CS 553 Project

Author: Jie Huang, Yinghao Li, Zhengyu An

Background:

With the rapid development of high technology, significant conveniences have been brought to our lives. While we enjoy this happiness, various criminal activities are also undermining this atmosphere. The security situation is particularly unstable in the United States, especially in large cities like LA, where public safety is a cause for concern. Considering various factors, we have chosen LA crime data (2020-) as the topic for our project this time.

The purpose of this project is to analyze and model crime data using machine learning methods, aiming to uncover correlations between different features, assess their impact on the target variable (such as crime status), and attempt to build a model capable of predicting or explaining crime status.

In order to efficiently manage the exploration and analysis within the available time frame, we have strategically opted to work with a subset of the original dataset, specifically selecting about 8,000 rows (1/100) for our analysis. This decision allows us to streamline the computational demands associated with processing a large dataset and facilitates a more rapid iteration through exploratory data analysis and modeling tasks. Despite the downsizing, this subset remains representative enough to capture key trends, patterns, and characteristics present in the larger dataset. This approach ensures that our analysis is both time-efficient and insightful, striking a balance between resource constraints and the depth of exploration required for meaningful insights.

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.moun
t("/content/drive", force_remount=True).
```

In [ ]:
```
!pip install umap-learn
import itertools
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import umap
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import OrdinalEncoder
from sklearn.neural_network import MLPClassifier
```

```
Requirement already satisfied: umap-learn in /usr/local/lib/python3.10/dist-packages (0.
5.5)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (f
rom umap-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.10/dist-packages
(from umap-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-pack
```

```
ages (from umap-learn) (1.2.2)
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.10/dist-packages
(from umap-learn) (0.58.1)
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.10/dist-packag
es (from umap-learn) (0.5.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from uma
p-learn) (4.66.1)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/python3.10/d
ist-packages (from numba>=0.51.2->umap-learn) (0.41.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages
(from pynndescent>=0.5->umap-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-pa
ckages (from scikit-learn>=0.22->umap-learn) (3.2.0)
```

In [ ]:
```
crime = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/sampled_csv_file.csv')
crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8117 entries, 0 to 8116
Data columns (total 28 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   DR_NO           8117 non-null   int64
 1   Date Rptd       8117 non-null   object
 2   DATE OCC        8117 non-null   object
 3   TIME OCC        8117 non-null   int64
 4   AREA            8117 non-null   int64
 5   AREA NAME       8117 non-null   object
 6   Rpt Dist No     8117 non-null   int64
 7   Part 1-2        8117 non-null   int64
 8   Crm Cd          8117 non-null   int64
 9   Crm Cd Desc     8117 non-null   object
 10  Mocodes         6960 non-null   object
 11  Vict Age        8117 non-null   int64
 12  Vict Sex        7019 non-null   object
 13  Vict Descent    7019 non-null   object
 14  Premis Cd       8116 non-null   float64
 15  Premis Desc     8112 non-null   object
 16  Weapon Used Cd  2758 non-null   float64
 17  Weapon Desc     2758 non-null   object
 18  Status          8117 non-null   object
 19  Status Desc     8117 non-null   object
 20  Crm Cd 1        8117 non-null   float64
 21  Crm Cd 2        602 non-null    float64
 22  Crm Cd 3        20 non-null     float64
 23  Crm Cd 4        1 non-null      float64
 24  LOCATION        8117 non-null   object
 25  Cross Street    1280 non-null   object
 26  LAT             8117 non-null   float64
 27  LON             8117 non-null   float64
dtypes: float64(8), int64(7), object(13)
memory usage: 1.7+ MB
```

Providing an initial exploration and understanding of the dataset.

1. The shape of the dataset, indicating the number of rows and columns. This quickly conveys the size of the dataset, helping to establish its dimensions.

2. The entire dataset, offering a visual inspection of the data's structure and content. This is useful for examining the first few rows and gaining an understanding of the data's format and feature values.

3. The column names of the dataset, representing the names of various features. This provides an overview of the dataset's structure, aiding in the interpretation of the meaning behind each feature.

In [ ]:
```
crime.shape
print(crime)
crime.columns
```

```
          DR_NO                Date Rptd                 DATE OCC  TIME OCC  \
0     211114745  10/29/2021 12:00:00 AM  09/20/2021 12:00:00 AM      1200
1     230710292  05/31/2023 12:00:00 AM  05/26/2023 12:00:00 AM      2000
2     231104968  01/25/2023 12:00:00 AM  01/25/2023 12:00:00 AM      1745
3     220105943  02/03/2022 12:00:00 AM  02/02/2022 12:00:00 AM      1915
4     221414988  08/10/2022 12:00:00 AM  08/09/2022 12:00:00 AM      1900
...         ...                     ...                     ...       ...
8112  221000791  07/18/2022 12:00:00 AM  07/18/2022 12:00:00 AM      1940
8113  221308749  03/28/2022 12:00:00 AM  03/28/2022 12:00:00 AM      1655
8114  211004254  01/08/2021 12:00:00 AM  01/05/2021 12:00:00 AM       906
8115  211613985  12/01/2021 12:00:00 AM  12/01/2021 12:00:00 AM      1400
8116  210610672  06/03/2021 12:00:00 AM  06/02/2021 12:00:00 AM      2300

      AREA    AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
0       11    Northeast         1132         2     354
1        7     Wilshire          745         1     520
2       11    Northeast         1127         2     940
3        1      Central          171         1     330
4       14      Pacific         1463         1     330
...    ...          ...          ...       ...     ...
8112    10  West Valley         1043         2     930
8113    13       Newton         1394         2     626
8114    10  West Valley         1017         1     440
8115    16     Foothill         1654         2     901
8116     6    Hollywood          636         1     236

                              Crm Cd Desc  ... Status   Status Desc  \
0                          THEFT OF IDENTITY  ...     IC   Invest Cont
1                  VEHICLE - ATTEMPT STOLEN  ...     IC   Invest Cont
2                                EXTORTION  ...     IC   Invest Cont
3                     BURGLARY FROM VEHICLE  ...     IC   Invest Cont
4                     BURGLARY FROM VEHICLE  ...     IC   Invest Cont
...                                     ...  ...    ...           ...
8112  CRIMINAL THREATS - NO WEAPON DISPLAYED  ...     AA  Adult Arrest
8113       INTIMATE PARTNER - SIMPLE ASSAULT  ...     AA  Adult Arrest
8114      THEFT PLAIN - PETTY ($950 & UNDER)  ...     AA  Adult Arrest
8115          VIOLATION OF RESTRAINING ORDER  ...     AA  Adult Arrest
8116   INTIMATE PARTNER - AGGRAVATED ASSAULT  ...     AA  Adult Arrest

     Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
0       354.0      NaN       NaN      NaN
1       520.0      NaN       NaN      NaN
2       940.0      NaN       NaN      NaN
3       330.0      NaN       NaN      NaN
4       330.0      NaN       NaN      NaN
...       ...      ...       ...      ...
8112    930.0    998.0       NaN      NaN
8113    626.0      NaN       NaN      NaN
8114    440.0      NaN       NaN      NaN
8115    901.0      NaN       NaN      NaN
8116    236.0    998.0       NaN      NaN

                                  LOCATION Cross Street      LAT       LON
0      3100    RIVERSIDE                      DR          NaN  34.1131 -118.2684
1      5600    WILSHIRE                       BL          NaN  34.0624 -118.3510
2      6000    FAYETTE                        ST          NaN  34.1194 -118.1889
3       800    JAMES M WOOD                   BL          NaN  34.0462 -118.2628
4      5300    PLAYA VISTA                    DR          NaN  33.9762 -118.4287
...                                         ...          ...      ...       ...
8112   5800    RESEDA                         BL          NaN  34.1769 -118.5384
```

```
8113    100 E   69TH                          ST          NaN  33.9769 -118.2739
8114   7200     PASO ROBLES                   AV          NaN  34.2028 -118.5022
8115  10100     RALSTON                       AV          NaN  34.2528 -118.4041
8116   1700 N   MCCADDEN                      PL          NaN  34.1016 -118.3374

[8117 rows x 28 columns]
```

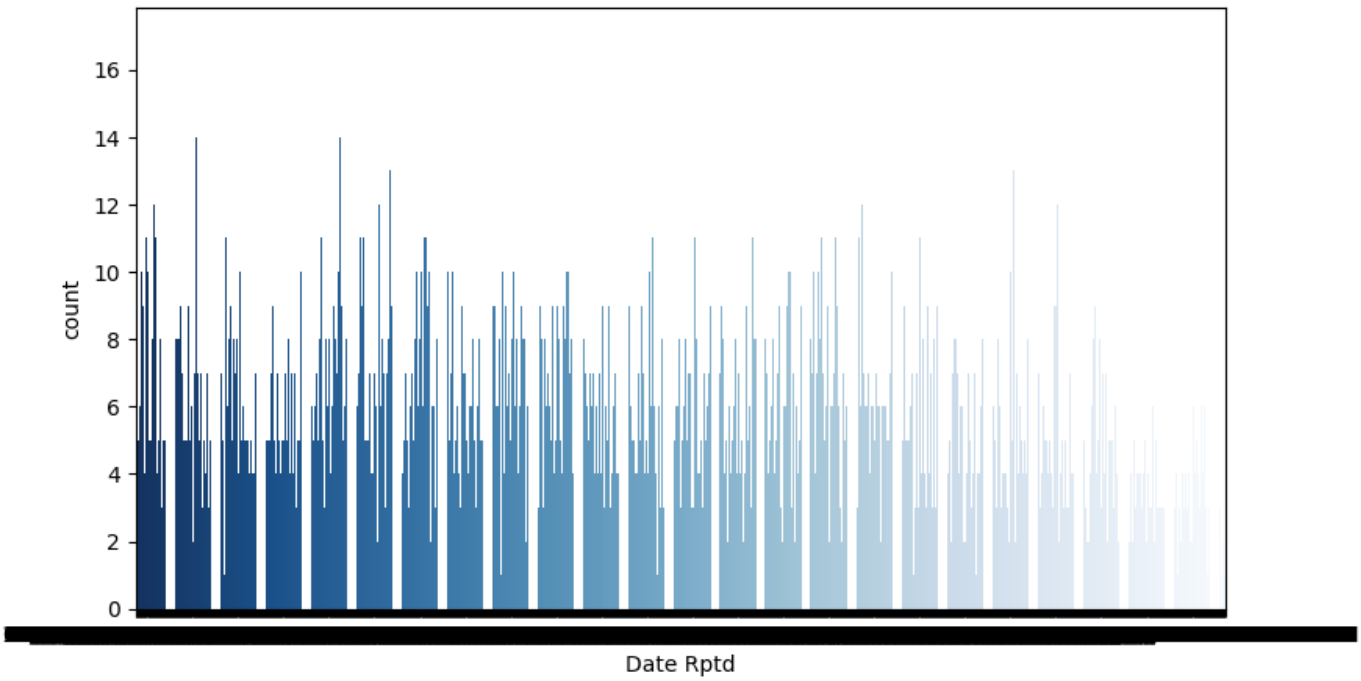
Out[ ]:

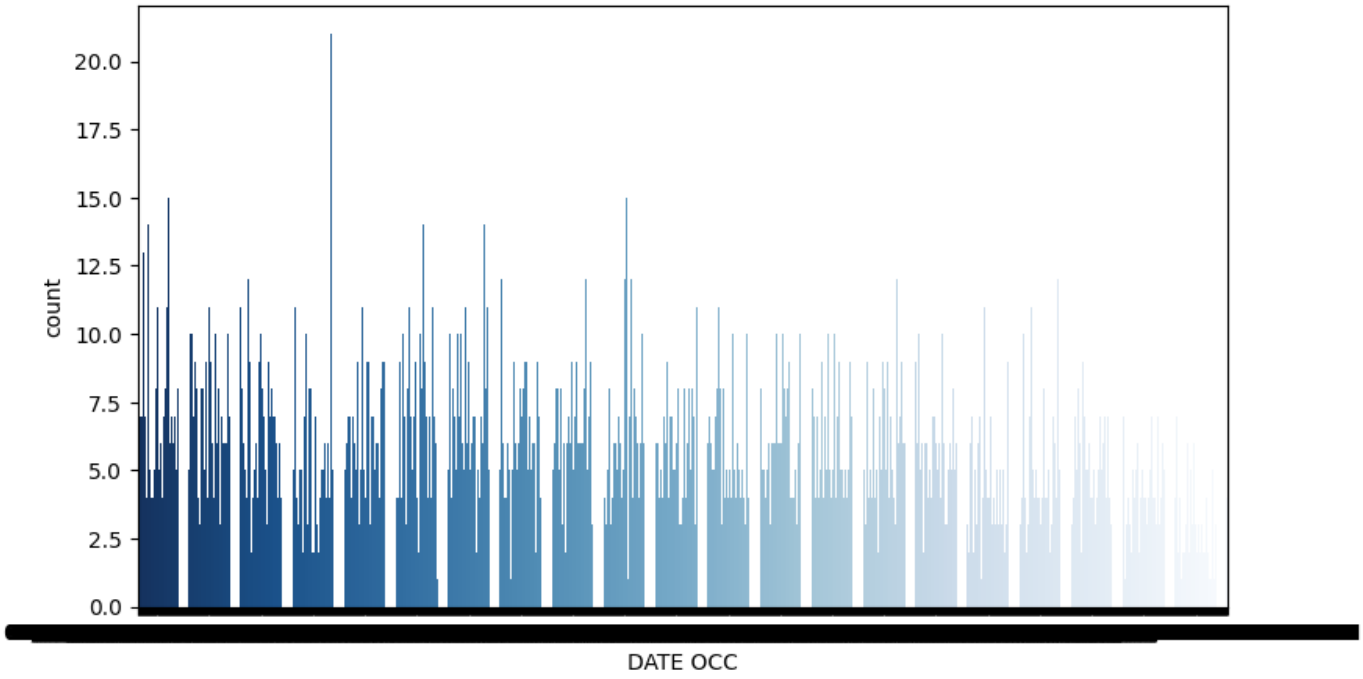
```
Index(['DR_NO', 'Date Rptd', 'DATE OCC', 'TIME OCC', 'AREA', 'AREA NAME',
       'Rpt Dist No', 'Part 1-2', 'Crm Cd', 'Crm Cd Desc', 'Mocodes',
       'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis Desc',
       'Weapon Used Cd', 'Weapon Desc', 'Status', 'Status Desc', 'Crm Cd 1',
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

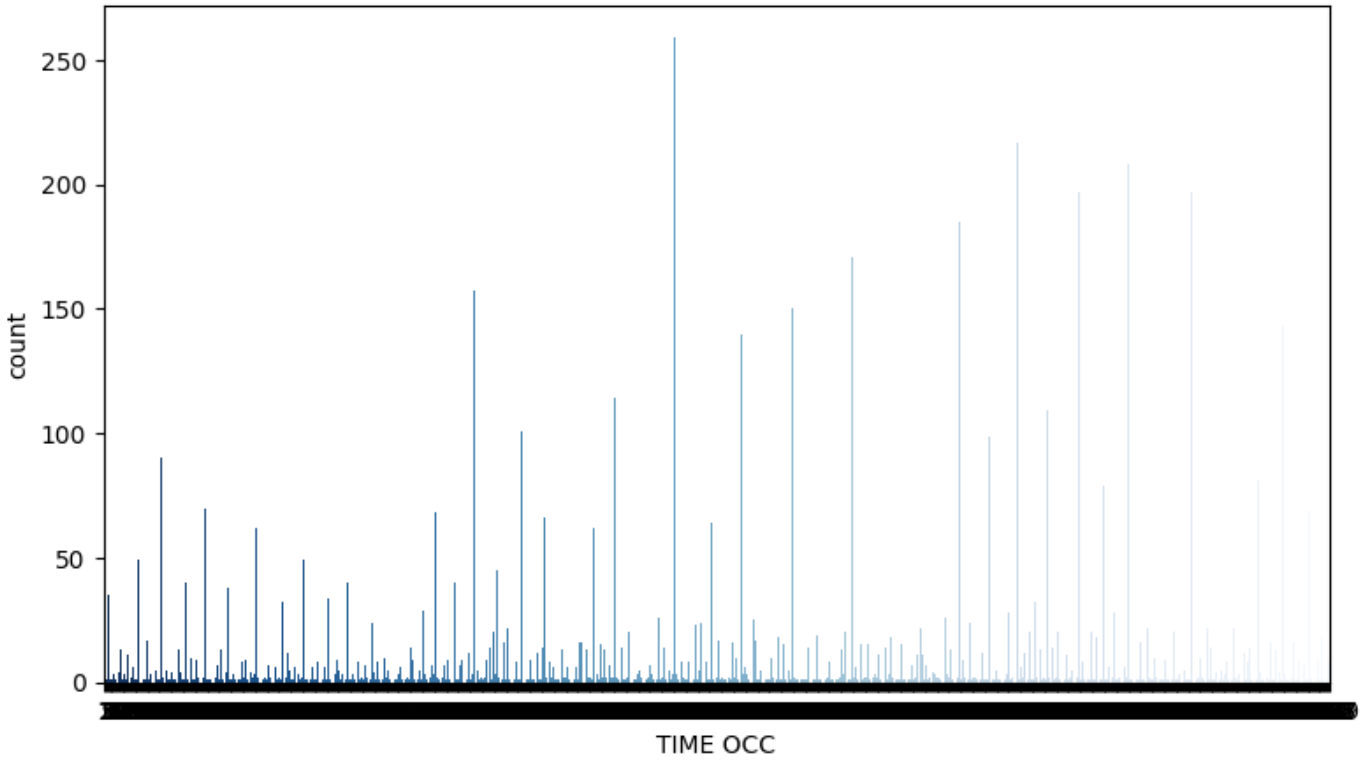
This code performs visual analysis on non-numeric (categorical) columns in the dataset. Specifically, it utilizes the Seaborn library to create bar plots for each non-numeric column, displaying the counts for each category.

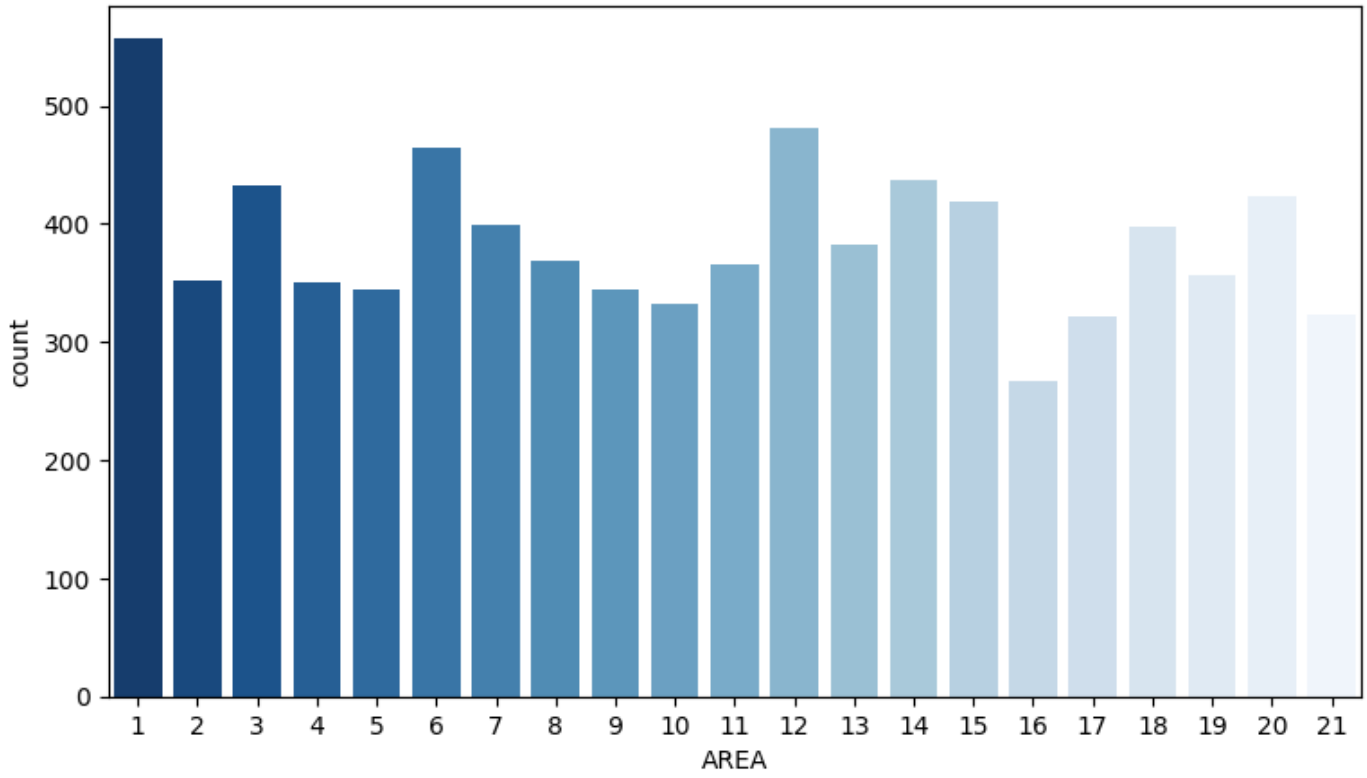
In [ ]:

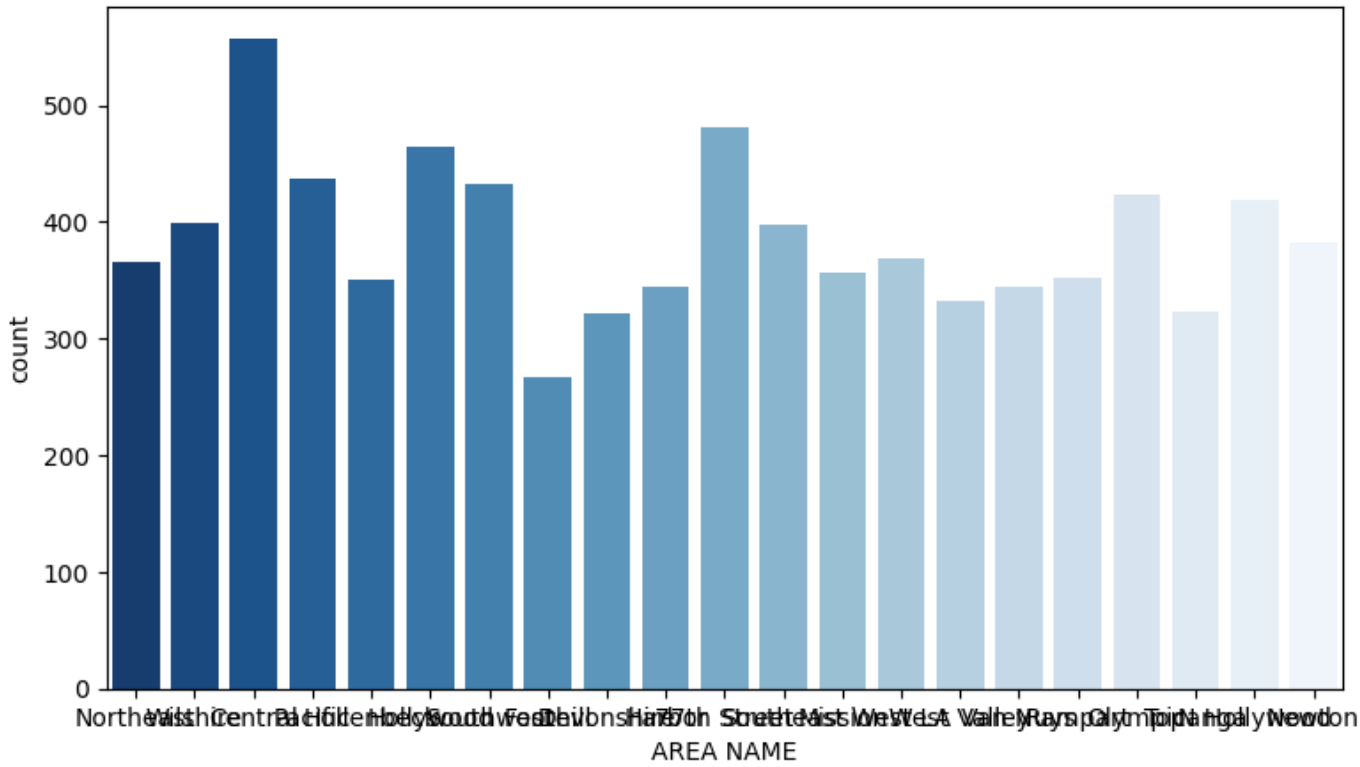
```
columns_nonnumeric=['Date Rptd','DATE OCC','TIME OCC','AREA','AREA NAME','Rpt Dist No','
for column in columns_nonnumeric:
    plt.figure(figsize=(9,5))
    sns.countplot(data=crime, x=column,palette="Blues_r")
```

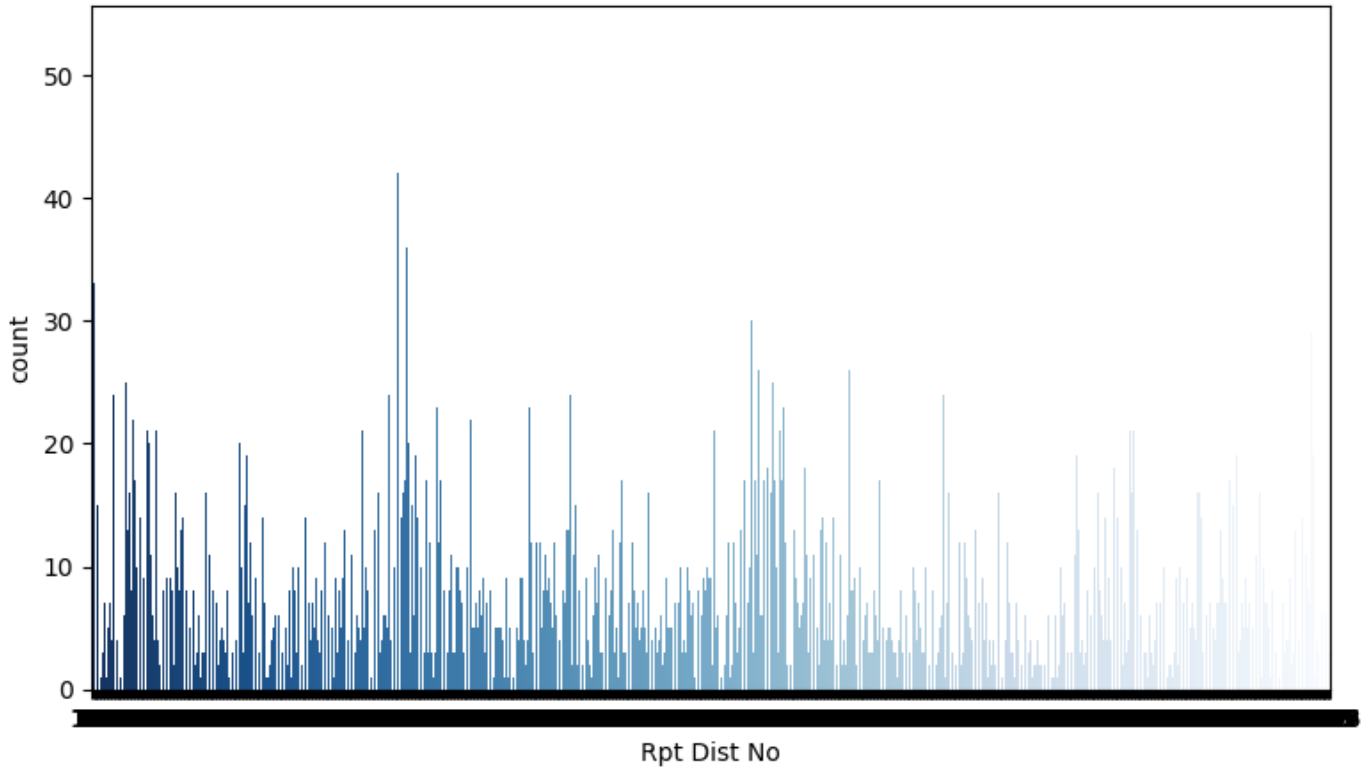


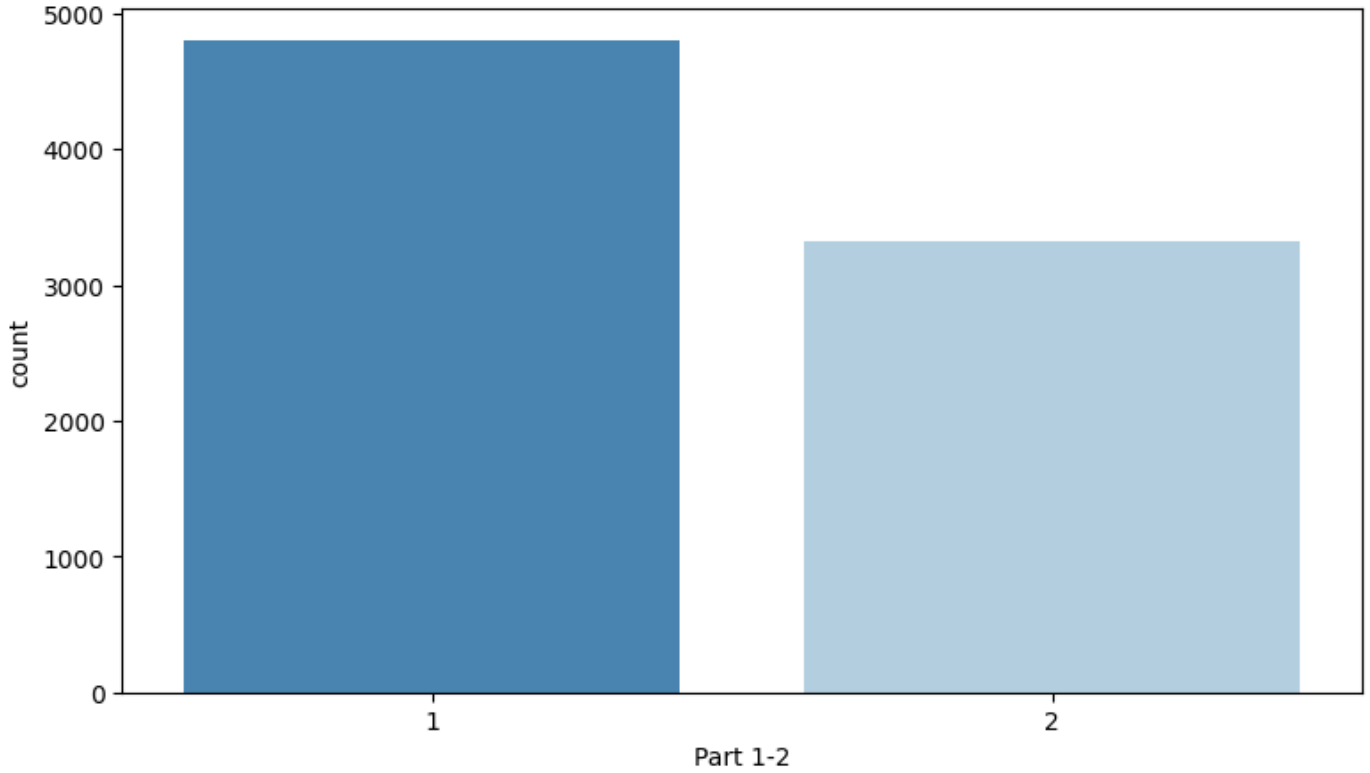

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
count
DATE OCC

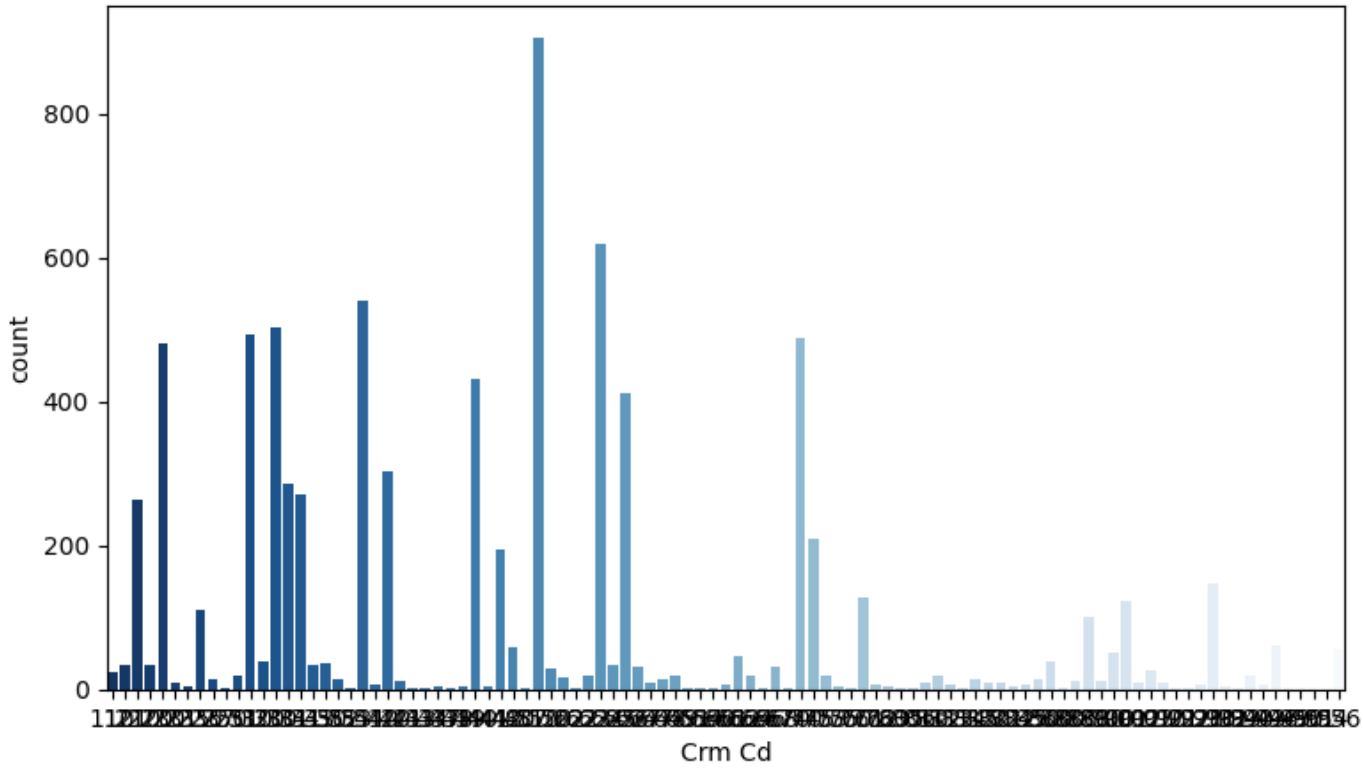

250
200
150
100
50
0
count
TIME OCC

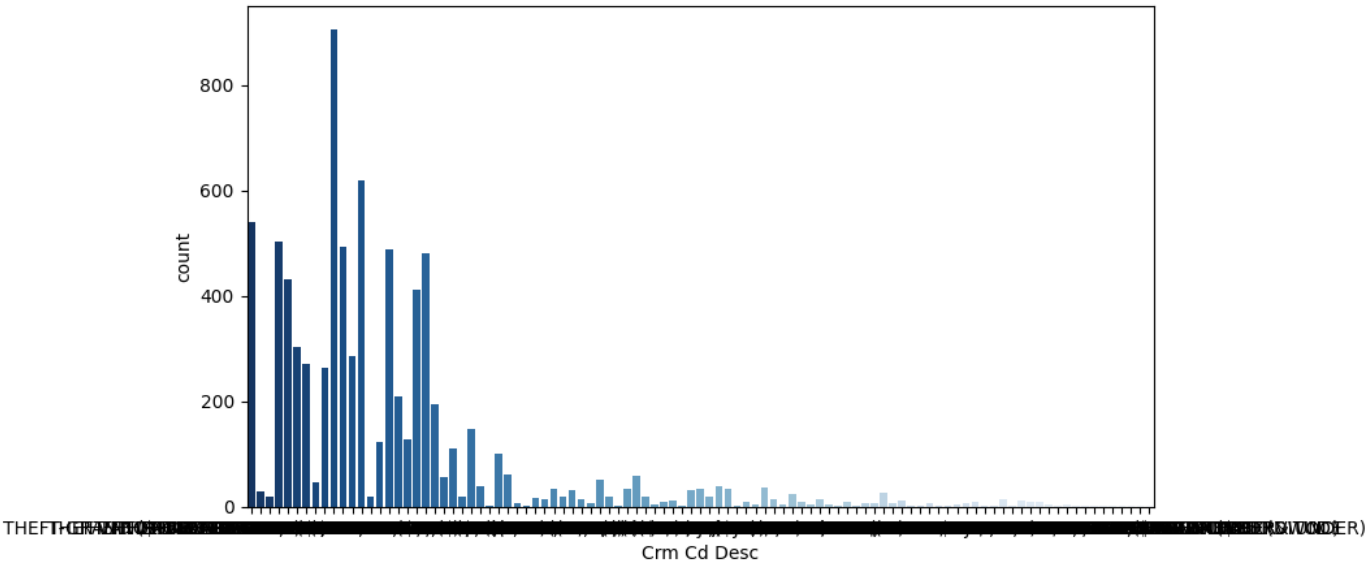

count
0
100
200
300
400
500
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
AREA

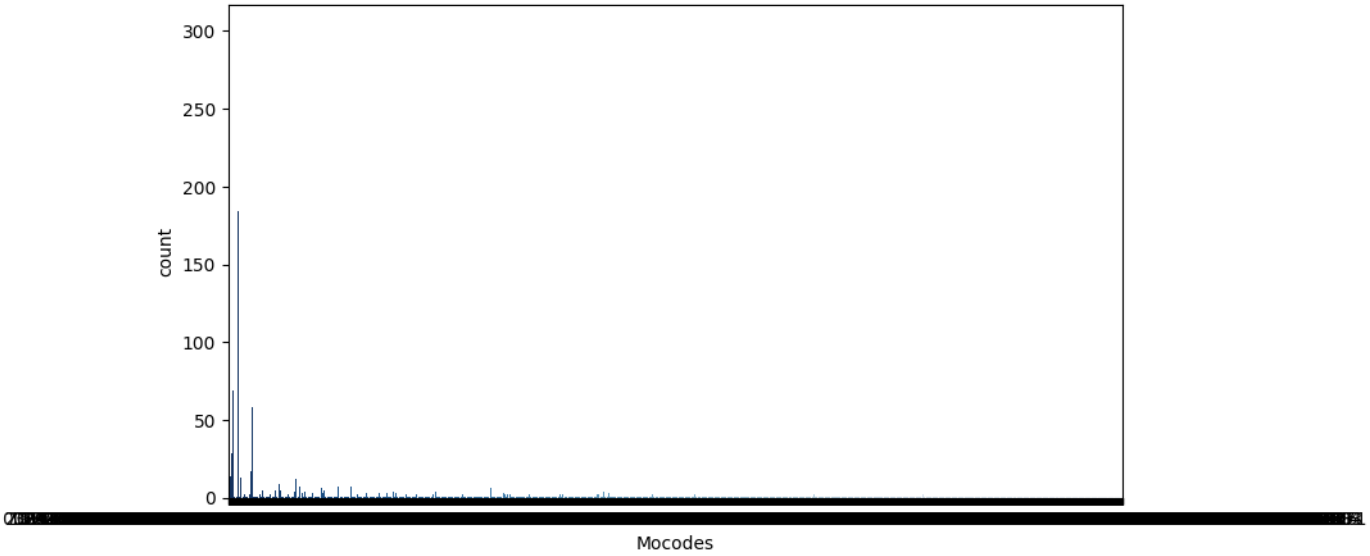

count
0
100
200
300
400
500
AREA NAME

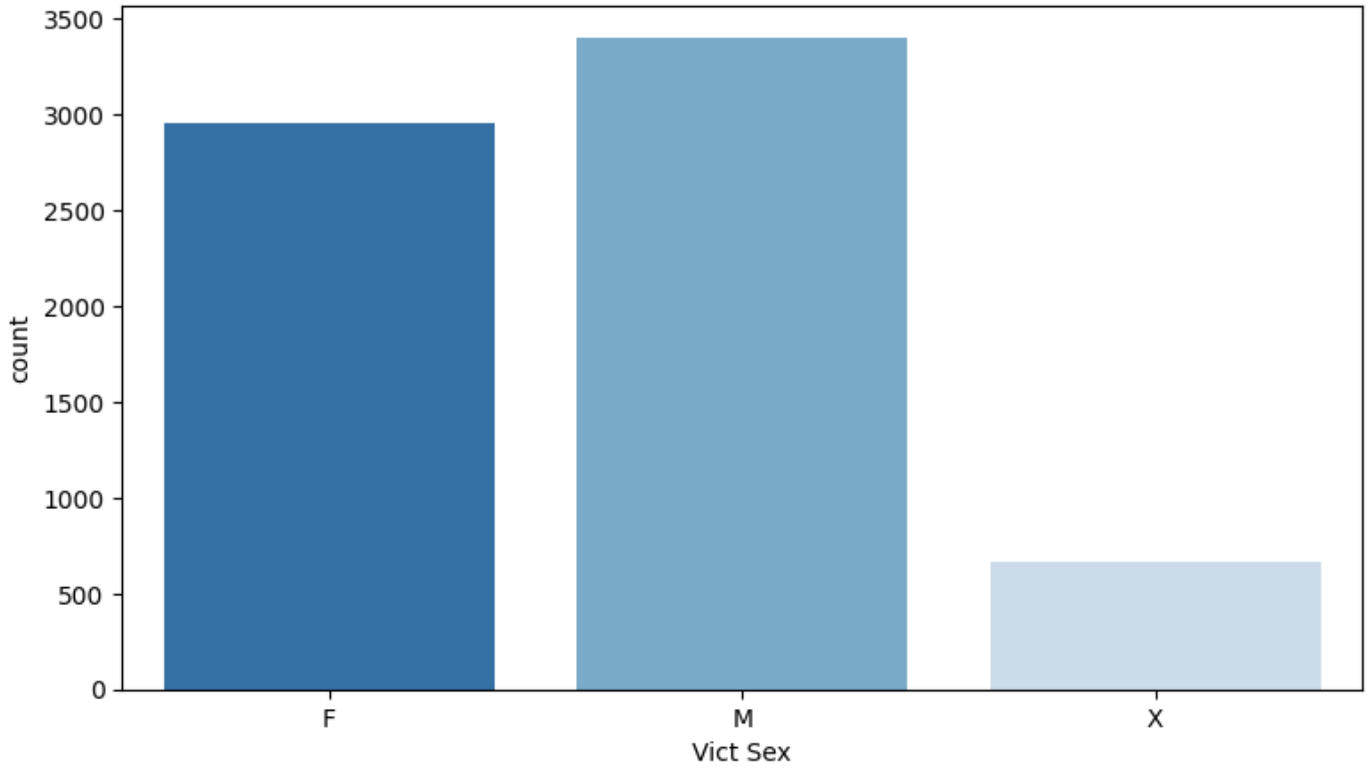

50
40
30
20
10
0
count
Rpt Dist No

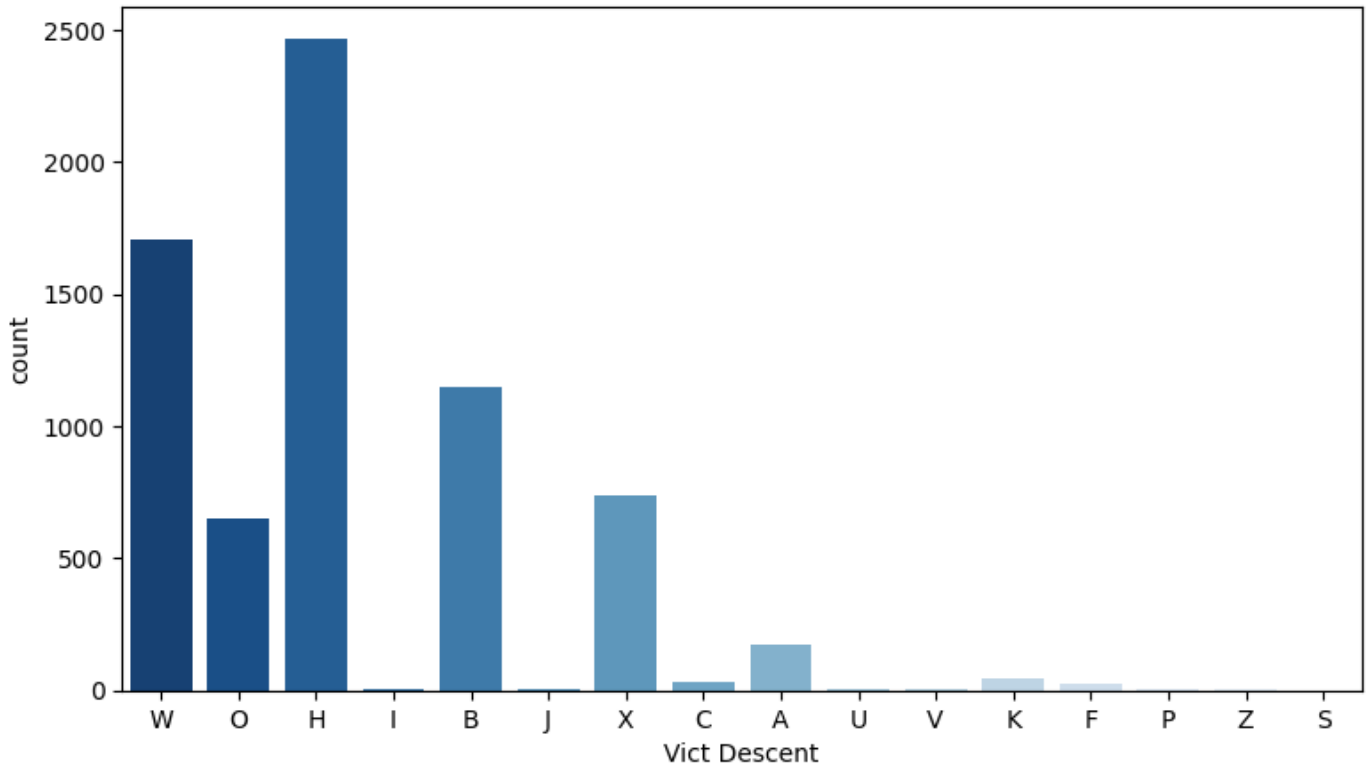

5000
4000
3000
2000
1000
0
count
1
2
Part 1-2

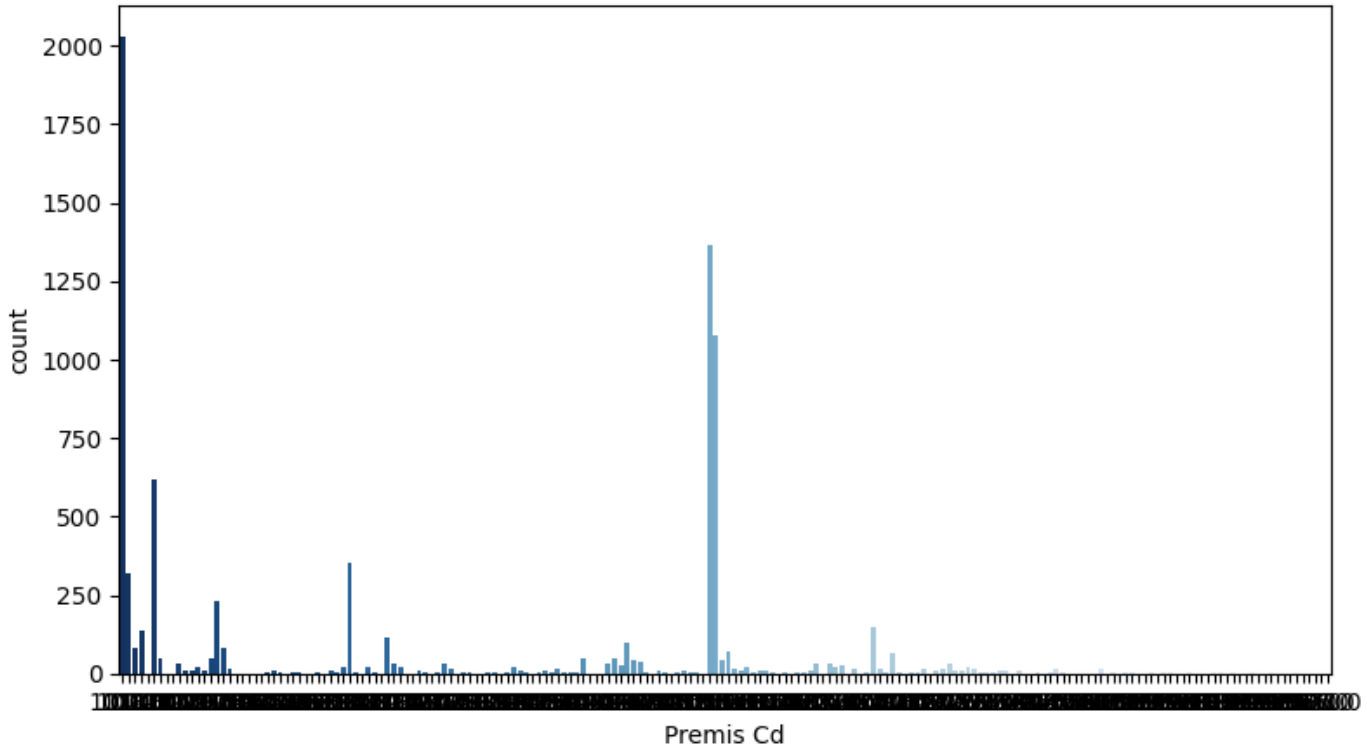

800
600
400
200
0
count
Crm Cd

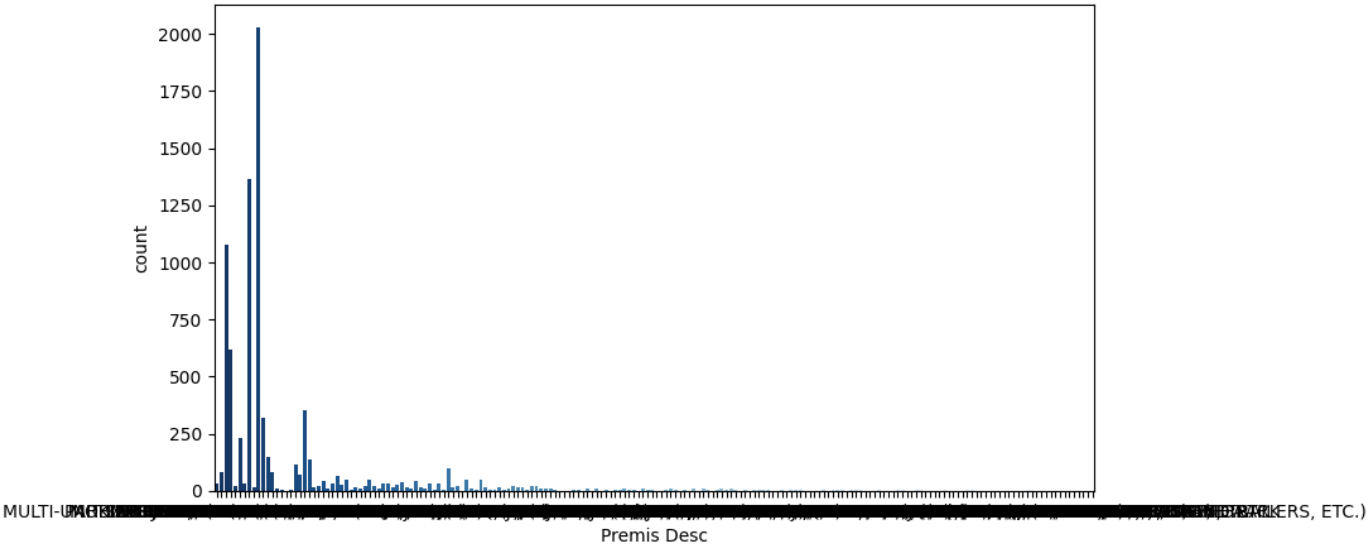

800
600
400
200
0
count
Crm Cd Desc

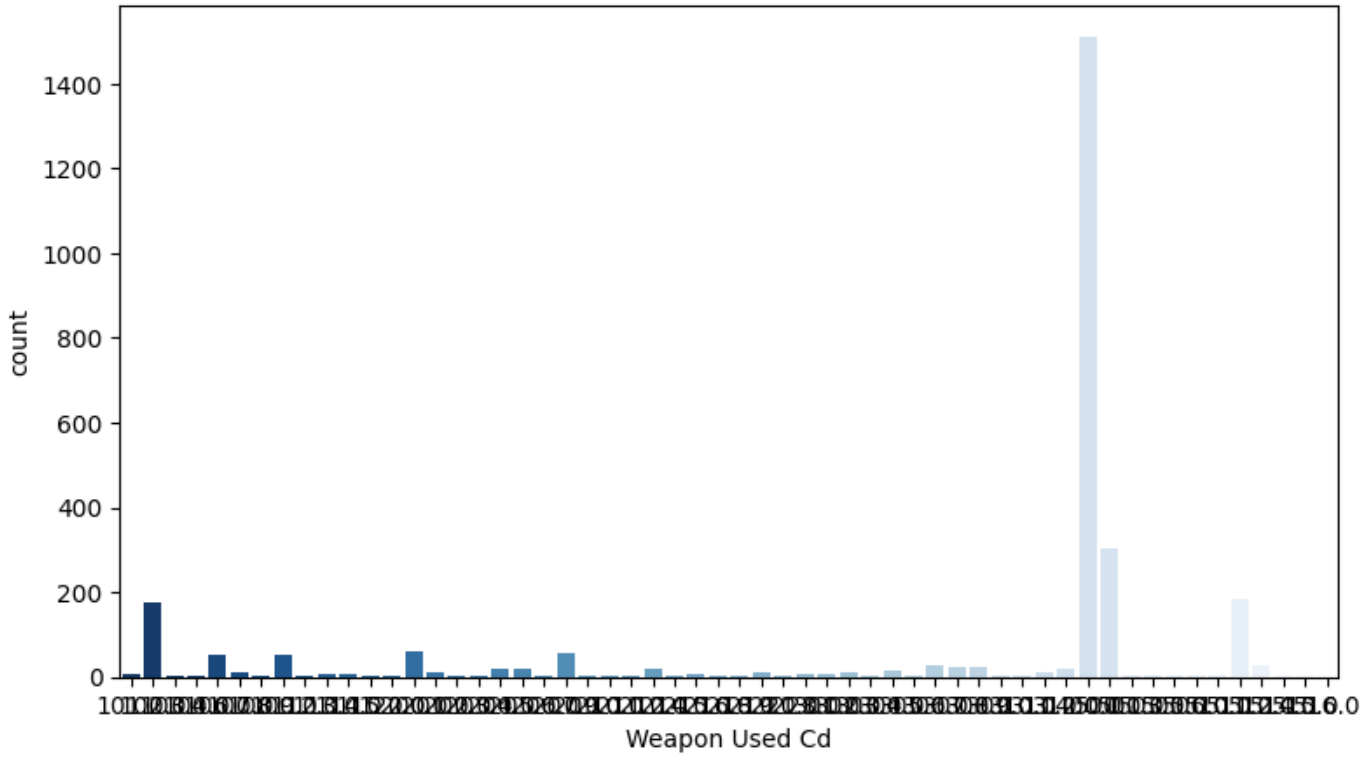

300
250
200
150
100
50
0
count
Mocodes

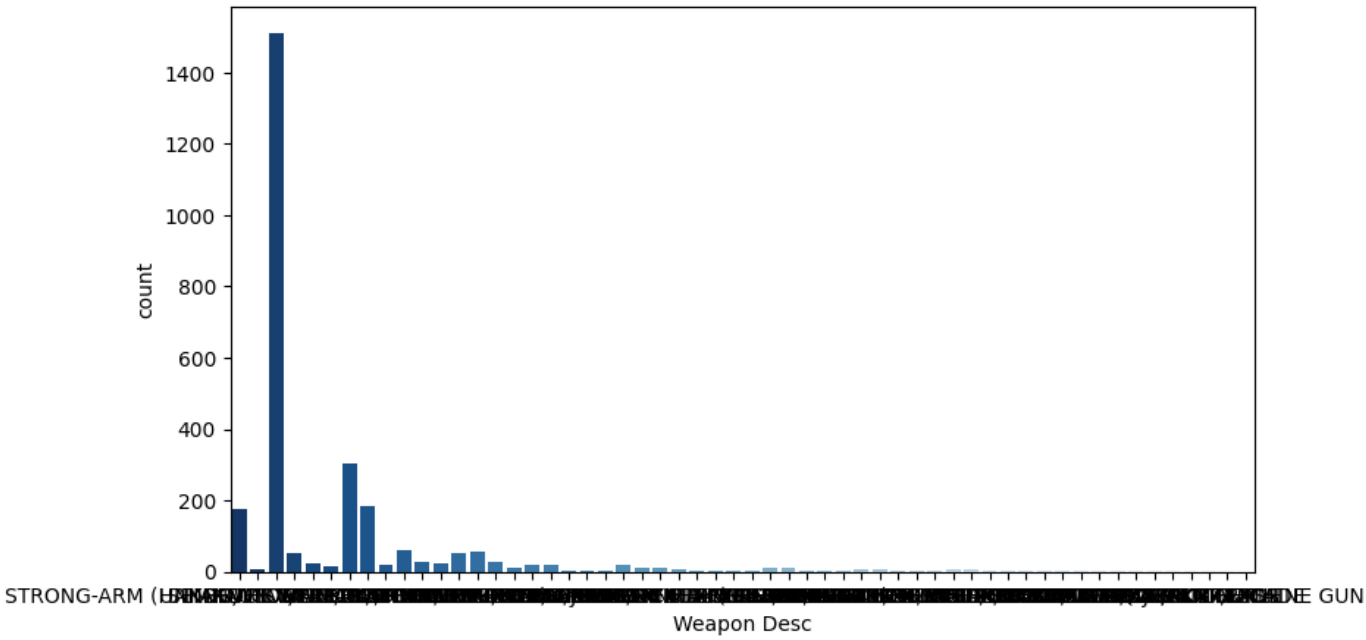

3500
3000
2500
2000
1500
1000
500
0
count
F
M
X
Vict Sex

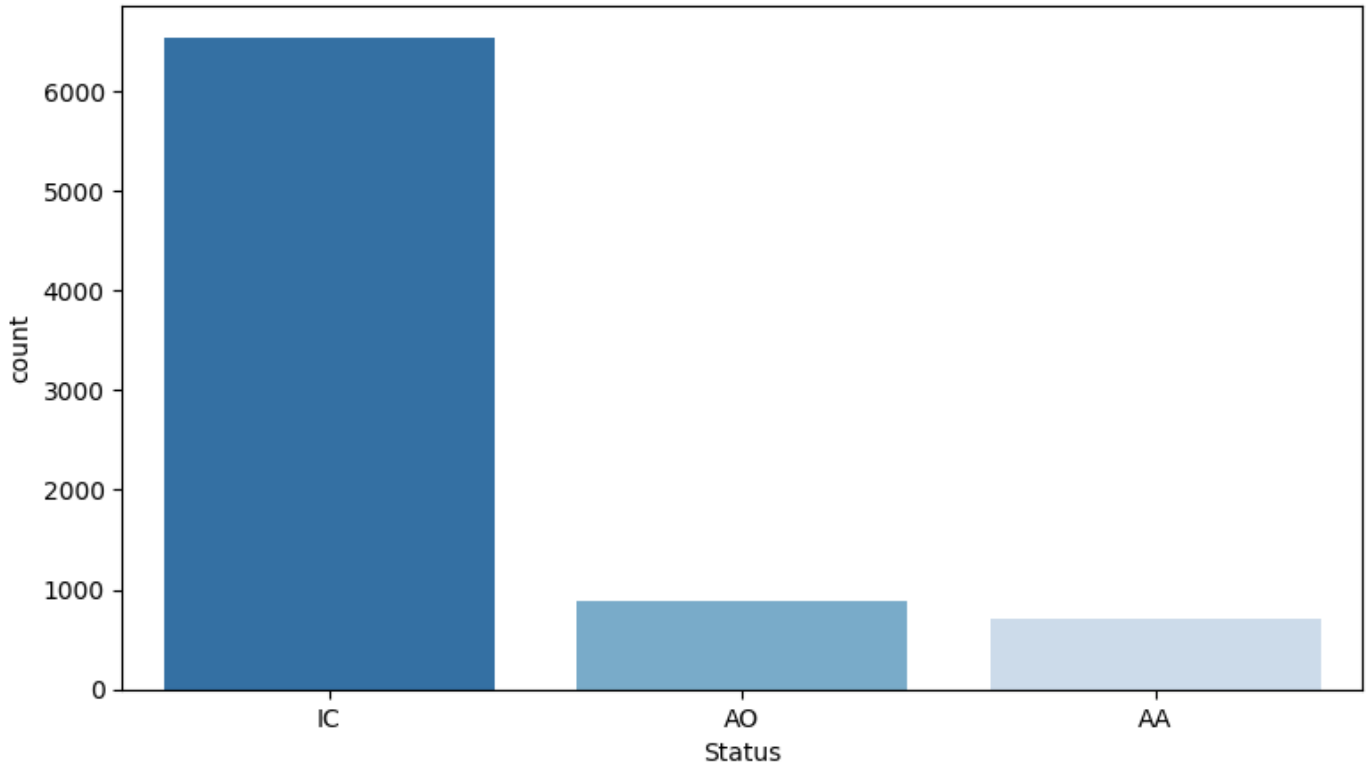

2500
2000
1500
1000
500
0
count
W
O
H
I
B
J
X
C
A
U
V
K
F
P
Z
S
Vict Descent

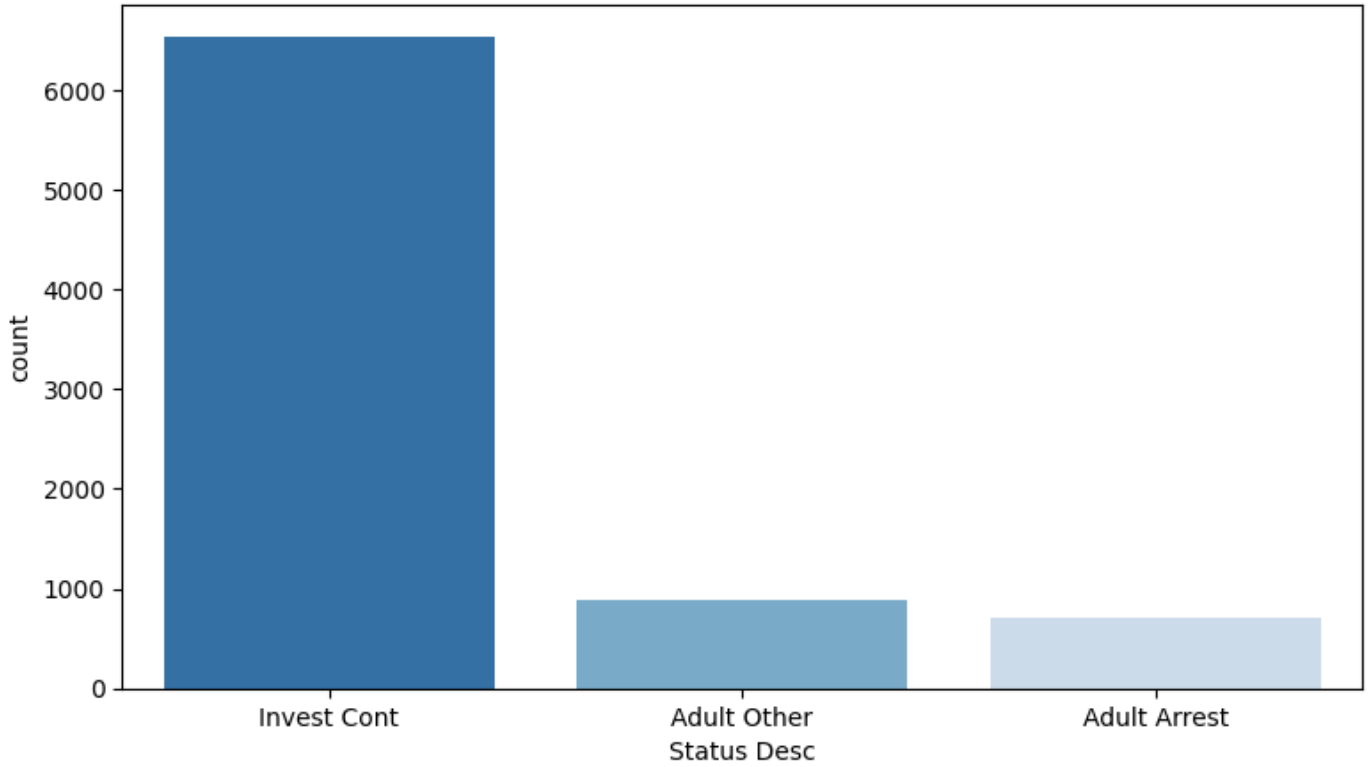

2000
1750
1500
1250
1000
750
500
250
0
count
Premis Cd

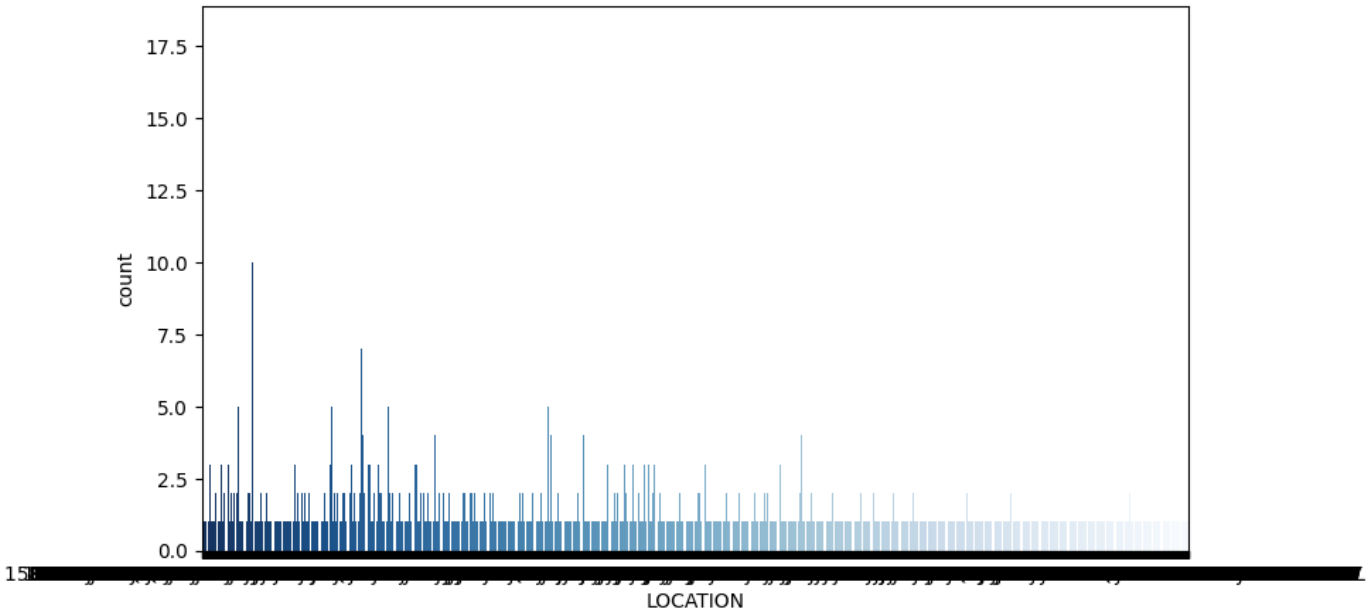

2000
1750
1500
1250
1000
750
500
250
0
count
ERS, ETC.)
Premis Desc

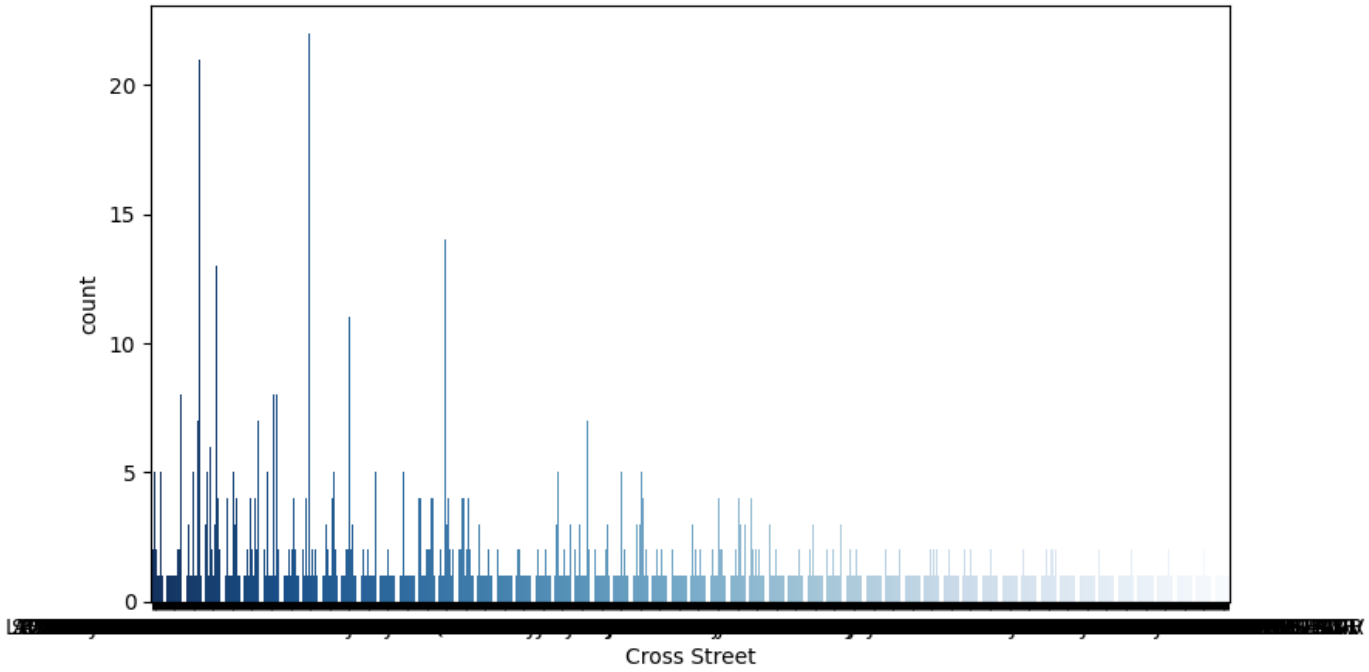

1400
1200
1000
800
600
400
200
0
count
Weapon Used Cd


1400
1200
1000
800
600
400
200
0
count
Weapon Desc


6000
5000
4000
3000
2000
1000
0
count
IC
AO
AA
Status

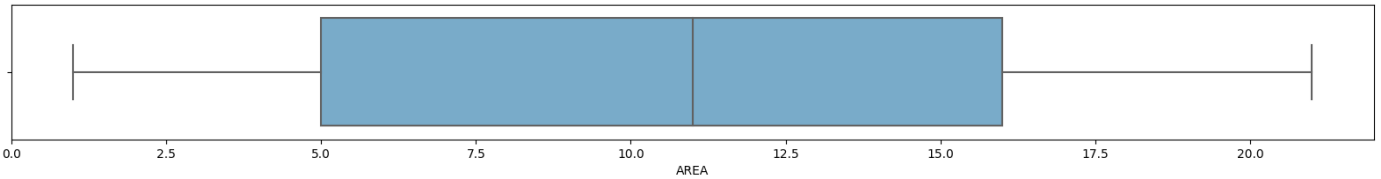

6000
5000
4000
3000
2000
1000
0
count
Invest Cont
Adult Other
Adult Arrest
Status Desc

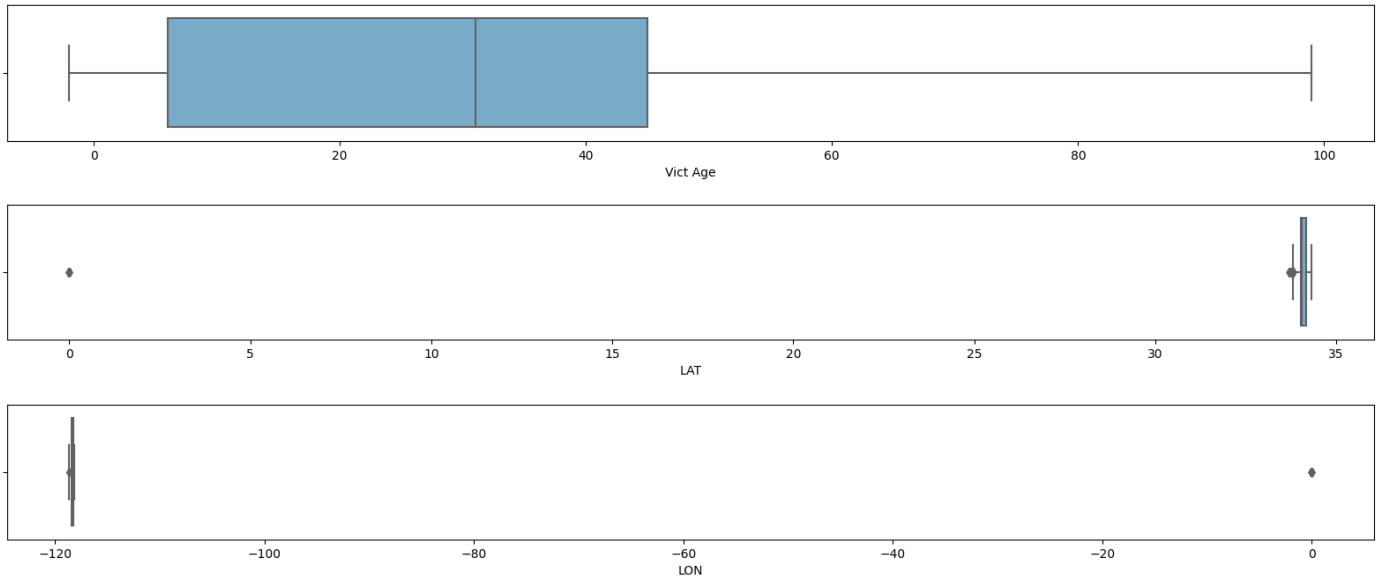




This code performs visual analysis on numeric columns in the dataset. It utilizes the Seaborn library to create boxplots for each numeric column, displaying the distribution characteristics of numerical values. The boxplots are employed to observe and understand the statistical information of these numeric values within each column.

In [ ]:
```
columns_numeric=['AREA','Vict Age','LAT','LON']
for column in columns_numeric:
        plt.figure(figsize=(20,2))
        sns.boxplot(data=crime, x=column,palette="Blues_r")
```





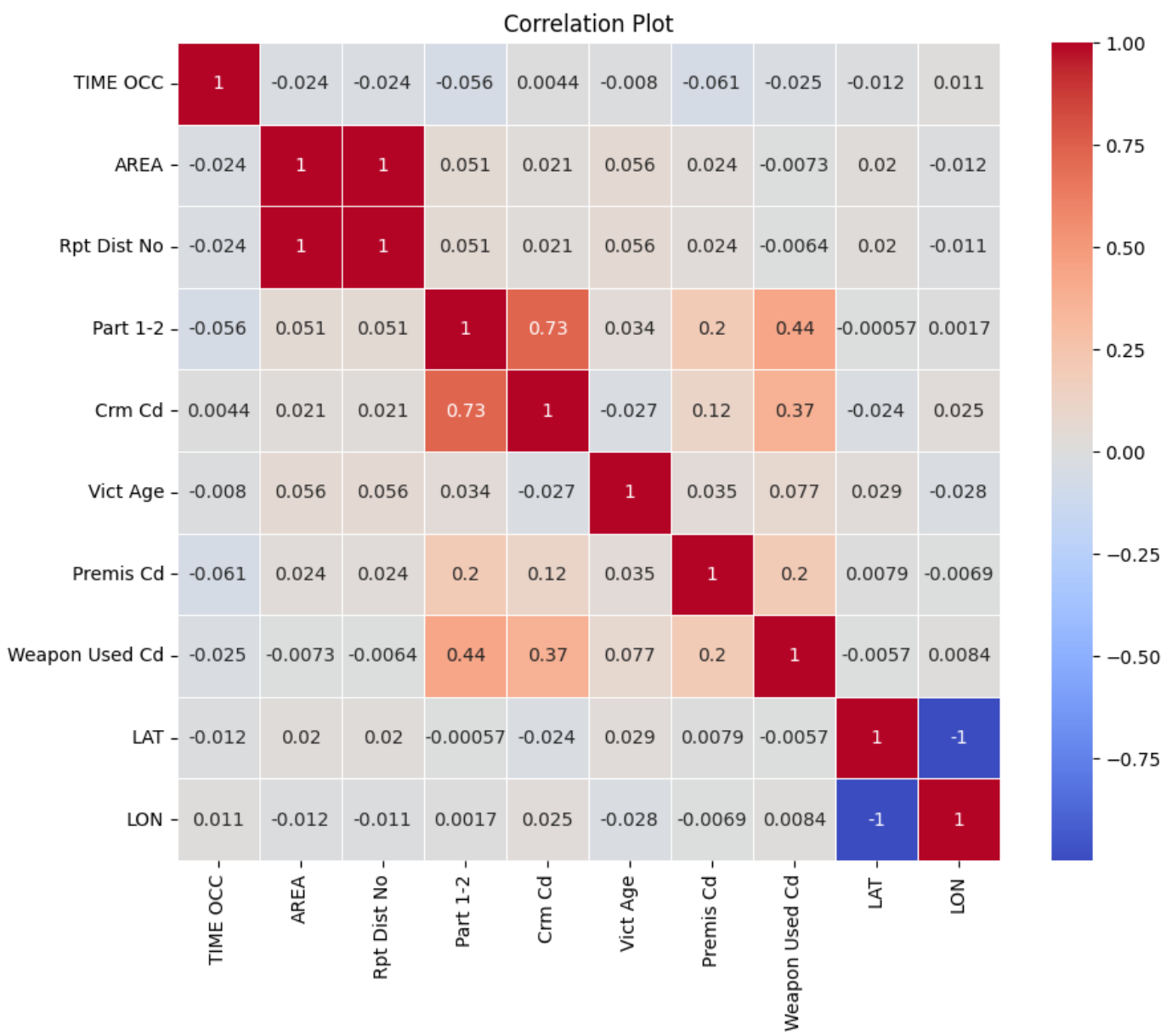
Correlation Matrix

Understand the relationships between numerical features through correlation analysis. Subsequently, it divides the dataset into sets of features and a target variable, preparing the data for the upcoming machine learning modeling. The visualization of the correlation matrix assists in identifying potential patterns and relationships between features.

In [ ]:
```
crime = crime.drop(columns=['DR_NO', 'Date Rptd', 'AREA NAME', 'Crm Cd Desc', 'Premis De
                            'Crm Cd 1', 'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4'])
crime = crime.dropna(subset=['Vict Sex', 'Vict Descent'])
crime = crime[~crime['Status'].isin(['JA', 'JO', 'CC'])]
#crime = crime.dropna()
crime = crime.reset_index(drop=True)
correlation_matrix = crime.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Plot")
plt.show()

# Split the data into training and testing sets
X = crime.drop("Status", axis=1)  # Features
y = crime["Status"]  # Target variable
```

```
<ipython-input-33-a7a0a6e6c13f>:7: FutureWarning: The default value of numeric_only in D
ataFrame.corr is deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = crime.corr()
```



Preprocesses the dataset using NumPy and Pandas libraries, along with the OrdinalEncoder class. It converts categorical variables 'Vict Sex' and 'Vict Descent' into integer encodings, handles missing values in the 'Weapon Used Cd' column, and similarly encodes the target variable y. These steps are aimed at providing more suitable input data for a machine learning model.

Ordinal Encoding

In [ ]:
```
unique_vict_sex = np.unique(X['Vict Sex'].astype(str))
unique_vict_descent = np.unique(X['Vict Descent'].astype(str))
unique_status = np.unique(y.astype(str))

x_encoder = OrdinalEncoder(categories=[unique_vict_sex, unique_vict_descent], encoded_mi
vict_encoded = x_encoder.fit_transform(X[['Vict Sex', 'Vict Descent']])
for idx in range(len(X)):
    X.loc[idx, 'Vict Sex'] = vict_encoded[idx][0]
    X.loc[idx, 'Vict Descent'] = vict_encoded[idx][1]

for idx, row in X.iterrows():
    if pd.isna(row['Weapon Used Cd']):
        X.loc[idx, 'Weapon Used Cd'] = -1

y_encoder = OrdinalEncoder(categories=[unique_status], encoded_missing_value=-1, dtype=n
```

```
reshaped_y = y.values.reshape(-1, 1)
y_encoded = y_encoder.fit_transform(reshaped_y)
y_encoded = y_encoded.reshape(y.values.shape[0])
X
```

Out[ ]:

	TIME OCC	AREA	Rpt Dist No	Part 1-2	Crm Cd	Vict Age	Vict Sex	Vict Descent	Premis Cd	Weapon Used Cd	LAT	LON
0	1200	11	1132	2	354	33	0	13	602.0	-1.0	34.1131	-118.2684
1	2000	7	745	1	520	49	0	8	123.0	-1.0	34.0624	-118.3510
2	1745	11	1127	2	940	17	1	4	502.0	-1.0	34.1194	-118.1889
3	1915	1	171	1	330	22	1	8	108.0	-1.0	34.0462	-118.2628
4	1900	14	1463	1	330	25	0	13	108.0	-1.0	33.9762	-118.4287
...	...	...	...	...	...	...	...	...	...	...	...	...
7014	1940	10	1043	2	930	29	0	13	502.0	511.0	34.1769	-118.5384
7015	1655	13	1394	2	626	26	1	4	101.0	400.0	33.9769	-118.2739
7016	906	10	1017	1	440	43	0	13	501.0	-1.0	34.2028	-118.5022
7017	1400	16	1654	2	901	63	0	4	501.0	-1.0	34.2528	-118.4041
7018	2300	6	636	1	236	25	0	1	502.0	400.0	34.1016	-118.3374

7019 rows × 12 columns

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_
```

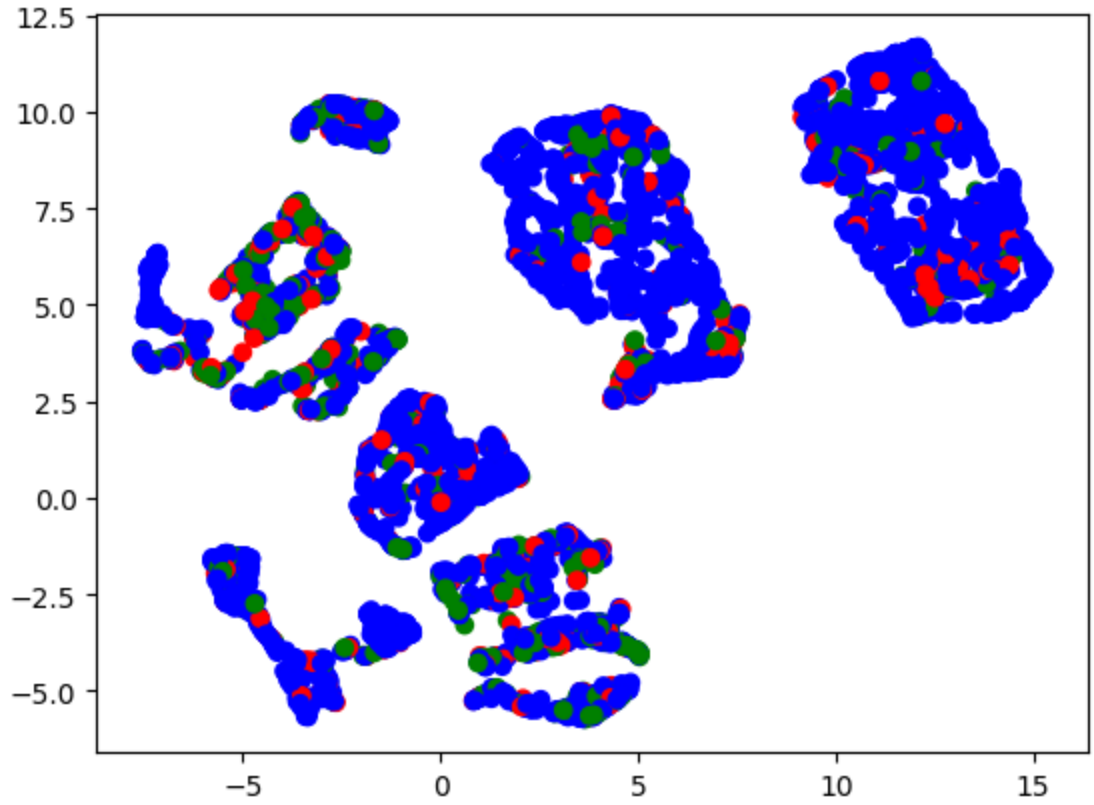
UMAP

Utilizes the UMAP (Uniform Manifold Approximation and Projection) algorithm to reduce the dimensionality of both training data (X_train) and testing data (X_test) to a two-dimensional space for visualization. The scatter plot depicts the distribution of training data in the two-dimensional space, with different classes represented by colors (red, green, and blue). It's important to note that when transforming the testing data, the existing UMAP model fitted on the training data should be used for consistency, ensuring a coherent representation.

In [ ]:
```
umap_model = umap.UMAP(n_components=2)
train_umap = umap_model.fit_transform(X_train)
test_umap = umap_model.fit_transform(X_test)
```

In [ ]:
```
colors = ['red', 'green', 'blue']
plt.scatter(train_umap[:,0], train_umap[:,1], c=[colors[i] for i in y_train])
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x784ee0ca2020>
```



The dispersion of samples from the same cluster into different categories or the clustering of different categories may suggest that the dimensionality reduction algorithm (such as UMAP) might not have effectively captured the intricate structure or non-linear relationships within the data. Fine-tuning UMAP hyperparameters, such as the number of neighbors and minimum distance, could be attempted to improve the representation. However, it's essential to consider whether the data itself exhibits complex structures, and further domain knowledge and feature engineering may be required to understand and address this complexity.

Neural Network

Utilizes the MLPClassifier from scikit-learn to train multiple Multi-Layer Perceptron (MLP) models. It iteratively tests various combinations of activation functions, hidden layer sizes, and hidden neuron counts. For each combination, the model is trained and performance metrics such as iteration count, F1 score, and confusion matrix are outputted. This process aids in identifying the optimal MLP model configuration for a given dataset.

In [ ]:
```
func = ['identity','logistic','relu','tanh']
n_layers = range(1,5,1)
n_hidden_neuron = range(5,20,5)
comb_list = itertools.product(func, n_layers, n_hidden_neuron)

for comb in comb_list:
    print(f'Now testing function {comb[0]}, {comb[1]} layers, {comb[2]} hidden neurons')
    mlp = MLPClassifier(hidden_layer_sizes=(comb[2],)*comb[1], activation=comb[0], verbo
    mlp = mlp.fit(train_umap, y_train)
    y_pred = mlp.predict(test_umap)
    print("iteration =", mlp.n_iter_)
    print("F1 score =,", metrics.f1_score(y_test, y_pred, average='micro'))
    print("Confusion Matrix =\n", metrics.confusion_matrix(y_test, y_pred))
```

```
Now testing function identity, 1 layers, 5 hidden neurons
iteration = 52
```

```
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 1 layers, 10 hidden neurons
iteration = 46
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 1 layers, 15 hidden neurons
iteration = 41
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 2 layers, 5 hidden neurons
iteration = 80
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 2 layers, 10 hidden neurons
iteration = 23
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 2 layers, 15 hidden neurons
iteration = 25
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 3 layers, 5 hidden neurons
iteration = 73
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 3 layers, 10 hidden neurons
iteration = 19
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 3 layers, 15 hidden neurons
iteration = 23
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 4 layers, 5 hidden neurons
iteration = 36
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
```

```
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 4 layers, 10 hidden neurons
iteration = 36
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function identity, 4 layers, 15 hidden neurons
iteration = 18
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 1 layers, 5 hidden neurons
iteration = 204
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 1 layers, 10 hidden neurons
iteration = 221
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 1 layers, 15 hidden neurons
iteration = 182
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 2 layers, 5 hidden neurons
iteration = 208
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 2 layers, 10 hidden neurons
iteration = 217
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 2 layers, 15 hidden neurons
iteration = 202
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 3 layers, 5 hidden neurons
iteration = 255
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 3 layers, 10 hidden neurons
```

```
iteration = 162
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 3 layers, 15 hidden neurons
iteration = 173
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 4 layers, 5 hidden neurons
iteration = 29
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 4 layers, 10 hidden neurons
iteration = 208
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function logistic, 4 layers, 15 hidden neurons
iteration = 149
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 1 layers, 5 hidden neurons
iteration = 115
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 1 layers, 10 hidden neurons
iteration = 187
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 1 layers, 15 hidden neurons
iteration = 134
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 2 layers, 5 hidden neurons
iteration = 208
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 2 layers, 10 hidden neurons
iteration = 138
F1 score =, 0.782051282051282
Confusion Matrix =
```

```
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 2 layers, 15 hidden neurons
iteration = 102
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 3 layers, 5 hidden neurons
iteration = 121
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 3 layers, 10 hidden neurons
iteration = 111
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 3 layers, 15 hidden neurons
iteration = 127
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 4 layers, 5 hidden neurons
iteration = 191
F1 score =, 0.7264957264957265
Confusion Matrix =
 [[   0    4  128]
 [   0    3  171]
 [   0   81 1017]]
Now testing function relu, 4 layers, 10 hidden neurons
iteration = 87
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function relu, 4 layers, 15 hidden neurons
iteration = 81
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function tanh, 1 layers, 5 hidden neurons
iteration = 121
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function tanh, 1 layers, 10 hidden neurons
iteration = 210
F1 score =, 0.6773504273504274
Confusion Matrix =
 [[  0  12 120]
 [  0   9 165]
 [  0 156 942]]
```

```
Now testing function tanh, 1 layers, 15 hidden neurons
iteration = 139
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function tanh, 2 layers, 5 hidden neurons
iteration = 153
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function tanh, 2 layers, 10 hidden neurons
iteration = 132
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function tanh, 2 layers, 15 hidden neurons
iteration = 110
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function tanh, 3 layers, 5 hidden neurons
iteration = 117
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function tanh, 3 layers, 10 hidden neurons
iteration = 103
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function tanh, 3 layers, 15 hidden neurons
iteration = 86
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function tanh, 4 layers, 5 hidden neurons
iteration = 69
F1 score =, 0.782051282051282
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
Now testing function tanh, 4 layers, 10 hidden neurons
iteration = 104
F1 score =, 0.6837606837606838
Confusion Matrix =
 [[  0   7 125]
 [  0   6 168]
 [  0 144 954]]
Now testing function tanh, 4 layers, 15 hidden neurons
iteration = 128
F1 score =, 0.782051282051282
```

```
Confusion Matrix =
 [[   0    0  132]
 [   0    0  174]
 [   0    0 1098]]
```

Systematically explores the performance of different configurations of MLP models by varying activation functions, hidden layer sizes, and neuron counts. While most configurations yield similar F1 scores around 0.78, there are variations in iteration counts and confusion matrices. Further fine-tuning may be necessary to optimize the model's performance on the specific dataset.

K-Fold Validation

Implements k-fold cross-validation for training and evaluating a Multilayer Perceptron (MLP) model. After reducing the dimensionality of the input data using UMAP, each fold undergoes weighted sampling. Multiple MLP models are trained and evaluated with different parameter combinations. The script outputs performance metrics such as iteration count, F1 score, and confusion matrix. The overall goal is to optimize the classification performance of the MLP on the reduced-dimensional data through cross-validation and model tuning.

In [ ]:
```
kf = KFold(n_splits=5)

data_umap = umap_model.fit_transform(X)

for train_index, test_index in kf.split(data_umap):
    X_train = data_umap[train_index]
    X_test = data_umap[test_index]
    y_train = y_encoded[train_index]
    y_test = y_encoded[test_index]

    n_train_sample = len(train_index)

    class_weights = []
    unique_status_value = np.unique(y_encoded)
    for i in range(len(unique_status_value)):
        count = np.count_nonzero(y_encoded == unique_status_value[i])
        ratio = count / len(train_index)
        weight = 1 - ratio
        class_weights.append(weight)

    data_weights = []
    for i in range(len(train_index)):
        data_weights.append(class_weights[y_encoded[train_index[i]]])

    data_weights = np.array(data_weights)
    data_weights /= np.sum(data_weights)

    selected_samples_indices = np.random.choice(train_index, size=len(train_index), p=da
    X_train_sampled = data_umap[selected_samples_indices]
    y_train_sampled = y_encoded[selected_samples_indices]

    func = ['identity','logistic','relu','tanh']
    n_layers = range(1,3,1)
    n_hidden_neuron = range(5,15,5)
    comb_list = itertools.product(func, n_layers, n_hidden_neuron)

    for comb in comb_list:
        print(f'Now testing function {comb[0]}, {comb[1]} layers, {comb[2]} hidden neuro
        mlp = MLPClassifier(hidden_layer_sizes=(comb[2],)*comb[1], activation=comb[0], v
        print(X_train_sampled.shape)
```

```
        mlp = mlp.fit(X_train_sampled, y_train_sampled)
        y_pred = mlp.predict(X_test)
        print("iteration =", mlp.n_iter_)
        print("F1 score =,", metrics.f1_score(y_test, y_pred, average='micro'))
        print("Confusion Matrix =\n", metrics.confusion_matrix(y_test, y_pred))
```

```
Now testing function identity, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 49
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [826 578   0]]
Now testing function identity, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 58
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [608 796   0]]
Now testing function identity, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 83
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [693 711   0]]
Now testing function identity, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 41
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [706 698   0]]
Now testing function logistic, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 52
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [786 618   0]]
Now testing function logistic, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 198
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [830 574   0]]
Now testing function logistic, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 59
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [790 614   0]]
Now testing function logistic, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 66
F1 score =, 0.0
```

```
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [807 597   0]]
Now testing function relu, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 262
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [829 575   0]]
Now testing function relu, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 184
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [928 476   0]]
Now testing function relu, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 198
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [866 538   0]]
Now testing function relu, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 166
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [925 479   0]]
Now testing function tanh, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 45
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [581 823   0]]
Now testing function tanh, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 198
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [901 503   0]]
Now testing function tanh, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 68
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [827 577   0]]
Now testing function tanh, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 218
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
```

```
 [  0   0   0]
 [860 544   0]]
Now testing function identity, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 51
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [633 771   0]]
Now testing function identity, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 53
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [782 622   0]]
Now testing function identity, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 73
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [702 702   0]]
Now testing function identity, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 55
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [817 587   0]]
Now testing function logistic, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 53
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [803 601   0]]
Now testing function logistic, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 114
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [828 576   0]]
Now testing function logistic, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 50
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [799 605   0]]
Now testing function logistic, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 193
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [834 570   0]]
```

```
Now testing function relu, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 238
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [839 565   0]]
Now testing function relu, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 221
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [807 597   0]]
Now testing function relu, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 110
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [813 591   0]]
Now testing function relu, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 165
F1 score =, 0.011396011396011395
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [776 612  16]]
Now testing function tanh, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 182
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [829 575   0]]
Now testing function tanh, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 186
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [732 672   0]]
Now testing function tanh, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 50
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [828 576   0]]
Now testing function tanh, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 164
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [744 660   0]]
Now testing function identity, 1 layers, 5 hidden neurons
(5615, 2)
```

```
iteration = 51
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [747 657   0]]
Now testing function identity, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 61
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [721 683   0]]
Now testing function identity, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 65
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [788 616   0]]
Now testing function identity, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 69
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [726 678   0]]
Now testing function logistic, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 56
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [804 600   0]]
Now testing function logistic, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 115
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [838 566   0]]
Now testing function logistic, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 66
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [817 587   0]]
Now testing function logistic, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 173
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [850 554   0]]
Now testing function relu, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 324
F1 score =, 0.0
```

```
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [836 568   0]]
Now testing function relu, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 180
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [981 423   0]]
Now testing function relu, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 188
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [848 556   0]]
Now testing function relu, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 141
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [845 559   0]]
Now testing function tanh, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 41
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [575 829   0]]
Now testing function tanh, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 163
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [848 556   0]]
Now testing function tanh, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 56
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [827 577   0]]
Now testing function tanh, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 162
F1 score =, 0.0
Confusion Matrix =
 [[  0   0   0]
 [  0   0   0]
 [854 550   0]]
Now testing function identity, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 44
F1 score =, 0.06623931623931624
Confusion Matrix =
 [[  0   0   0]
```

```
 [ 36  93   0]
 [770 505   0]]
Now testing function identity, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 53
F1 score =, 0.06623931623931624
Confusion Matrix =
 [[  0   0   0]
 [ 36  93   0]
 [770 505   0]]
Now testing function identity, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 76
F1 score =, 0.057692307692307696
Confusion Matrix =
 [[  0   0   0]
 [ 48  81   0]
 [820 455   0]]
Now testing function identity, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 47
F1 score =, 0.06623931623931624
Confusion Matrix =
 [[  0   0   0]
 [ 36  93   0]
 [775 500   0]]
Now testing function logistic, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 75
F1 score =, 0.06837606837606838
Confusion Matrix =
 [[  0   0   0]
 [ 33  96   0]
 [727 548   0]]
Now testing function logistic, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 146
F1 score =, 0.06837606837606838
Confusion Matrix =
 [[  0   0   0]
 [ 33  96   0]
 [729 546   0]]
Now testing function logistic, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 66
F1 score =, 0.06837606837606838
Confusion Matrix =
 [[  0   0   0]
 [ 33  96   0]
 [729 546   0]]
Now testing function logistic, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 69
F1 score =, 0.06837606837606838
Confusion Matrix =
 [[  0   0   0]
 [ 33  96   0]
 [738 537   0]]
Now testing function relu, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 190
F1 score =, 0.06623931623931624
Confusion Matrix =
 [[  0   0   0]
 [ 36  93   0]
 [754 521   0]]
```

```
Now testing function relu, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 214
F1 score =, 0.06196581196581197
Confusion Matrix =
 [[  0   0   0]
 [ 42  87   0]
 [905 370   0]]
Now testing function relu, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 115
F1 score =, 0.06766381766381767
Confusion Matrix =
 [[  0   0   0]
 [ 34  95   0]
 [741 534   0]]
Now testing function relu, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 175
F1 score =, 0.0633903133903134
Confusion Matrix =
 [[  0   0   0]
 [ 40  89   0]
 [838 437   0]]
Now testing function tanh, 1 layers, 5 hidden neurons
(5615, 2)
iteration = 141
F1 score =, 0.06837606837606838
Confusion Matrix =
 [[  0   0   0]
 [ 33  96   0]
 [723 552   0]]
Now testing function tanh, 1 layers, 10 hidden neurons
(5615, 2)
iteration = 109
F1 score =, 0.06552706552706553
Confusion Matrix =
 [[  0   0   0]
 [ 37  92   0]
 [800 475   0]]
Now testing function tanh, 2 layers, 5 hidden neurons
(5615, 2)
iteration = 61
F1 score =, 0.06837606837606838
Confusion Matrix =
 [[  0   0   0]
 [ 33  96   0]
 [735 540   0]]
Now testing function tanh, 2 layers, 10 hidden neurons
(5615, 2)
iteration = 234
F1 score =, 0.0641025641025641
Confusion Matrix =
 [[  0   0   0]
 [ 39  90   0]
 [808 467   0]]
Now testing function identity, 1 layers, 5 hidden neurons
(5616, 2)
iteration = 27
F1 score =, 0.3029223093371347
Confusion Matrix =
 [[  0 296 381]
 [  0 425 301]
 [  0   0   0]]
Now testing function identity, 1 layers, 10 hidden neurons
(5616, 2)
```

```
iteration = 35
F1 score =, 0.26585887384176765
Confusion Matrix =
 [[  0 251 426]
 [  0 373 353]
 [  0   0   0]]
Now testing function identity, 2 layers, 5 hidden neurons
(5616, 2)
iteration = 67
F1 score =, 0.27726300784034213
Confusion Matrix =
 [[  0 263 414]
 [  0 389 337]
 [  0   0   0]]
Now testing function identity, 2 layers, 10 hidden neurons
(5616, 2)
iteration = 27
F1 score =, 0.22238061297220243
Confusion Matrix =
 [[  0 220 457]
 [  0 312 414]
 [  0   0   0]]
Now testing function logistic, 1 layers, 5 hidden neurons
(5616, 2)
iteration = 440
F1 score =, 0.3556664290805417
Confusion Matrix =
 [[  0 306 371]
 [  0 499 227]
 [  0   0   0]]
Now testing function logistic, 1 layers, 10 hidden neurons
(5616, 2)
iteration = 317
F1 score =, 0.3378474697077691
Confusion Matrix =
 [[  0 298 379]
 [  0 474 252]
 [  0   0   0]]
Now testing function logistic, 2 layers, 5 hidden neurons
(5616, 2)
iteration = 90
F1 score =, 0.3955808980755524
Confusion Matrix =
 [[  0 372 305]
 [  0 555 171]
 [  0   0   0]]
Now testing function logistic, 2 layers, 10 hidden neurons
(5616, 2)
iteration = 195
F1 score =, 0.38987883107626514
Confusion Matrix =
 [[  0 364 313]
 [  0 547 179]
 [  0   0   0]]
Now testing function relu, 1 layers, 5 hidden neurons
(5616, 2)
iteration = 300
F1 score =, 0.3884533143264433
Confusion Matrix =
 [[  0 363 314]
 [  0 545 181]
 [  0   0   0]]
Now testing function relu, 1 layers, 10 hidden neurons
(5616, 2)
iteration = 282
F1 score =, 0.36279401282965074
```

```
Confusion Matrix =
 [[  0 318 359]
 [  0 509 217]
 [  0   0   0]]
Now testing function relu, 2 layers, 5 hidden neurons
(5616, 2)
iteration = 143
F1 score =, 0.3321454027084818
Confusion Matrix =
 [[  0 296 381]
 [  0 466 260]
 [  0   0   0]]
Now testing function relu, 2 layers, 10 hidden neurons
(5616, 2)
iteration = 240
F1 score =, 0.27726300784034213
Confusion Matrix =
 [[  0 235 442]
 [  0 389 337]
 [  0   0   0]]
Now testing function tanh, 1 layers, 5 hidden neurons
(5616, 2)
iteration = 102
F1 score =, 0.3941553813257306
Confusion Matrix =
 [[  0 370 307]
 [  0 553 173]
 [  0   0   0]]
Now testing function tanh, 1 layers, 10 hidden neurons
(5616, 2)
iteration = 347
F1 score =, 0.34640057020669995
Confusion Matrix =
 [[  0 313 364]
 [  0 486 240]
 [  0   0   0]]
Now testing function tanh, 2 layers, 5 hidden neurons
(5616, 2)
iteration = 272
F1 score =, 0.3406985032074127
Confusion Matrix =
 [[  0 299 378]
 [  0 478 248]
 [  0   0   0]]
Now testing function tanh, 2 layers, 10 hidden neurons
(5616, 2)
iteration = 412
F1 score =, 0.3570919458303635
Confusion Matrix =
 [[  0 316 361]
 [  0 501 225]
 [  0   0   0]]
```

Conclusion

We utilizes k-fold cross-validation to train and evaluate Multi-Layer Perceptron (MLP) models with various configurations, including different activation functions (identity, logistic, relu, tanh), numbers of hidden layers (1 or 2 layers), and varying hidden neuron counts. However, the majority of configurations exhibit F1 scores of 0, indicating suboptimal learning of data patterns. Possible reasons include insufficient data features, inappropriate model configurations, or the need for more training iterations.

The overall results highlight the complex task of improving model performance, requiring careful tuning and optimization.

GitHub Link

https://github.com/YFA22SCM49L/IITFA23CS584Project.git