# DATA MINING

**LA - 2**

## CRIME PREDICTION USING k-NN CLASSIFIER

*Bachelor Of Engineering*
*in*
*Information Science and Engineering*

SUBMITTED BY:

**CHHAVI VERMA**

Under the Guidance of Mr. AS
Assistant Professor, Dept of ISE

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**
(Accredited by NBA Tier – 1)

# LA2 – TOPIC

**ALLOTED SATEMENT FOR “Crime Prediction” :**

Use a k-NN classifier to predict the likelihood of a crime occurring in a specific location. Use a dataset of crime incidents & their corresponding location & time to train the classifier. Then use the trained model to predict the likelihood of a crime occurring in a new location based on its nearest neighbors:

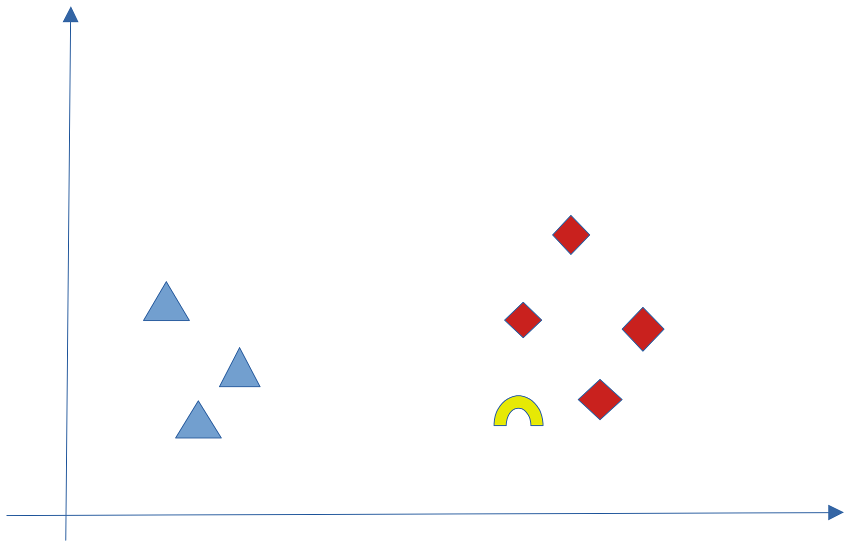
## WHAT IS k-NN

A. KNN classifier is a machine learning algorithm used for classification and regression problems.

*It works by finding the K nearest points in the training dataset and uses their class to predict the class or value of a new data point. It can handle complex data and is also easy to implement, which is why despite* its simplicity, KNN can outperform more powerful classifiers & therefore *has become a popular tool in the field of artificial intelligence.*

## HOW DOES k-NN WORK

Consider the following figure. As shown, we have a total of 8 data points (3 blue and 4 red). Red data points belong to ‘class1’ and blue data points belong to ‘class2’. And yellow data point in a feature space represents the new point for which a class is to be predicted. Obviously, we say it belongs to ‘class1’ (red points)



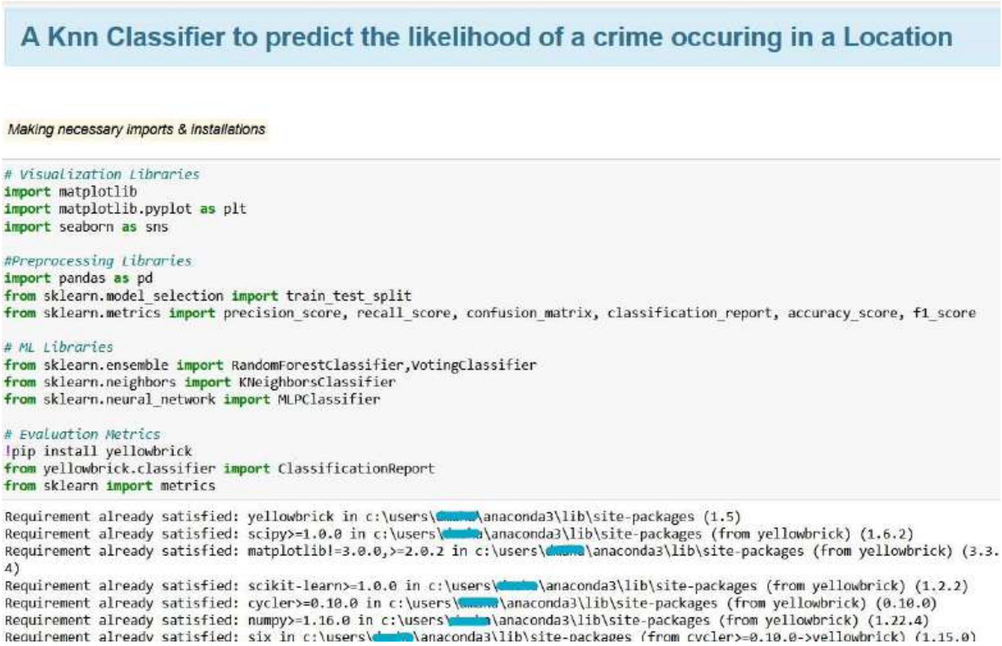
<u>Why?</u>

***Because its nearest neighbors belong to that class!***

Here, nearest neighbors are those data points that have minimum distance in feature space from our new data point. And K is the number of such data points we consider in our implementation of the algorithm. Therefore, distance metric and K value are two important considerations while using the KNN algorithm. Euclidean distance is the most popular distance metric. & we can use many more!

## CODE EXPLANATION WITH SCREENSHOTS:

### 1. Installing necessary Libraries here



A Knn Classifier to predict the likelihood of a crime occuring in a Location

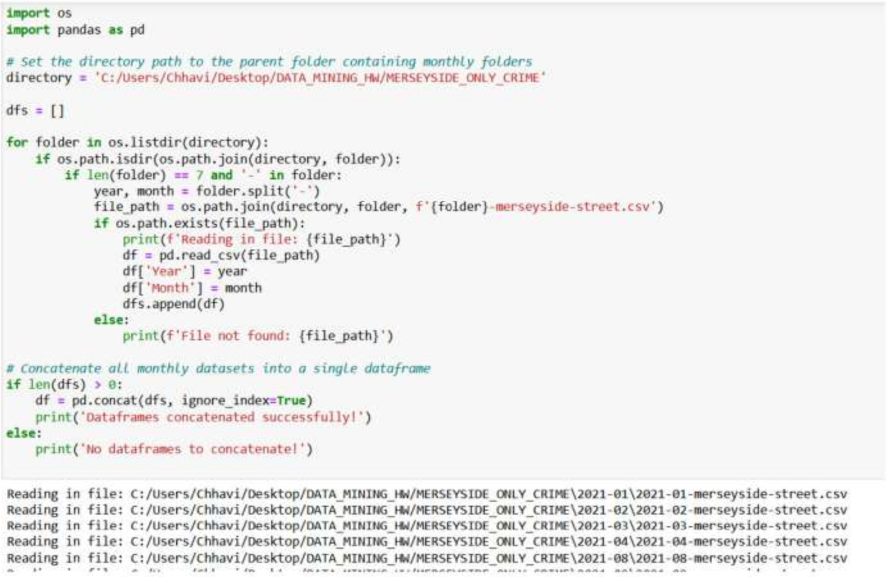
*Making necessary imports & installations*

```
# Visualization Libraries
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

#Preprocessing Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, confusion_matrix, classification_report, accuracy_score, f1_score

# ML Libraries
from sklearn.ensemble import RandomForestClassifier,VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier

# Evaluation Metrics
!pip install yellowbrick
from yellowbrick.classifier import ClassificationReport
from sklearn import metrics
```

```
Requirement already satisfied: yellowbrick in c:\users\[redacted]\anaconda3\lib\site-packages (1.5)
Requirement already satisfied: scipy>=1.0.0 in c:\users\[redacted]\anaconda3\lib\site-packages (from yellowbrick) (1.6.2)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in c:\users\[redacted]\anaconda3\lib\site-packages (from yellowbrick) (3.3.
4)
Requirement already satisfied: scikit-learn>=1.0.0 in c:\users\[redacted]\anaconda3\lib\site-packages (from yellowbrick) (1.2.2)
Requirement already satisfied: cycler>=0.10.0 in c:\users\[redacted]\anaconda3\lib\site-packages (from yellowbrick) (0.10.0)
Requirement already satisfied: numpy>=1.16.0 in c:\users\[redacted]\anaconda3\lib\site-packages (from yellowbrick) (1.22.4)
Requirement already satisfied: six in c:\users\[redacted]\anaconda3\lib\site-packages (from cycler>=0.10.0->yellowbrick) (1.15.0)
```

## 2. Segregating the data from multiple csv files into a single one.

(As UK Crime website's data has folders from year 2020 to 2023 for all 12 months & this data is contained in their corresponding csv files.      )

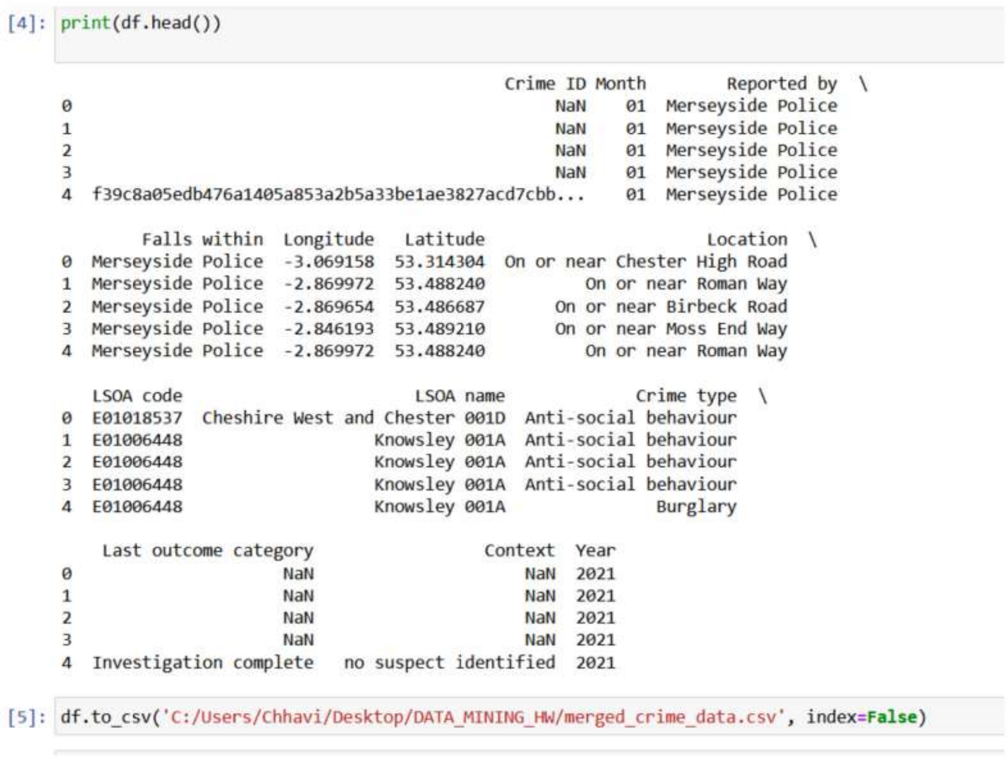


```
import os
import pandas as pd

# Set the directory path to the parent folder containing monthly folders
directory = 'C:/Users/Chhavi/Desktop/DATA_MINING_HW/MERSEYSIDE_ONLY_CRIME'

dfs = []

for folder in os.listdir(directory):
    if os.path.isdir(os.path.join(directory, folder)):
        if len(folder) == 7 and '-' in folder:
            year, month = folder.split('-')
            file_path = os.path.join(directory, folder, f'{folder}-merseyside-street.csv')
            if os.path.exists(file_path):
                print(f'Reading in file: {file_path}')
                df = pd.read_csv(file_path)
                df['Year'] = year
                df['Month'] = month
                dfs.append(df)
            else:
                print(f'File not found: {file_path}')

# Concatenate all monthly datasets into a single dataframe
if len(dfs) > 0:
    df = pd.concat(dfs, ignore_index=True)
    print('Dataframes concatenated successfully!')
else:
    print('No dataframes to concatenate!')
```

```
Reading in file: C:/Users/Chhavi/Desktop/DATA_MINING_HW/MERSEYSIDE_ONLY_CRIME\2021-01\2021-01-merseyside-street.csv
Reading in file: C:/Users/Chhavi/Desktop/DATA_MINING_HW/MERSEYSIDE_ONLY_CRIME\2021-02\2021-02-merseyside-street.csv
Reading in file: C:/Users/Chhavi/Desktop/DATA_MINING_HW/MERSEYSIDE_ONLY_CRIME\2021-03\2021-03-merseyside-street.csv
Reading in file: C:/Users/Chhavi/Desktop/DATA_MINING_HW/MERSEYSIDE_ONLY_CRIME\2021-04\2021-04-merseyside-street.csv
Reading in file: C:/Users/Chhavi/Desktop/DATA_MINING_HW/MERSEYSIDE_ONLY_CRIME\2021-08\2021-08-merseyside-street.csv
```

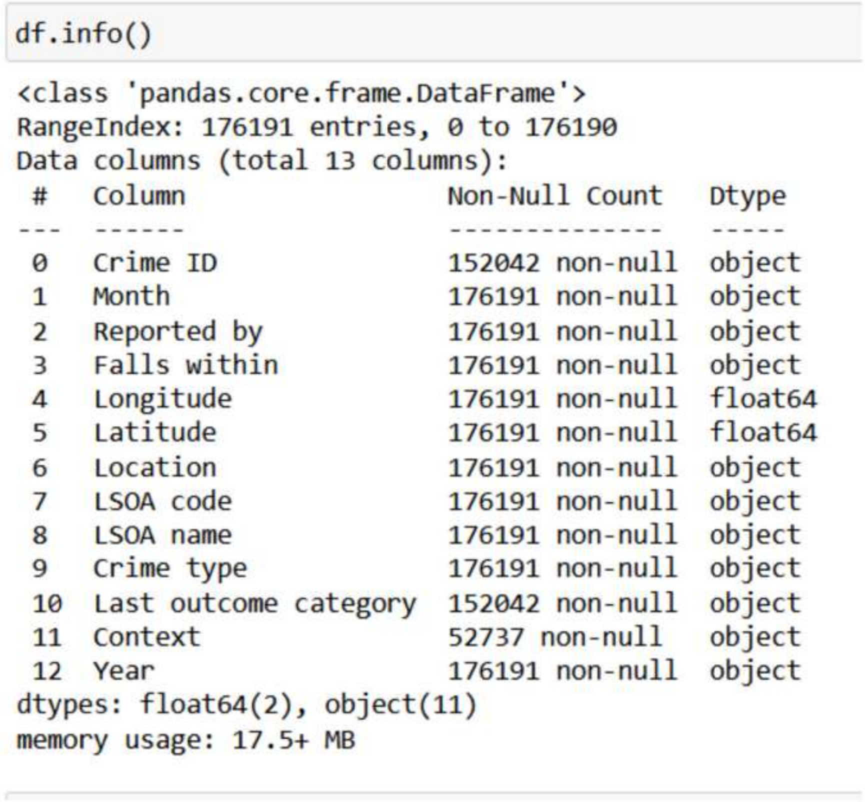
## 3. Printing top 5 rows from the dataframe that has the csv data

(Later saving all merged csv's into a single csv named merged_crime_data.csv)



```
[4]: print(df.head())

                                            Crime ID Month       Reported by  \
0                                                NaN    01  Merseyside Police
1                                                NaN    01  Merseyside Police
2                                                NaN    01  Merseyside Police
3                                                NaN    01  Merseyside Police
4  f39c8a05edb476a1405a853a2b5a33be1ae3827acd7cbb...    01  Merseyside Police

        Falls within  Longitude   Latitude                 Location  \
0  Merseyside Police  -3.069158  53.314304  On or near Chester High Road
1  Merseyside Police  -2.869972  53.488240         On or near Roman Way
2  Merseyside Police  -2.869654  53.486687       On or near Birbeck Road
3  Merseyside Police  -2.846193  53.489210       On or near Moss End Way
4  Merseyside Police  -2.869972  53.488240         On or near Roman Way

   LSOA code                    LSOA name           Crime type  \
0  E01018537  Cheshire West and Chester 001D  Anti-social behaviour
1  E01006448                 Knowsley 001A  Anti-social behaviour
2  E01006448                 Knowsley 001A  Anti-social behaviour
3  E01006448                 Knowsley 001A  Anti-social behaviour
4  E01006448                 Knowsley 001A              Burglary

    Last outcome category               Context  Year
0                     NaN                   NaN  2021
1                     NaN                   NaN  2021
2                     NaN                   NaN  2021
3                     NaN                   NaN  2021
4  Investigation complete  no suspect identified  2021

[5]: df.to_csv('C:/Users/Chhavi/Desktop/DATA_MINING_HW/merged_crime_data.csv', index=False)
```

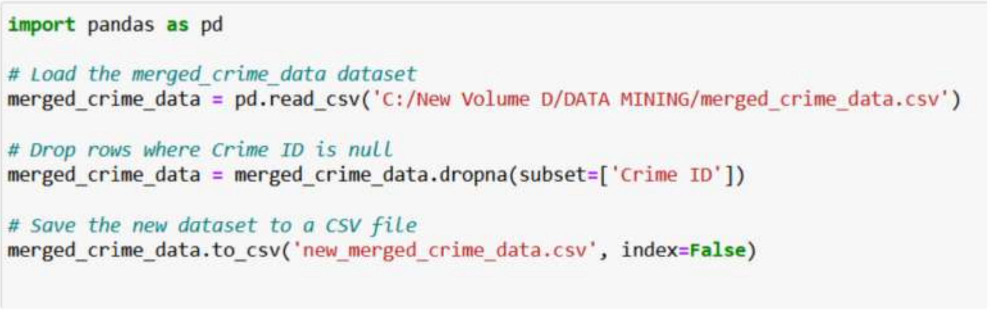
## 4. Dataframe information



```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 176191 entries, 0 to 176190
Data columns (total 13 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   Crime ID               152042 non-null  object 
 1   Month                  176191 non-null  object 
 2   Reported by            176191 non-null  object 
 3   Falls within           176191 non-null  object 
 4   Longitude              176191 non-null  float64
 5   Latitude               176191 non-null  float64
 6   Location               176191 non-null  object 
 7   LSOA code              176191 non-null  object 
 8   LSOA name              176191 non-null  object 
 9   Crime type             176191 non-null  object 
 10  Last outcome category  152042 non-null  object 
 11  Context                52737 non-null   object 
 12  Year                   176191 non-null  object 
dtypes: float64(2), object(11)
memory usage: 17.5+ MB
```

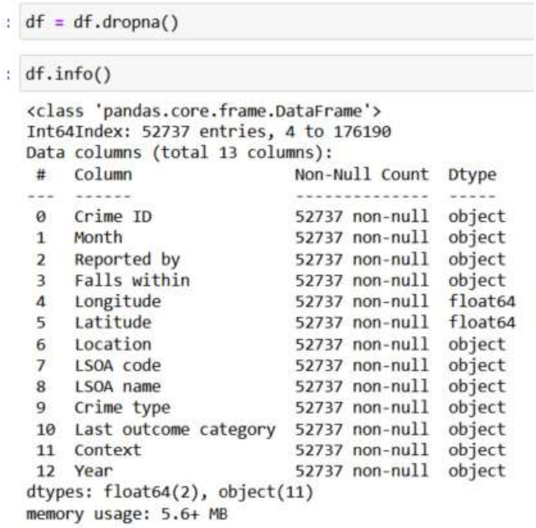
## 5. Removing missing values from both rows & column if present

## & saving in new_merged_csv.csv file

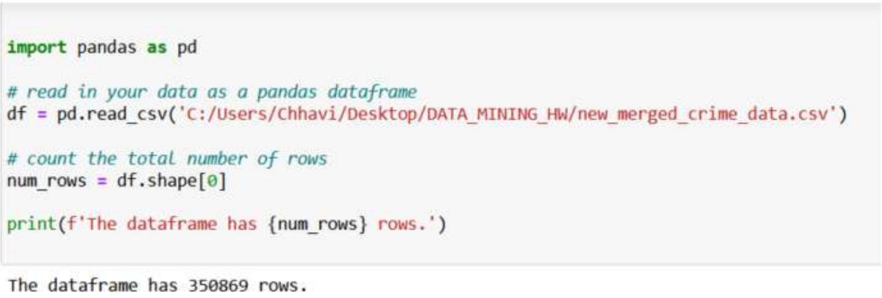


```
import pandas as pd

# Load the merged_crime_data dataset
merged_crime_data = pd.read_csv('C:/New Volume D/DATA MINING/merged_crime_data.csv')

# Drop rows where Crime ID is null
merged_crime_data = merged_crime_data.dropna(subset=['Crime ID'])

# Save the new dataset to a CSV file
merged_crime_data.to_csv('new_merged_crime_data.csv', index=False)
```

## 6. Basics dataframe information from this final new_merged_csv.csv

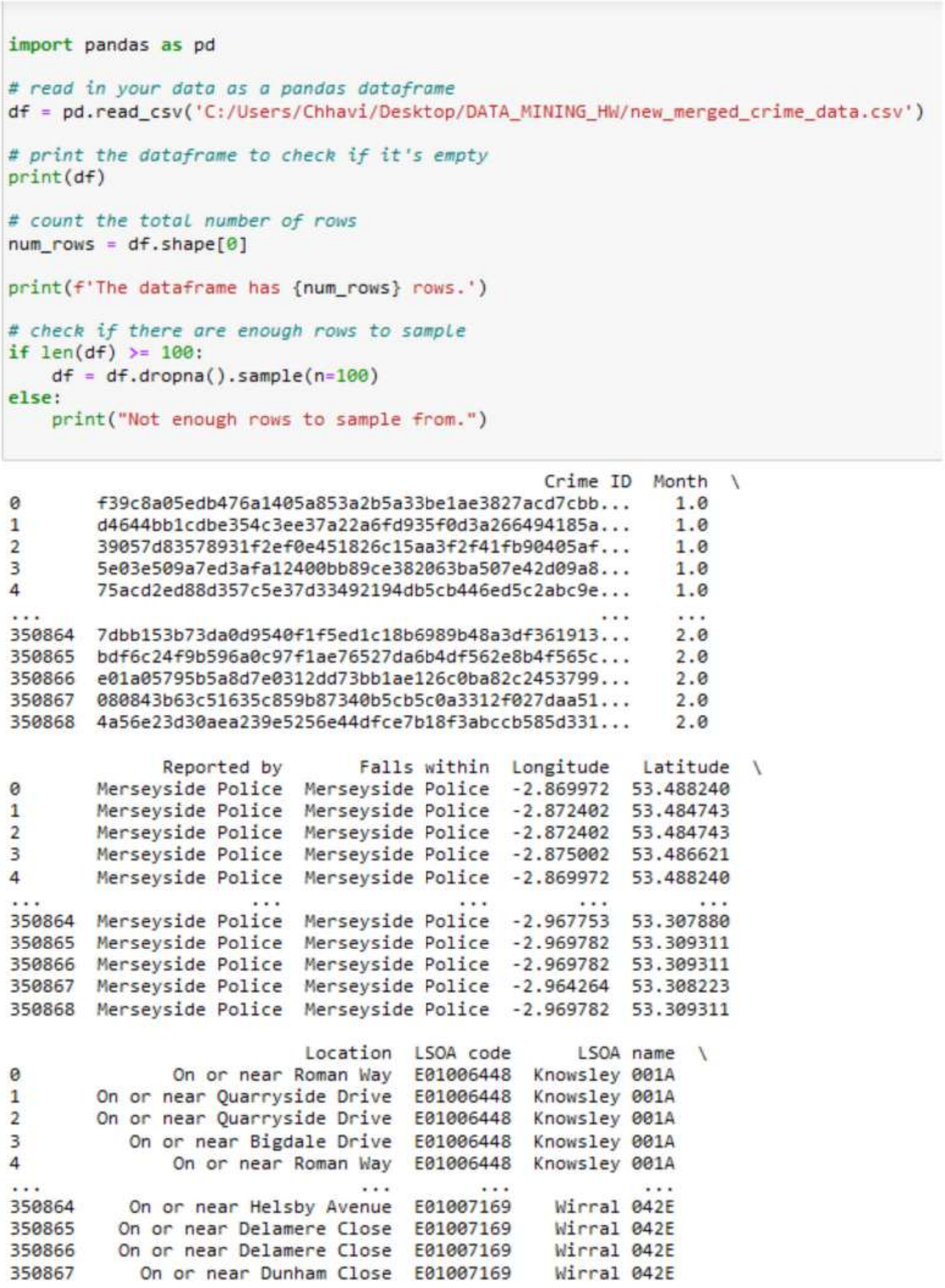


```
: df = df.dropna()

: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 52737 entries, 4 to 176190
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Crime ID               52737 non-null  object 
 1   Month                  52737 non-null  object 
 2   Reported by            52737 non-null  object 
 3   Falls within           52737 non-null  object 
 4   Longitude              52737 non-null  float64
 5   Latitude               52737 non-null  float64
 6   Location               52737 non-null  object 
 7   LSOA code              52737 non-null  object 
 8   LSOA name              52737 non-null  object 
 9   Crime type             52737 non-null  object 
 10  Last outcome category  52737 non-null  object 
 11  Context                52737 non-null  object 
 12  Year                   52737 non-null  object 
dtypes: float64(2), object(11)
memory usage: 5.6+ MB
```



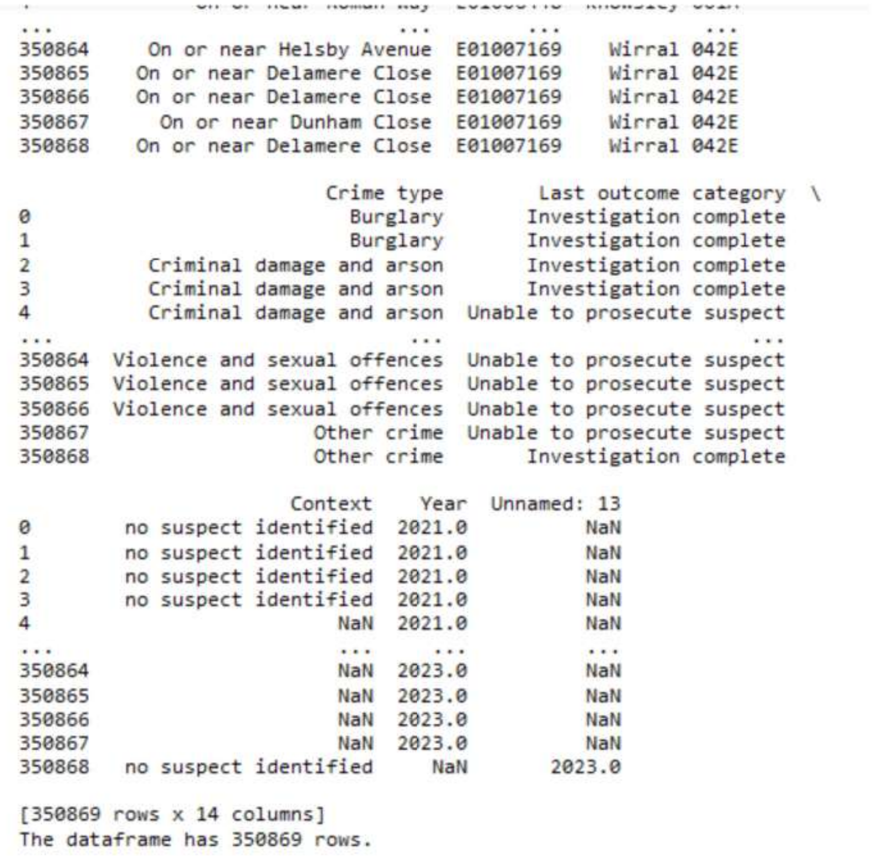
```
import pandas as pd

# read in your data as a pandas dataframe
df = pd.read_csv('C:/Users/Chhavi/Desktop/DATA_MINING_HW/new_merged_crime_data.csv')

# count the total number of rows
num_rows = df.shape[0]

print(f'The dataframe has {num_rows} rows.')

The dataframe has 350869 rows.
```

## 7. Checking if sufficient rows are present to sample the data from



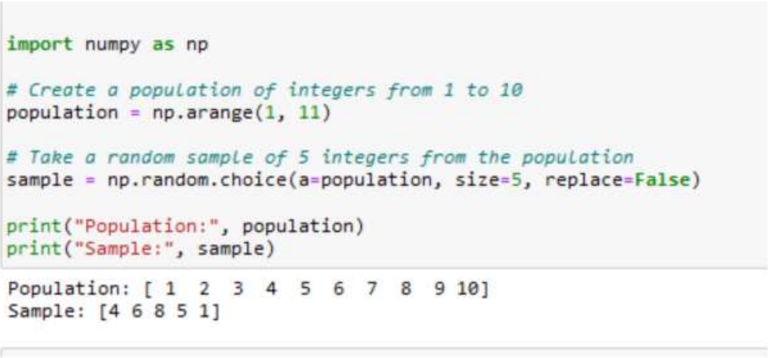
```python
import pandas as pd

# read in your data as a pandas dataframe
df = pd.read_csv('C:/Users/Chhavi/Desktop/DATA_MINING_HW/new_merged_crime_data.csv')

# print the dataframe to check if it's empty
print(df)

# count the total number of rows
num_rows = df.shape[0]

print(f'The dataframe has {num_rows} rows.')

# check if there are enough rows to sample
if len(df) >= 100:
    df = df.dropna().sample(n=100)
else:
    print("Not enough rows to sample from.")
```

```
                                                 Crime ID  Month  \
0       f39c8a05edb476a1405a853a2b5a33be1ae3827acd7cbb...    1.0   
1       d4644bb1cdbe354c3ee37a22a6fd935f0d3a266494185a...    1.0   
2       39057d83578931f2ef0e451826c15aa3f2f41fb90405af...    1.0   
3       5e03e509a7ed3afa12400bb89ce382063ba507e42d09a8...    1.0   
4       75acd2ed88d357c5e37d33492194db5cb446ed5c2abc9e...    1.0   
...                                                   ...    ...   
350864  7dbb153b73da0d9540f1f5ed1c18b6989b48a3df361913...    2.0   
350865  bdf6c24f9b596a0c97f1ae76527da6b4df562e8b4f565c...    2.0   
350866  e01a05795b5a8d7e0312dd73bb1ae126c0ba82c2453799...    2.0   
350867  080843b63c51635c859b87340b5cb5c0a3312f027daa51...    2.0   
350868  4a56e23d30aea239e5256e44dfce7b18f3abccb585d331...    2.0   

              Reported by       Falls within  Longitude   Latitude  \
0       Merseyside Police  Merseyside Police  -2.869972  53.488240   
1       Merseyside Police  Merseyside Police  -2.872402  53.484743   
2       Merseyside Police  Merseyside Police  -2.872402  53.484743   
3       Merseyside Police  Merseyside Police  -2.875002  53.486621   
4       Merseyside Police  Merseyside Police  -2.869972  53.488240   
...                   ...                ...        ...        ...   
350864  Merseyside Police  Merseyside Police  -2.967753  53.307880   
350865  Merseyside Police  Merseyside Police  -2.969782  53.309311   
350866  Merseyside Police  Merseyside Police  -2.969782  53.309311   
350867  Merseyside Police  Merseyside Police  -2.964264  53.308223   
350868  Merseyside Police  Merseyside Police  -2.969782  53.309311   

                        Location  LSOA code     LSOA name  \
0             On or near Roman Way  E01006448  Knowsley 001A   
1       On or near Quarryside Drive  E01006448  Knowsley 001A   
2       On or near Quarryside Drive  E01006448  Knowsley 001A   
3          On or near Bigdale Drive  E01006448  Knowsley 001A   
4             On or near Roman Way  E01006448  Knowsley 001A   
...                           ...        ...           ...   
350864     On or near Helsby Avenue  E01007169    Wirral 042E   
350865   On or near Delamere Close  E01007169    Wirral 042E   
350866   On or near Delamere Close  E01007169    Wirral 042E   
350867      On or near Dunham Close  E01007169    Wirral 042E   
```



```
...                              ...         ...          ...
350864     On or near Helsby Avenue  E01007169    Wirral 042E
350865    On or near Delamere Close  E01007169    Wirral 042E
350866    On or near Delamere Close  E01007169    Wirral 042E
350867      On or near Dunham Close  E01007169    Wirral 042E
350868    On or near Delamere Close  E01007169    Wirral 042E

                         Crime type        Last outcome category  \
0                          Burglary       Investigation complete
1                          Burglary       Investigation complete
2         Criminal damage and arson       Investigation complete
3         Criminal damage and arson       Investigation complete
4         Criminal damage and arson  Unable to prosecute suspect
...                             ...                          ...
350864  Violence and sexual offences  Unable to prosecute suspect
350865  Violence and sexual offences  Unable to prosecute suspect
350866  Violence and sexual offences  Unable to prosecute suspect
350867                  Other crime  Unable to prosecute suspect
350868                  Other crime       Investigation complete

                      Context    Year  Unnamed: 13
0       no suspect identified  2021.0          NaN
1       no suspect identified  2021.0          NaN
2       no suspect identified  2021.0          NaN
3       no suspect identified  2021.0          NaN
4                         NaN  2021.0          NaN
...                       ...     ...          ...
350864                    NaN  2023.0          NaN
350865                    NaN  2023.0          NaN
350866                    NaN  2023.0          NaN
350867                    NaN  2023.0          NaN
350868  no suspect identified     NaN       2023.0

[350869 rows x 14 columns]
The dataframe has 350869 rows.
```

**8. Here we, generate a population of integers from 1 to 10 and take a random sample of 5 integers from the population in an array**

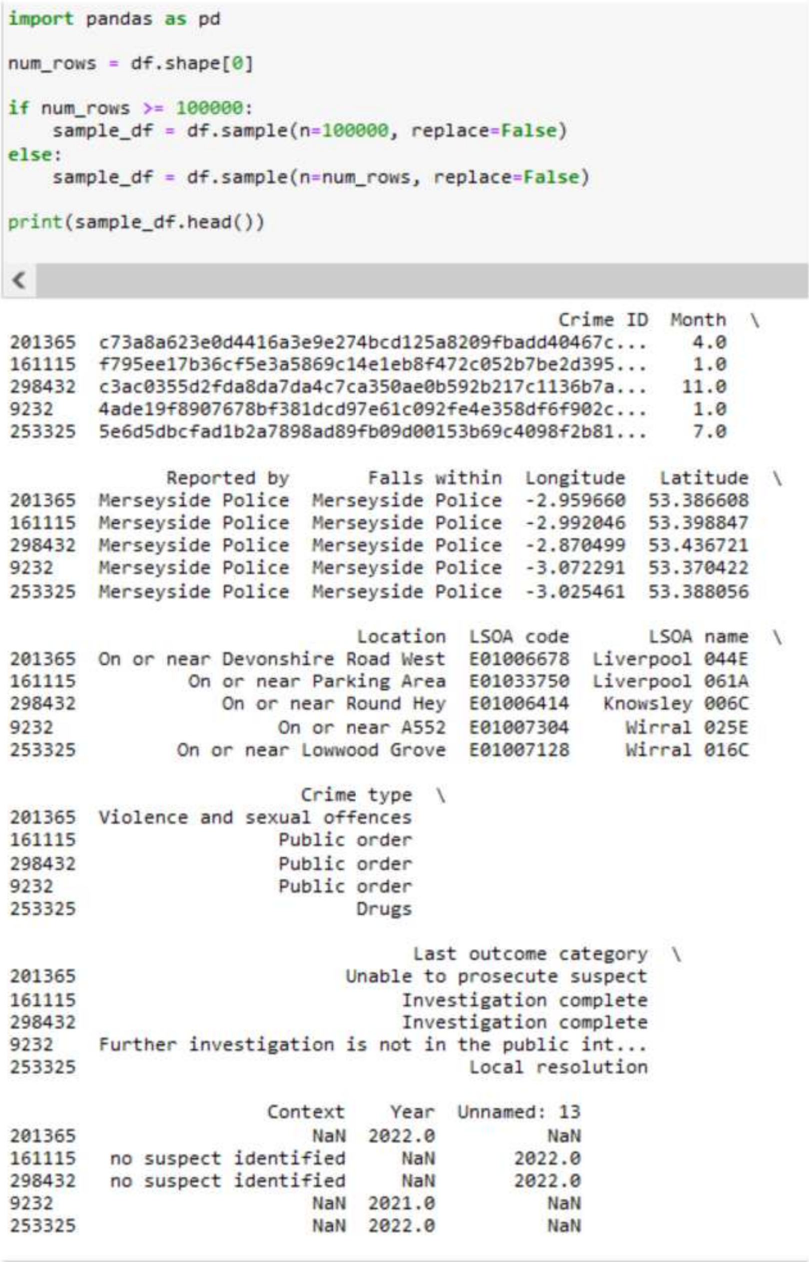


```python
import numpy as np

# Create a population of integers from 1 to 10
population = np.arange(1, 11)

# Take a random sample of 5 integers from the population
sample = np.random.choice(a=population, size=5, replace=False)

print("Population:", population)
print("Sample:", sample)
```

```
Population: [ 1  2  3  4  5  6  7  8  9 10]
Sample: [4 6 8 5 1]
```

**9. Here we'll use pandas to sample 100,000 rows from a DataFrame object df without replacement.**

Then we’ll create a new DataFrame object ‘sample_df’ containing a random sample of 100,000 rows from original df.



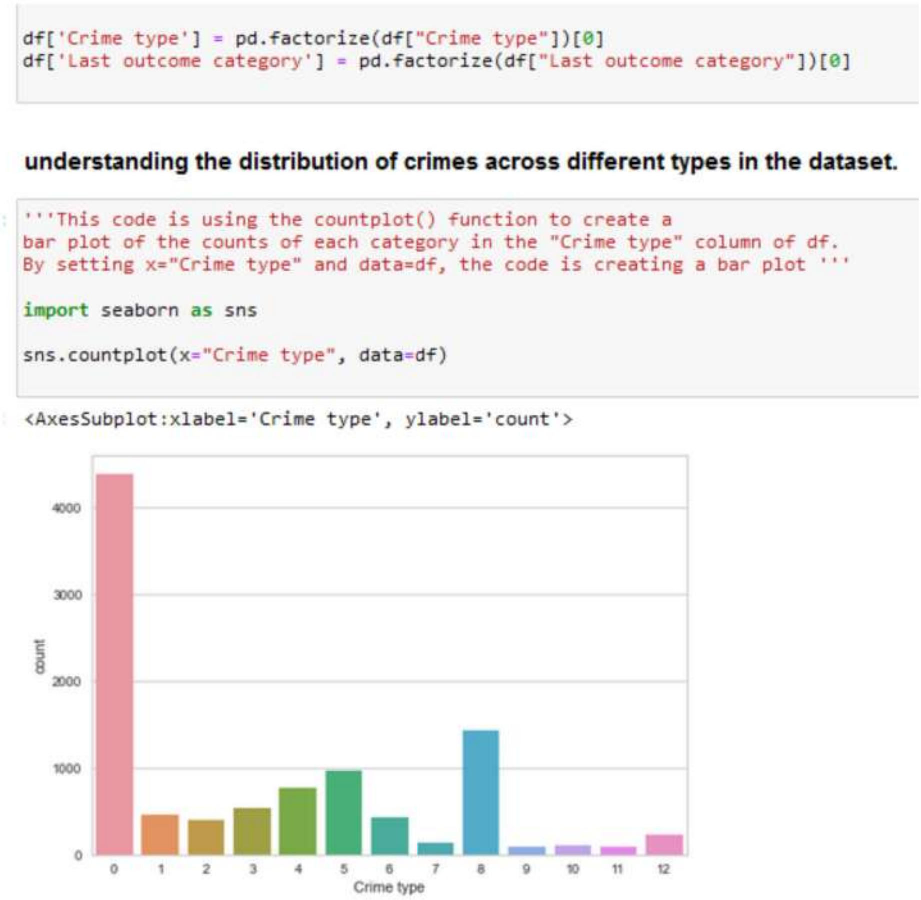
```python
import pandas as pd

num_rows = df.shape[0]

if num_rows >= 100000:
    sample_df = df.sample(n=100000, replace=False)
else:
    sample_df = df.sample(n=num_rows, replace=False)

print(sample_df.head())
```

```
                                                 Crime ID  Month  \
201365  c73a8a623e0d4416a3e9e274bcd125a8209fbadd40467c...    4.0   
161115  f795ee17b36cf5e3a5869c14e1eb8f472c052b7be2d395...    1.0   
298432  c3ac0355d2fda8da7da4c7ca350ae0b592b217c1136b7a...   11.0   
9232    4ade19f8907678bf381dcd97e61c092fe4e358df6f902c...    1.0   
253325  5e6d5dbcfad1b2a7898ad89fb09d00153b69c4098f2b81...    7.0   

              Reported by       Falls within  Longitude   Latitude  \
201365  Merseyside Police  Merseyside Police  -2.959660  53.386608   
161115  Merseyside Police  Merseyside Police  -2.992046  53.398847   
298432  Merseyside Police  Merseyside Police  -2.870499  53.436721   
9232    Merseyside Police  Merseyside Police  -3.072291  53.370422   
253325  Merseyside Police  Merseyside Police  -3.025461  53.388056   

                               Location  LSOA code       LSOA name  \
201365  On or near Devonshire Road West  E01006678  Liverpool 044E   
161115          On or near Parking Area  E01033750  Liverpool 061A   
298432             On or near Round Hey  E01006414   Knowsley 006C   
9232                    On or near A552  E01007304     Wirral 025E   
253325         On or near Lowwood Grove  E01007128     Wirral 016C   

                          Crime type  \
201365  Violence and sexual offences   
161115                  Public order   
298432                  Public order   
9232                    Public order   
253325                         Drugs   

                                     Last outcome category  \
201365                       Unable to prosecute suspect   
161115                            Investigation complete   
298432                            Investigation complete   
9232    Further investigation is not in the public int...   
253325                                  Local resolution   

                      Context    Year  Unnamed: 13  
201365                    NaN  2022.0          NaN  
161115  no suspect identified     NaN       2022.0  
298432  no suspect identified     NaN       2022.0  
9232                      NaN  2021.0          NaN  
253325                    NaN  2022.0          NaN  
```

**10 . Here factorize() is converting the selected column names from text-based**

**categorical values to numerical values, which will make it easier to work with.**

In part 2 of code we created a bar plot using the df we created by converting text based categorical values to numeric based.



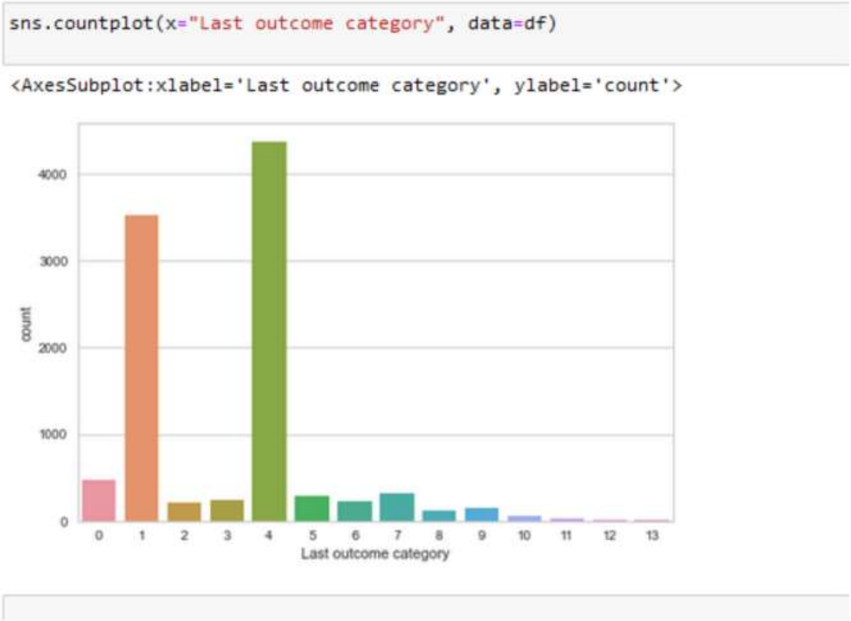
```
df['Crime type'] = pd.factorize(df["Crime type"])[0]
df['Last outcome category'] = pd.factorize(df["Last outcome category"])[0]
```

**understanding the distribution of crimes across different types in the dataset.**

```
'''This code is using the countplot() function to create a
bar plot of the counts of each category in the "Crime type" column of df.
By setting x="Crime type" and data=df, the code is creating a bar plot '''

import seaborn as sns

sns.countplot(x="Crime type", data=df)
```

```
<AxesSubplot:xlabel='Crime type', ylabel='count'>
```

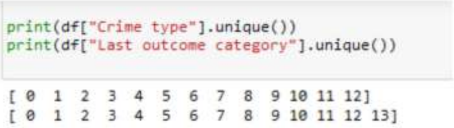
## 11. Contains bar plot of last outcome category column



```
sns.countplot(x="Last outcome category", data=df)
```

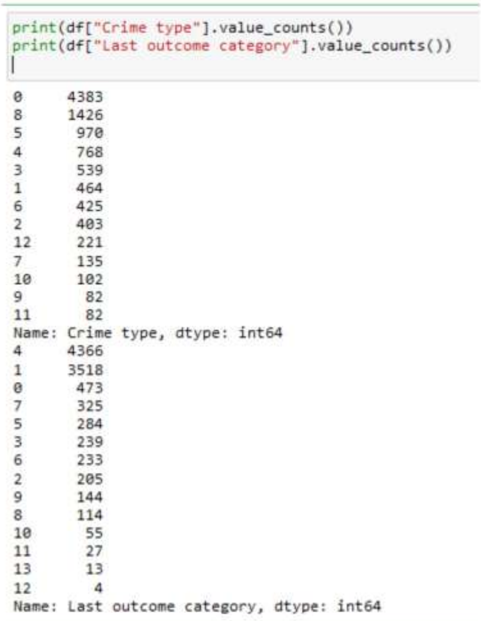
```
<AxesSubplot:xlabel='Last outcome category', ylabel='count'>
```

## 12. Here, we print the unique values of the "Crime type" and "Last outcome category, later we count total values for each column names selected



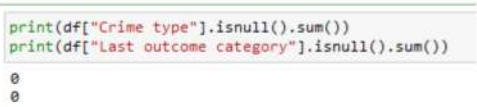
```
print(df["Crime type"].unique())
print(df["Last outcome category"].unique())
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12]
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13]
```



```
print(df["Crime type"].value_counts())
print(df["Last outcome category"].value_counts())
|
```

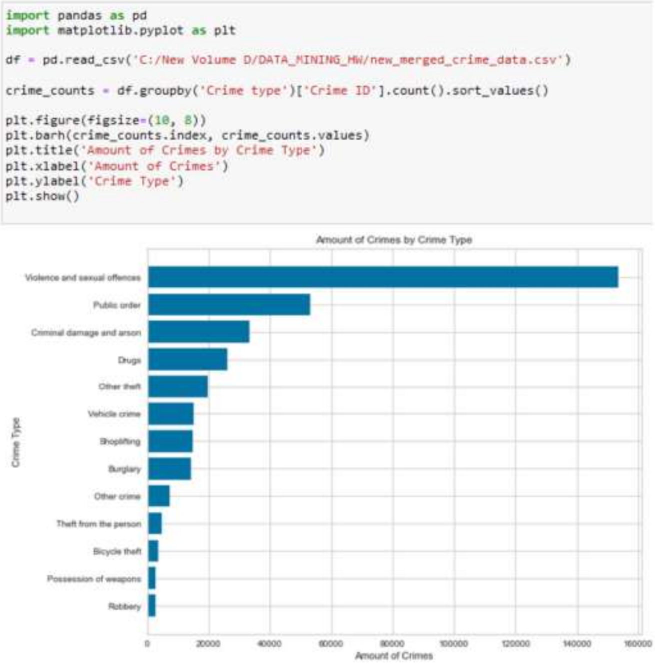
```
0     4383
8     1426
5      970
4      768
3      539
1      464
6      425
2      403
12     221
7      135
10     102
9       82
11      82
Name: Crime type, dtype: int64
4     4366
1     3518
0      473
7      325
5      284
3      239
6      233
2      205
9      144
8      114
10      55
11      27
13      13
12       4
Name: Last outcome category, dtype: int64
```



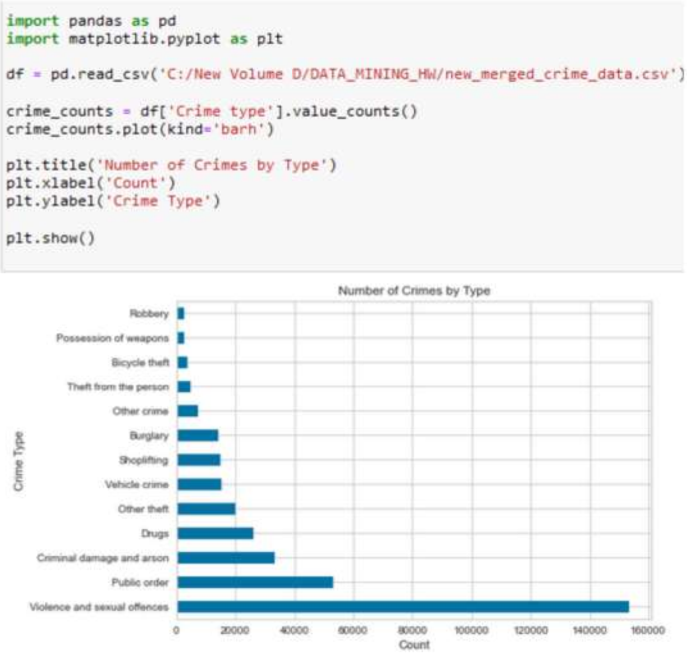
```
print(df["Crime type"].isnull().sum())
print(df["Last outcome category"].isnull().sum())
```

```
0
0
```

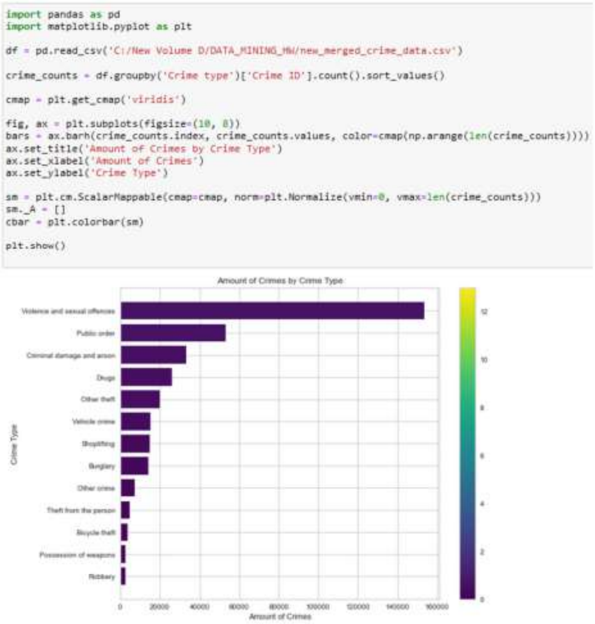
## 13. Creating a horizontal bar plot for ease of visualization



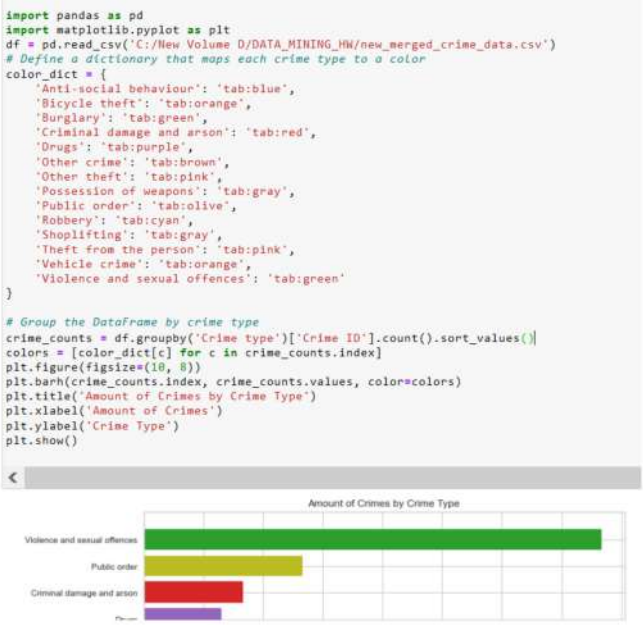
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

crime_counts = df.groupby('Crime type')['Crime ID'].count().sort_values()

plt.figure(figsize=(10, 8))
plt.barh(crime_counts.index, crime_counts.values)
plt.title('Amount of Crimes by Crime Type')
plt.xlabel('Amount of Crimes')
plt.ylabel('Crime Type')
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

crime_counts = df['Crime type'].value_counts()
crime_counts.plot(kind='barh')

plt.title('Number of Crimes by Type')
plt.xlabel('Count')
plt.ylabel('Crime Type')

plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

crime_counts = df.groupby('Crime type')['Crime ID'].count().sort_values()

cmap = plt.get_cmap('viridis')

fig, ax = plt.subplots(figsize=(10, 8))
bars = ax.barh(crime_counts.index, crime_counts.values, color=cmap(np.arange(len(crime_counts))))
ax.set_title('Amount of Crimes by Crime Type')
ax.set_xlabel('Amount of Crimes')
ax.set_ylabel('Crime Type')

sm = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=0, vmax=len(crime_counts)))
sm._A = []
cbar = plt.colorbar(sm)

plt.show()
```

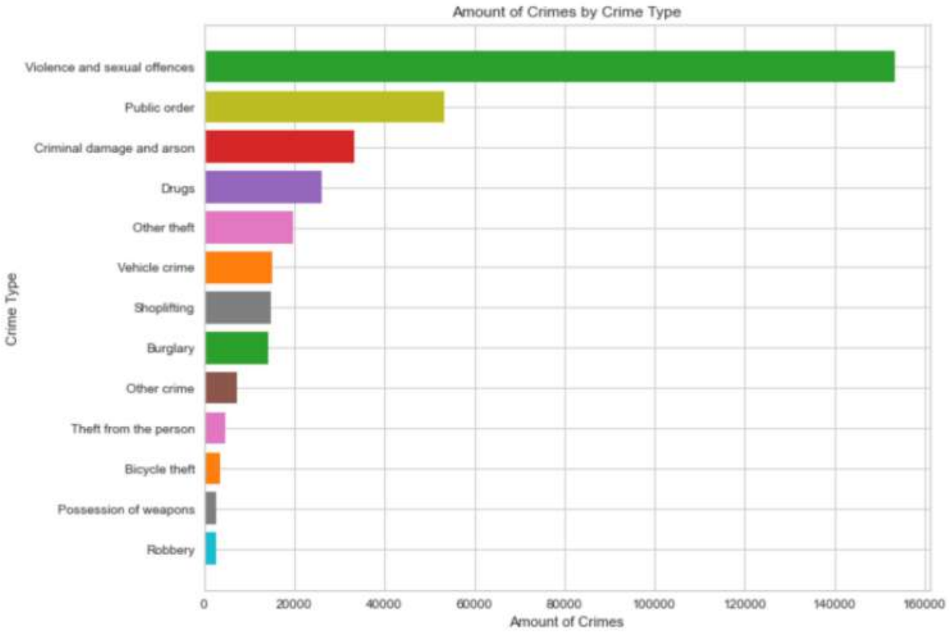


```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')
# Define a dictionary that maps each crime type to a color
color_dict = {
    'Anti-social behaviour': 'tab:blue',
    'Bicycle theft': 'tab:orange',
    'Burglary': 'tab:green',
    'Criminal damage and arson': 'tab:red',
    'Drugs': 'tab:purple',
    'Other crime': 'tab:brown',
    'Other theft': 'tab:pink',
    'Possession of weapons': 'tab:gray',
    'Public order': 'tab:olive',
    'Robbery': 'tab:cyan',
    'Shoplifting': 'tab:gray',
    'Theft from the person': 'tab:pink',
    'Vehicle crime': 'tab:orange',
    'Violence and sexual offences': 'tab:green'
}

# Group the DataFrame by crime type
crime_counts = df.groupby('Crime type')['Crime ID'].count().sort_values()
colors = [color_dict[c] for c in crime_counts.index]
plt.figure(figsize=(10, 8))
plt.barh(crime_counts.index, crime_counts.values, color=colors)
plt.title('Amount of Crimes by Crime Type')
plt.xlabel('Amount of Crimes')
plt.ylabel('Crime Type')
plt.show()
```

Amount of Crimes by Crime Type
Violence and sexual offences
Public order
Criminal damage and arson

# 14. WAYS OF CREATING A BAR PLOT

Here we used dictionary that maps each crime type to a color, groups the DataFrame by crime type and counts the number of occurrences of each crime type, and creates a horizontal bar plot with different colors for each crime type:

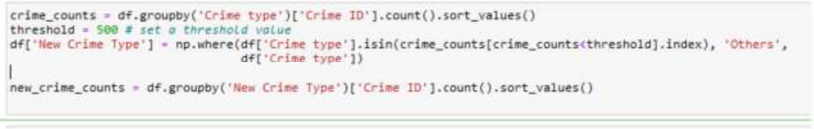


METHOD 2:

Line 1 : Grouping the df DataFrame by the Crime type column and counting the number of occurrences of each crime type using the count()

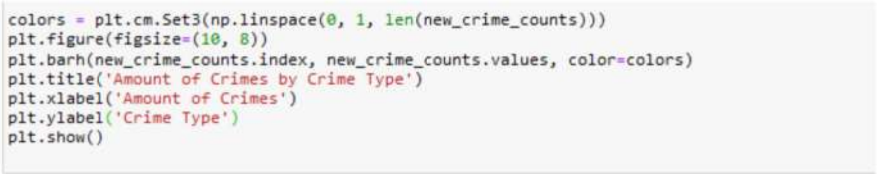
Line 2 & 3 : The values in this column 'New Crime Type'are determined by the values in the 'Crime type' column, where if the count of a particular crime type is less than the threshold value of 500, the value in the 'New Crime Type' column will be 'Others'. If the count of the crime type is greater than or equal to the threshold value, the value in the 'New Crime Type' column will be the same as the value in the 'Crime type' column.

Line 4 : This new DataFrame can be used to plot the counts of the new crime types.'''

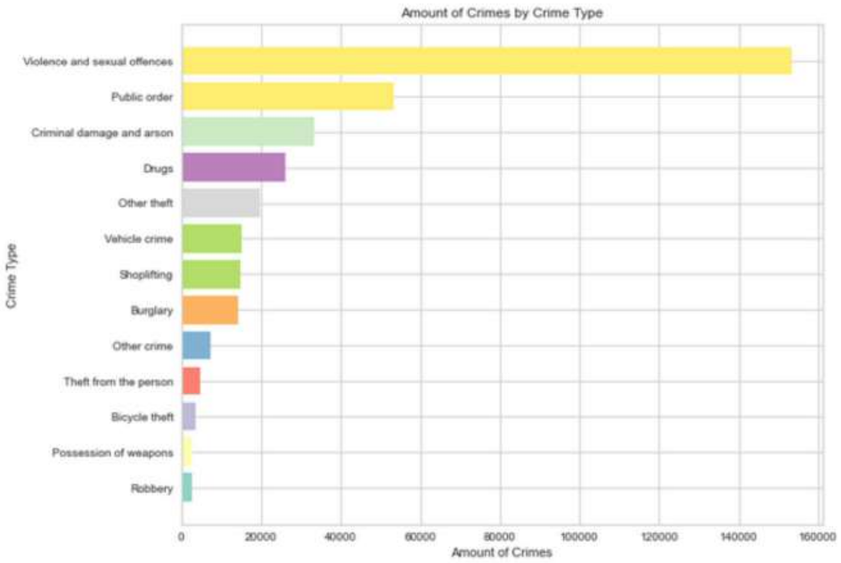


```
crime_counts = df.groupby('Crime type')['Crime ID'].count().sort_values()
threshold = 500 # set a threshold value
df['New Crime Type'] = np.where(df['Crime type'].isin(crime_counts[crime_counts<threshold].index), 'Others',
                                df['Crime type'])

new_crime_counts = df.groupby('New Crime Type')['Crime ID'].count().sort_values()
```

'plt.cm.Set3’ function generates a color map with a range of colors, and the np.linspace function is used to evenly divide the range of colors

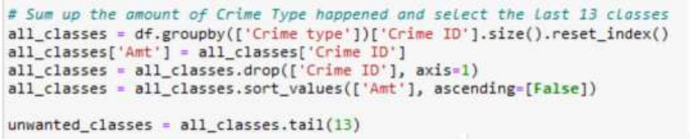


```
colors = plt.cm.Set3(np.linspace(0, 1, len(new_crime_counts)))
plt.figure(figsize=(10, 8))
plt.barh(new_crime_counts.index, new_crime_counts.values, color=colors)
plt.title('Amount of Crimes by Crime Type')
plt.xlabel('Amount of Crimes')
plt.ylabel('Crime Type')
plt.show()
```

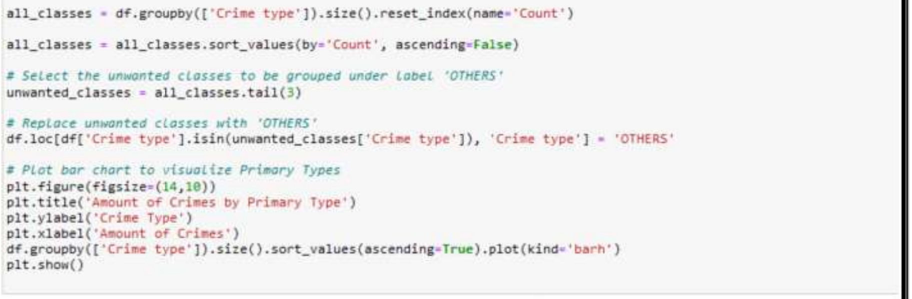


## METHOD 3:

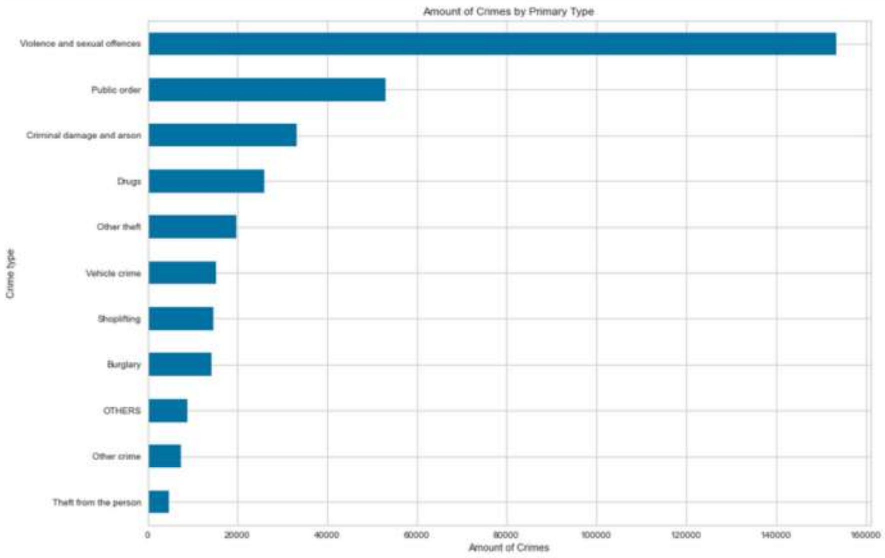
Doing this before the new more sorted bar plot. Sorting the DataFrame in descending order of 'Amt' using all_classes.sort_values Then selecting the last 13 classes using & assigns them to unwanted_classes variable.



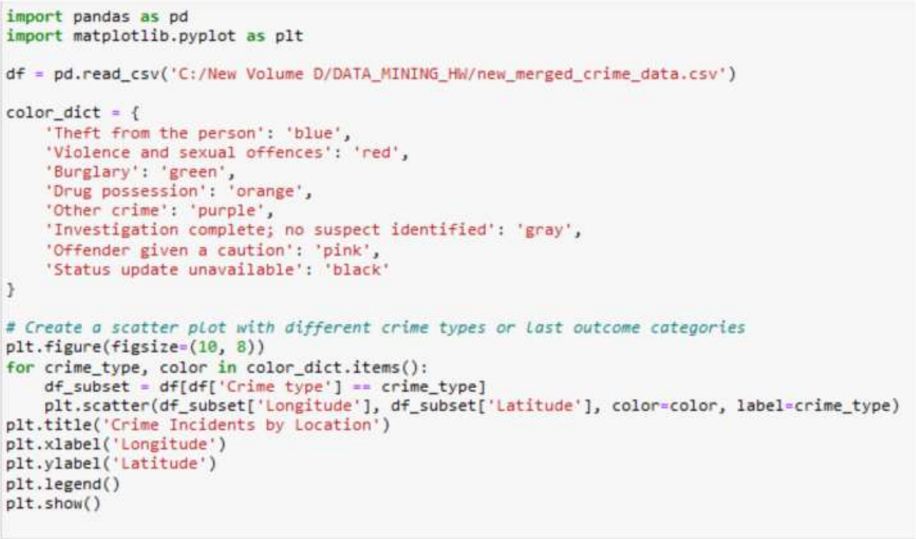
```
# Sum up the amount of Crime Type happened and select the last 13 classes
all_classes = df.groupby(['Crime type'])['Crime ID'].size().reset_index()
all_classes['Amt'] = all_classes['Crime ID']
all_classes = all_classes.drop(['Crime ID'], axis=1)
all_classes = all_classes.sort_values(['Amt'], ascending=[False])

unwanted_classes = all_classes.tail(13)
```



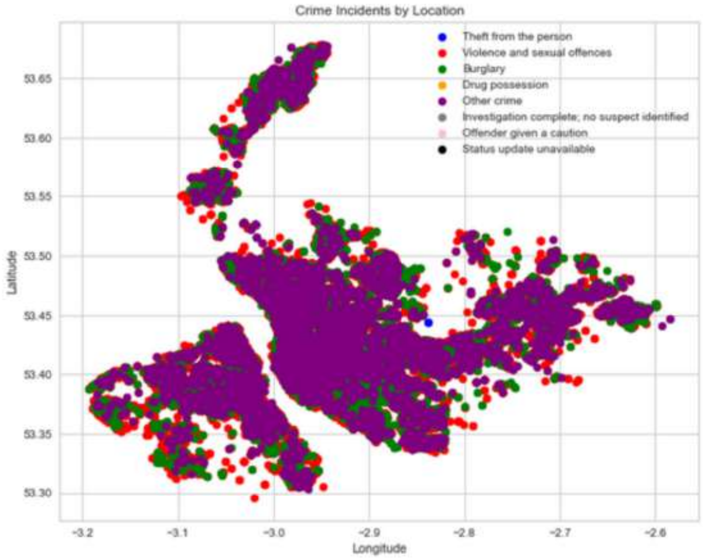
```
all_classes = df.groupby(['Crime type']).size().reset_index(name='Count')

all_classes = all_classes.sort_values(by='Count', ascending=False)

# Select the unwanted classes to be grouped under label 'OTHERS'
unwanted_classes = all_classes.tail(3)

# Replace unwanted classes with 'OTHERS'
df.loc[df['Crime type'].isin(unwanted_classes['Crime type']), 'Crime type'] = 'OTHERS'

# Plot bar chart to visualize Primary Types
plt.figure(figsize=(14,10))
plt.title('Amount of Crimes by Primary Type')
plt.ylabel('Crime Type')
plt.xlabel('Amount of Crimes')
df.groupby(['Crime type']).size().sort_values(ascending=True).plot(kind='barh')
plt.show()
```



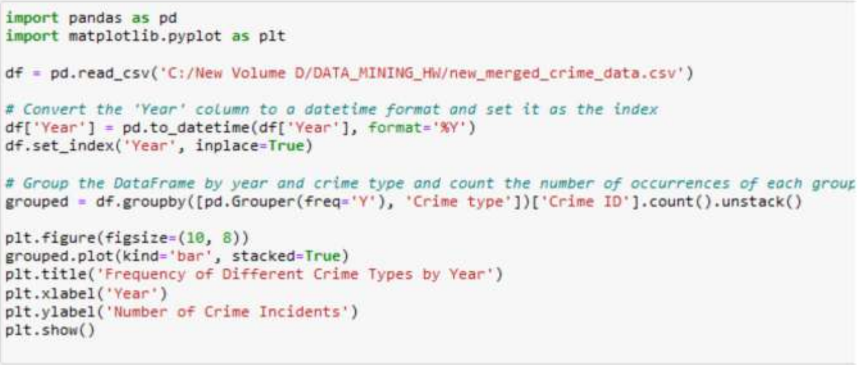
## 15. Creating a scatter plot with different crime types



```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

color_dict = {
    'Theft from the person': 'blue',
    'Violence and sexual offences': 'red',
    'Burglary': 'green',
    'Drug possession': 'orange',
    'Other crime': 'purple',
    'Investigation complete; no suspect identified': 'gray',
    'Offender given a caution': 'pink',
    'Status update unavailable': 'black'
}

# Create a scatter plot with different crime types or last outcome categories
plt.figure(figsize=(10, 8))
for crime_type, color in color_dict.items():
    df_subset = df[df['Crime type'] == crime_type]
    plt.scatter(df_subset['Longitude'], df_subset['Latitude'], color=color, label=crime_type)
plt.title('Crime Incidents by Location')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend()
plt.show()
```

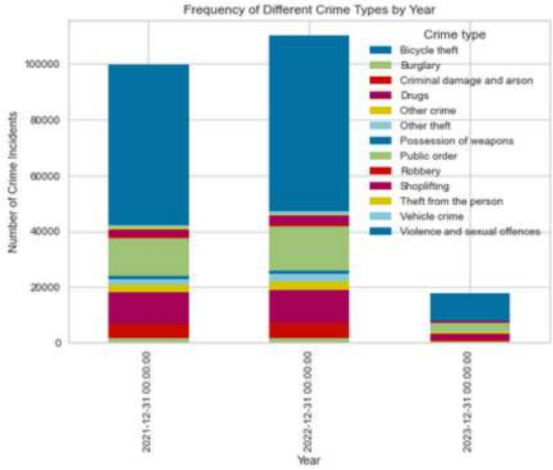


## 16. Grouped bar plot from year 2021 to 2023

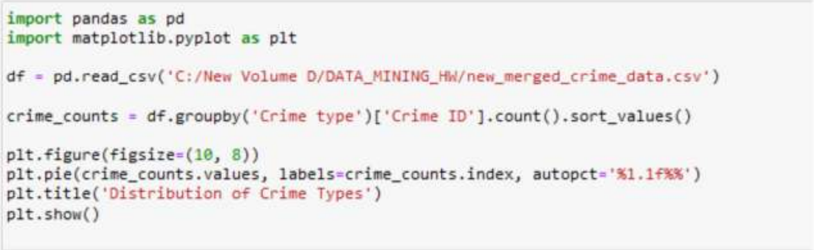


```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Convert the 'Year' column to a datetime format and set it as the index
df['Year'] = pd.to_datetime(df['Year'], format='%Y')
df.set_index('Year', inplace=True)

# Group the DataFrame by year and crime type and count the number of occurrences of each grouf
grouped = df.groupby([pd.Grouper(freq='Y'), 'Crime type'])['Crime ID'].count().unstack()

plt.figure(figsize=(10, 8))
grouped.plot(kind='bar', stacked=True)
plt.title('Frequency of Different Crime Types by Year')
plt.xlabel('Year')
plt.ylabel('Number of Crime Incidents')
plt.show()
```

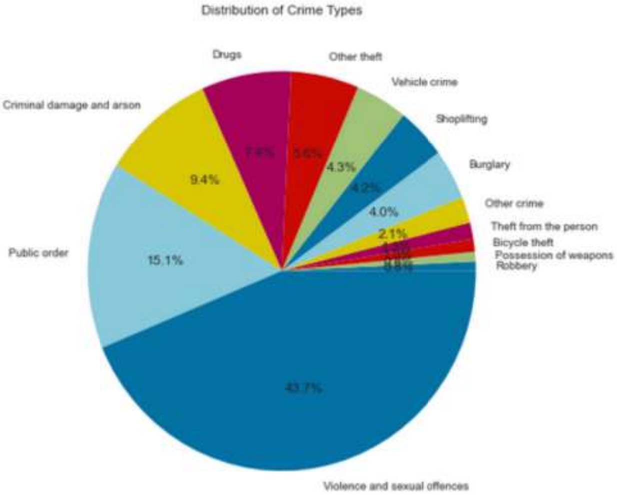
```
<Figure size 720x576 with 0 Axes>
```



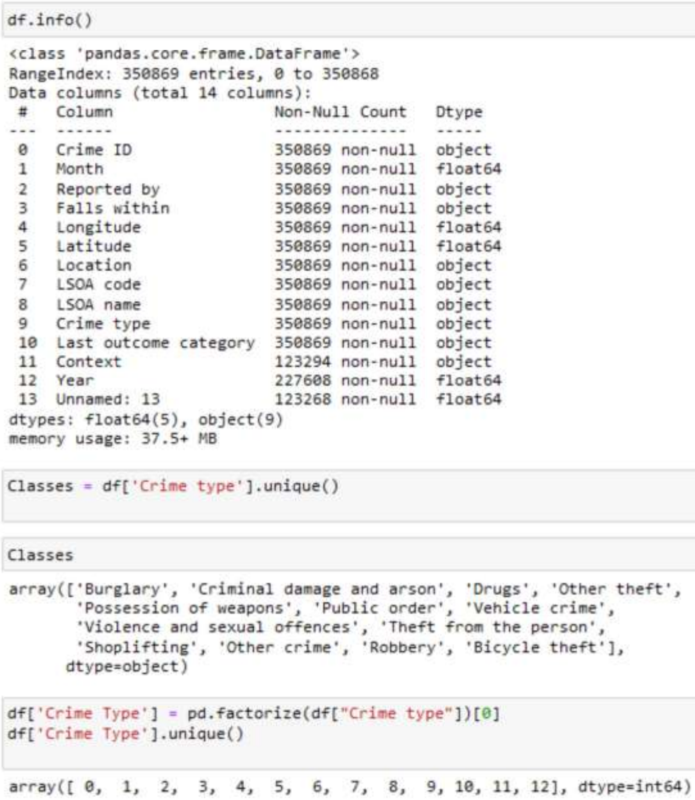
## 17. Pie Chart



```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

crime_counts = df.groupby('Crime type')['Crime ID'].count().sort_values()

plt.figure(figsize=(10, 8))
plt.pie(crime_counts.values, labels=crime_counts.index, autopct='%1.1f%%')
plt.title('Distribution of Crime Types')
plt.show()
```



## 18. Listing the unique classes



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350869 entries, 0 to 350868
Data columns (total 14 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   Crime ID               350869 non-null  object 
 1   Month                  350869 non-null  float64
 2   Reported by            350869 non-null  object 
 3   Falls within           350869 non-null  object 
 4   Longitude              350869 non-null  float64
 5   Latitude               350869 non-null  float64
 6   Location               350869 non-null  object 
 7   LSOA code              350869 non-null  object 
 8   LSOA name              350869 non-null  object 
 9   Crime type             350869 non-null  object 
 10  Last outcome category  350869 non-null  object 
 11  Context                123294 non-null  object 
 12  Year                   227608 non-null  float64
 13  Unnamed: 13            123268 non-null  float64
dtypes: float64(5), object(9)
memory usage: 37.5+ MB
```

```
Classes = df['Crime type'].unique()
```

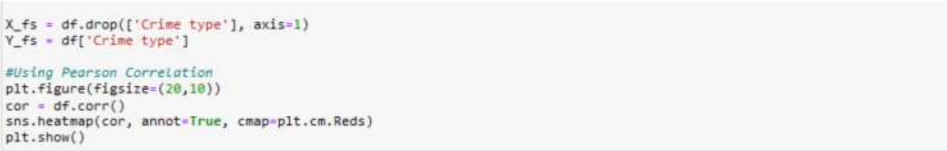
```
Classes
```

```
array(['Burglary', 'Criminal damage and arson', 'Drugs', 'Other theft',
       'Possession of weapons', 'Public order', 'Vehicle crime',
       'Violence and sexual offences', 'Theft from the person',
       'Shoplifting', 'Other crime', 'Robbery', 'Bicycle theft'],
      dtype=object)
```

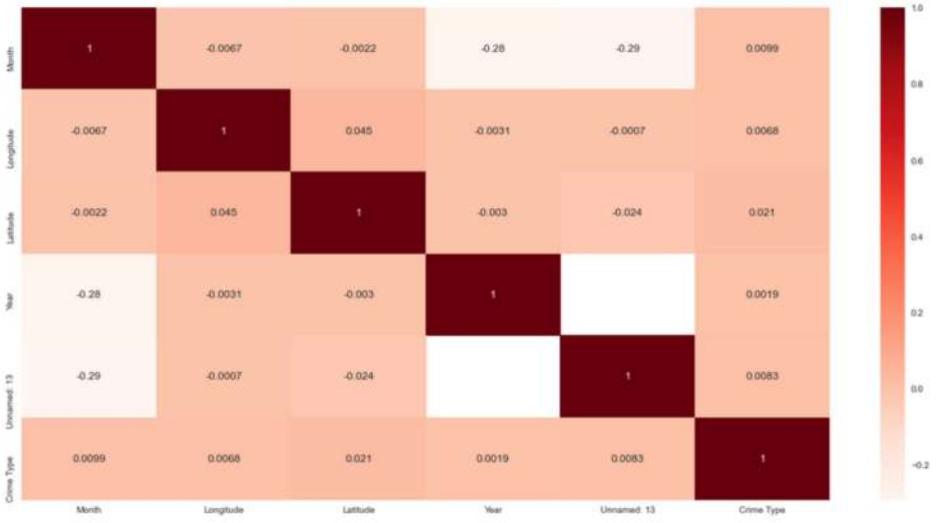
```
df['Crime Type'] = pd.factorize(df["Crime type"])[0]
df['Crime Type'].unique()
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12], dtype=int64)
```

## 19. HeatMap

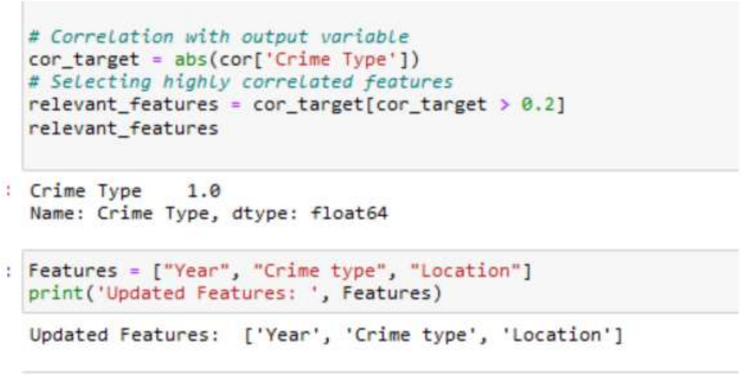


```
X_fs = df.drop(['Crime type'], axis=1)
Y_fs = df['Crime type']

#Using Pearson Correlation
plt.figure(figsize=(20,10))
cor = df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



## 20. Implementing feature selection based on correlation with the target variable "Crime Type".

This helps in identifying features that have a strong linear correlation with the target variable "Crime Type" using a correlation threshold of 0.2



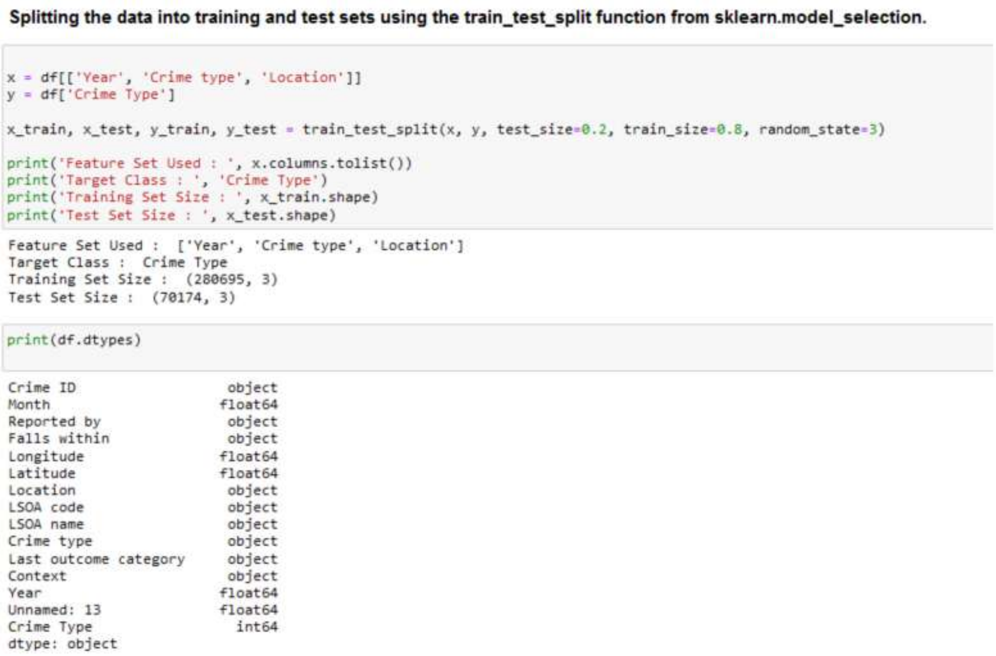
```
# Correlation with output variable
cor_target = abs(cor['Crime Type'])
# Selecting highly correlated features
relevant_features = cor_target[cor_target > 0.2]
relevant_features
```

```
: Crime Type    1.0
  Name: Crime Type, dtype: float64
```

```
: Features = ["Year", "Crime type", "Location"]
  print('Updated Features: ', Features)
```

```
Updated Features:  ['Year', 'Crime type', 'Location']
```

## 21. BUILDING A PREDICTIVE MODEL by :



**Splitting the data into training and test sets using the train_test_split function from sklearn.model_selection.**

```
x = df[['Year', 'Crime type', 'Location']]
y = df['Crime Type']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, train_size=0.8, random_state=3)

print('Feature Set Used : ', x.columns.tolist())
print('Target Class : ', 'Crime Type')
print('Training Set Size : ', x_train.shape)
print('Test Set Size : ', x_test.shape)
```

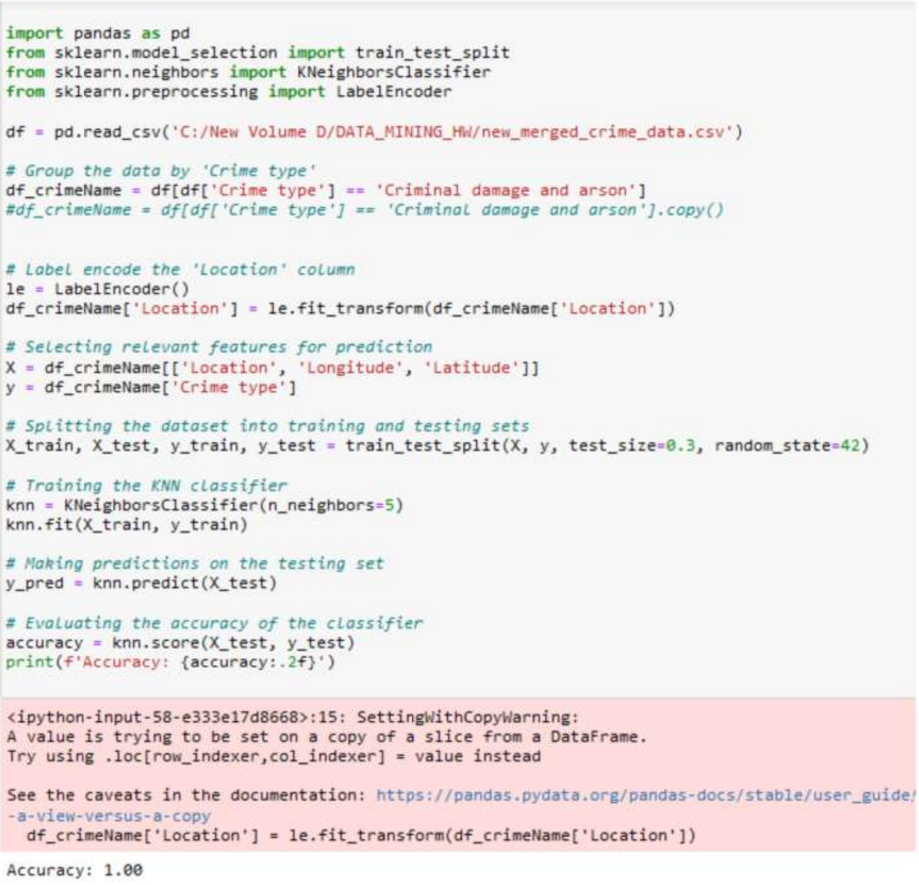
```
Feature Set Used :  ['Year', 'Crime type', 'Location']
Target Class :  Crime Type
Training Set Size :  (280695, 3)
Test Set Size :  (70174, 3)
```

```
print(df.dtypes)
```

```
Crime ID                  object
Month                    float64
Reported by               object
Falls within              object
Longitude                float64
Latitude                 float64
Location                  object
LSOA code                 object
LSOA name                 object
Crime type                object
Last outcome category     object
Context                   object
Year                     float64
Unnamed: 13              float64
Crime Type                 int64
dtype: object
```

The training set size is set to 80% (train_size=0.8) and the test set size is set to 20% (test_size=0.2). Selecting a subset of features and the target variable from the df & splitting the data . The resulting feature set and target variable can be used for building a predictive model for crime type classification.

## 22. Making predictions on the testing set & Evaluating the accuracy of the classifier



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Group the data by 'Crime type'
df_crimeName = df[df['Crime type'] == 'Criminal damage and arson']
#df_crimeName = df[df['Crime type'] == 'Criminal damage and arson'].copy()


# Label encode the 'Location' column
le = LabelEncoder()
df_crimeName['Location'] = le.fit_transform(df_crimeName['Location'])

# Selecting relevant features for prediction
X = df_crimeName[['Location', 'Longitude', 'Latitude']]
y = df_crimeName['Crime type']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Training the KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Making predictions on the testing set
y_pred = knn.predict(X_test)

# Evaluating the accuracy of the classifier
accuracy = knn.score(X_test, y_test)
print(f'Accuracy: {accuracy:.2f}')
```

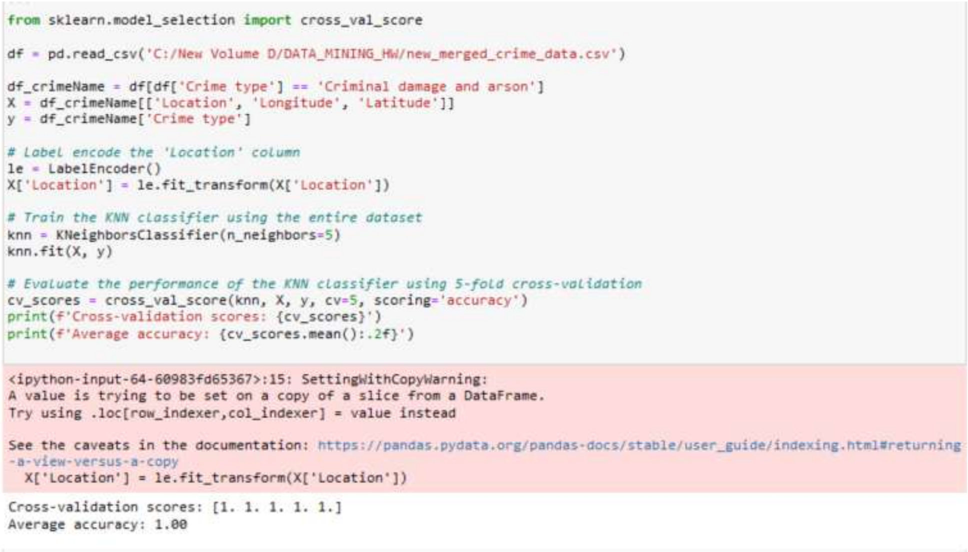
```
<ipython-input-58-e333e17d8668>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/
-a-view-versus-a-copy
  df_crimeName['Location'] = le.fit_transform(df_crimeName['Location'])
```

```
Accuracy: 1.00
```

## 23. Evaluating the performance of the KNN classifier using 5-fold cross-validation.

(There can be a case where model may be overfitting ie. it’s possible that model will perform good on training set but not on test set. And that is why we need to cross-validate the model by either precison, F1-Score, Recall rather than accuracy. The results will reveall how well the model performs)

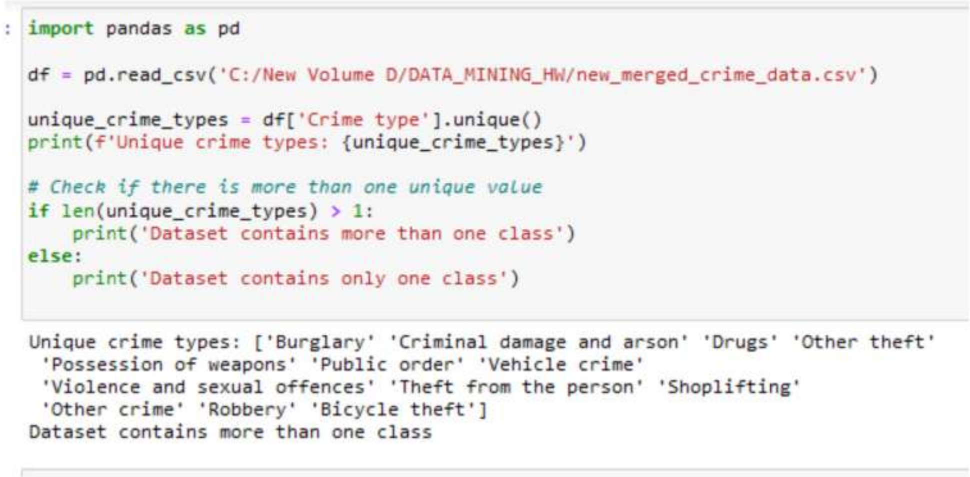
(The function spits the dataset into k folds & evaluates it on each fold)



```
from sklearn.model_selection import cross_val_score

df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

df_crimeName = df[df['Crime type'] == 'Criminal damage and arson']
X = df_crimeName[['Location', 'Longitude', 'Latitude']]
y = df_crimeName['Crime type']

# Label encode the 'Location' column
le = LabelEncoder()
X['Location'] = le.fit_transform(X['Location'])

# Train the KNN classifier using the entire dataset
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X, y)

# Evaluate the performance of the KNN classifier using 5-fold cross-validation
cv_scores = cross_val_score(knn, X, y, cv=5, scoring='accuracy')
print(f'Cross-validation scores: {cv_scores}')
print(f'Average accuracy: {cv_scores.mean():.2f}')
```

```
<ipython-input-64-60983fd65367>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  X['Location'] = le.fit_transform(X['Location'])
Cross-validation scores: [1. 1. 1. 1. 1.]
Average accuracy: 1.00
```

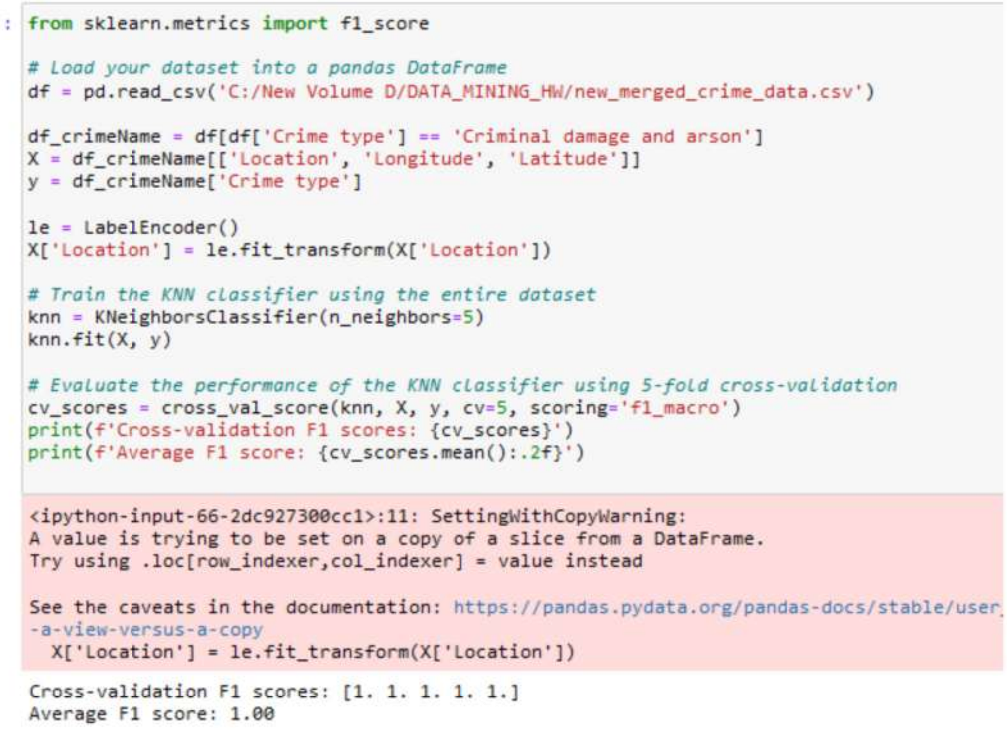
## 24. Unique crime types



```
: import pandas as pd

  df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

  unique_crime_types = df['Crime type'].unique()
  print(f'Unique crime types: {unique_crime_types}')

  # Check if there is more than one unique value
  if len(unique_crime_types) > 1:
      print('Dataset contains more than one class')
  else:
      print('Dataset contains only one class')
```

```
Unique crime types: ['Burglary' 'Criminal damage and arson' 'Drugs' 'Other theft'
 'Possession of weapons' 'Public order' 'Vehicle crime'
 'Violence and sexual offences' 'Theft from the person' 'Shoplifting'
 'Other crime' 'Robbery' 'Bicycle theft']
Dataset contains more than one class
```

## 25. Checking the F1 score of the model

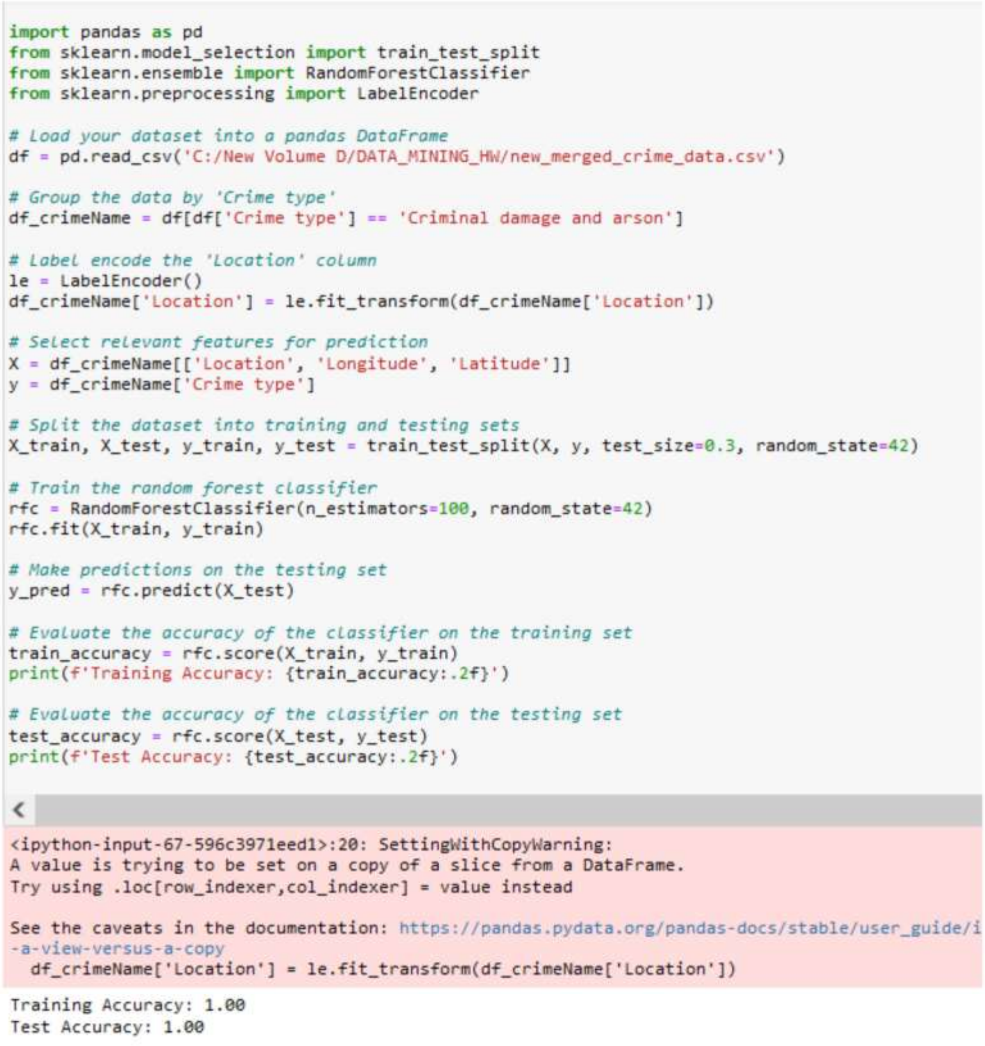


```
from sklearn.metrics import f1_score

# Load your dataset into a pandas DataFrame
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

df_crimeName = df[df['Crime type'] == 'Criminal damage and arson']
X = df_crimeName[['Location', 'Longitude', 'Latitude']]
y = df_crimeName['Crime type']

le = LabelEncoder()
X['Location'] = le.fit_transform(X['Location'])

# Train the KNN classifier using the entire dataset
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X, y)

# Evaluate the performance of the KNN classifier using 5-fold cross-validation
cv_scores = cross_val_score(knn, X, y, cv=5, scoring='f1_macro')
print(f'Cross-validation F1 scores: {cv_scores}')
print(f'Average F1 score: {cv_scores.mean():.2f}')
```

```
<ipython-input-66-2dc927300cc1>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
-a-view-versus-a-copy
  X['Location'] = le.fit_transform(X['Location'])
```

```
Cross-validation F1 scores: [1. 1. 1. 1. 1.]
Average F1 score: 1.00
```

## 26. Using a different classifier to check for overfitting like Random Forest

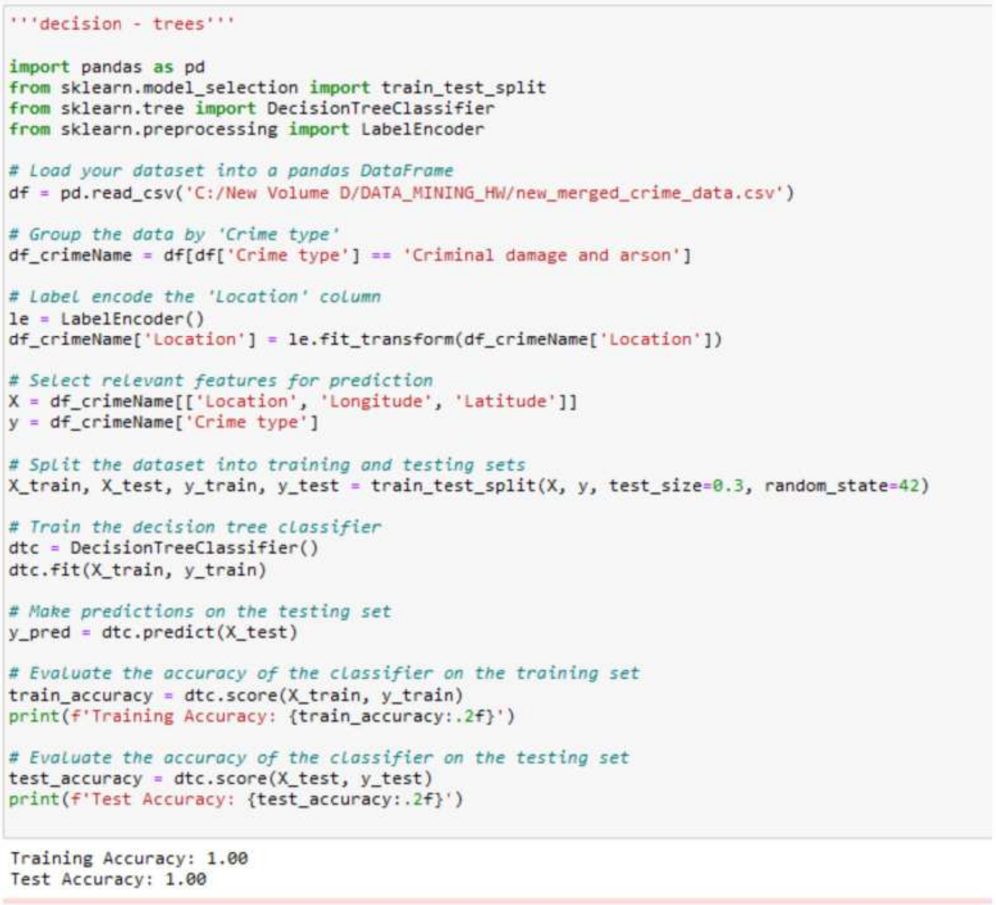


```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder

# Load your dataset into a pandas DataFrame
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Group the data by 'Crime type'
df_crimeName = df[df['Crime type'] == 'Criminal damage and arson']

# Label encode the 'Location' column
le = LabelEncoder()
df_crimeName['Location'] = le.fit_transform(df_crimeName['Location'])

# Select relevant features for prediction
X = df_crimeName[['Location', 'Longitude', 'Latitude']]
y = df_crimeName['Crime type']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train the random forest classifier
rfc = RandomForestClassifier(n_estimators=100, random_state=42)
rfc.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = rfc.predict(X_test)

# Evaluate the accuracy of the classifier on the training set
train_accuracy = rfc.score(X_train, y_train)
print(f'Training Accuracy: {train_accuracy:.2f}')

# Evaluate the accuracy of the classifier on the testing set
test_accuracy = rfc.score(X_test, y_test)
print(f'Test Accuracy: {test_accuracy:.2f}')
```

```
<ipython-input-67-596c3971eed1>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
-a-view-versus-a-copy
  df_crimeName['Location'] = le.fit_transform(df_crimeName['Location'])
```

```
Training Accuracy: 1.00
Test Accuracy: 1.00
```

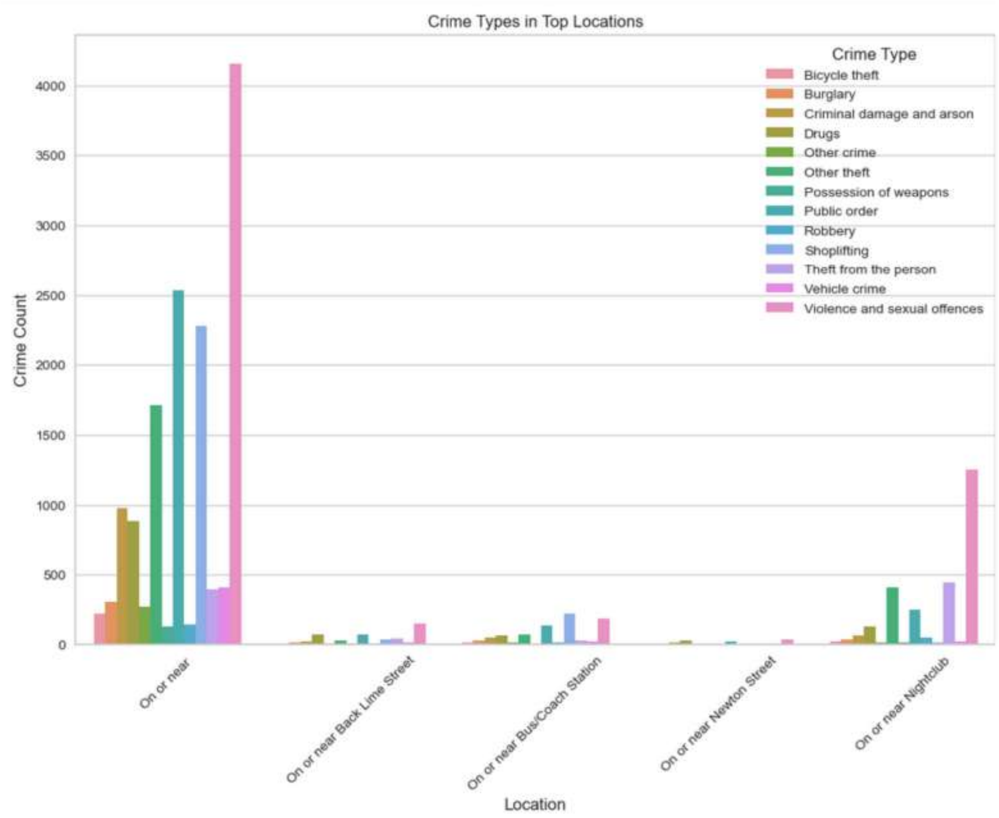
## 27. Using a different classifier to check for overfitting like Decision Trees



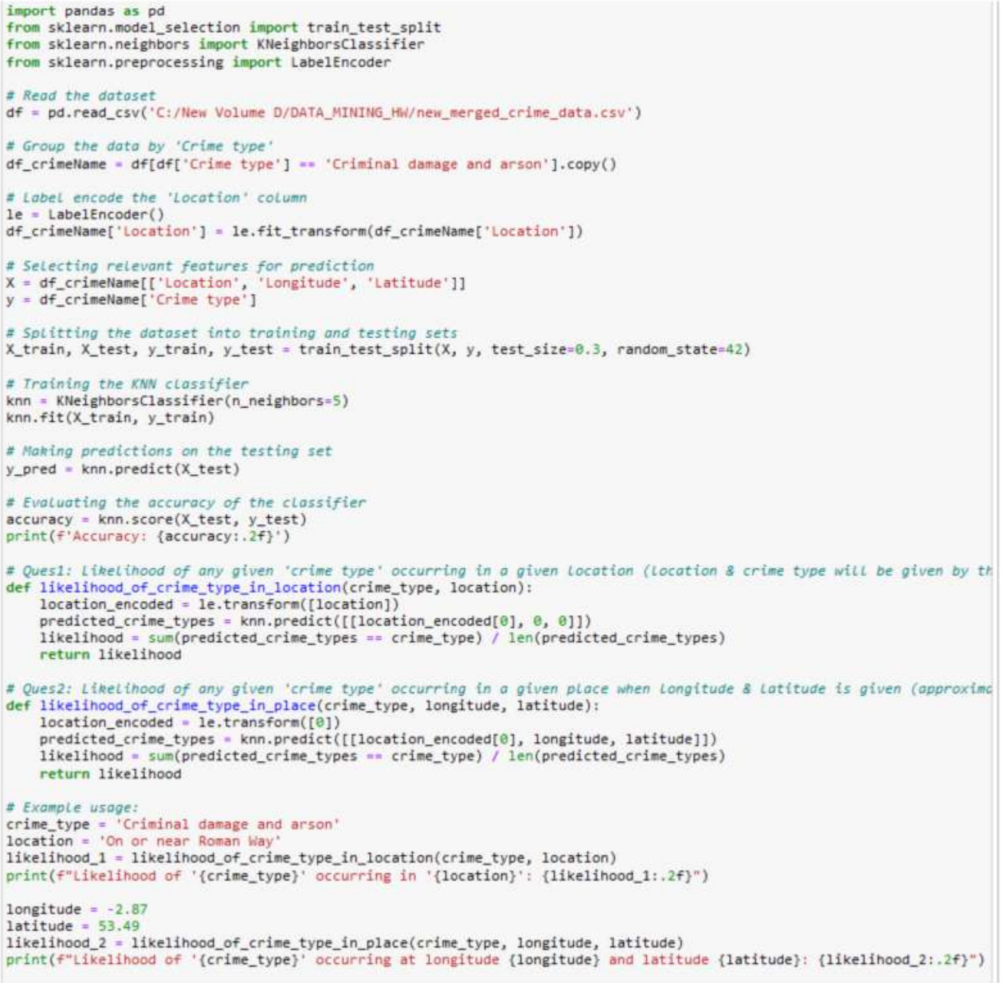
```
'''decision - trees'''

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder

# Load your dataset into a pandas DataFrame
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Group the data by 'Crime type'
df_crimeName = df[df['Crime type'] == 'Criminal damage and arson']

# Label encode the 'Location' column
le = LabelEncoder()
df_crimeName['Location'] = le.fit_transform(df_crimeName['Location'])

# Select relevant features for prediction
X = df_crimeName[['Location', 'Longitude', 'Latitude']]
y = df_crimeName['Crime type']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train the decision tree classifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = dtc.predict(X_test)

# Evaluate the accuracy of the classifier on the training set
train_accuracy = dtc.score(X_train, y_train)
print(f'Training Accuracy: {train_accuracy:.2f}')

# Evaluate the accuracy of the classifier on the testing set
test_accuracy = dtc.score(X_test, y_test)
print(f'Test Accuracy: {test_accuracy:.2f}')
```

```
Training Accuracy: 1.00
Test Accuracy: 1.00
```

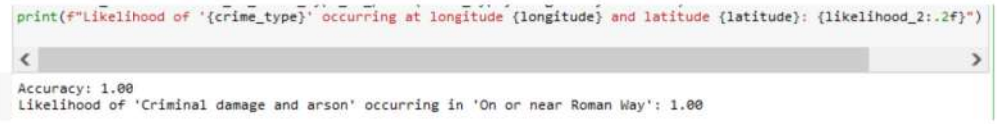
## 28. Location vs Crime count including the list of top crime types



## 29. Likelihood of any given 'crime type' occurring in a given location along with accuracy



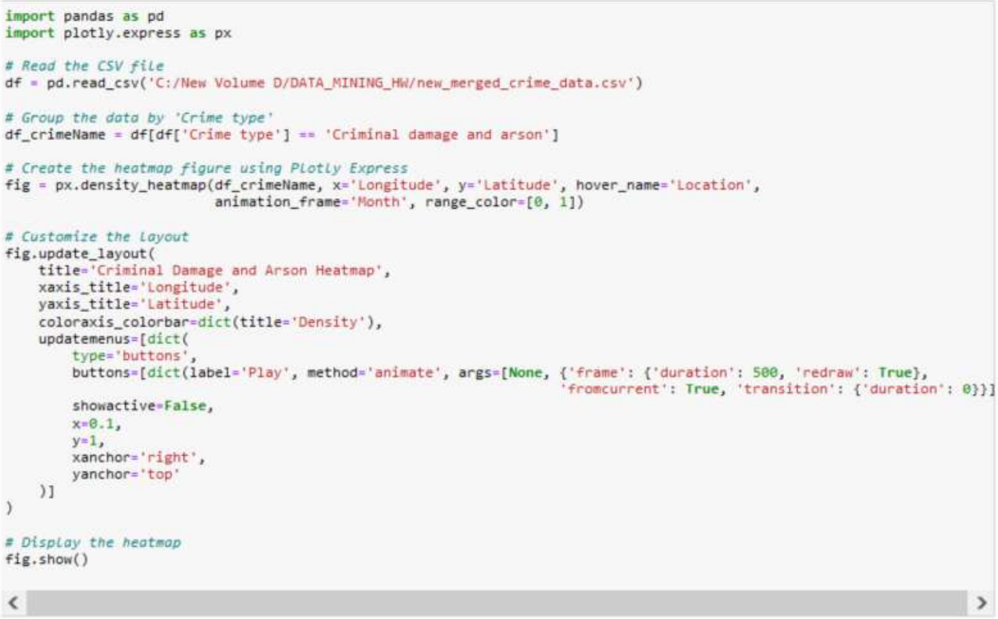
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder

# Read the dataset
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Group the data by 'Crime type'
df_crimeName = df[df['Crime type'] == 'Criminal damage and arson'].copy()

# Label encode the 'Location' column
le = LabelEncoder()
df_crimeName['Location'] = le.fit_transform(df_crimeName['Location'])

# Selecting relevant features for prediction
X = df_crimeName[['Location', 'Longitude', 'Latitude']]
y = df_crimeName['Crime type']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Training the KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Making predictions on the testing set
y_pred = knn.predict(X_test)

# Evaluating the accuracy of the classifier
accuracy = knn.score(X_test, y_test)
print(f'Accuracy: {accuracy:.2f}')

# Ques1: Likelihood of any given 'crime type' occurring in a given location (location & crime type will be given by th
def likelihood_of_crime_type_in_location(crime_type, location):
    location_encoded = le.transform([location])
    predicted_crime_types = knn.predict([[location_encoded[0], 0, 0]])
    likelihood = sum(predicted_crime_types == crime_type) / len(predicted_crime_types)
    return likelihood

# Ques2: Likelihood of any given 'crime type' occurring in a given place when longitude & latitude is given (approxima
def likelihood_of_crime_type_in_place(crime_type, longitude, latitude):
    location_encoded = le.transform([0])
    predicted_crime_types = knn.predict([[location_encoded[0], longitude, latitude]])
    likelihood = sum(predicted_crime_types == crime_type) / len(predicted_crime_types)
    return likelihood

# Example usage:
crime_type = 'Criminal damage and arson'
location = 'On or near Roman Way'
likelihood_1 = likelihood_of_crime_type_in_location(crime_type, location)
print(f"Likelihood of '{crime_type}' occurring in '{location}': {likelihood_1:.2f}")

longitude = -2.87
latitude = 53.49
likelihood_2 = likelihood_of_crime_type_in_place(crime_type, longitude, latitude)
print(f"Likelihood of '{crime_type}' occurring at longitude {longitude} and latitude {latitude}: {likelihood_2:.2f}")
```



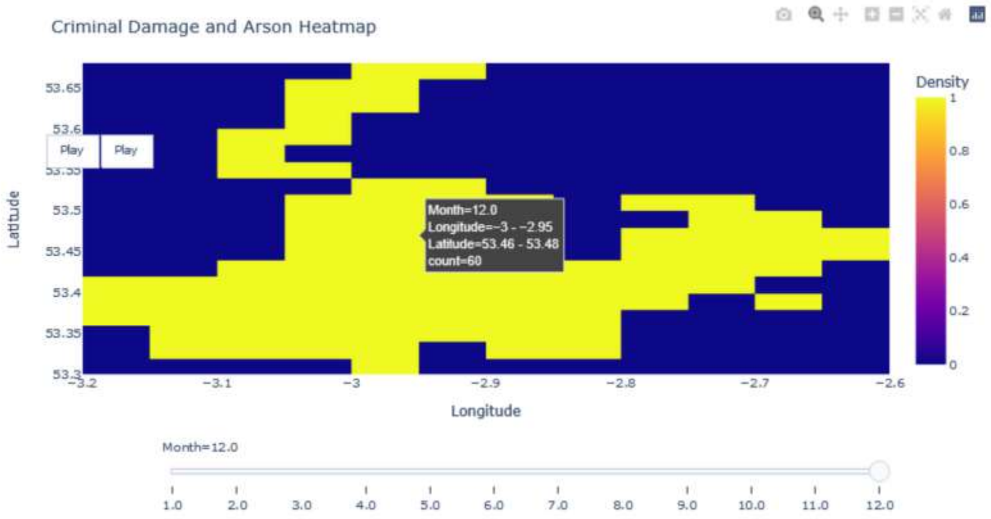
```python
print(f"Likelihood of '{crime_type}' occurring at longitude {longitude} and latitude {latitude}: {likelihood_2:.2f}")
```

```
Accuracy: 1.00
Likelihood of 'Criminal damage and arson' occurring in 'On or near Roman Way': 1.00
```

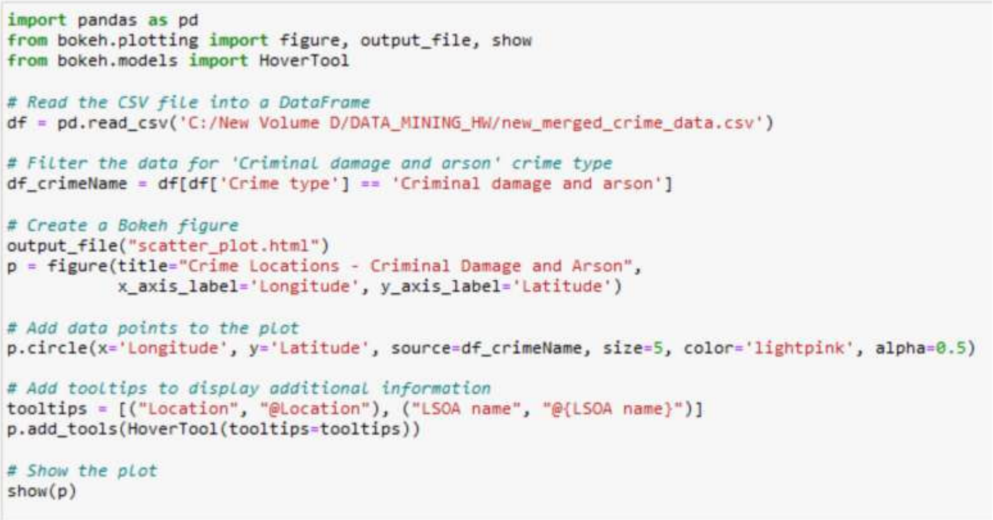
# 30. Interactive plotly heatmap



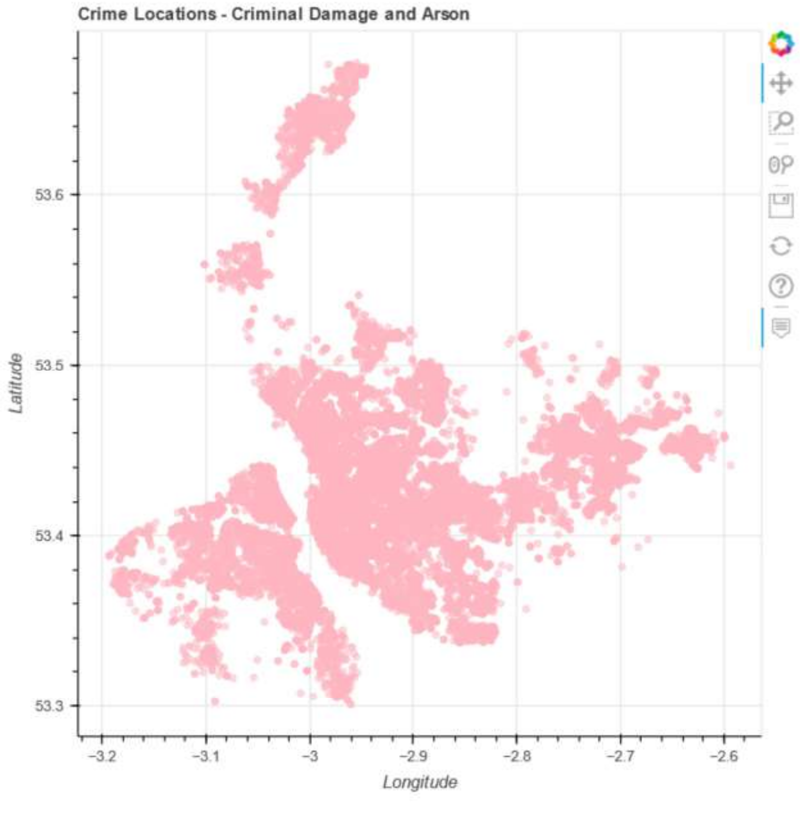
```
import pandas as pd
import plotly.express as px

# Read the CSV file
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Group the data by 'Crime type'
df_crimeName = df[df['Crime type'] == 'Criminal damage and arson']

# Create the heatmap figure using Plotly Express
fig = px.density_heatmap(df_crimeName, x='Longitude', y='Latitude', hover_name='Location',
                         animation_frame='Month', range_color=[0, 1])

# Customize the Layout
fig.update_layout(
    title='Criminal Damage and Arson Heatmap',
    xaxis_title='Longitude',
    yaxis_title='Latitude',
    coloraxis_colorbar=dict(title='Density'),
    updatemenus=[dict(
        type='buttons',
        buttons=[dict(label='Play', method='animate', args=[None, {'frame': {'duration': 500, 'redraw': True},
                                                          'fromcurrent': True, 'transition': {'duration': 0}}]
        showactive=False,
        x=0.1,
        y=1,
        xanchor='right',
        yanchor='top'
    )]
)

# Display the heatmap
fig.show()
```

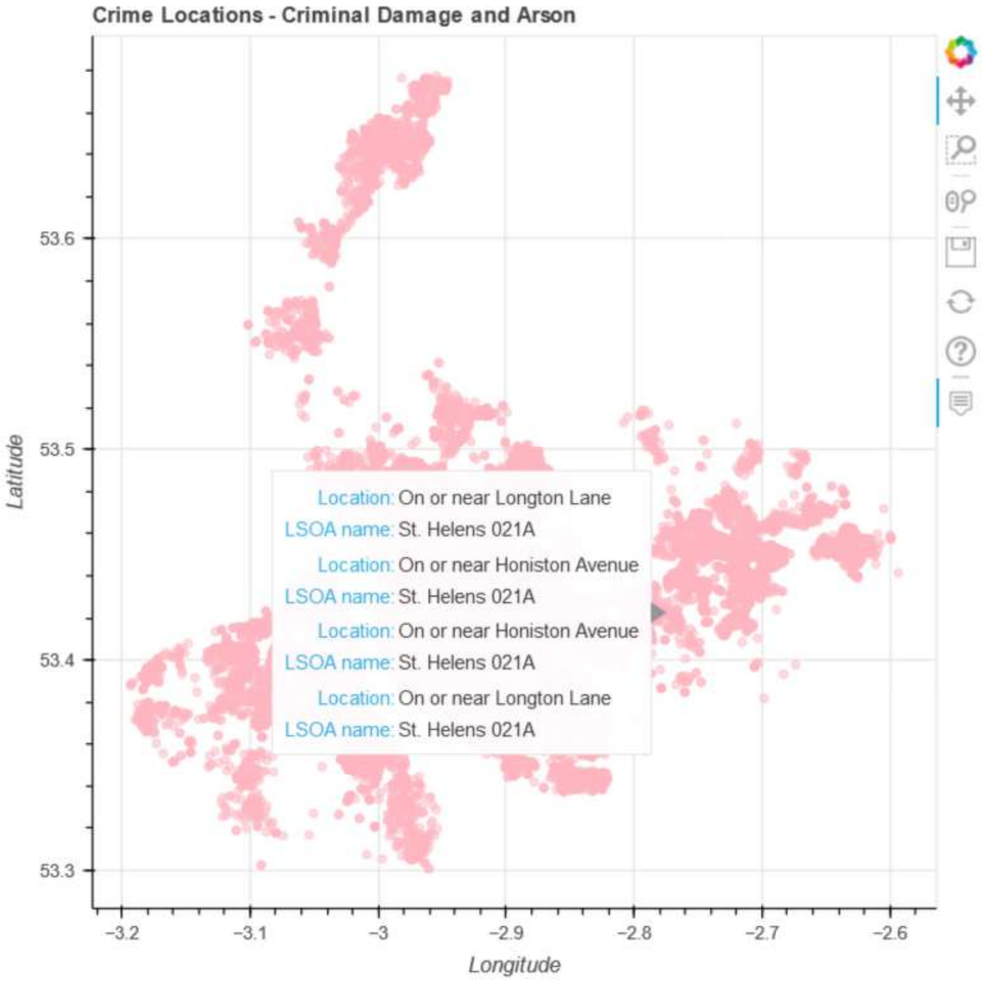


## 31. Interactive Bokeh Scatter plot, which lists the LSOA name at every location



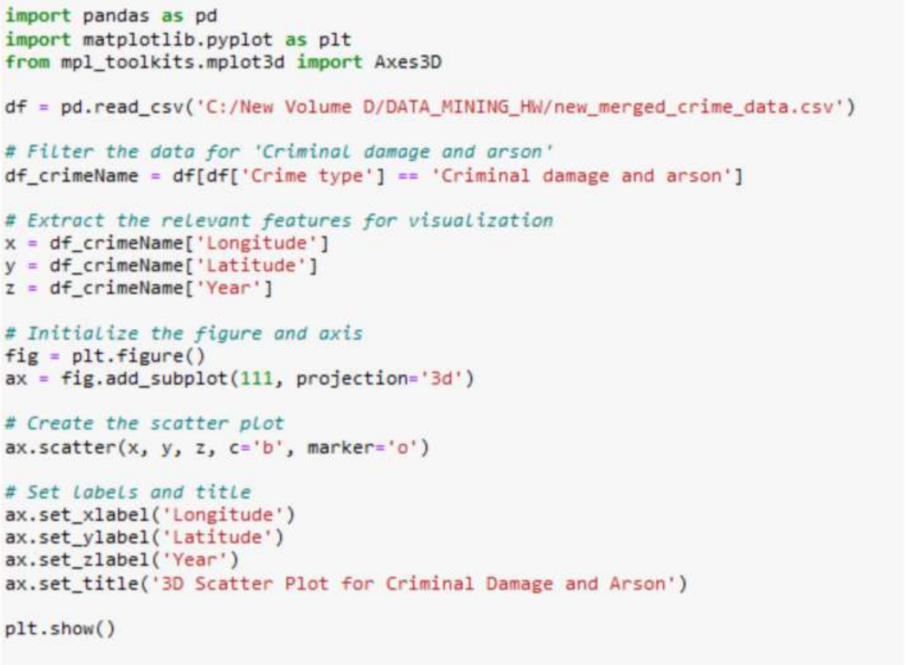
```
import pandas as pd
from bokeh.plotting import figure, output_file, show
from bokeh.models import HoverTool

# Read the CSV file into a DataFrame
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Filter the data for 'Criminal damage and arson' crime type
df_crimeName = df[df['Crime type'] == 'Criminal damage and arson']

# Create a Bokeh figure
output_file("scatter_plot.html")
p = figure(title="Crime Locations - Criminal Damage and Arson",
           x_axis_label='Longitude', y_axis_label='Latitude')

# Add data points to the plot
p.circle(x='Longitude', y='Latitude', source=df_crimeName, size=5, color='lightpink', alpha=0.5)

# Add tooltips to display additional information
tooltips = [("Location", "@Location"), ("LSOA name", "@{LSOA name}")]
p.add_tools(HoverTool(tooltips=tooltips))

# Show the plot
show(p)
```



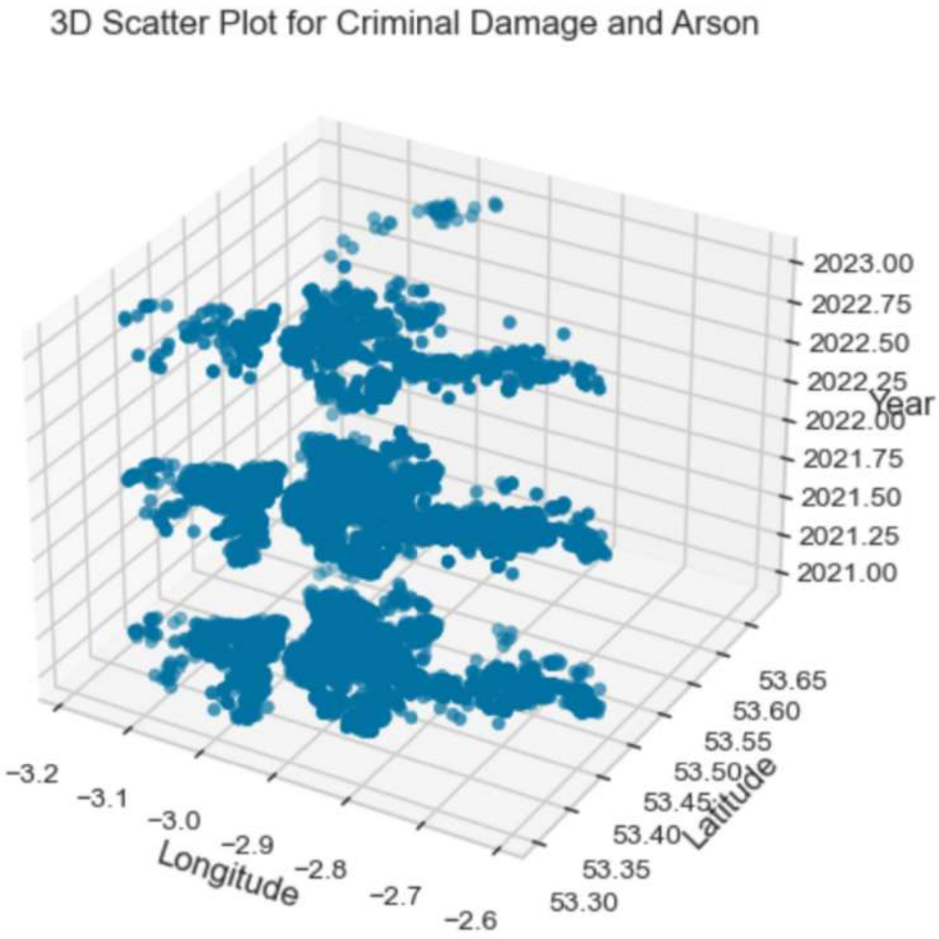


Location: On or near Longton Lane
LSOA name: St. Helens 021A
Location: On or near Honiston Avenue
LSOA name: St. Helens 021A
Location: On or near Honiston Avenue
LSOA name: St. Helens 021A
Location: On or near Longton Lane
LSOA name: St. Helens 021A

## 32. A 3D Scatter Plot

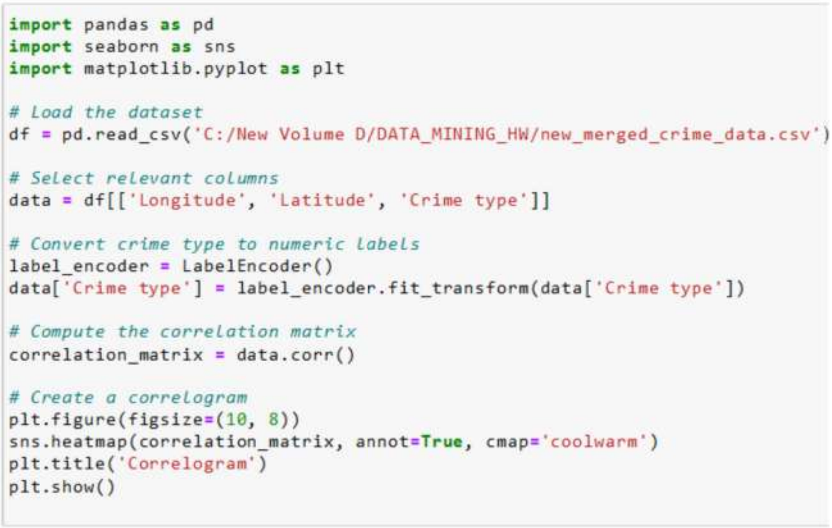


```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Filter the data for 'Criminal damage and arson'
df_crimeName = df[df['Crime type'] == 'Criminal damage and arson']

# Extract the relevant features for visualization
x = df_crimeName['Longitude']
y = df_crimeName['Latitude']
z = df_crimeName['Year']

# Initialize the figure and axis
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Create the scatter plot
ax.scatter(x, y, z, c='b', marker='o')

# Set labels and title
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_zlabel('Year')
ax.set_title('3D Scatter Plot for Criminal Damage and Arson')

plt.show()
```

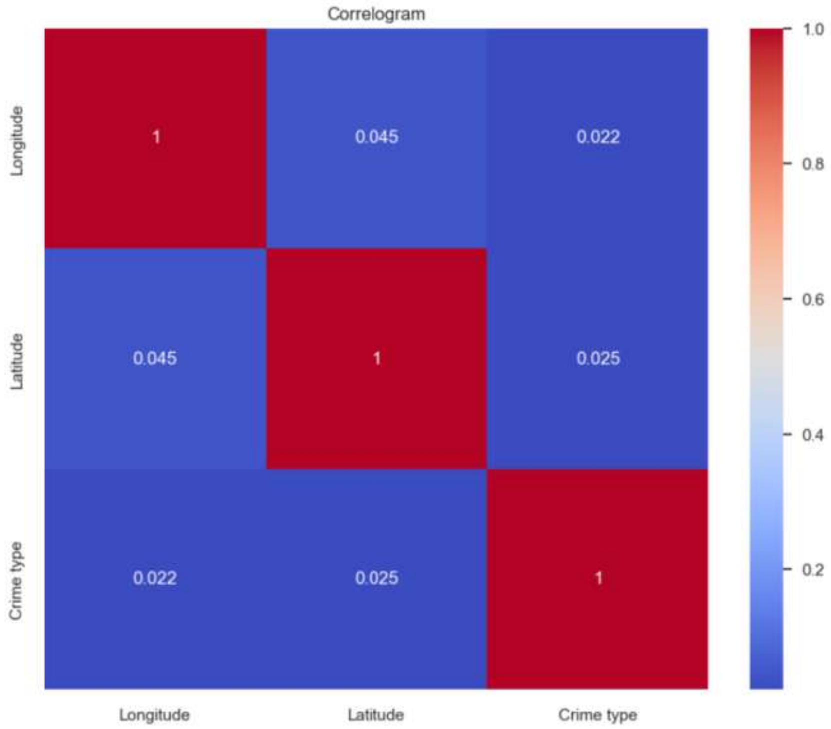


3D Scatter Plot for Criminal Damage and Arson

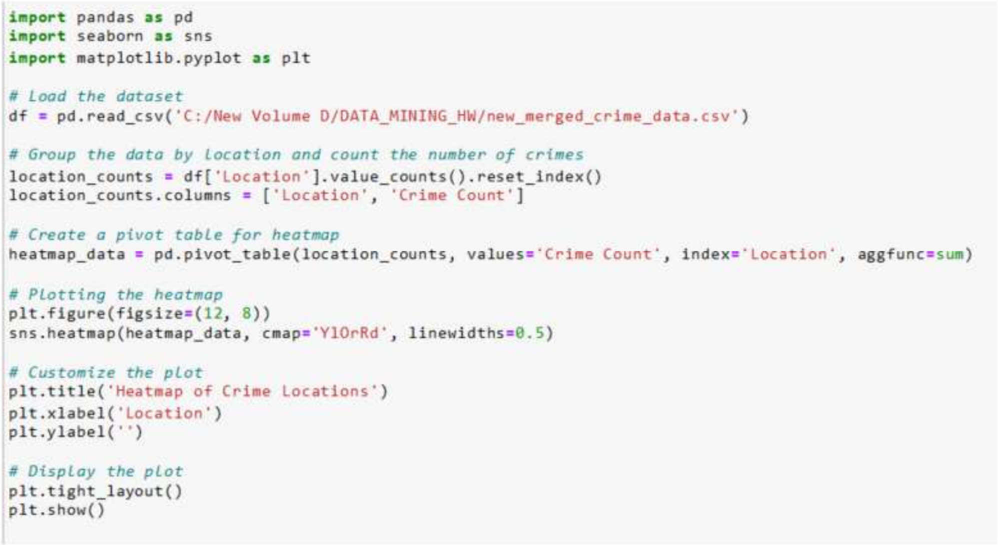
## 33. A correlogram (Like a heatmap)



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

# Load the dataset
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Select relevant columns
data = df[['Longitude', 'Latitude', 'Crime type']]

# Convert crime type to numeric labels
label_encoder = LabelEncoder()
data['Crime type'] = label_encoder.fit_transform(data['Crime type'])

# Compute the correlation matrix
correlation_matrix = data.corr()

# Create a correlogram
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlogram')
plt.show()
```



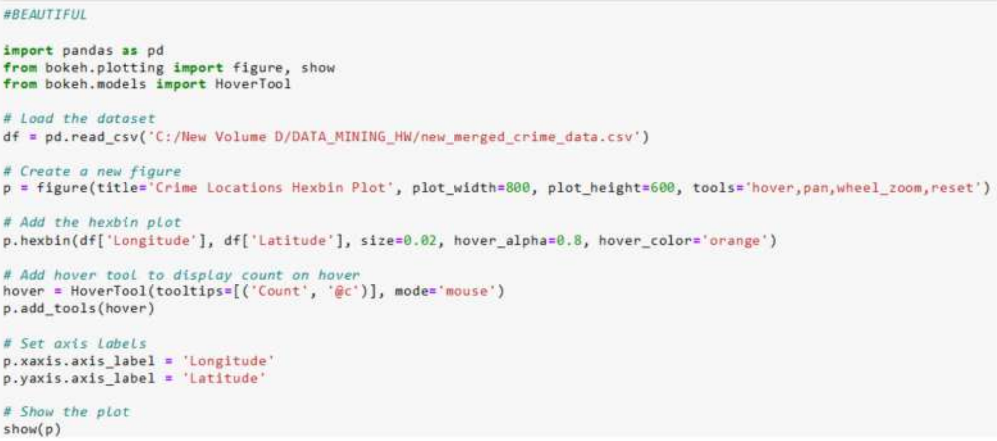
## 34. HeatMap of crime locations



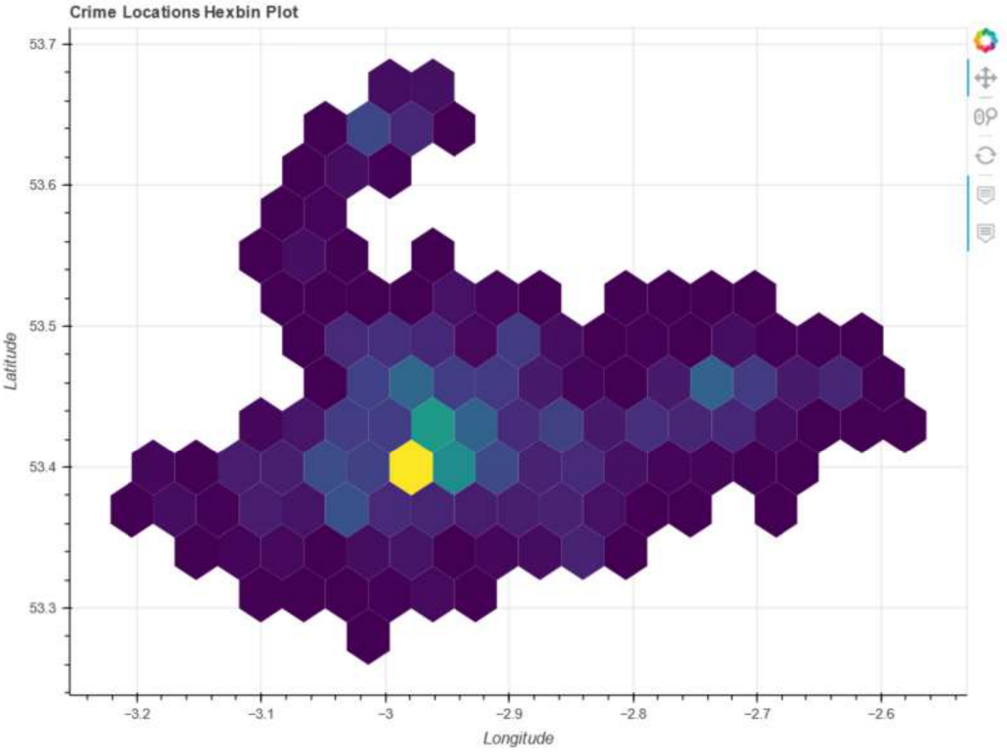
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Group the data by location and count the number of crimes
location_counts = df['Location'].value_counts().reset_index()
location_counts.columns = ['Location', 'Crime Count']

# Create a pivot table for heatmap
heatmap_data = pd.pivot_table(location_counts, values='Crime Count', index='Location', aggfunc=sum)

# Plotting the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, cmap='YlOrRd', linewidths=0.5)

# Customize the plot
plt.title('Heatmap of Crime Locations')
plt.xlabel('Location')
plt.ylabel('')

# Display the plot
plt.tight_layout()
plt.show()
```

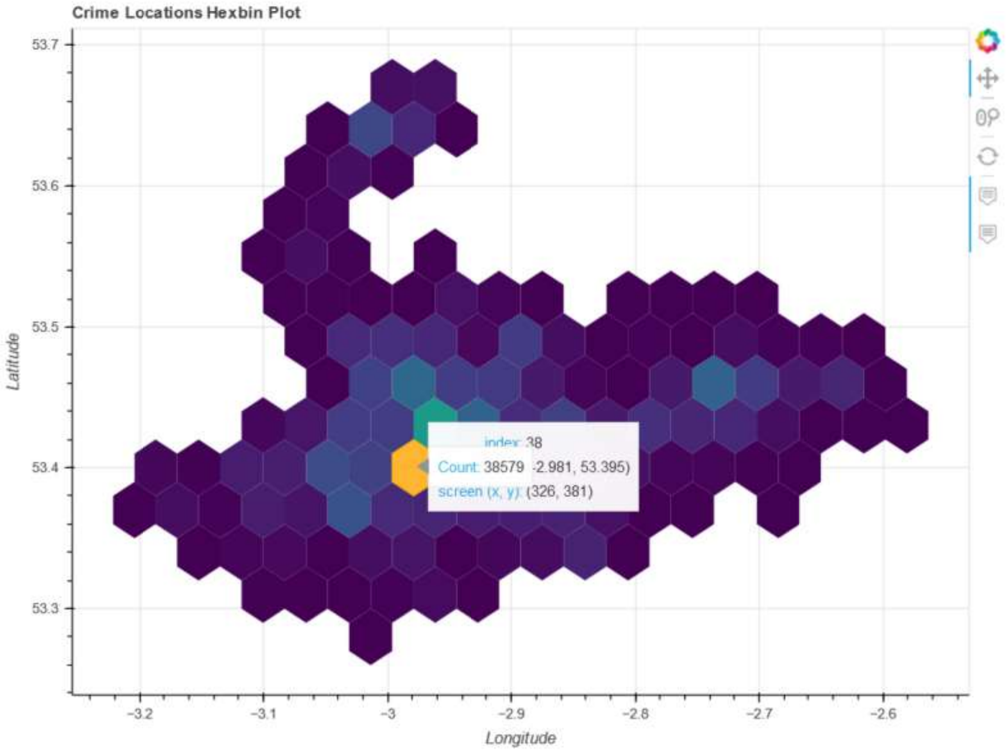
Heatmap of Crime Locations
On or near
On or near Alundale Road
On or near Autumn Grove
On or near Bayhorse Lane
On or near Birkin Close
On or near Brewster Street
On or near Burrell Drive
On or near Catonfield Road
On or near Cirencester Avenue
On or near Corwen Close
On or near Damerham Mews
On or near Dorothy Drive
On or near Effingham Street
On or near Falkland Road
On or near Frankby Road
On or near Gooch Drive
On or near Guildhall Road
On or near Hawthorne Close
On or near Hinson Street
On or near Iris Avenue
On or near Kingsbury Close
On or near Leaway
On or near Longfield Avenue
On or near Maple Crescent
On or near Merepark Drive
On or near Mossley Avenue
On or near Northern Perimeter Road
On or near Oxford Close
On or near Phillips Close
On or near Queen Mary'S Drive
On or near Ridley Lane
On or near Rusland Avenue
On or near Sevenoak Grove
On or near Southport Road
On or near Stanhope Street
On or near Tadlow Close
On or near Thornham Close
On or near Vallance Road
On or near Well Lane
On or near Wilton Road
14000
12000
10000
8000
6000
4000
2000
Crime Count
Location

## 35. Interactive Hexbin Plot



```
#BEAUTIFUL

import pandas as pd
from bokeh.plotting import figure, show
from bokeh.models import HoverTool

# Load the dataset
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Create a new figure
p = figure(title='Crime Locations Hexbin Plot', plot_width=800, plot_height=600, tools='hover,pan,wheel_zoom,reset')

# Add the hexbin plot
p.hexbin(df['Longitude'], df['Latitude'], size=0.02, hover_alpha=0.8, hover_color='orange')

# Add hover tool to display count on hover
hover = HoverTool(tooltips=[('Count', '@c')], mode='mouse')
p.add_tools(hover)

# Set axis labels
p.xaxis.axis_label = 'Longitude'
p.yaxis.axis_label = 'Latitude'

# Show the plot
show(p)
```



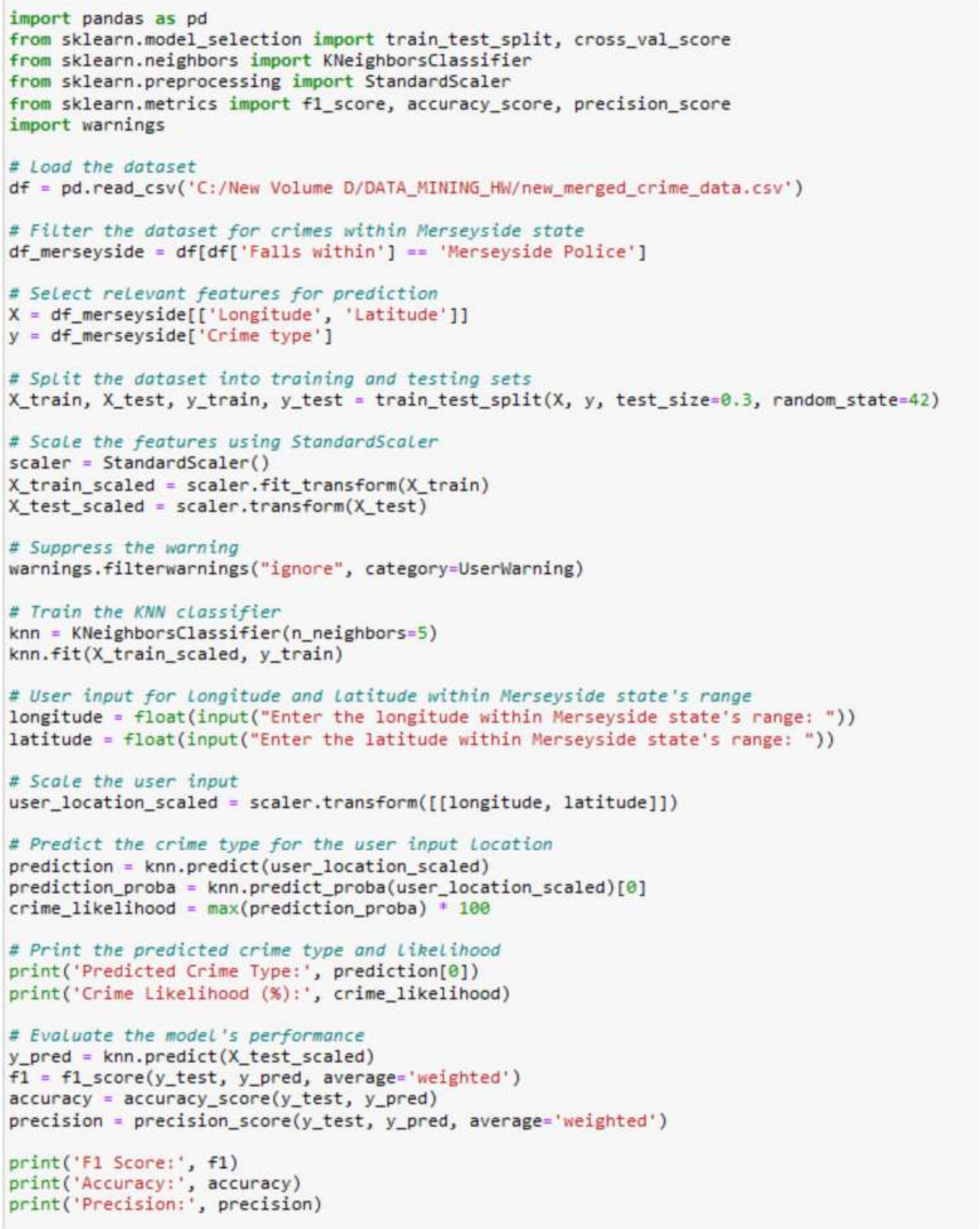

Crime Locations Hexbin Plot
Latitude
Longitude
53.7
53.6
53.5
53.4
53.3
-3.2
-3.1
-3
-2.9
-2.8
-2.7
-2.6
index: 38
Count: 38579 (-2.981, 53.395)
screen (x, y): (326, 381)

Here we can notice that Count of crimes along with longitude & latitude is present

# 36. FINAL ANSWER WITH VARIATIONS

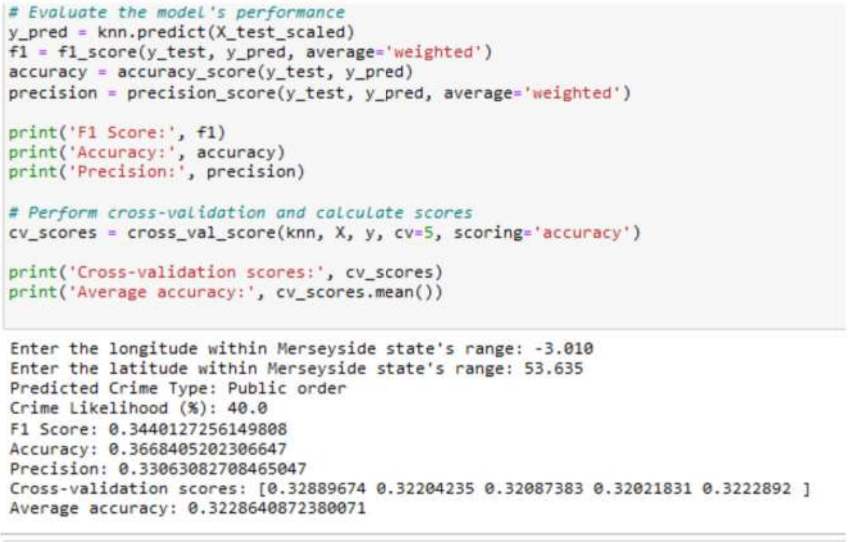
*To check the quality of the code lets add the same values in each code cell ie. longitude = -3.010 & latitude = 53.635*

## A) Function to predict crime type only for Merseyside state



```
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score, accuracy_score, precision_score
import warnings

# Load the dataset
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Filter the dataset for crimes within Merseyside state
df_merseyside = df[df['Falls within'] == 'Merseyside Police']

# Select relevant features for prediction
X = df_merseyside[['Longitude', 'Latitude']]
y = df_merseyside['Crime type']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Scale the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Suppress the warning
warnings.filterwarnings("ignore", category=UserWarning)

# Train the KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

# User input for longitude and latitude within Merseyside state's range
longitude = float(input("Enter the longitude within Merseyside state's range: "))
latitude = float(input("Enter the latitude within Merseyside state's range: "))

# Scale the user input
user_location_scaled = scaler.transform([[longitude, latitude]])

# Predict the crime type for the user input location
prediction = knn.predict(user_location_scaled)
prediction_proba = knn.predict_proba(user_location_scaled)[0]
crime_likelihood = max(prediction_proba) * 100

# Print the predicted crime type and likelihood
print('Predicted Crime Type:', prediction[0])
print('Crime Likelihood (%):', crime_likelihood)

# Evaluate the model's performance
y_pred = knn.predict(X_test_scaled)
f1 = f1_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')

print('F1 Score:', f1)
print('Accuracy:', accuracy)
print('Precision:', precision)
```

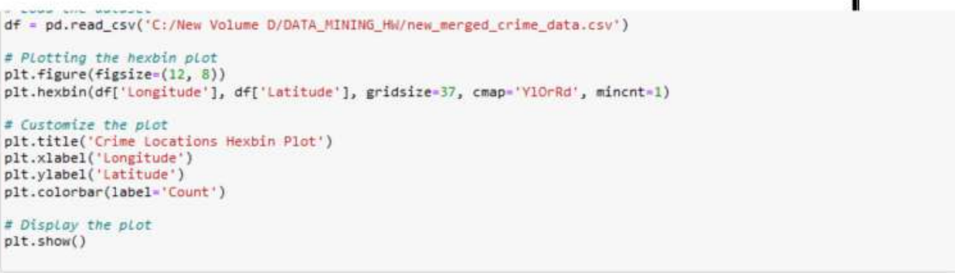
**MERSEYSIDE LIKELIHOOD RESULTS → on next page**



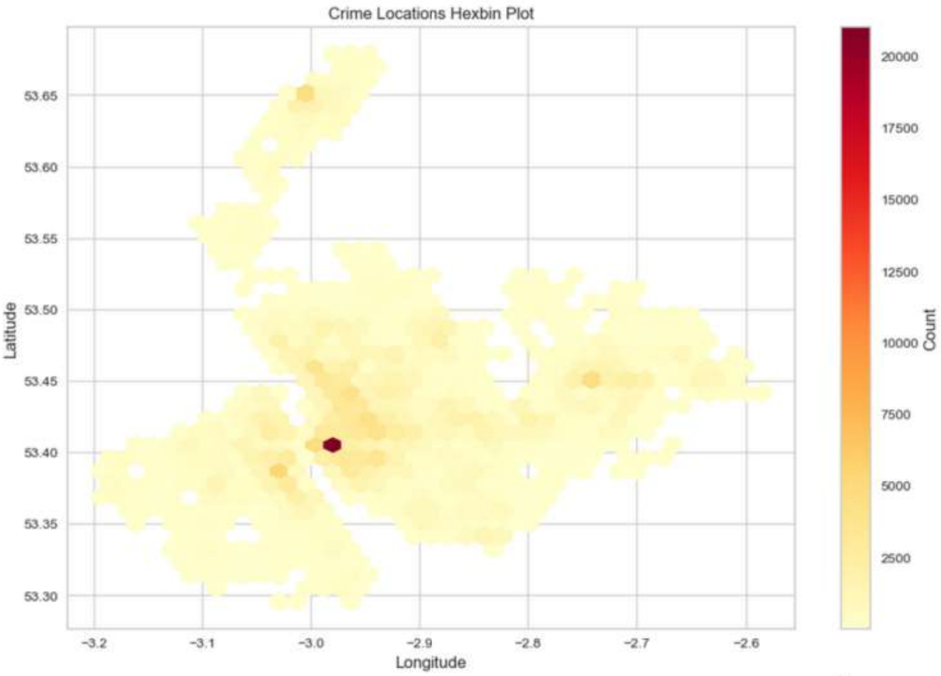
```
# Evaluate the model's performance
y_pred = knn.predict(X_test_scaled)
f1 = f1_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')

print('F1 Score:', f1)
print('Accuracy:', accuracy)
print('Precision:', precision)

# Perform cross-validation and calculate scores
cv_scores = cross_val_score(knn, X, y, cv=5, scoring='accuracy')

print('Cross-validation scores:', cv_scores)
print('Average accuracy:', cv_scores.mean())
```

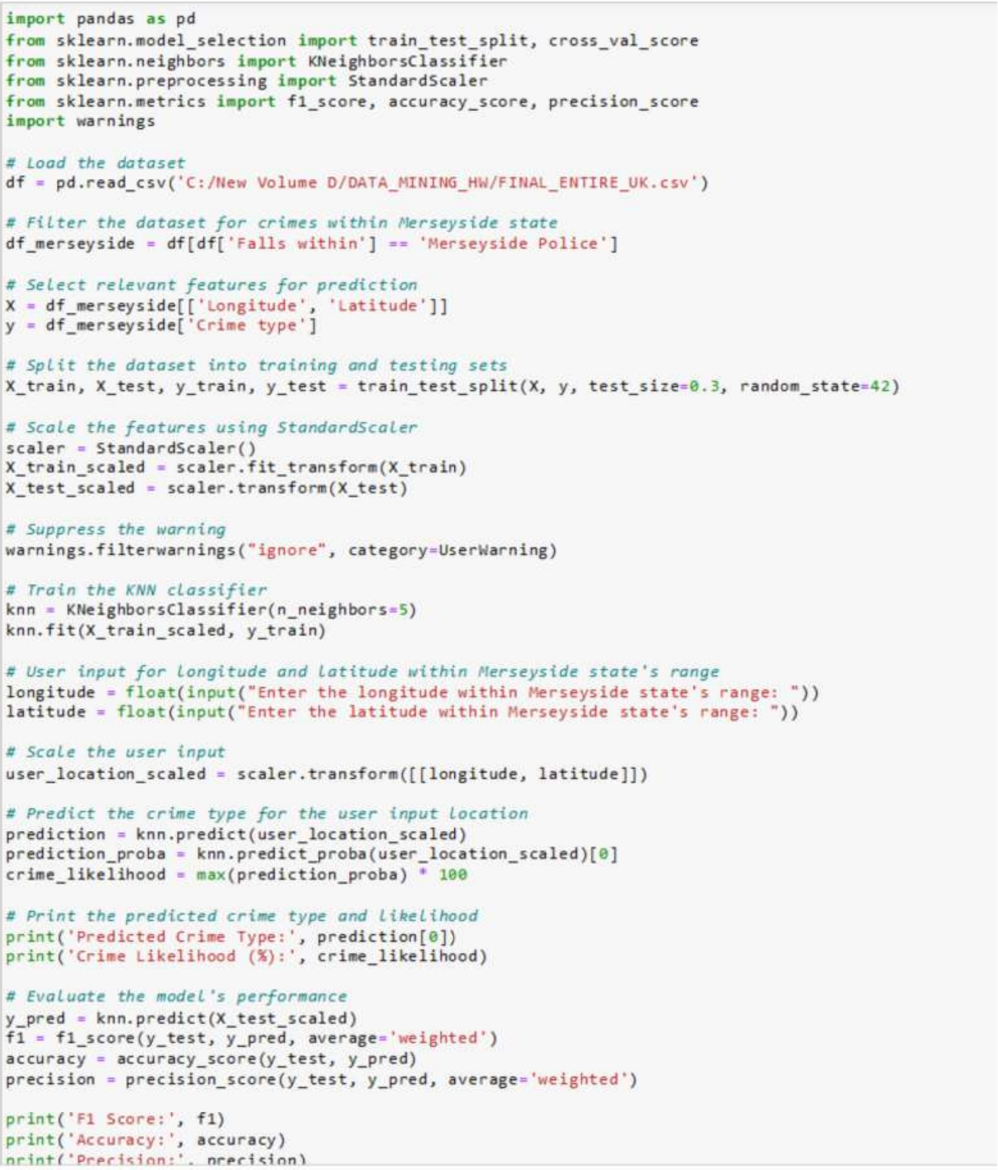
```
Enter the longitude within Merseyside state's range: -3.010
Enter the latitude within Merseyside state's range: 53.635
Predicted Crime Type: Public order
Crime Likelihood (%): 40.0
F1 Score: 0.3440127256149808
Accuracy: 0.3668405202306647
Precision: 0.33063082708465047
Cross-validation scores: [0.32889674 0.32204235 0.32087383 0.32021831 0.3222892 ]
Average accuracy: 0.3228640872380071
```

## B) Function to predict crime type only for Entire UK



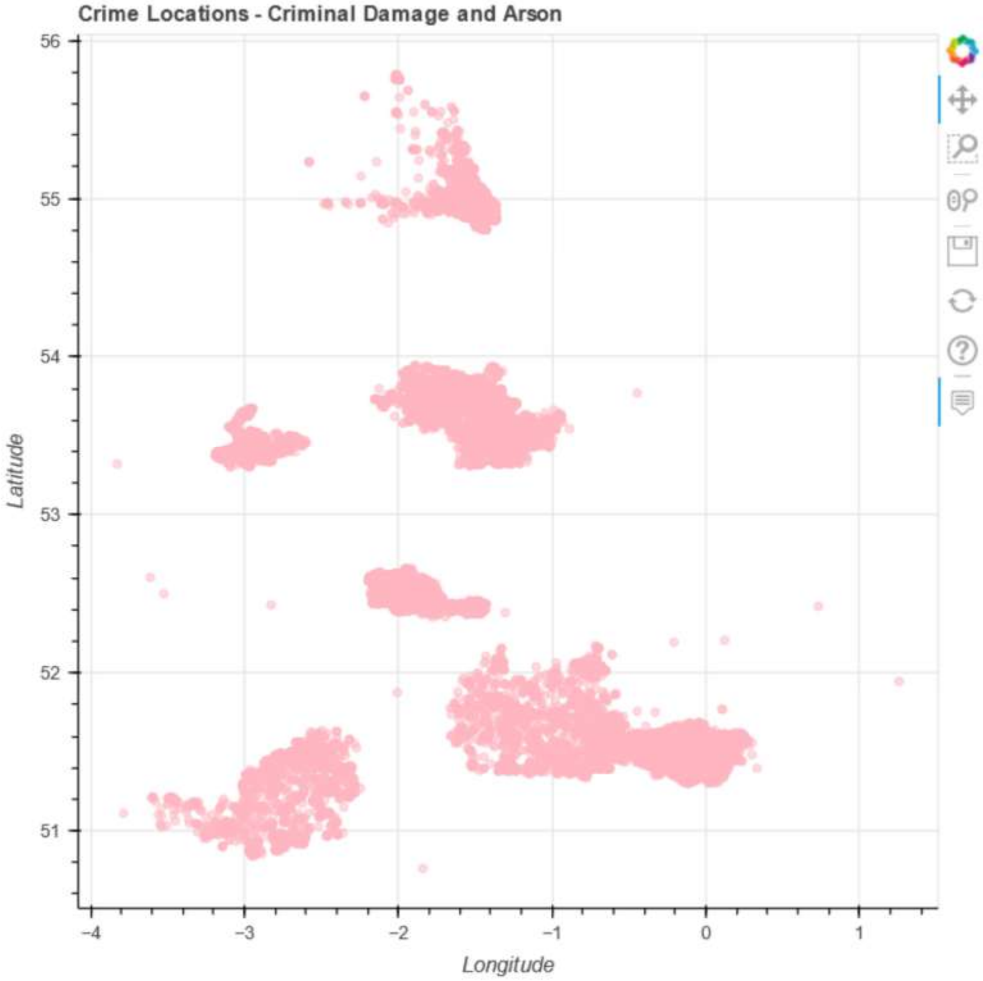
```
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')

# Plotting the hexbin plot
plt.figure(figsize=(12, 8))
plt.hexbin(df['Longitude'], df['Latitude'], gridsize=37, cmap='YlOrRd', mincnt=1)

# Customize the plot
plt.title('Crime Locations Hexbin Plot')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.colorbar(label='Count')

# Display the plot
plt.show()
```





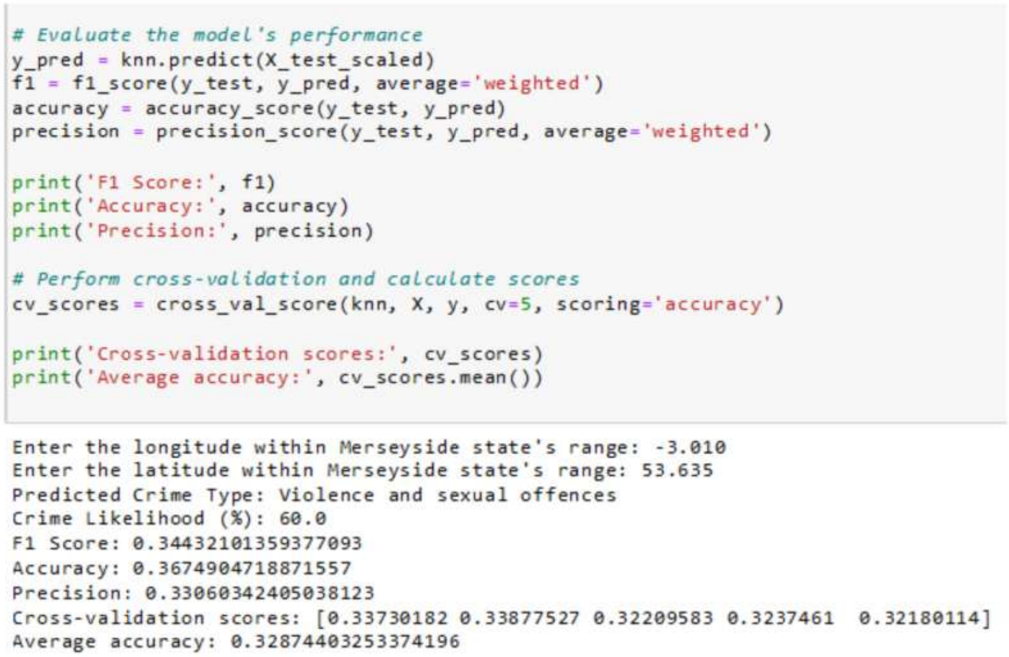
```
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score, accuracy_score, precision_score
import warnings

# Load the dataset
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/FINAL_ENTIRE_UK.csv')

# Filter the dataset for crimes within Merseyside state
df_merseyside = df[df['Falls within'] == 'Merseyside Police']

# Select relevant features for prediction
X = df_merseyside[['Longitude', 'Latitude']]
y = df_merseyside['Crime type']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Scale the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Suppress the warning
warnings.filterwarnings("ignore", category=UserWarning)

# Train the KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

# User input for longitude and latitude within Merseyside state's range
longitude = float(input("Enter the longitude within Merseyside state's range: "))
latitude = float(input("Enter the latitude within Merseyside state's range: "))

# Scale the user input
user_location_scaled = scaler.transform([[longitude, latitude]])

# Predict the crime type for the user input location
prediction = knn.predict(user_location_scaled)
prediction_proba = knn.predict_proba(user_location_scaled)[0]
crime_likelihood = max(prediction_proba) * 100

# Print the predicted crime type and likelihood
print('Predicted Crime Type:', prediction[0])
print('Crime Likelihood (%):', crime_likelihood)

# Evaluate the model's performance
y_pred = knn.predict(X_test_scaled)
f1 = f1_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')

print('F1 Score:', f1)
print('Accuracy:', accuracy)
print('Precision:', precision)
```

UK-MAP VISUALIZATION PICTURES:



**ENTIRE UNITED KINGDOM LIKELIHOOD RESULTS**

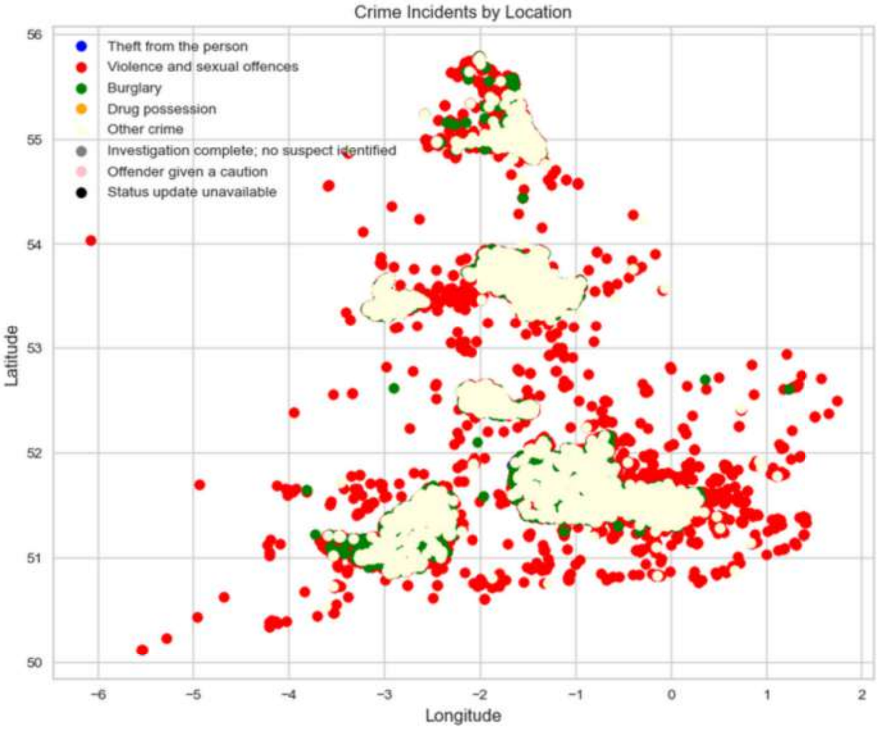
**→ on next page**

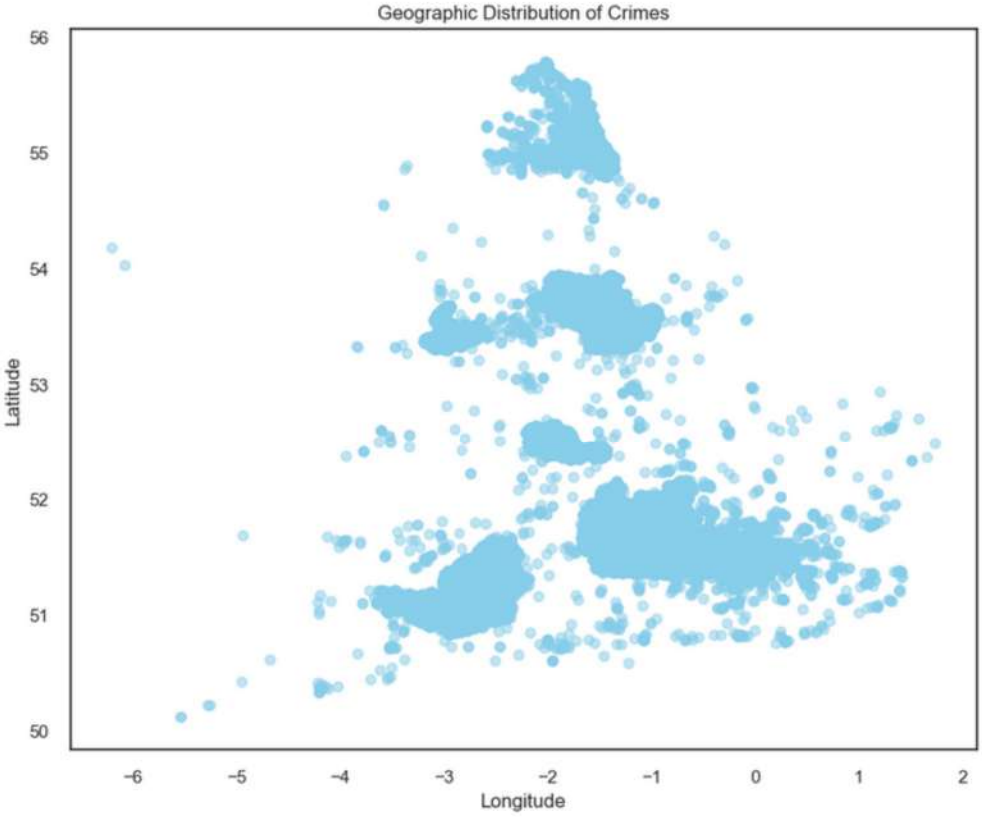


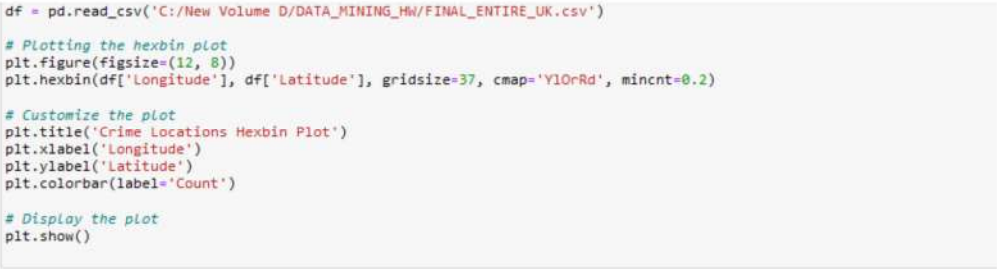
```
# Evaluate the model's performance
y_pred = knn.predict(X_test_scaled)
f1 = f1_score(y_test, y_pred, average='weighted')
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')

print('F1 Score:', f1)
print('Accuracy:', accuracy)
print('Precision:', precision)

# Perform cross-validation and calculate scores
cv_scores = cross_val_score(knn, X, y, cv=5, scoring='accuracy')

print('Cross-validation scores:', cv_scores)
print('Average accuracy:', cv_scores.mean())
```

```
Enter the longitude within Merseyside state's range: -3.010
Enter the latitude within Merseyside state's range: 53.635
Predicted Crime Type: Violence and sexual offences
Crime Likelihood (%): 60.0
F1 Score: 0.34432101359377093
Accuracy: 0.3674904718871557
Precision: 0.33060342405038123
Cross-validation scores: [0.33730182 0.33877527 0.32209583 0.3237461  0.32180114]
Average accuracy: 0.32874403253374196
```

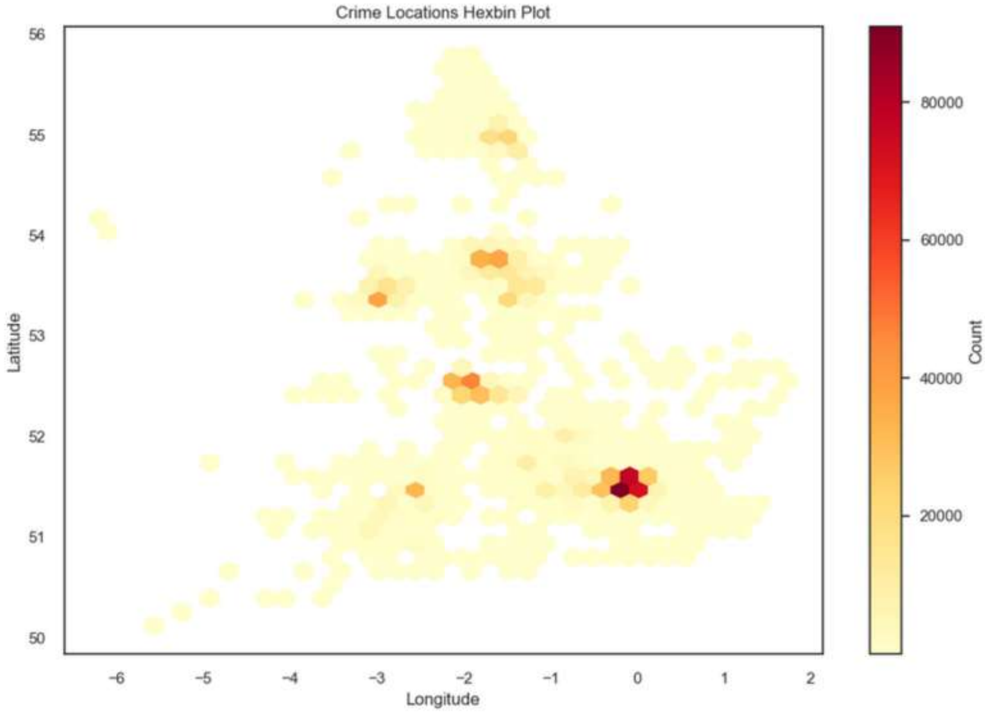
SOME ENTIRE UK's VISUALIZATION:




Geographic Distribution of Crimes
Latitude
56
55
54
53
52
51
50
Longitude
−6
−5
−4
−3
−2
−1
0
1
2



```
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/FINAL_ENTIRE_UK.csv')

# Plotting the hexbin plot
plt.figure(figsize=(12, 8))
plt.hexbin(df['Longitude'], df['Latitude'], gridsize=37, cmap='YlOrRd', mincnt=0.2)

# Customize the plot
plt.title('Crime Locations Hexbin Plot')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.colorbar(label='Count')

# Display the plot
plt.show()
```



# CONCLUSION:

PREDICTION RESULTS:

As we can see that after using the same values for ***longitude = -3.010 and latitude = 53.635 in Merseyside state of United Kingdom*** our results were as follows:

1. Predicted Crime Type: Public order

2. Crime Likelihood (%): 40.0
3. F1 Score: 0.3440
4. Accuracy: 0.3668
5. Precision: 0.3306
6. Cross-validation scores: [0.328  0.322  0.320  0.320  0.322]
7. Average accuracy: 0.3228

While using the same values for ***longitude = -3.010 and  latitude = 53.635 in entire United Kingdom tweaked*** our results as follows:

1. Predicted Crime Type: Violence and sexual offences

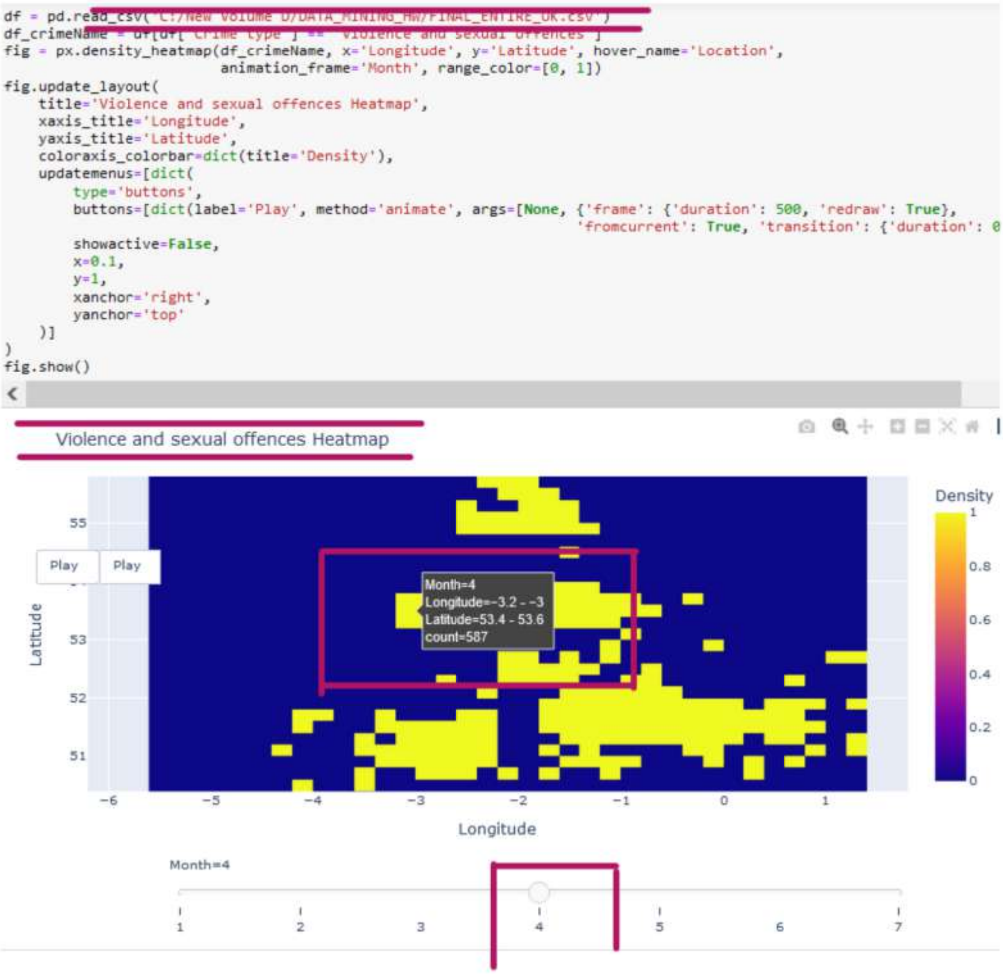
2. Crime Likelihood (%): 60.0
3. F1 Score: 0.3443
4. Accuracy: 0.3674
5. Precision: 0.3306
6. Cross-validation scores: [0.337  0.338  0.322  0.323  0.321]
7. Average accuracy: 0.3287

REAL DATASET RESULTS:

1. Entire UK Dataset :

In Prediction result it showed that ‘Violence & sexual offences’ have more likelihood on the given longitude & latitude (-3.010, 53.635).

So, lets see what actual dataset’s Heatmap shows us in a particular month:

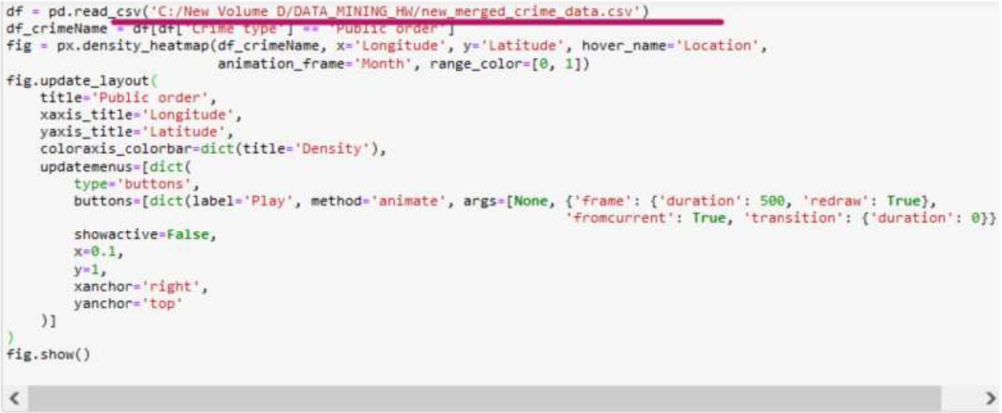


```
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/FINAL_ENTIRE_UK.csv')
df_crimeName = df[df['Crime type'] == 'Violence and sexual offences']
fig = px.density_heatmap(df_crimeName, x='Longitude', y='Latitude', hover_name='Location',
                              animation_frame='Month', range_color=[0, 1])
fig.update_layout(
    title='Violence and sexual offences Heatmap',
    xaxis_title='Longitude',
    yaxis_title='Latitude',
    coloraxis_colorbar=dict(title='Density'),
    updatemenus=[dict(
        type='buttons',
        buttons=[dict(label='Play', method='animate', args=[None, {'frame': {'duration': 500, 'redraw': True},
                                                                    'fromcurrent': True, 'transition': {'duration': 0

        showactive=False,
        x=0.1,
        y=1,
        xanchor='right',
        yanchor='top'
    )]
)
fig.show()
```

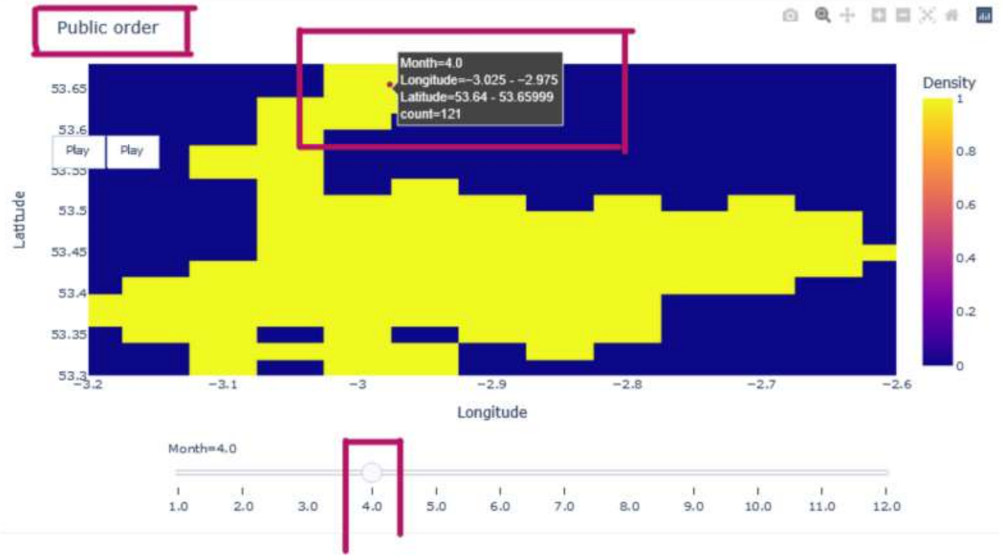
2. Only Merseyside state from United Kingdom :

In Prediction result it showed that ‘Public Order’ have more likelihood on the given longitude & latitude (-3.010, 53.635).

So, lets see what actual dataset’s Heatmap shows us in a particular month:



```
df = pd.read_csv('C:/New Volume D/DATA_MINING_HW/new_merged_crime_data.csv')
df_crimeName = df[df['Crime type'] == 'Public order']
fig = px.density_heatmap(df_crimeName, x='Longitude', y='Latitude', hover_name='Location',
                         animation_frame='Month', range_color=[0, 1])
fig.update_layout(
    title='Public order',
    xaxis_title='Longitude',
    yaxis_title='Latitude',
    coloraxis_colorbar=dict(title='Density'),
    updatemenus=[dict(
        type='buttons',
        buttons=[dict(label='Play', method='animate', args=[None, {'frame': {'duration': 500, 'redraw': True},
                                                                  'fromcurrent': True, 'transition': {'duration': 0}}
        showactive=False,
        x=0.1,
        y=1,
        xanchor='right',
        yanchor='top'
    )]
)
fig.show()
```



**As, both ‘Crime type’ are occurring to some given extent in their respective original datasets!**

**Therefore, I conclude that prediction results are more on the satisfactory side.**

**And they show better chances to predict the future outcomes**

**THANK YOU**