```
#Importing the Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Community_Crime_Statistics_20240524.csv to
Community_Crime_Statistics_20240524 (1).csv
```

```
df = pd.read_csv('/content/Community_Crime_Statistics_20240524.csv')
df.head()
```


{"summary":"{\n \"name\": \"df\",\n \"rows\": 70661,\n \"fields\":
[\n   {\n   \"column\": \"Community\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 296,\n
\"samples\": [\n   \"UNIVERSITY OF CALGARY\",\n   \"LAKE
BONAVISTA\",\n   \"COUNTRY HILLS VILLAGE\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Category\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
9,\n   \"samples\": [\n   \"Commercial Robbery\",\n
\"Break & Enter - Commercial\",\n   \"Violence\\u00a0 'Other'
(Non-domestic)\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Crime Count\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 3,\n   \"min\": 1,\n
\"max\": 111,\n   \"num_unique_values\": 77,\n
\"samples\": [\n   5,\n   39,\n   10\n   ],\
n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Year\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 1,\n
\"min\": 2018,\n   \"max\": 2024,\n   \"num_unique_values\":
7,\n   \"samples\": [\n   2022,\n   2019,\n
2021\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Month\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 3,\n   \"min\": 1,\n   \"max\": 12,\n
\"num_unique_values\": 12,\n   \"samples\": [\n   5,\n
9,\n   11\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}


```
df.tail()
```


{"summary":"{\n \"name\": \"df\",\n \"rows\": 5,\n \"fields\": [\n
{\n   \"column\": \"Community\",\n   \"properties\": {\n

```
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"01H\",\n          \"SCARBORO/ SUNALTA
WEST\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Category\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Theft OF Vehicle\",\n          \"Violence\\u00a0
'Other' (Non-domestic)\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Crime Count\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 1,\n
\"max\": 1,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 2018,\n        \"max\": 2021,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          2018\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Month\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 4,\n
\"min\": 1,\n        \"max\": 11,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```


```
#shape of the dataset
df.shape
```

```
(70661, 5)
```

```
#checking for missing values
df.isnull().sum()
```

```
Community      0
Category       0
Crime Count    0
Year           0
Month          0
dtype: int64
```

```
#checking for the datatypes
df.dtypes
```

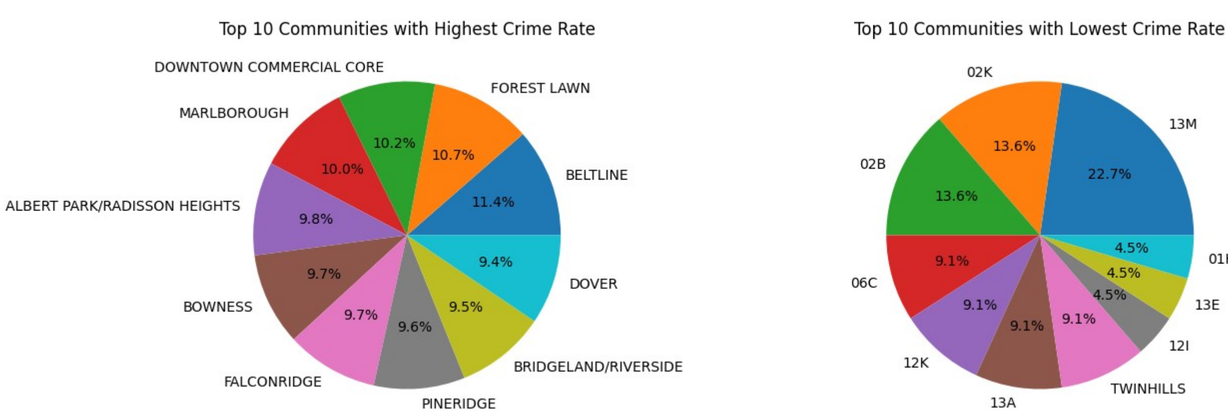
```
Community      object
Category       object
Crime Count     int64
Year            int64
Month           int64
dtype: object
```

```
#describing the data
df.describe()
```


{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n
{\n   \"column\": \"Crime Count\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 24976.175893301483,\n
\"min\": 1.0,\n   \"max\": 70661.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   70661.0,\
n   2.855747866574206,\n   3.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Year\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 24380.196590501935,\n
\"min\": 1.8253299721556513,\n   \"max\": 70661.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
2020.6186156437072,\n   2021.0,\n   70661.0\n   ],\
n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Month\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
24980.37462098201,\n   \"min\": 1.0,\n   \"max\": 70661.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
6.3692418731690745,\n   6.0,\n   70661.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

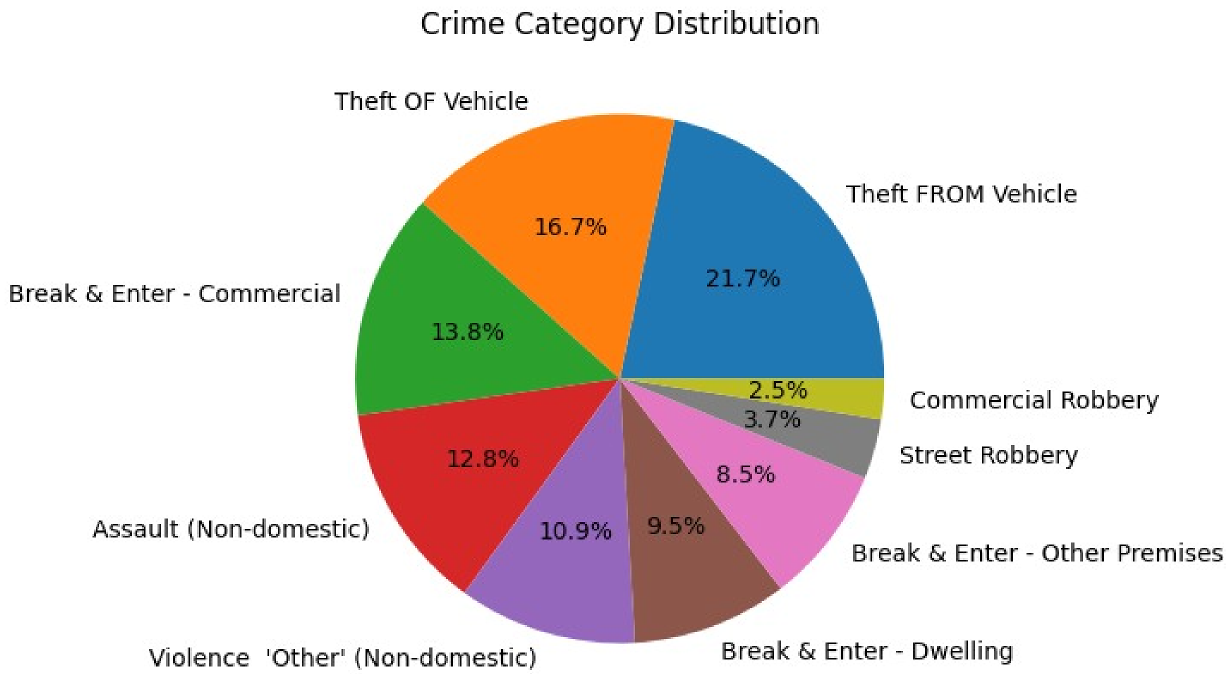

```
fig, ax = plt.subplots(1, 2, figsize=(15, 5))
#Top 10 Communities with Highest Crime Rate
df['Community'].value_counts().head(10).plot.pie(autopct='%1.1f%%', ax
= ax[0])
ax[0].set_title('Top 10 Communities with Highest Crime Rate')
ax[0].set_ylabel('')
#Top 10 Communities with Lowest Crime Rate
df['Community'].value_counts().tail(10).plot.pie(autopct='%1.1f%%', ax
= ax[1])
ax[1].set_title('Top 10 Communities with Lowest Crime Rate')
ax[1].set_ylabel('')
```

```
Text(0, 0.5, '')
```

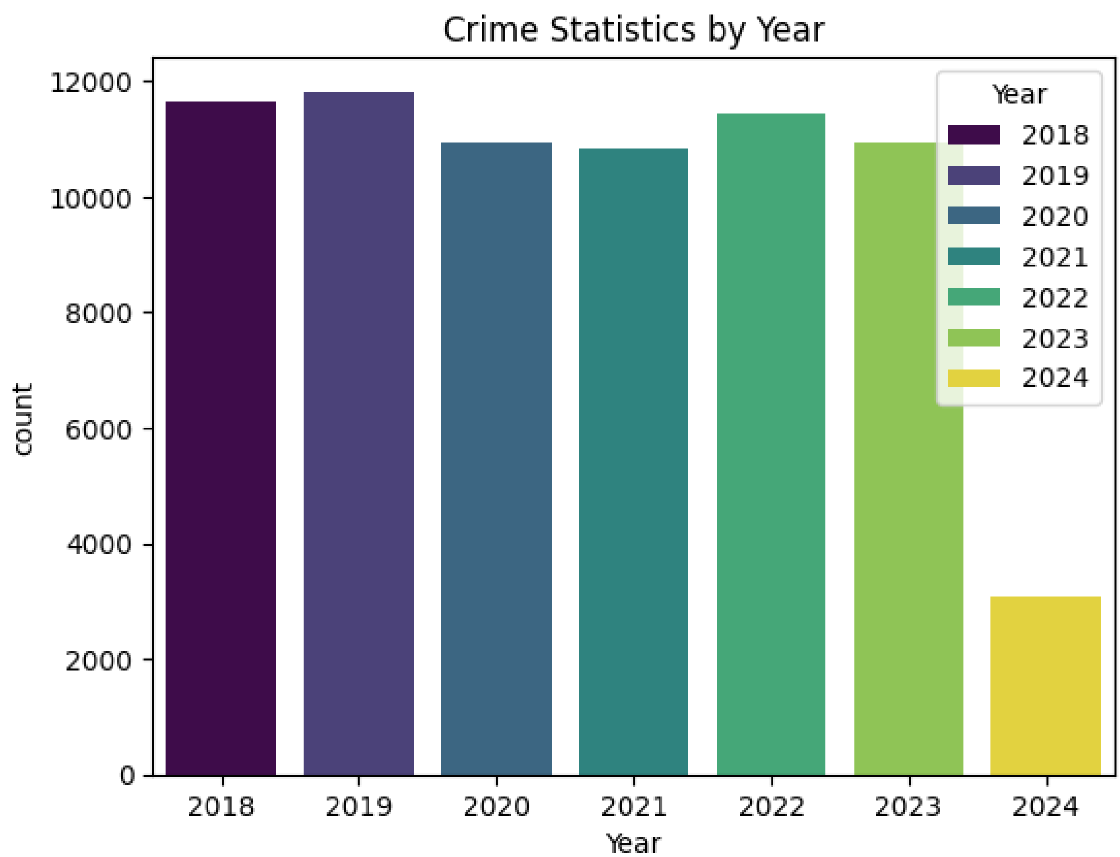


```
plt.figure(figsize=(5, 5))
df['Category'].value_counts().plot.pie(autopct='%1.1f%%')
plt.title('Crime Category Distribution')
plt.ylabel('')
```

```
Text(0, 0.5, '')
```

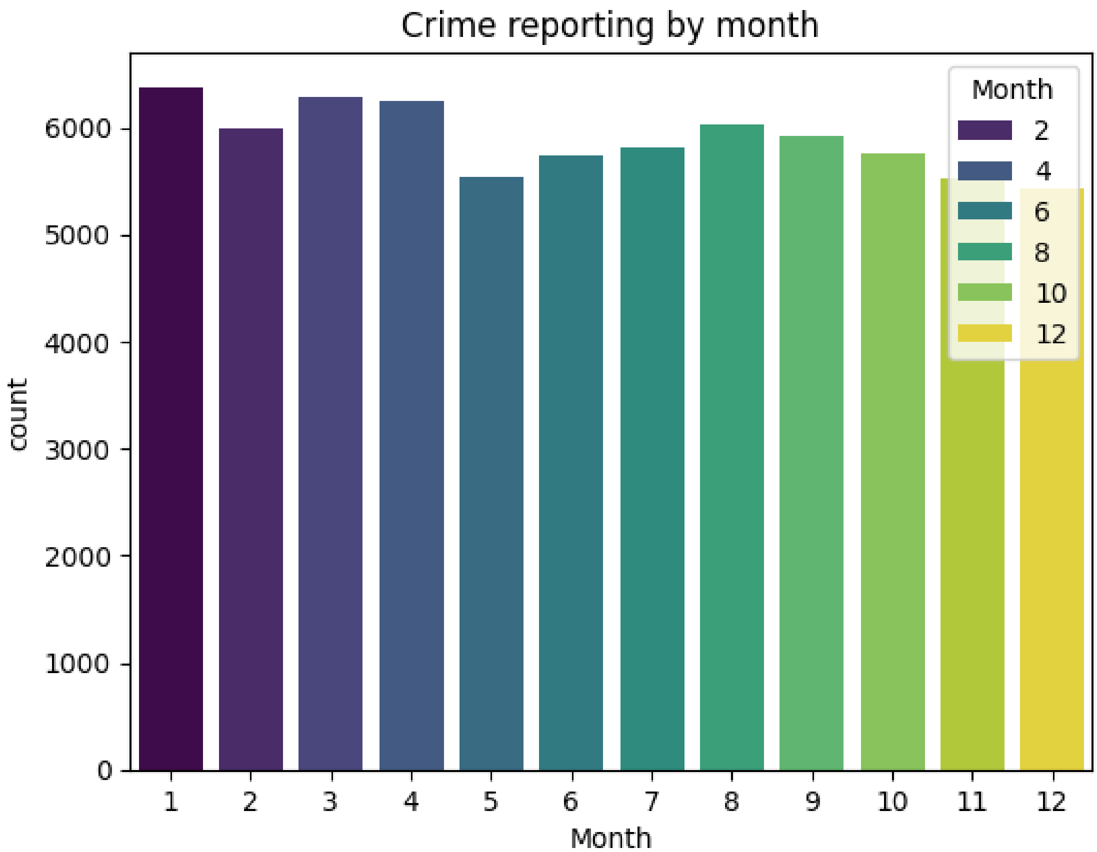



```
sns.countplot(x='Year', data=df, hue='Year',
palette='viridis').set_title('Crime Statistics by Year')
plt.show()
```

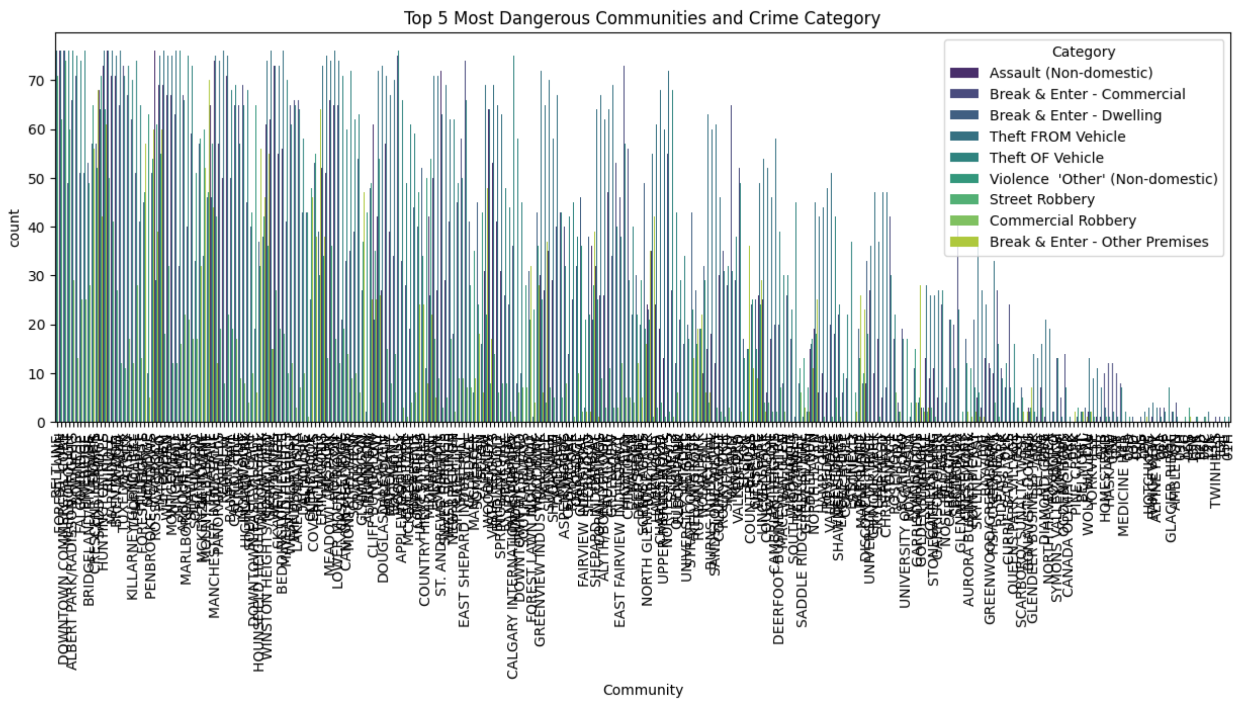



```
sns.countplot(x = 'Month', data = df, hue = 'Month',
palette='viridis').set_title('Crime reporting by month')

Text(0.5, 1.0, 'Crime reporting by month')
```

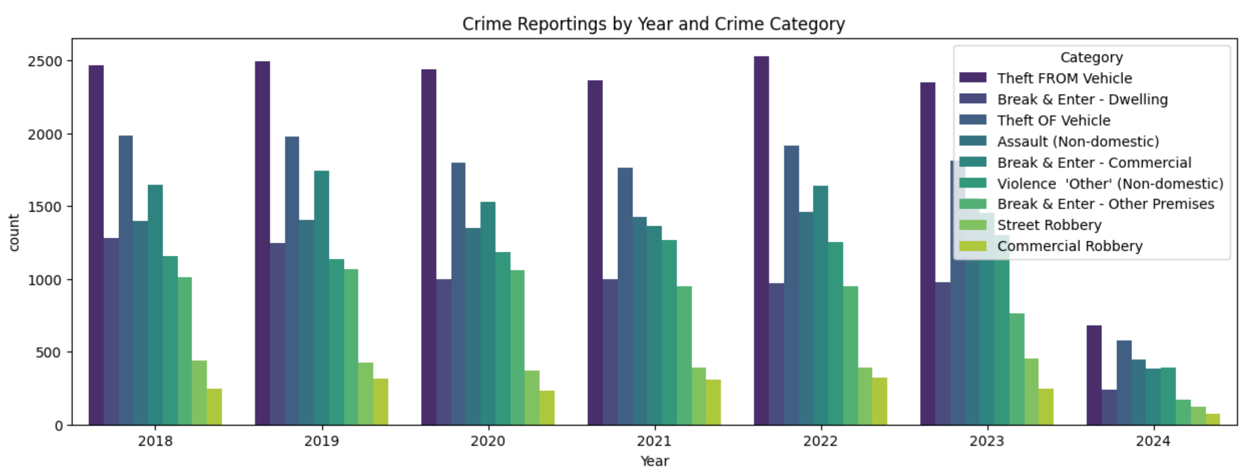



```
plt.figure(figsize=(15, 5))
sns.countplot(x='Community', data=df, hue='Category',
palette='viridis', order=df['Community'].value_counts().index)
plt.title('Top 5 Most Dangerous Communities and Crime Category')
plt.xticks(rotation=90)
sns.move_legend(plt.gca(), "upper right")
plt.show()
```



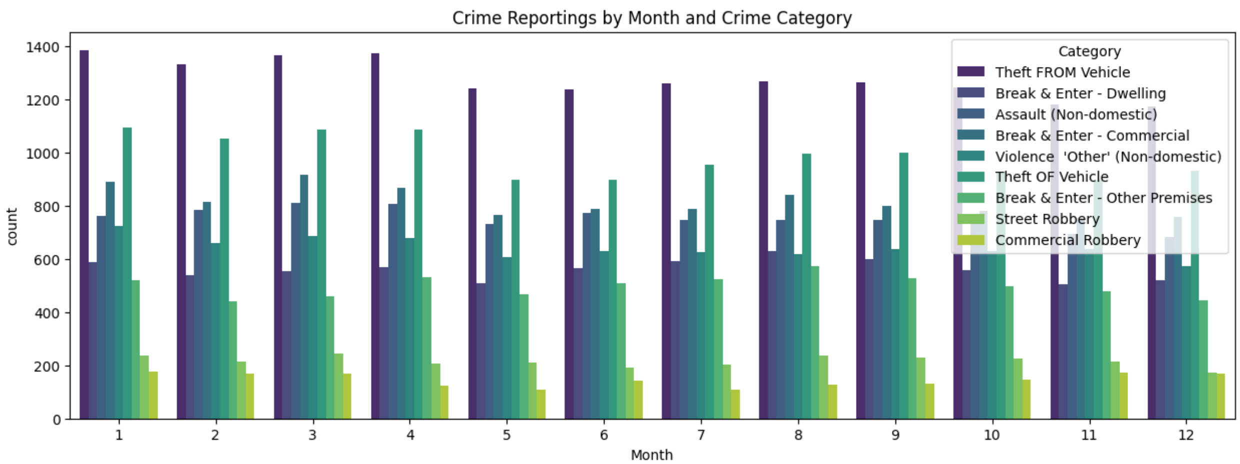

```python
plt.figure(figsize=(15, 5))
sns.countplot(x = 'Year', data = df, hue = 'Category',
palette='viridis').set_title('Crime Reportings by Year and Crime
Category')
```

```
Text(0.5, 1.0, 'Crime Reportings by Year and Crime Category')
```




```python
plt.figure(figsize=(15, 5))
sns.countplot(x = 'Month', data = df, hue = 'Category',
palette='viridis').set_title('Crime Reportings by Month and Crime
Category')
```

```
Text(0.5, 1.0, 'Crime Reportings by Month and Crime Category')
```




```
from sklearn.preprocessing import LabelEncoder
#Label Encoding Object
le = LabelEncoder()
#Object type columns
object_type_columns = df.select_dtypes(include='object').columns
#Label Encoding
for col in object_type_columns:
 df[col] = le.fit_transform(df[col])
df.head()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 70661,\n \"fields\":
[\n   {\n   \"column\": \"Community\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 77,\n   \"min\": 0,\n
\"max\": 295,\n   \"num_unique_values\": 296,\n
\"samples\": [\n   276,\n   156,\n   85\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Category\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 2,\n   \"min\": 0,\n
\"max\": 8,\n   \"num_unique_values\": 9,\n   \"samples\":
[\n   4,\n   1,\n   8\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Crime Count\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
3,\n   \"min\": 1,\n   \"max\": 111,\n
\"num_unique_values\": 77,\n   \"samples\": [\n   5,\n
39,\n   10\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Year\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1,\n   \"min\": 2018,\n   \"max\": 2024,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   2022,\n
2019,\n   2021\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Month\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 3,\n   \"min\": 1,\n   \"max\": 12,\n

```
\"num_unique_values\": 12,\n        \"samples\": [\n          5,\n
9,\n          11\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}

# Prepare sequences for LSTM
def create_sequences(data, seq_length):
 xs = []
 ys = []
 for i in range(len(data) - seq_length):
  x = data.iloc[i:(i + seq_length)].to_numpy()
  y = data.iloc[i + seq_length]['Crime Count']
  xs.append(x)
  ys.append(y)
 return np.array(xs), np.array(ys)

seq_length = 3
X, y = create_sequences(df, seq_length)

from sklearn.model_selection import train_test_split

# Split the dataset into training and temporary sets (70% training,
30% temporary)
X_train, X_temp, y_train, y_temp = train_test_split(X, y,
test_size=0.3, random_state=42)

# Split the temporary set into validation and test sets (15% each)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam

from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense
from keras.optimizers import Adam

# Build the LSTM model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(seq_length,
X_train.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(1))

# Compile the model
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='mse')

# Train the model
```

```
history = model.fit(X_train, y_train, epochs=100,
validation_data=(X_val, y_val))
```

```
Epoch 1/100
1546/1546 [==============================] - 9s 5ms/step - loss:
764.0525 - val_loss: 10.9579
Epoch 2/100
1546/1546 [==============================] - 6s 4ms/step - loss:
12.3547 - val_loss: 9.2604
Epoch 3/100
1546/1546 [==============================] - 8s 5ms/step - loss:
10.1108 - val_loss: 7.7126
Epoch 4/100
1546/1546 [==============================] - 6s 4ms/step - loss:
7.6738 - val_loss: 6.2204
Epoch 5/100
1546/1546 [==============================] - 8s 5ms/step - loss:
6.5437 - val_loss: 5.4582
Epoch 6/100
1546/1546 [==============================] - 6s 4ms/step - loss:
6.4304 - val_loss: 4.9851
Epoch 7/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.8836 - val_loss: 5.2189
Epoch 8/100
1546/1546 [==============================] - 7s 5ms/step - loss:
5.6928 - val_loss: 5.0456
Epoch 10/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.6835 - val_loss: 5.1393
Epoch 11/100
1546/1546 [==============================] - 7s 5ms/step - loss:
5.8599 - val_loss: 4.9128
Epoch 12/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5817 - val_loss: 4.8480
Epoch 13/100
1546/1546 [==============================] - 7s 5ms/step - loss:
5.6911 - val_loss: 4.7536
Epoch 14/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.7244 - val_loss: 4.7677
Epoch 15/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.6584 - val_loss: 5.3889
Epoch 16/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.7333 - val_loss: 4.7090
Epoch 17/100
1546/1546 [==============================] - 8s 5ms/step - loss:
```

```
5.6759 - val_loss: 4.8193
Epoch 18/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4113 - val_loss: 4.9972
Epoch 19/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.6801 - val_loss: 4.9851
Epoch 20/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.6006 - val_loss: 4.9837
Epoch 21/100
1546/1546 [==============================] - 7s 5ms/step - loss:
5.6414 - val_loss: 4.7675
Epoch 22/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.6319 - val_loss: 5.0435
Epoch 23/100
1546/1546 [==============================] - 7s 5ms/step - loss:
5.5363 - val_loss: 4.8311
Epoch 24/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4180 - val_loss: 4.7787
Epoch 25/100
1546/1546 [==============================] - 7s 5ms/step - loss:
5.4284 - val_loss: 5.7353
Epoch 26/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.6086 - val_loss: 4.8736
Epoch 27/100
1546/1546 [==============================] - 7s 5ms/step - loss:
5.5770 - val_loss: 4.8036
Epoch 28/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5408 - val_loss: 4.6840
Epoch 29/100
1546/1546 [==============================] - 7s 5ms/step - loss:
5.5422 - val_loss: 4.8889
Epoch 30/100
1546/1546 [==============================] - 7s 4ms/step - loss:
5.4842 - val_loss: 4.7117
Epoch 31/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4441 - val_loss: 4.8252
Epoch 32/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4935 - val_loss: 4.7938
Epoch 33/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.2441 - val_loss: 5.2977
```

```
Epoch 34/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5035 - val_loss: 5.1587
Epoch 35/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.6563 - val_loss: 4.9895
Epoch 36/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5581 - val_loss: 5.0774
Epoch 37/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4212 - val_loss: 5.4891
Epoch 38/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4642 - val_loss: 4.6506
Epoch 39/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4278 - val_loss: 4.8788
Epoch 40/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4731 - val_loss: 5.3527
Epoch 41/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.3990 - val_loss: 4.8528
Epoch 42/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.3872 - val_loss: 5.0234
Epoch 43/100
1546/1546 [==============================] - 13s 9ms/step - loss:
5.5952 - val_loss: 5.1624
Epoch 44/100
1546/1546 [==============================] - 11s 7ms/step - loss:
5.4371 - val_loss: 4.9906
Epoch 45/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5465 - val_loss: 4.7348
Epoch 46/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.3693 - val_loss: 4.6951
Epoch 47/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.2868 - val_loss: 4.8267
Epoch 48/100
1546/1546 [==============================] - 10s 6ms/step - loss:
5.4386 - val_loss: 4.8102
Epoch 49/100
1546/1546 [==============================] - 11s 7ms/step - loss:
5.2202 - val_loss: 4.9014
Epoch 50/100
```

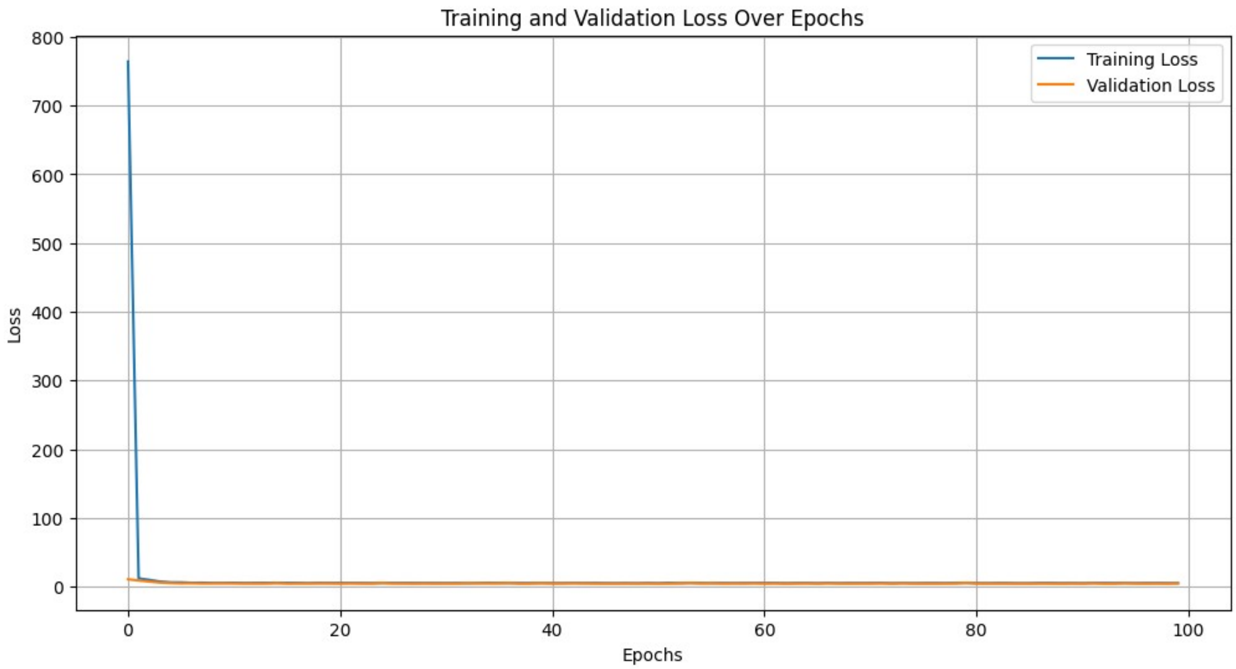
```
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5354 - val_loss: 4.8299
Epoch 51/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.3014 - val_loss: 4.7224
Epoch 52/100
1546/1546 [==============================] - 7s 4ms/step - loss:
5.6830 - val_loss: 4.6693
Epoch 53/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4020 - val_loss: 5.0737
Epoch 54/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5169 - val_loss: 5.6681
Epoch 55/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.5024 - val_loss: 4.9283
Epoch 56/100
1546/1546 [==============================] - 7s 4ms/step - loss:
5.3505 - val_loss: 5.2838
Epoch 57/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.5203 - val_loss: 4.7472
Epoch 58/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4418 - val_loss: 4.9993
Epoch 59/100
1546/1546 [==============================] - 7s 5ms/step - loss:
5.4791 - val_loss: 4.8155
Epoch 60/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5800 - val_loss: 5.1209
Epoch 61/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.5466 - val_loss: 4.6777
Epoch 62/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5820 - val_loss: 4.9439
Epoch 63/100
1546/1546 [==============================] - 7s 5ms/step - loss:
5.3121 - val_loss: 4.6314
Epoch 64/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5073 - val_loss: 4.8074
Epoch 65/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.5564 - val_loss: 4.9823
Epoch 66/100
1546/1546 [==============================] - 6s 4ms/step - loss:
```

```
5.3135 - val_loss: 4.6997
Epoch 67/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.6037 - val_loss: 5.1239
Epoch 68/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.3397 - val_loss: 4.8810
Epoch 69/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.5089 - val_loss: 4.6393
Epoch 70/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4779 - val_loss: 4.7786
Epoch 71/100
1546/1546 [==============================] - 9s 6ms/step - loss:
5.5078 - val_loss: 4.7532
Epoch 72/100
1546/1546 [==============================] - 15s 10ms/step - loss:
5.6545 - val_loss: 5.0286
Epoch 73/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.2690 - val_loss: 4.6689
Epoch 74/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4850 - val_loss: 5.1560
Epoch 75/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.3442 - val_loss: 4.8327
Epoch 76/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4346 - val_loss: 4.6587
Epoch 77/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4859 - val_loss: 4.6390
Epoch 78/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4108 - val_loss: 4.8495
Epoch 79/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5472 - val_loss: 4.8331
Epoch 80/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4715 - val_loss: 5.9805
Epoch 81/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4253 - val_loss: 4.8367
Epoch 82/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.2953 - val_loss: 4.9087
```

```
Epoch 83/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5011 - val_loss: 4.7534
Epoch 84/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4477 - val_loss: 4.7365
Epoch 85/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.2331 - val_loss: 4.8152
Epoch 86/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.2672 - val_loss: 4.7378
Epoch 87/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.5495 - val_loss: 4.7605
Epoch 88/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.5413 - val_loss: 4.6933
Epoch 89/100
1546/1546 [==============================] - 7s 4ms/step - loss:
5.3721 - val_loss: 4.6558
Epoch 90/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4420 - val_loss: 4.8174
Epoch 91/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.2852 - val_loss: 4.7336
Epoch 92/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.5496 - val_loss: 5.1305
Epoch 93/100
1546/1546 [==============================] - 7s 4ms/step - loss:
5.5337 - val_loss: 4.6353
Epoch 94/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.4506 - val_loss: 4.6754
Epoch 95/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4979 - val_loss: 5.2219
Epoch 96/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.3498 - val_loss: 4.8620
Epoch 97/100
1546/1546 [==============================] - 6s 4ms/step - loss:
5.4576 - val_loss: 4.6721
Epoch 98/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.5181 - val_loss: 4.8039
Epoch 99/100
1546/1546 [==============================] - 6s 4ms/step - loss:
```

```
5.3918 - val_loss: 4.7106
Epoch 100/100
1546/1546 [==============================] - 8s 5ms/step - loss:
5.3492 - val_loss: 5.1677
```

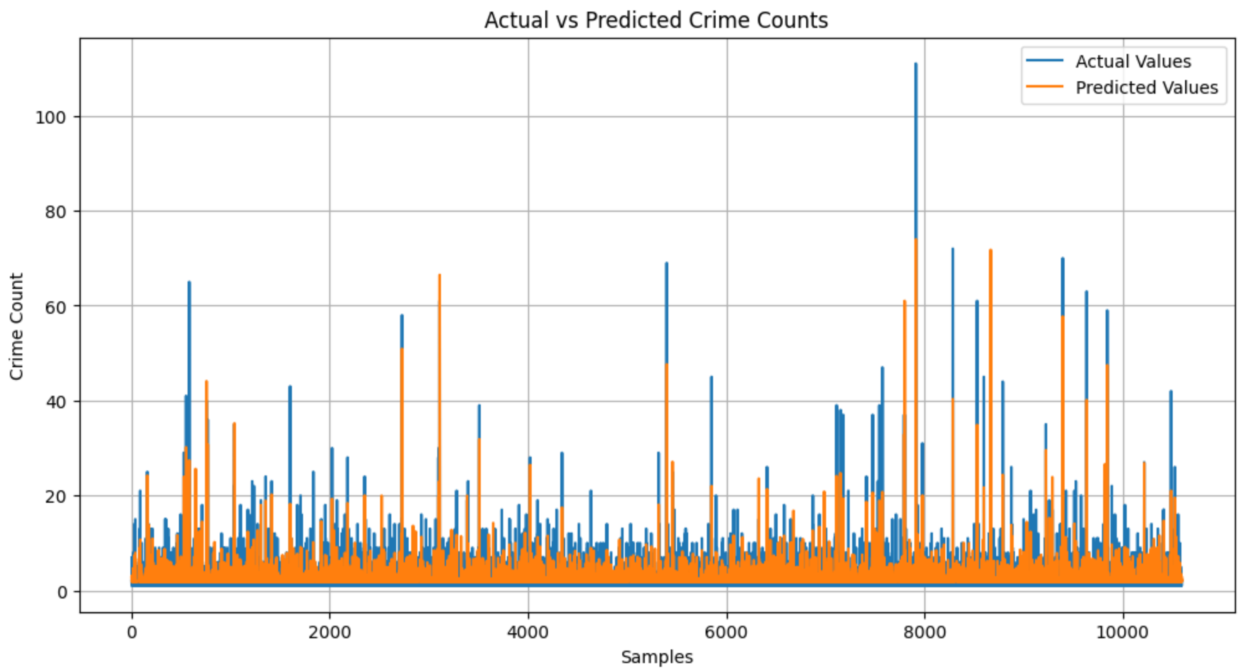
```
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()
```



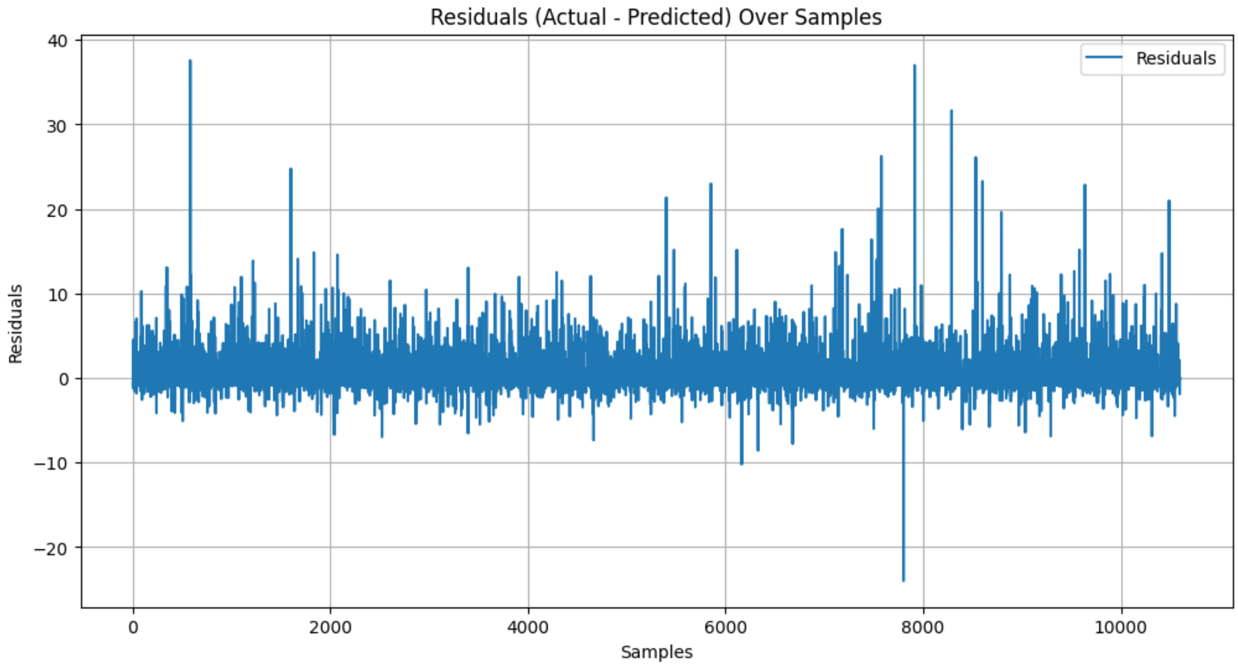

```
# Evaluate the model
test_loss = model.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss}')
# Predictions
y_pred = model.predict(X_test)
print(f'Predictions: {y_pred.flatten()}')
print(f'True Values: {y_test.flatten()}')
```

```
332/332 [==============================] - 1s 4ms/step - loss: 5.1366
Test Loss: 5.136634349822998
332/332 [==============================] - 2s 4ms/step
Predictions: [2.6913044 1.8409897 1.8409897 ... 2.896269  1.8409897
2.1454394]
True Values: [2 1 1 ... 1 2 2]
```

```
# Plotting Actual vs Predicted Values
plt.figure(figsize=(12, 6))
plt.plot(y_test, label='Actual Values')
plt.plot(y_pred, label='Predicted Values')
plt.title('Actual vs Predicted Crime Counts')
plt.xlabel('Samples')
plt.ylabel('Crime Count')
plt.legend()
plt.grid(True)
plt.show()
```




```
# Calculating residuals
residuals = y_test.flatten() - y_pred.flatten()
# Plotting residuals
plt.figure(figsize=(12, 6))
plt.plot(residuals, label='Residuals')
plt.title('Residuals (Actual - Predicted) Over Samples')
plt.xlabel('Samples')
plt.ylabel('Residuals')
plt.legend()
plt.grid(True)
plt.show()
```



Residuals (Actual - Predicted) Over Samples
Residuals
40
30
20
10
0
−10
−20
0
2000
4000
6000
8000
10000
Samples
Residuals