes: title={'center': '17'}>, <Axes: title={'center': '18'}>,
        <Axes: title={'center': '19'}>],
       [<Axes: title={'center': '20'}>, <Axes: title={'center': '21'}>,
        <Axes: title={'center': '22'}>, <Axes: title={'center': '23'}>,
        <Axes: title={'center': '24'}>],
       [<Axes: title={'center': '25'}>, <Axes: title={'center': '26'}>,
        <Axes: title={'center': '27'}>, <Axes: title={'center': '28'}>,
        <Axes: >]], dtype=object)
```

RMSE

```
[ ]  from sklearn.metrics import mean_squared_error as mse
     from sklearn.linear_model import LinearRegression

     lm = LinearRegression().fit(X_train, y_train)
     mse(lm.predict(X_train), y_train, squared=False), mse(lm.predict(X_val), y_val, squared=False)
```

```
(20.224810406982435, 20.251087000108253)
```

Trying Linear Regression

```
[ ]  from sklearn.metrics import mean_squared_error as mse
     from sklearn.linear_model import LinearRegression

     # Assuming you have X_train, y_train, X_val, and y_val

     # Create and fit a linear regression model
     lm = LinearRegression()
     lm.fit(X_train, y_train)

     # Predict on the training and validation sets
     y_train_pred = lm.predict(X_train)
     y_val_pred = lm.predict(X_val)

     # Calculate the RMSE for the training and validation sets
     rmse_train = mse(y_train_pred, y_train, squared=False)
     rmse_val = mse(y_val_pred, y_val, squared=False)

     print("RMSE on training set:", rmse_train)
     print("RMSE on validation set:", rmse_val)
```

Metric Evaluation

Environment | History | Connections | Tutorial

Import Dataset | 1.42 GiB | List

R | Global Environment

Data	
clean_df	11286 obs. of 20 variables
clean_df2	11286 obs. of 20 variables
clean_df2_final	19 obs. of 29 variables
clean_df2_shuffled	11286 obs. of 19 variables
df	525598 obs. of 21 variables
df2	11287 obs. of 20 variables
dummy_df	11286 obs. of 21 variables
dummy_vars	Large matrix (237006 elements, 2.6 MB)
test_pd	7 obs. of 29 variables
train_pd	10 obs. of 29 variables
val_pd	2 obs. of 29 variables
values2	List of 2
X_test	7 obs. of 28 variables
X_train	10 obs. of 28 variables
X_val	2 obs. of 28 variables
Values	
area_name_counts	'table' int [1:21(1d)] 748 817 501 371 462 418 509 509 544 62…
column_index	integer (empty)
column_name	"Southwest"
columns_to_drop	num [1:10] 1 2 3 4 8 10 11 13 18 19
n_rows	19L
n_test	7L
n_train	10L

Files | Plots | Packages | Help | Viewer | Presentation

New Folder | New Blank File | Delete | Rename | More

Home > DBSCAN

Name	Size	Modified
..		
Crime_Data_from_2020_to_Present.csv	192.2 MB	Oct 8, 2023, 12:59 PM
DBSCAN.R	3.7 KB	Oct 8, 2023, 1:33 PM
DBSCAN.Rproj	205 B	Oct 8, 2023, 11:01 PM
df_final3.csv	931.2 MB	Oct 8, 2023, 8:16 PM
DT.R	7.1 KB	Oct 8, 2023, 7:03 PM
DT.RData	4.2 MB	Oct 8, 2023, 10:00 PM
DT2ndAttempt.R	2.4 KB	Oct 8, 2023, 11:35 PM
DTree.RData	30 MB	Oct 8, 2023, 5:56 PM
FirstDBSCANRplot.png	42 KB	Oct 8, 2023, 1:17 PM
X_test.csv	186.8 MB	Oct 8, 2023, 9:09 PM
X_train.csv	745.1 MB	Oct 8, 2023, 9:11 PM
y_test.csv	181.9 KB	Oct 8, 2023, 9:08 PM
y_train.csv	722.5 KB	Oct 8, 2023, 9:08 PM

Metric Evaluation

```
[ ] from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, explained_variance_score
    from sklearn.linear_model import LinearRegression

    # Assuming you have X_train, y_train, X_val, y_val, X_test, and y_test

    # Create and fit a linear regression model
    lm = LinearRegression()
    lm.fit(X_train, y_train)

    # Predict on the training, validation, and test sets
    y_train_pred = lm.predict(X_train)
    y_val_pred = lm.predict(X_val)
    y_test_pred = lm.predict(X_test)

    # Calculate evaluation metrics
    rmse_train = mean_squared_error(y_train, y_train_pred, squared=False)
    rmse_val = mean_squared_error(y_val, y_val_pred, squared=False)
    rmse_test = mean_squared_error(y_test, y_test_pred, squared=False)

    mae_train = mean_absolute_error(y_train, y_train_pred)
    mae_val = mean_absolute_error(y_val, y_val_pred)
```

```
r2_test = r2_score(y_test, y_test_pred)

explained_variance_train = explained_variance_score(y_train, y_train_pred)
explained_variance_val = explained_variance_score(y_val, y_val_pred)
explained_variance_test = explained_variance_score(y_test, y_test_pred)

# Print the evaluation metrics
print("RMSE on training set:", rmse_train)
print("RMSE on validation set:", rmse_val)
print("RMSE on test set:", rmse_test)

print("MAE on training set:", mae_train)
print("MAE on validation set:", mae_val)
print("MAE on test set:", mae_test)

print("R-squared on training set:", r2_train)
print("R-squared on validation set:", r2_val)
print("R-squared on test set:", r2_test)

print("Explained Variance on training set:", explained_variance_train)
print("Explained Variance on validation set:", explained_variance_val)
print("Explained Variance on test set:", explained_variance_test)
```

```
MSE : 1432.222, RMSE : 1432.222
Variance Score : -2.461
Accuracy Score : 0.160
ROC_AUC Score : 0.585
```

Metric Evaluation

6.4 DBSCAN

AIM: The aim is to determine areas with higher crime rates and with lower crime rates. This execution requires clustering the areas based on the latitude and longitude of the area of crime.

INPUT: The input file is a .csv file that has the following attributes:

- DR_NO: Division of Records Number: Official file number made up of a 2-digit year, area ID, and 5 digits.
- Date Rptd: Date at which the crime was reported. This value is in MM/DD/YYYY format.
- DATE OCC: Date at which the crime occurred. This value is in MM/DD/YYYY format.
- TIME OCC: It stands for the time at which the crime occurred. This value is in 24-hour military time.
- AREA: The LAPD has 21 Community Police Stations referred to as Geographic Areas within the department. These Geographic Areas are sequentially numbered from 1-21.
- AREA NAME: The 21 Geographic Areas or Patrol Divisions are also given a name designation that references a landmark or the surrounding community that it is responsible for. For example, the 77th Street Division is located at the intersection of South Broadway and 77th Street, serving neighborhoods in South Los Angeles.
- Rpt Dist No: A four-digit code that represents a sub-area within a Geographic Area. All crime records reference the "RD" that it occurred in for statistical comparisons. Find LAPD Reporting Districts on the LA City GeoHub.
  - Part 1-2
  - Crm Cd: It indicates the crime committed. It is also referred as Crime Code 1
  - Crm Cd Desc: It stands for the description of the crime code (Crm Cd).
- Mocodes: It stands for the Modus Operandi Codes. It refers to Activities associated with the suspect in commission of the crime.See attached PDF for list of MO Codes in numerical order.
  - Vict Age: It stands for age of the victim.
- Vict Descent: It stands for the race of the victim. It is referred to by Descent Code. Descent Codes are as follows:
  - Vict Sex: It stands for the sex of the victim.
  - Premis Cd: It stands for Premise code.

- Premis Desc: It defines the Premise description.
- Weapon Used Cd: It refers to the type of weapon used.
- Weapon Desc: It describes the weapon code defined.

- Status: It refers to the status of the case. It has values IC, AA, AO, CC, JA, JO.
- Status Desc: It describes the status mentioned in the Status column. IC stands for Investigation Cont, AA stands for Adult Arrest, Adult Other, Juv(enile) Arrest, Juv(enile) Other, and UNK.
- Crm cd 1: It indicates the crime committed. Crime Code 1 is the primary and most serious one. Crime Codes 2, 3, and 4 are respectively less serious offenses. Lower crime class numbers are more serious.
- Crm cd 2: May contain a code for an additional crime, less serious than Crime Code 1.
- Crm cd 3: May contain a code for an additional crime, less serious than Crime Code 1.
- Crm cd 4: May contain a code for an additional crime, less serious than Crime Code 1.
- LOCATION: The street address of the crime incident was rounded to the nearest hundred blocks to maintain anonymity.
- Cross Street: It stands for Cross Street of rounded Address.
- LAT: Latitude
- LONG: Longitude

From the above attributes, we will be working on the attributes DATE OCC, LAT, and LONG.

RSTUDIO

Loading Necessary Libraries

```r
library("ggplot2")
library("dplyr")
```

Pre-Processing Data

```r
# Convert 'DATE OCC' column to Date format
df$DATE.OCC <- as.Date(df$DATE.OCC, format = "%m/%d/%Y")

# Create 'Day', 'Month', and 'Year' columns
df$Day <- format(df$DATE.OCC, "%A")
df$Month <- format(df$DATE.OCC, "%B")
df$Year <- format(df$DATE.OCC, "%Y")


# Define a function to convert 'TIME OCC' to time format
convert_to_time <- function(value) {
  hours <- as.integer(value %/% 100)
  minutes <- as.integer(value %% 100)
  return(paste0(formatC(hours, width = 2, format = "d", flag = "0"), ":",
                formatC(minutes, width = 2, format = "d", flag = "0")))
}
# Apply the 'convert_to_time' function to 'TIME OCC' column
df$TIME.OCC <- lapply(df$TIME.OCC, convert_to_time)



# Convert 'TIME OCC' to character and pad with leading zeros
df$TIME.OCC <- format(df$TIME.OCC, format = "%H%M")

# Drop columns by specifying the column indices to keep
df <- df[, -c(1, 2)]



# Filter rows where 'Year' is greater than 2022
df <- df[df$Year > 2022, ]

# Filter rows where 'Month' is equal to 'February'
df <- df[df$Month == 'February', ]

# Define values1 for filling missing values
values1 <- list('Crm Cd 2' = 0, 'Crm Cd 3' = 0, 'Crm Cd 4' = 0)

# Fill missing values using values1
clean_df <- df
clean_df[is.na(clean_df$`Crm Cd 2`), 'Crm Cd 2'] <- values1$`Crm Cd 2`
clean_df[is.na(clean_df$`Crm Cd 3`), 'Crm Cd 3'] <- values1$`Crm Cd 3`
clean_df[is.na(clean_df$`Crm Cd 4`), 'Crm Cd 4'] <- values1$`Crm Cd 4`

# Drop rows with missing values in specified columns
clean_df <- clean_df[!is.na(clean_df$Crm.Cd.1), ]
clean_df <- clean_df[!is.na(clean_df$Mocodes), ]

# Define values2 for filling missing values
values2 <- list('Vict Sex' = 'X', 'Vict Descent' = 'X')

# Fill missing values using values2
clean_df2 <- clean_df
clean_df2$Vict.Sex[is.na(clean_df2$Vict.Sex)] <- values2$Vict.Sex
clean_df2$Vict.Descent[is.na(clean_df2$Vict.Descent)] <- values2$Vict.Descent

# Replace 'H' with 'X' in 'Vict Descent' column
clean_df2$Vict.Descent[clean_df2$Vict.Descent == 'H'] <- 'X'

# Drop rows with missing values in specified columns

clean_df2 <- clean_df2[!is.na(clean_df2$Premis.Cd), ]
clean_df2 <- clean_df2[!is.na(clean_df2$Premis.Desc), ]
# Drop repeat columns

clean_df2 <- clean_df2 %>%
  select(-c(22, 23, 24))
```

PYTHON

Loading Necessary Libraries

```python
import pandas as pd
import numpy as np
import matplotlib as plt
```

Pre-Processing Data

```python
df['DATE OCC'] = df['DATE OCC'].str.extract(r'(\d{2}/\d{2}/\d{4})')
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'], format='%m/%d/%Y')
df['Day'] = df['DATE OCC'].dt.day_name()
df['Month'] = df['DATE OCC'].dt.month_name()
df['Year'] = df['DATE OCC'].dt.year

from datetime import time


def convert_to_time(value):
    hours = value // 100
    minutes = value % 100
    return time(hours, minutes)


df['TIME OCC'] = df['TIME OCC'].apply(convert_to_time)

df['TIME OCC'] = df['TIME OCC'].astype(str).str.zfill(4)

df.drop(df.columns[[0, 1]], axis = 1, inplace = True)
df = df[['Day',
       'Month', 'Year', 'TIME OCC', 'AREA', 'AREA NAME', 'Rpt Dist No',
       'Crm Cd', 'Crm Cd Desc',
       'Mocodes', 'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd',
       'Premis Desc', 'Crm Cd 1', 'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4',
       'LAT', 'LON']]

df = df[df['Year'] > 2022]
```

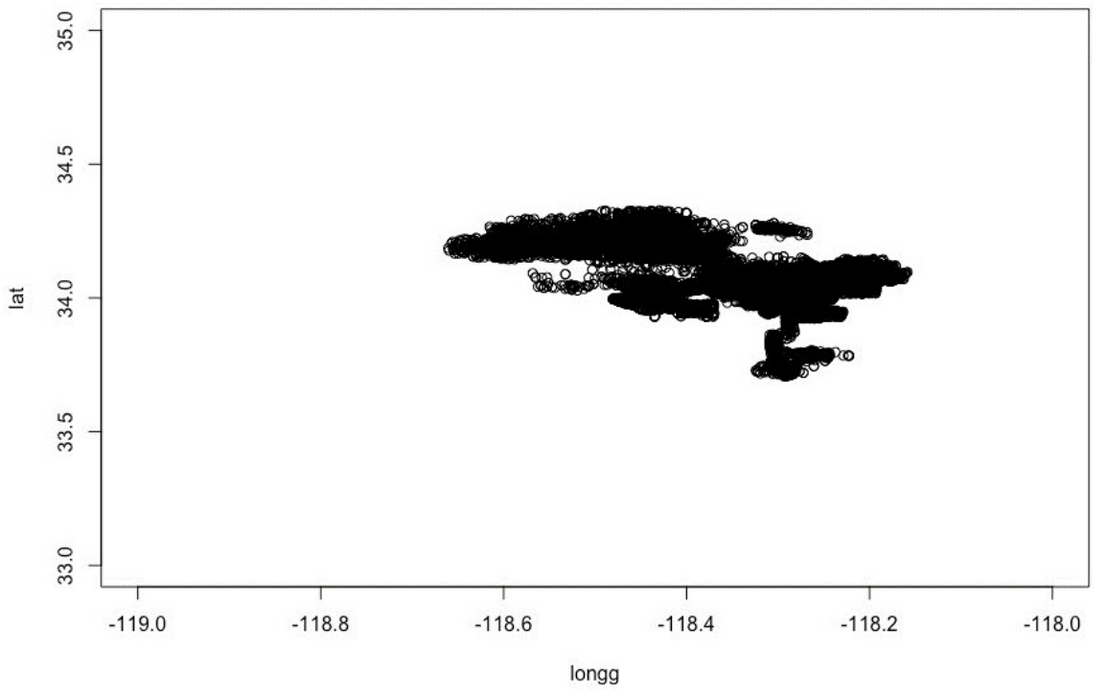
```python
df = df[df['Month'] == 'February']
```

```python
values1 = {'Crm Cd 2': 0, 'Crm Cd 3': 0, 'Crm Cd 4': 0}
clean_df = df.fillna(value = values1)
clean_df.head(10)

clean_df.dropna(subset = ['Crm Cd 1', 'Mocodes'], inplace = True)
clean_df

values2 = {'Vict Sex': 'X', 'Vict Descent': 'X'}
clean_df2 = clean_df.fillna(value = values2)
clean_df2.replace(['H'], 'X', inplace = True)
clean_df2

clean_df2.dropna(subset = ['Premis Cd', 'Premis Desc'], inplace = True)
clean_df2
```

Extracting the latitude and longitude for plotting the Scatter Plot

```r
# Extract the 'LAT' and 'LON' columns into a data frame 'lat_long'
lat_long <- clean_df2[c('LAT', 'LON')]

# Extract 'LAT' and 'LON' into separate vectors 'lat' and 'longg'
lat <- clean_df2$LAT
longg <- clean_df2$LON

# Set xlim and ylim for the plot
xlim <- c(-119, -118)
ylim <- c(33, 35)

# Create a scatter plot
plot(longg, lat, xlim = xlim, ylim = ylim)
```

Scatter Plot



Loading DBSACN Package

```r
# Load the dbscan package if not already loaded
library(dbscan)
```

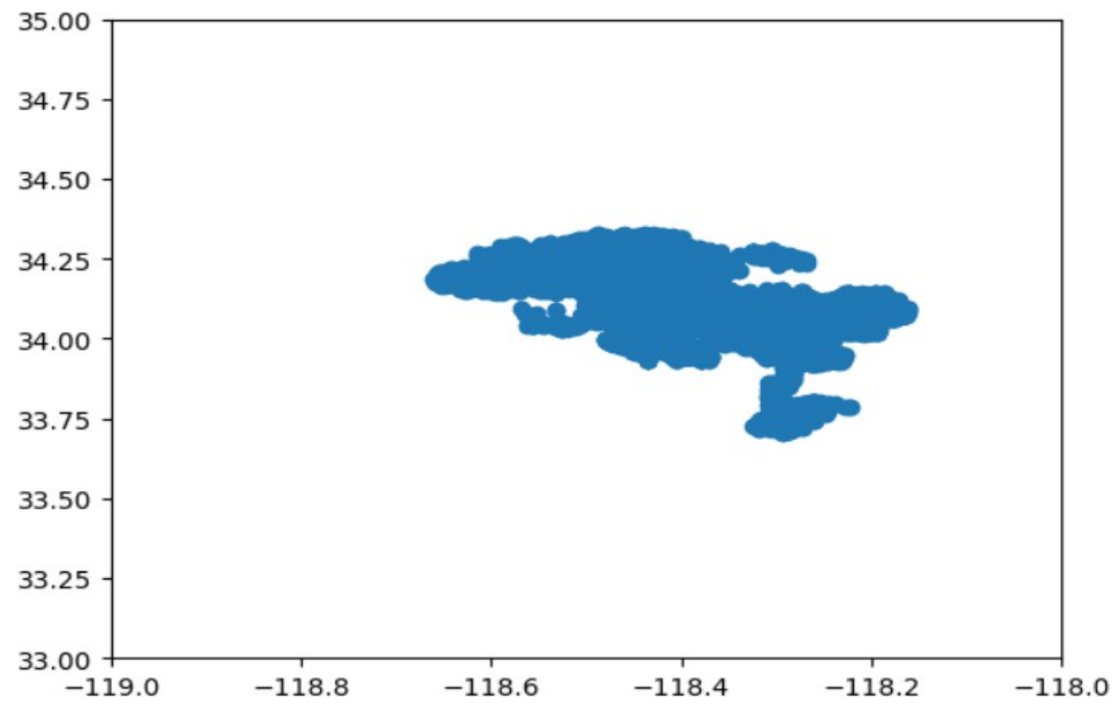
Executing DBSCAN Clustering

```r
# Convert 'lat_long' data frame to a matrix
X <- as.matrix(lat_long)

# Run DBSCAN clustering
eps <- 0.01
min_samples <- 15
# Run DBSCAN clustering
dbscan_result <- dbscan(X, eps = eps, minPts = min_samples)


# Get the dimensions of the resulting cluster assignments
cluster_assignments <- dbscan_result$cluster
n_clusters <- length(unique(cluster_assignments))

# Add cluster labels to the data frame
clean_df2$cluster <- cluster_assignments

# Calculate the counts of unique values in the 'cluster' column
cluster_counts <- table(clean_df2$cluster)
```

Extracting the latitude and longitude for plotting the Scatter Plot

```python
lat_long = clean_df2[['LAT', 'LON']]
lat, longg = clean_df2['LAT'], clean_df2['LON']

plt.xlim(-119, -118)
plt.ylim(33, 35)
plt.scatter(longg, lat)
```

Scatter Plot



Loading DBSCAN Package

```python
from sklearn.cluster import DBSCAN
```

Executing DBSCAN Clustering

```
X = lat_long.to_numpy()
X.shape
```

```
(15775, 2)
```

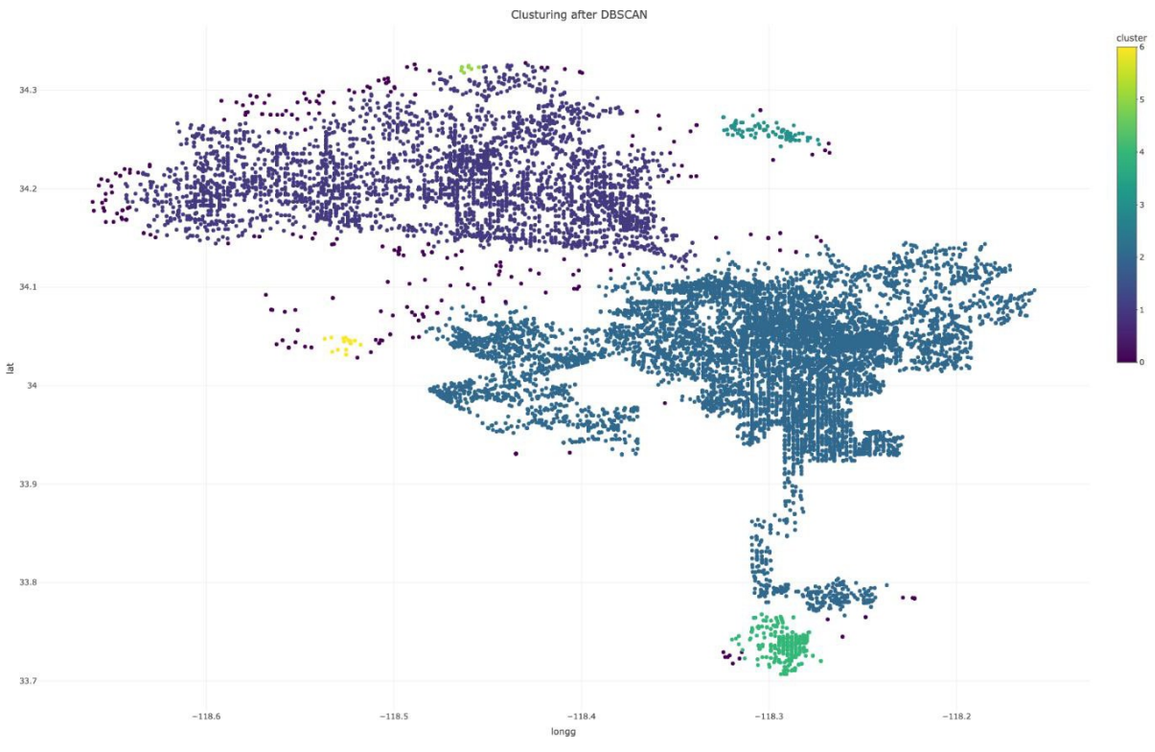
```
dbscan_cluster_model = DBSCAN(eps=0.01, min_samples=15).fit(X)
dbscan_cluster_model
```

```
DBSCAN(eps=0.01, min_samples=15)
```

```
clean_df2['cluster'] = dbscan_cluster_model.labels_

clean_df2
```

Scatter Plot with color-coded cluster

```
library(plotly)

# Create a scatter plot with color-coded clusters
plot <- plot_ly(data = clean_df2, x = ~longg, y = ~lat, color = ~cluster) %>%
  add_markers() %>%
  layout(title = "Clusturing after DBSCAN")

# Show the plot
plot
```



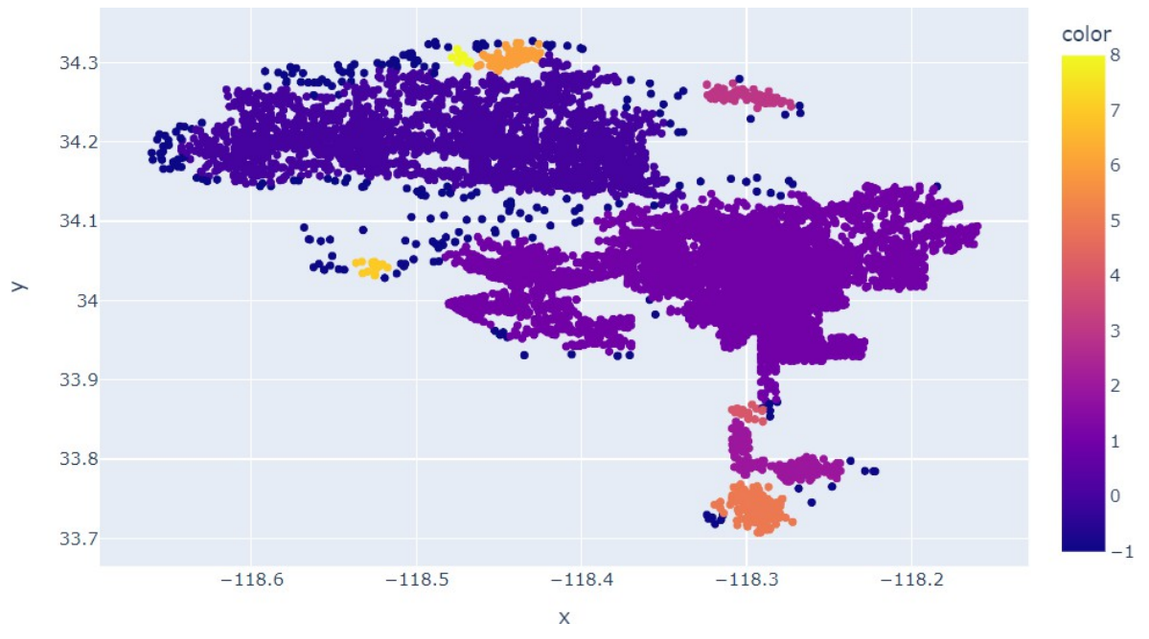
Silhouette Score

```
#Silhoutte score

library(fpc)

dist_matrix <- dist(X)

silhoutte_score <- cluster.stats(dist_matrix, cluster_assignments)$avg.silwidth

print(silhoutte_score)
```

```
> print(silhoutte_score)
[1] 0.294954
```

Scatter Plot with color-coded cluster

```
import plotly.express as px

fig = px.scatter(x=longg, y=lat, color=clean_df2['cluster'])

fig.show()
```



Silhouette Score

```
from sklearn.metrics import silhouette_score as ss

ss(X, clean_df2['cluster'])
```

```
0.2277931342361696
```

7. KEY RESULTS AND MATRICES OF EACH ALGORITHM

7.1 Performance Metrics

We have used various performance metrics for each algorithm.

K-Means:

For K-Means, we have used Silhouette Score and Cluster Size.

Performance Metrics	Python	R Studio
Silhouette Score	0.0091	0.0081
Cluster Size	All sizes are the same	All sizes are the same except for 1

Decision Tree

For the Decision Tree, we have used the Accuracy of the code.

While executing the code Python we observed the following:

- MSE: 1432.222
- RMSE: 1432.222
- Variance Score: -2.461
- Accuracy Score: 0.160
- ROC_AUC Score: 0.585
- Gini Accuracy: 0.9993

While executing the code in R Studio, we got an accuracy score of 84.6%.

In decision tree algorithms, Gini impurity and entropy are criteria used to measure the impurity or disorder of data sets when deciding how to split nodes in the tree.

Gini impurity calculates the probability of misclassifying a randomly chosen data point based on class label distribution, while entropy measures the level of uncertainty in a data set. Gini impurity is computationally efficient, robust to class imbalances, and often produces similar results to entropy in practice.

Gini impurity is often chosen as the default criterion in popular machine learning libraries like scikit-learn. Gini impurity is computationally more efficient than entropy because it doesn't involve logarithmic calculations. If efficiency is a concern in large datasets, Gini may be preferred. Since we are operating on a large dataset, we choose gini.

DBSCAN:

I have used Silhouette Score for the code. When executing in Python, I got a Silhouette Score of 0.2277 whereas for R Studio we got a Score of 0.2949. It is clearly observed that the Silhouette score obtained after executing the code in R Studio is better than the one observed in Python.

Eps chosen was 0.01 and min samples as 15.

Eps is a positive numeric value that defines the maximum distance between two data points for one data point to be considered as in the neighborhood of the other. In other words, it sets the radius of the neighborhood around each data point.

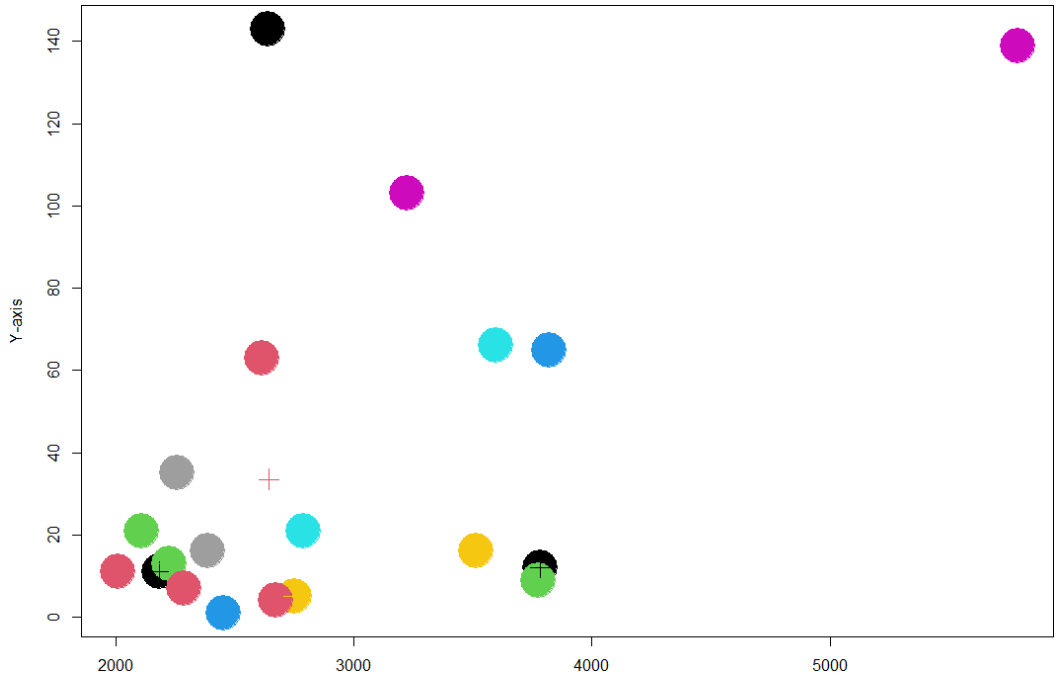
min_samples is an integer value that represents the minimum number of data points required to form a dense region (core point) that can be considered a cluster. For a data point to be classified as a core point, it must have at least min_samples data points (including itself) within its eps-radius neighborhood.

The choice of eps and min_samples depends on the specific characteristics of your dataset. It often involves a trial-and-error process or data exploration to find suitable values. In our case, we performed the analysis using multiple eps and min samples value in python before finalizing the values. The same value was continued in R.

7.2 Plots and Graphs in Python V/S R Studio

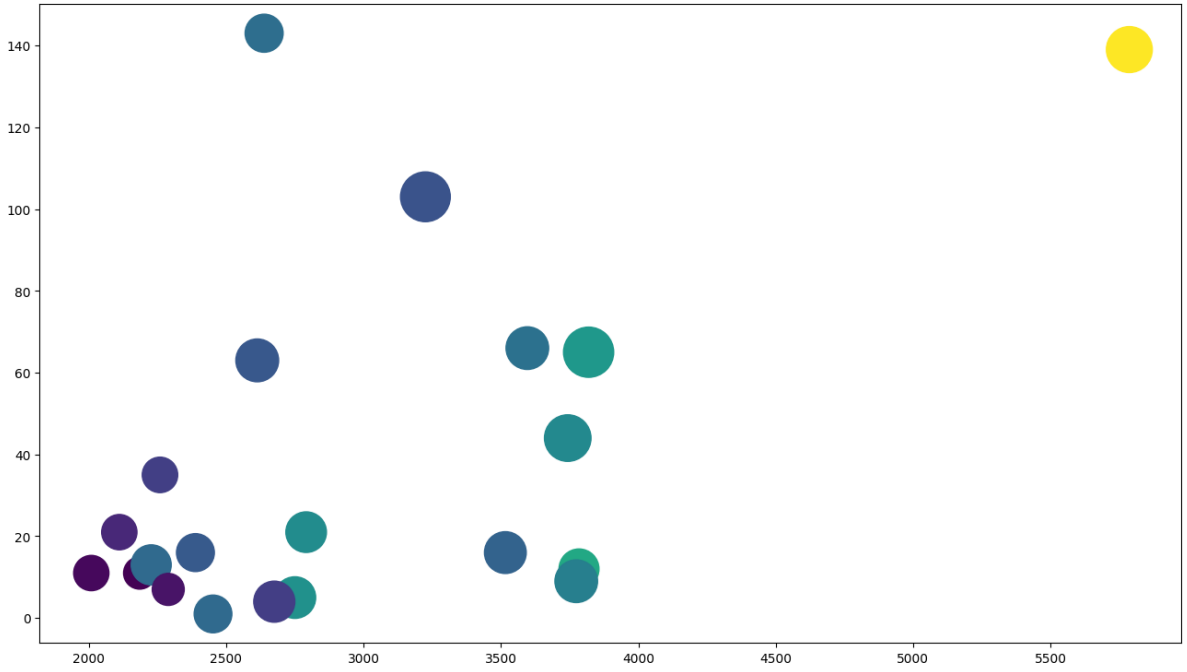
Scatter Plot for K-Means

PYTHON



:

R STUDIO



We have observed one extra cluster while executing in R Studio.

While comparing the clusters sizes, we observed the following:

CLUSTER SIZE FOR PYTHON:

```
Cluster 1 size: 1
Cluster 2 size: 1
Cluster 3 size: 1
Cluster 4 size: 1
Cluster 5 size: 1
Cluster 6 size: 1
Cluster 7 size: 1
Cluster 8 size: 1
Cluster 9 size: 1
Cluster 10 size: 1
Cluster 11 size: 1
Cluster 12 size: 1
Cluster 13 size: 1
Cluster 14 size: 1
Cluster 15 size: 1
Cluster 16 size: 1
Cluster 17 size: 1
Cluster 18 size: 1
Cluster 19 size: 1
```

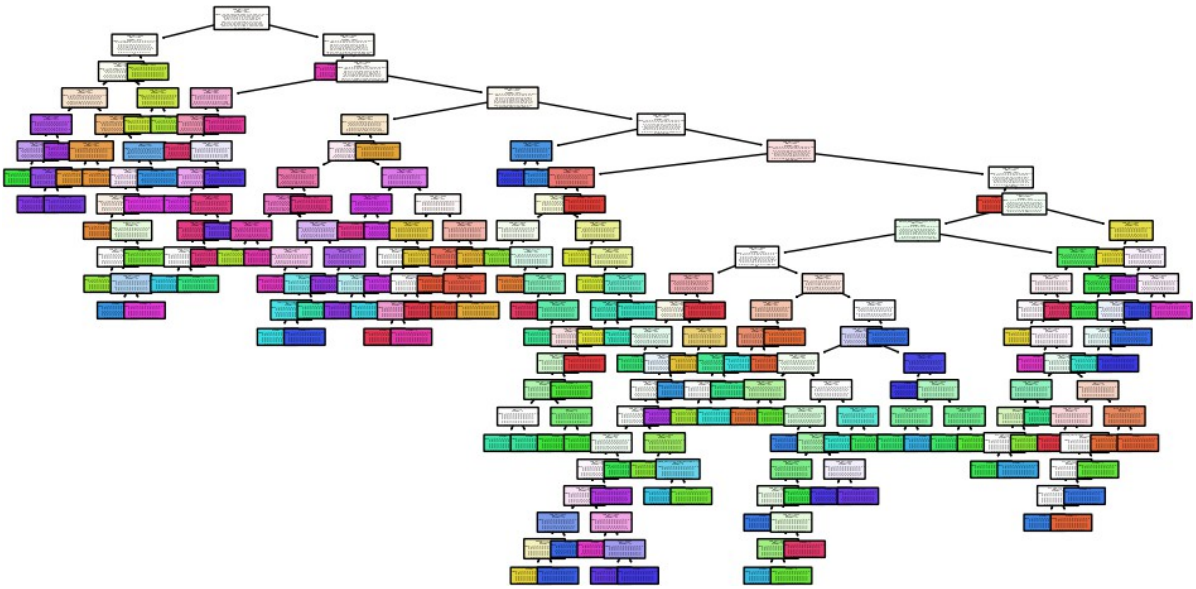
CLUSTER SIZE FOR R STUDIO

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1

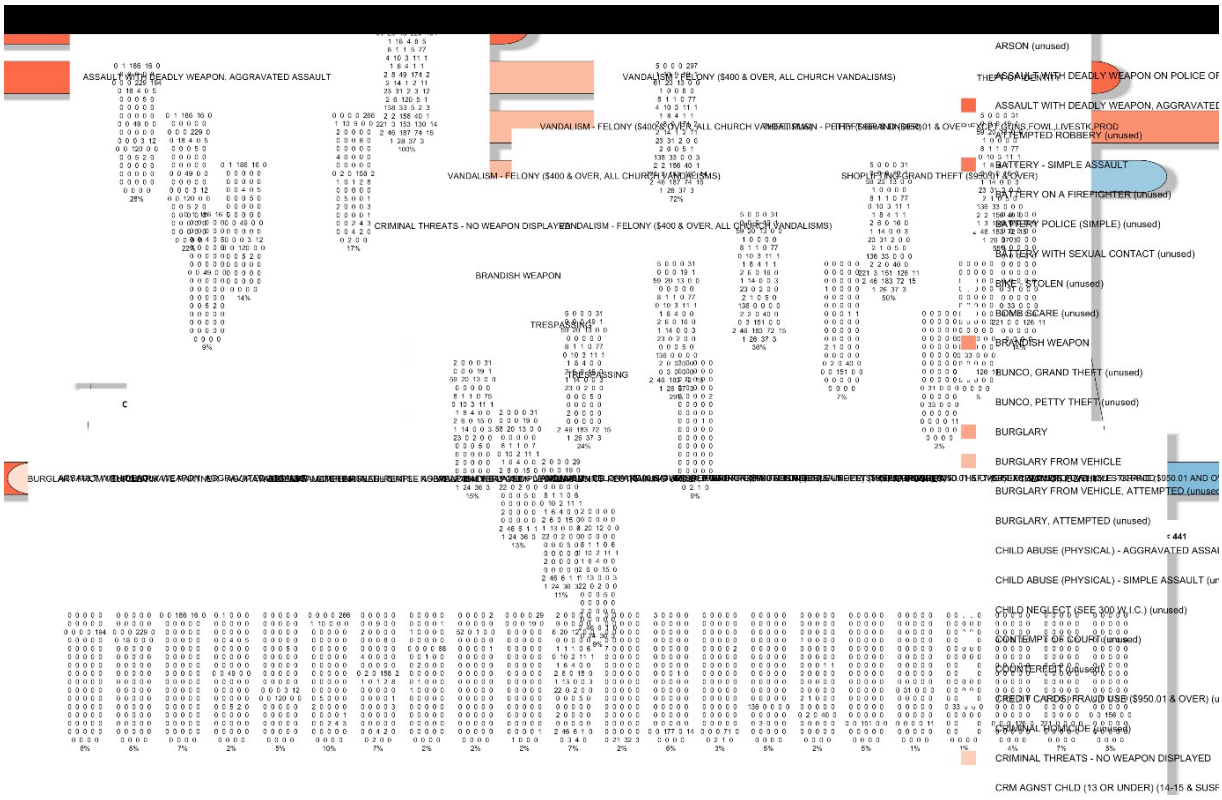
From the above cluster size, we observe that Python code provided us with consistent cluster size while R Studio had one outlier.

Decision Tree

PYTHON



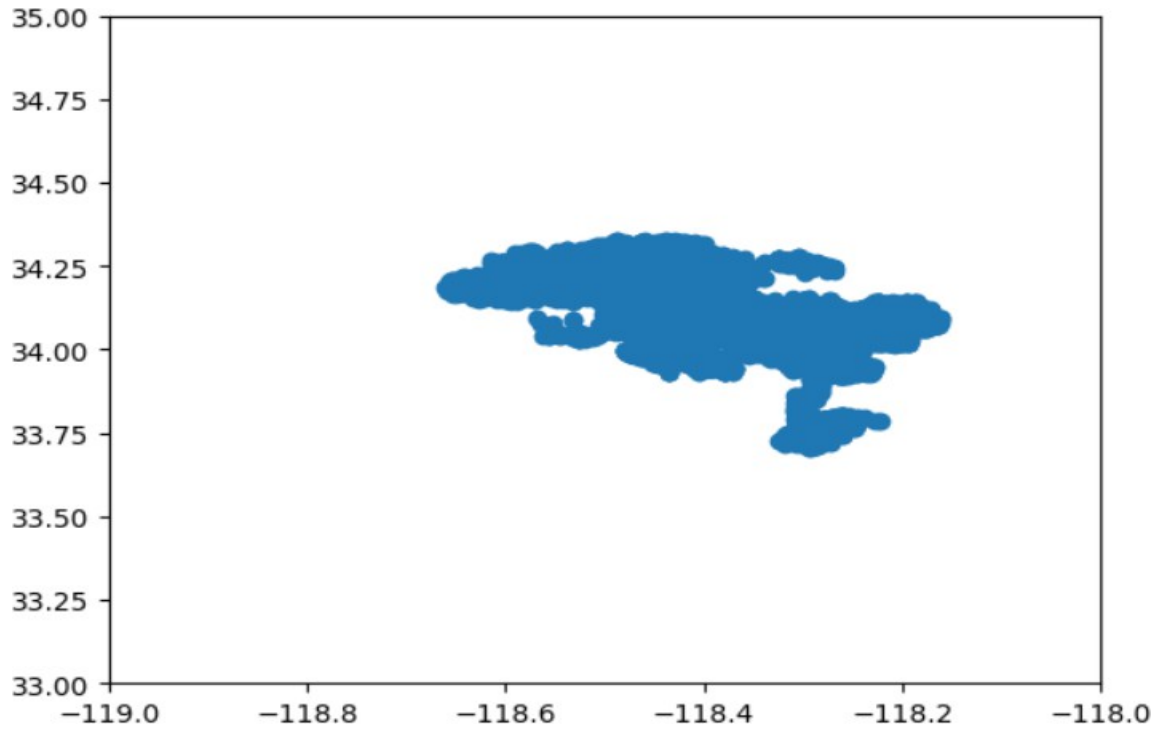
R STUDIO

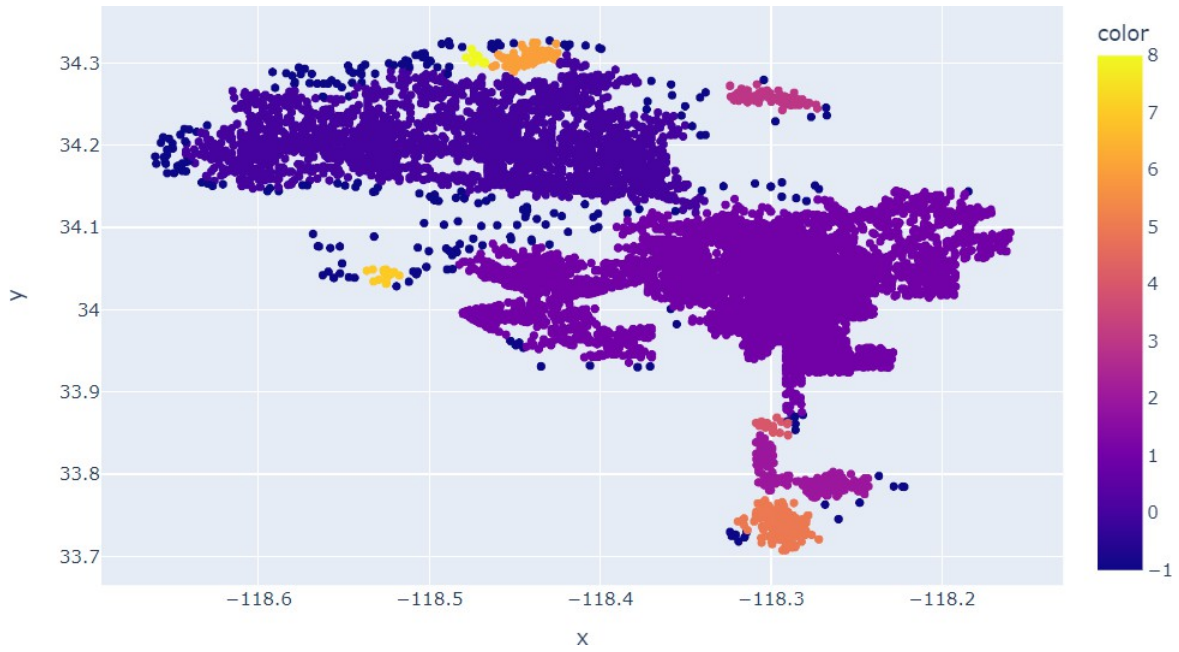


I was able to make a perfect decision tress when the code was executed in Python whereas the same was not observed in R Studio. The data pre-processing in R was not executed perfectly due to which we have implemented the visualization in Python

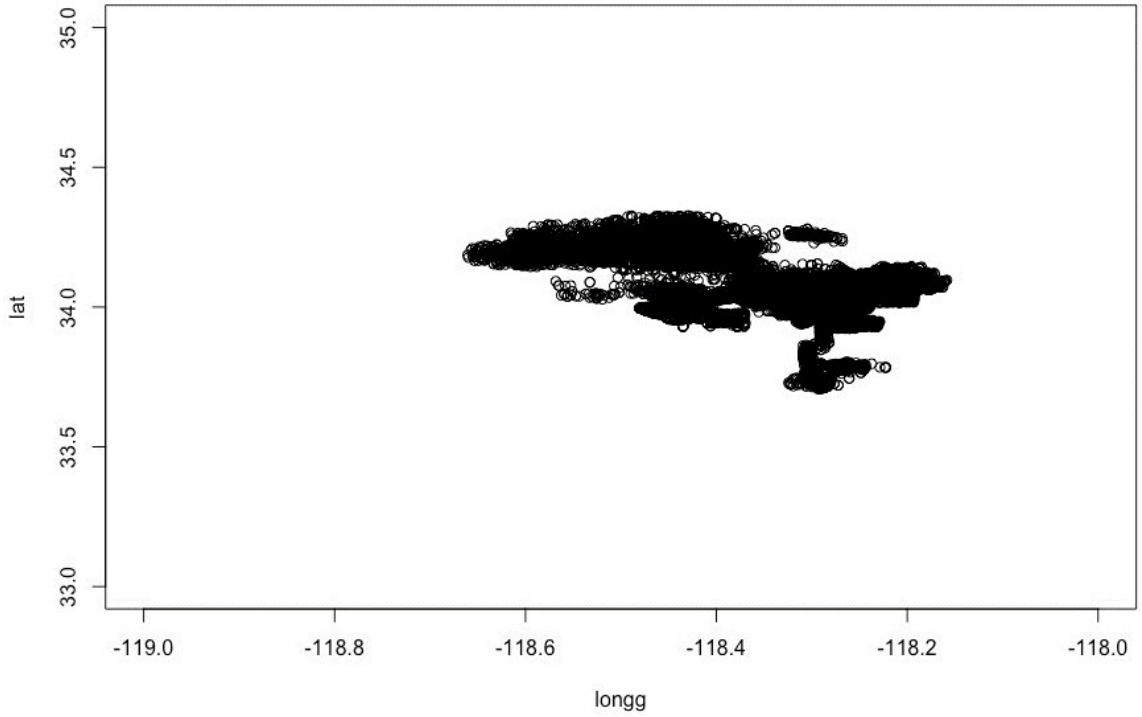
DBSCAN

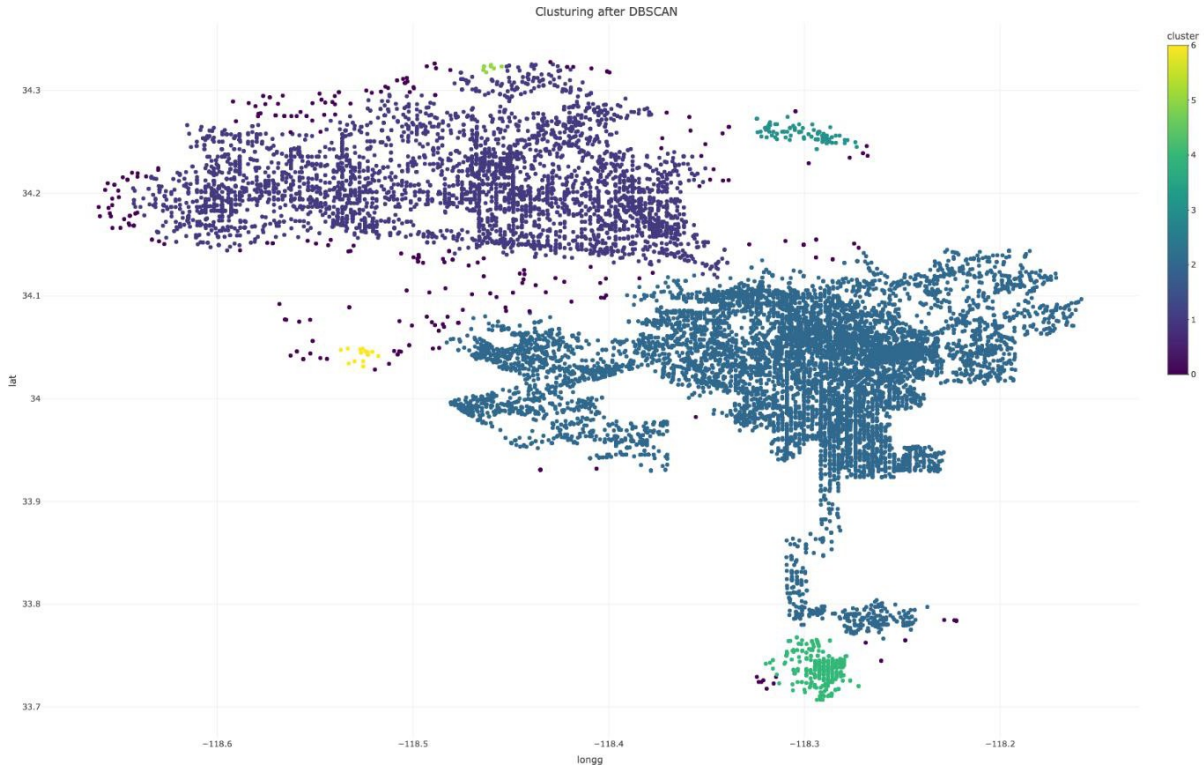
SCATTER PLOT FOR PYTHON





SCATTER PLOT FOR R STUDIO





More detailed Plots were created with R Studio, providing a better visualization of the clusters.

8. Summary

The data mining project began by selecting a dataset from the LAPD containing information about crimes committed in Los Angeles from 2020 to 2023. This dataset comprised 28 attributes (columns) and a substantial 788,768 instances (rows). The focus was on investigating crimes that occurred in the year 2023, so I extracted and analyzed relevant data for this year.

In the initial exploratory analysis before diving into machine learning, we have uncovered valuable insights. We can identify the most frequently occurring crimes, examine the distribution of gender and race among the victims, determine the frequency patterns of crimes, and pinpoint the peak times when crimes tend to happen.

Moving on to the second part of the project, I set out to apply various machine learning algorithms, including K-means, Linear Regression, Logistic Regression, DBSCAN, and Decision Trees. I opted to utilize both Python and R for implementing these algorithms, and this choice led to some interesting observations and differences in their outcomes.

- ❖ K – Means:

When implementing K-means clustering, we observed an intriguing difference between Python and R Studio. In Python, we identified one extra cluster compared to R Studio. This variation piqued our curiosity and warranted further investigation.

Additionally, it was noteworthy that in Python, the cluster sizes remained consistent, offering stability and predictability in our results. In contrast, in R Studio, we observed differences in the cluster sizes for one of the clusters, indicating potential sensitivity to initial conditions.

- ❖ Linear and Logistic Regression:

Our objective was to classify relationships between the area of crime committed and other attributes. Implementing these algorithms in R proved challenging, with unsatisfactory results and cumbersome data preprocessing.

Python, on the other hand, provided more favorable outcomes, smoother data handling, and more interpretable results. Plots and visualizations generated in Python were clearer and more informative.

- ❖ DBSCAN:

For DBSCAN, our aim was to cluster crime data based on latitude and longitude coordinates. Remarkably, the implementation in R was straightforward and efficient.

The results and observations from the analysis in R were consistent with those from Python, showing the formation of clusters resembling the geographical shape of Los Angeles.

Decision Tree:

In the case of Decision Trees, our goal was to organize crimes based on crime code descriptions. The implementation proved to be somewhat tedious in both Python and R.

However, Python outshines R in terms of result clarity and interpretability. The outcomes, plots, and visualizations generated in Python were notably more legible and comprehensible.

In summary, our project journey involved data selection, pre-machine learning analysis, and the application of various machine learning algorithms using both Python and R. While R exhibited efficiency in some respects, Python generally outperformed in terms of result quality, data preprocessing ease, and visualization clarity, making it the preferred choice for our data mining tasks. These findings provided valuable insights into crime patterns in Los Angeles in 2023 and the efficacy of different tools for data analysis.