Deep Learning
Assignment 0: Checkpoint
Yavar Khan (50592324)

Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? Provide the main statistics about the entries of the dataset (mean, std, number of missing values, etc.)

About the dataset: The **Buffalo Crime Incidents Dataset** contains detailed records of crime incidents reported within the city of Buffalo, NY. The dataset is maintained by local law enforcement and public safety agencies to provide transparency and aid in crime analysis. It includes key information such as incident type, location, date and time, and reporting agency. The dataset helps analyze crime patterns, trends, and hotspot areas, making it useful for public safety initiatives, law enforcement strategies, and urban policy planning.

Main Statistics:

1. **Before preprocessing:**

```
df.describe()
```

	Incident ID	Hour of Day	updated_at
count	0.0	318673.000000	0.0
mean	NaN	11.923122	NaN
std	NaN	7.194647	NaN
min	NaN	0.000000	NaN
25%	NaN	6.000000	NaN
50%	NaN	13.000000	NaN
75%	NaN	18.000000	NaN
max	NaN	23.000000	NaN

The statistical summary of the Hour of Day column is shown because many other columns contained irrelevant or empty data that needed preprocessing. The dataset initially lacked proper data type assignments, which required significant cleaning and transformation to make it usable for analysis.

```
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 318673 entries, 0 to 318672
Data columns (total 31 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   Case Number             318673 non-null  object 
 1   Incident Datetime       318673 non-null  object 
 2   Incident ID             0 non-null       float64
 3   Incident Type Primary   318673 non-null  object 
 4   Incident Description    318673 non-null  object 
 5   Parent Incident Type    318673 non-null  object 
 6   Hour of Day             318673 non-null  int64  
 7   Day of Week             318673 non-null  object 
 8   Address                 318635 non-null  object 
 9   City                    318673 non-null  object 
 10  State                   318673 non-null  object 
 11  Location                312119 non-null  object 
 12  Latitude                317398 non-null  object 
 13  Longitude               317398 non-null  object 
 14  Created At              74609 non-null   object 
 15  updated_at              0 non-null       float64
 16  zip_code                316014 non-null  object 
 17  neighborhood            315050 non-null  object 
 18  Council District        315954 non-null  object 
 19  Council District 2011   316014 non-null  object 
 20  Census Tract            315050 non-null  object 
 21  Census Block Group      315050 non-null  object 
 22  Census Block            315050 non-null  object 
 23  2010 Census Tract       315050 non-null  object 
 24  2010 Census Block Group 315050 non-null  object 
 25  2010 Census Block       315050 non-null  object 
 26  Police District         315050 non-null  object 
 27  TRACTCE20               315187 non-null  object 
 28  GEOID20_tract           315187 non-null  object 
 29  GEOID20_blockgroup      315187 non-null  object 
 30  GEOID20_block           315187 non-null  object 
dtypes: float64(2), int64(1), object(28)
memory usage: 75.4+ MB
```

The dataset comprises 31 columns with 318,673 entries, but several columns contained irrelevant or incomplete data, such as Incident ID, updated_at, and other census-related columns.

	Case Number	Incident Datetime	Incident ID	Incident Type Primary	Incident Description	Parent Incident Type	Hour of Day	Day of Week	Address	City	...	Census Block Group	Census Block	2010 Census Tract	2010 Census Block Group	Ce
0	16-1660403	06/14/2016 01:20:00 AM	NaN	ASSAULT	ASSAULT	Assault	1	Tuesday	E AMHERST ST & E AMHERST ST	Buffalo	...	2	2003	55	2	
1	16-3480266	12/13/2016 05:00:00 AM	NaN	LARCENY/THEFT	LARCENY/THEFT	Theft	5	Tuesday	1000 Block E LOVEJOY ST	Buffalo	...	4	4001	23	4	
2	20-2010167	07/19/2020 03:09:00 AM	NaN	ASSAULT	Buffalo Police are investigating this report o...	Assault	3	Sunday	GRIDER ST & KENSINGTON WB	Buffalo	...	NaN	NaN	NaN	NaN	
3	14-3210732	11/17/2014 08:08:00 AM	NaN	LARCENY/THEFT	LARCENY/THEFT	Theft	8	Monday	2100 Block ELMWOOD AV	Buffalo	...	2	2007	56	2	
4	15-1100268	04/20/2015 10:22:00 AM	NaN	LARCENY/THEFT	LARCENY/THEFT	Theft	10	Monday	2100 Block ELMWOOD AV	Buffalo	...	2	2007	56	2	

5 rows × 31 columns

The initial rows of the dataset reveal raw, unprocessed data, with important fields like Incident Datetime, Incident Type Primary, and Day of Week visible. However, due to the presence of noisy and incomplete columns, preprocessing was essential to extract meaningful insights. For example, the location coordinates and categorical data were standardized for proper analysis.

2. **After preprocessing:**

	Hour of Day	Day of Week	Latitude	Longitude	zip_code	Census Tract	Census Block Group	Census Block	Year	Mor
count	297014.000000	297014.000000	297014.000000	297014.000000	297014.000000	297014.000000	297014.000000	297014.000000	297014.000000	297014.0000
mean	11.905382	4.008451	42.911723	-78.849840	14210.992738	70.006820	2.231518	2237.559846	2013.620718	6.8082
std	7.274902	1.994915	0.028314	0.031294	5.389089	292.084581	1.228350	1227.033130	5.157585	3.3099
min	0.000000	1.000000	42.828000	-78.910000	14201.000000	1.100000	1.000000	1000.000000	1910.000000	1.0000
25%	6.000000	2.000000	42.893000	-78.878000	14207.000000	33.020000	1.000000	1007.000000	2009.000000	4.0000
50%	13.000000	4.000000	42.913000	-78.849000	14211.000000	47.020000	2.000000	2004.000000	2013.000000	7.0000
75%	18.000000	6.000000	42.935000	-78.821000	14215.000000	67.020000	3.000000	3004.000000	2018.000000	10.0000
max	23.000000	7.000000	42.966000	-78.799000	14225.000000	9805.000000	7.000000	7005.000000	2025.000000	12.0000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 297014 entries, 0 to 318672
Data columns (total 18 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Incident Type Primary 297014 non-null  object
 1   Hour of Day           297014 non-null  int64
 2   Day of Week           297014 non-null  int64
 3   Address               297014 non-null  object
 4   Latitude              297014 non-null  float64
 5   Longitude             297014 non-null  float64
 6   zip_code              297014 non-null  float64
 7   neighborhood          297014 non-null  object
 8   Council District      297014 non-null  object
 9   Census Tract          297014 non-null  float64
 10  Census Block Group    297014 non-null  float64
 11  Census Block          297014 non-null  float64
 12  2010 Census Tract     297014 non-null  object
 13  Police District       297014 non-null  object
 14  Year                  297014 non-null  int32
 15  Month                 297014 non-null  int32
 16  Part of Day           297014 non-null  object
 17  LocationCluster       297014 non-null  int32
dtypes: float64(6), int32(3), int64(2), object(7)
memory usage: 39.7+ MB
```

```
df.head()
```

	Incident Type Primary	Hour of Day	Latitude	Longitude	neighborhood	Council District	2010 Census Tract	Police District	Year	Month	Part of Day	LocationCluster	Season	Crime Density
0	ASSAULT	1	0.797101	0.189189	Grant-Amherst	NORTH	55	4	2016	6	Night	7	Summer	Medium Crime
1	LARCENY/THEFT	5	0.442029	0.909910	Lovejoy	LOVEJOY	23	3	2016	12	Night	3	Winter	Low Crime
3	LARCENY/THEFT	8	0.913043	0.279279	West Hertel	NORTH	56	4	2014	11	Morning	1	Fall	Medium Crime
4	LARCENY/THEFT	10	0.913043	0.279279	West Hertel	NORTH	56	4	2015	4	Morning	1	Spring	Medium Crime
5	BURGLARY	3	0.615942	0.558559	Masten Park	MASTEN	33.02	3	2015	4	Night	8	Spring	Low Crime

After preprocessing, the dataset shows significant improvements. We fixed incorrect data types, dropped multiple irrelevant and empty columns, and handled incomplete entries to ensure a cleaner dataset. Additionally, several new features were added as part of feature engineering to enhance the dataset's analytical potential. These improvements are evident in the processed metrics and will be discussed in detail in a later section.

<u>What kind of preprocessing techniques have you applied to this dataset?</u>

1. Handling Missing Data:
    - Columns with a high percentage of missing values (e.g., Incident ID, updated_at, Created At) were dropped.
    - Rows with missing values in critical columns were removed, while some missing values (e.g., Latitude and Longitude) were filled using the mean of their respective neighborhood.

```python
df = df.drop(columns=['Incident ID', 'updated_at'])
```

```python
#Filling null values of latitude and longitude using neighborhood data

df['Latitude'] = df.groupby('neighborhood')['Latitude'].transform(lambda x: x.fillna(x.mean()))
df['Longitude'] = df.groupby('neighborhood')['Longitude'].transform(lambda x: x.fillna(x.mean()))
```

2. Irrelevant Columns:
    - Several columns that were irrelevant for analysis (e.g., Case Number, City, State, Address, GEOID20_block, Council District 2011) were dropped to reduce noise in the dataset.

3. Duplicate Entries:
    - Duplicate rows were identified and removed to prevent redundant data from skewing results.

4. Data Type Conversion:
    - Object columns like Latitude, Longitude, zip_code, and census-related fields were converted to numeric types.
    - Categorical columns like Day of Week were mapped to numerical values for easier processing.

Handling mismatched string formats

```
# Converting data types of 'Longitude', 'Latitude', and 'zip_code'
df['Longitude'] = pd.to_numeric(df['Longitude'], errors='coerce')
df['Latitude'] = pd.to_numeric(df['Latitude'], errors='coerce')
df['zip_code'] = pd.to_numeric(df['zip_code'], errors='coerce')
df['Census Tract'] = pd.to_numeric(df['Census Tract'], errors='coerce')
df['Census Block Group'] = pd.to_numeric(df['Census Block Group'], errors='coerce')
df['Census Block'] = pd.to_numeric(df['Census Block'], errors='coerce')
```

```
#Extracting month and year from 'Incident Datetime' and converting to integers
df['Year'] = pd.to_datetime(df['Incident Datetime']).dt.year
df['Month'] = pd.to_datetime(df['Incident Datetime']).dt.month

# Droping the 'Incident Datetime' column --> not required anymore
df = df.drop(columns=['Incident Datetime'])
```

```
# Converting 'Hour of Day' to integer
df['Hour of Day'] = df['Hour of Day'].astype(int)
```

```
 Converting 'Day of Week' to integers (1-7, assuming Monday=1, etc.)
ay_mapping = {'Monday': 1, 'Tuesday': 2, 'Wednesday': 3, 'Thursday': 4, 'Friday': 5, 'Saturday': 6, 'Sunday': 7}
f['Day of Week'] = df['Day of Week'].map(day_mapping)
```

```
# Dropping 'Incident Description' and 'Parent Incident Type' --> highly correlated to Incident Type
df = df.drop(columns=['Incident Description', 'Parent Incident Type'])
```

```
# Dropping column "Created At" as it has approx 70% null values
# Dropping case number, city, state and others as they are irrelevant for our model

df = df.drop(columns=["Created At","City","State","Case Number","GEOID20_block","GEOID20_blockgroup","GEOID20_tr
df = df.drop(columns=["2010 Census Block Group","2010 Census Block"])

```

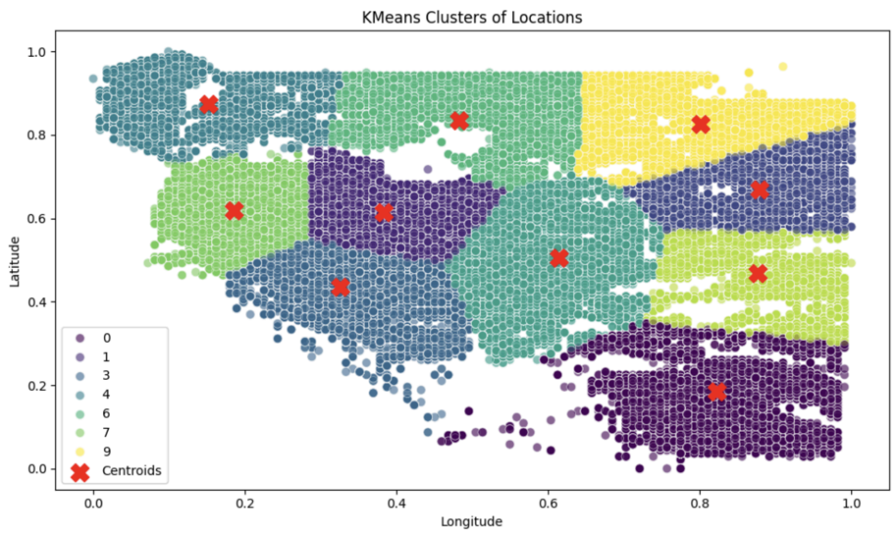
5. Feature Engineering:
    - Datetime Features: Extracted Year and Month from the Incident Datetime column and created a new feature, Season, based on the month.

```
# 1. Creating seasonal feature
# Reason: Crime rates may vary by season (e.g., higher in sum
def assign_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Fall'

df['Season'] = df['Month'].apply(assign_season)
```

    - Time-Based Feature: Added Part of Day by mapping Hour of Day into categories such as Morning, Afternoon, Evening, and Night.

```python
# Adding new feature

def part_of_day(hour):
    if 6 <= hour < 12:
        return 'Morning'
    elif 12 <= hour < 16:
        return 'Afternoon'
    elif 16 <= hour < 22:
        return 'Evening'
    else:
        return 'Night'

df['Part of Day'] = df['Hour of Day'].apply(part_of_day)
```

- Location Clustering: Applied KMeans clustering on Latitude and Longitude to group locations into clusters (LocationCluster).



- Crime Density: Grouped zip_code by crime density into Low Crime, Medium Crime, and High Crime categories.

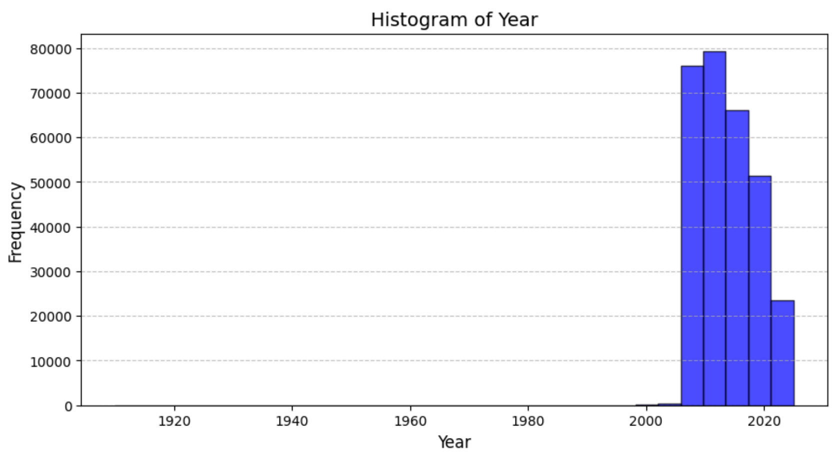
```python
def zip_code_density_group(zip_code_counts):
    if zip_code_counts >= 40000:
        return 'High Crime'
    elif zip_code_counts >= 20000:
        return 'Medium Crime'
    else:
        return 'Low Crime'

zip_code_counts = df['zip_code'].value_counts()
df['Crime Density'] = df['zip_code'].map(zip_code_counts).apply(zip_code_density_group)
```

6. Data Normalization and Standardization:
    - Normalized Latitude and Longitude values using Min-Max Scaling to bring them to a uniform range.

```python
# Normalising Latitude and Longitude

from sklearn.preprocessing import MinMaxScaler

# Initialize scaler
scaler = MinMaxScaler()

# Apply Min-Max Scaling
df[["Latitude", "Longitude"]] = scaler.fit_transform(df[["Latitude", "Longitude"]])

```

7. Correlation-Based Feature Reduction:
   - Dropped highly skewed or less meaningful features, such as Census Block, Census Tract, and 2010 Census Block Group, based on correlation analysis.

8. Filtering Outliers:
   - Removed older data before 2000 to focus on recent crime patterns for meaningful insights.



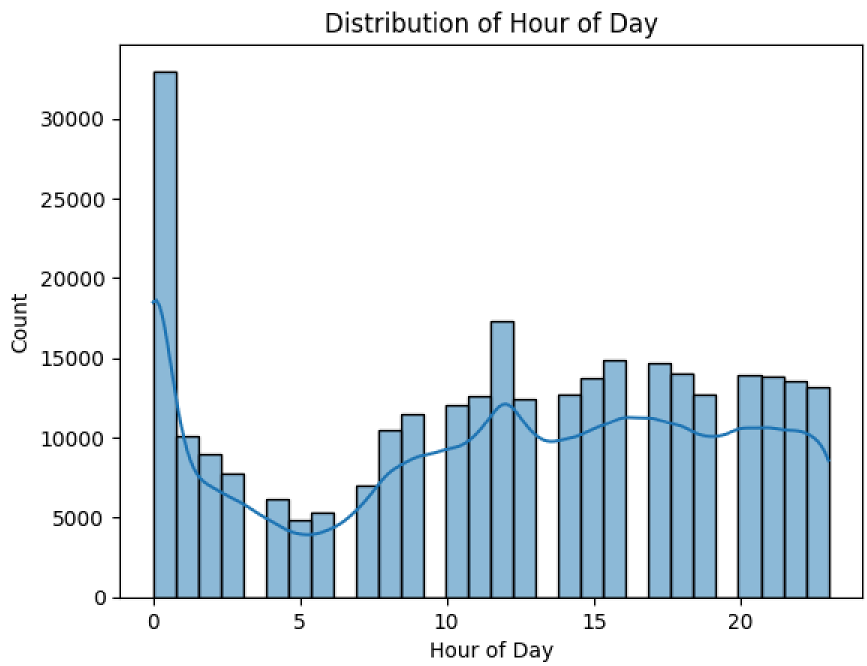
9. Categorical Encoding:
   - Converted categorical variables (e.g., neighborhood, Council District, Season, Crime Density) into numerical labels using Label Encoding.

10. Removing Noise and Unknown Values:
    - Rows containing "UNKNOWN" or "NaN" in any column were removed.

```python
# Removing rows where any column has the value "UNKNOWN"
df = df[~df.isin(["UNKNOWN"]).any(axis=1)]
df = df[~df.isin(["NaN"]).any(axis=1)]


print(df.info())
```

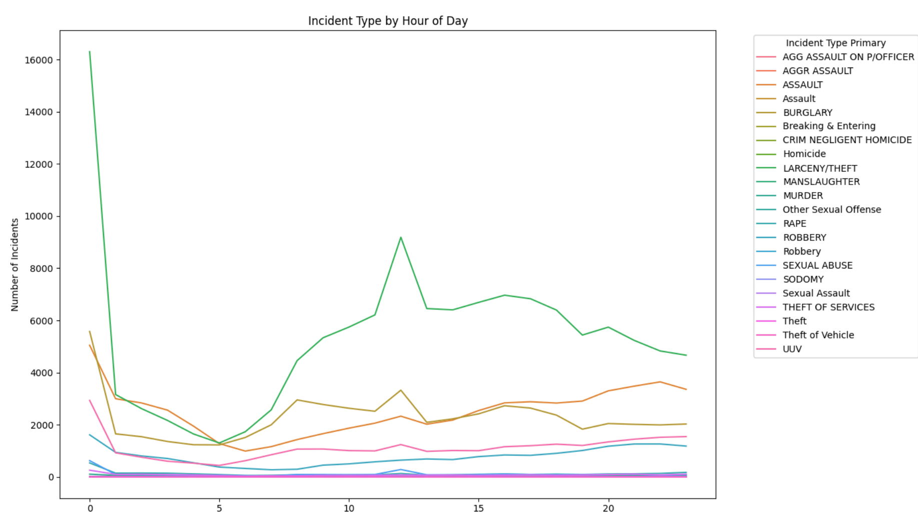
Provide at least 5 visualization graphs with a brief description for each graph, e.g. discuss if there are any interesting patterns or correlations.

1.



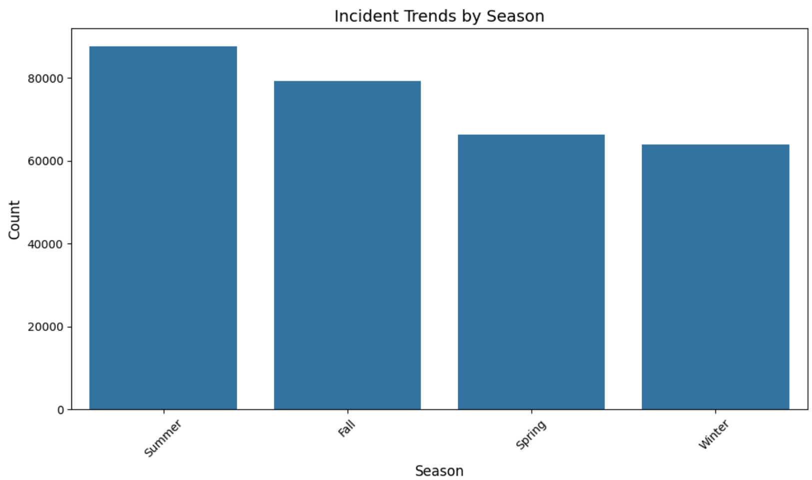
This histogram with a density plot shows the frequency of crime incidents across different hours of the day. A sharp peak is observed at midnight (Hour 0), indicating that a significant number of crimes occur late at night. The frequency gradually decreases during the early morning hours and picks up again during the afternoon and evening. This suggests that nighttime is a critical period for crime prevention.

2.



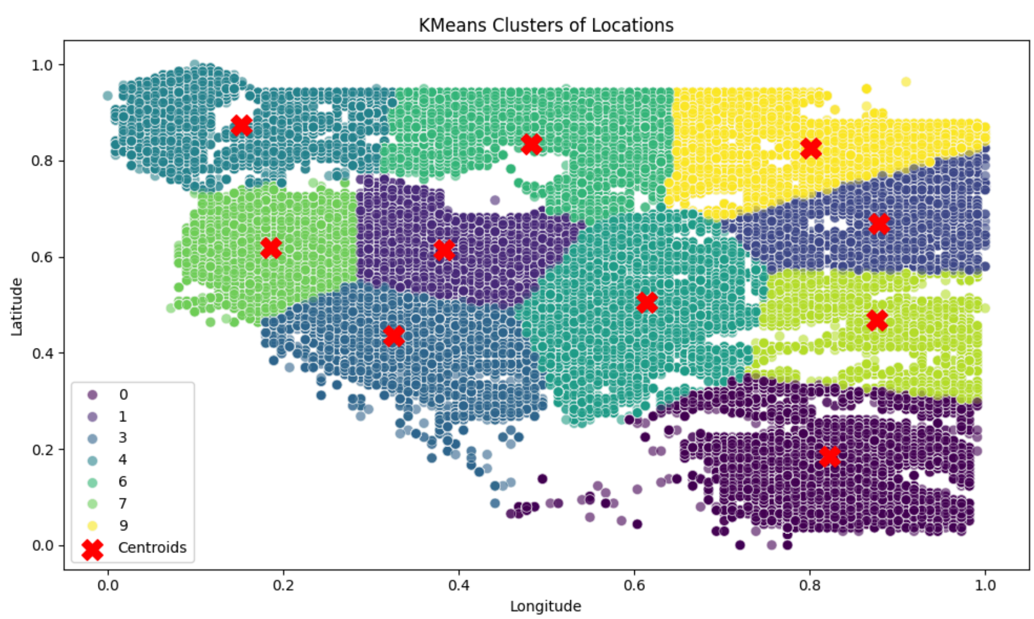
This line plot shows the distribution of different crime types throughout the day. Larceny/Theft dominates as the most frequent crime type, particularly at midnight. Other crimes like Assault and Burglary show peaks during specific times, such as late evenings and nights, suggesting a temporal dependency for different crime types.

3.



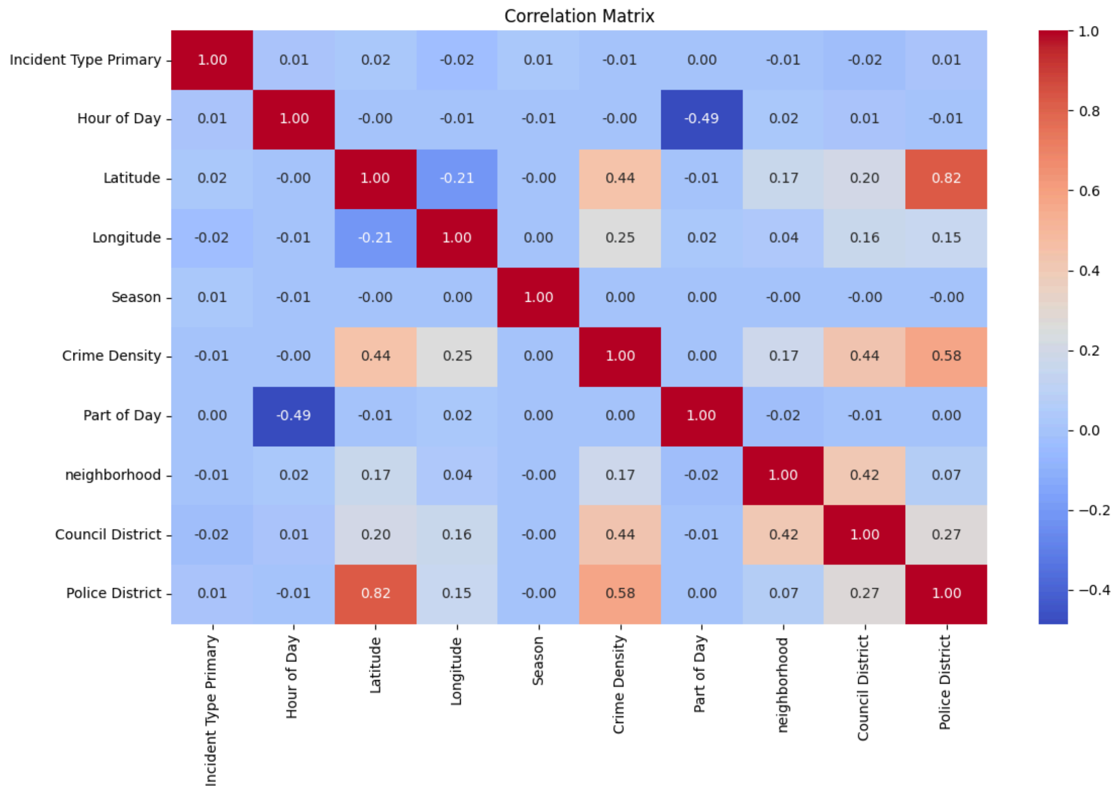
This bar plot illustrates the number of incidents across different seasons. Summer has the highest crime rate, likely due to increased outdoor activity and gatherings, which create opportunities for certain crimes. Fall and Spring have moderate crime rates, while Winter sees the least crime, possibly due to weather conditions keeping people indoors.

4.



This scatter plot visualizes the geographical distribution of incidents using KMeans clustering. Each cluster represents a high-crime zone, with centroids marked in red. The clustering highlights distinct hotspots, allowing for targeted interventions in specific areas to reduce crime.

5.



The heatmap displays the correlation between numerical and categorical variables. Notable insights include a strong correlation between Latitude and Police District, indicating geographical clustering of law enforcement zones. Crime Density shows a moderate correlation with Latitude and Council District, suggesting crime hotspots are influenced by geographical and administrative factors. Additionally, the correlation matrix was instrumental in identifying highly correlated features, such as Census Block and Census Tract, which were dropped during preprocessing to avoid redundancy and improve the dataset's usability. Features with low or no correlation to the target variables were also excluded to simplify the dataset without losing critical information.

Provide brief details and mathematical representation of the ML methods you have used. What are the key features? What are the advantages/disadvantages?

1. **Logistic Regression**

   Mathematical Formula:

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n)}}$$

Where, $P(y = 1|X)$ is the probability of the positive class, and β_0 is the bias term, and $\beta_{1,} \beta_2 \ldots \beta_n$ are feature weights.

   Characteristics:
   - Simple and interpretable.
   - Best suited for data that is linearly separable.

   Pros:
   - Quick to train and evaluate.
   - Provides probabilities for classifications.

   Cons:
   - Limited in handling non-linear relationships.
   - Assumes independence between predictors and log-odds.

2. **Gradient Boosting**

   Mathematical Formula:

$$F_m(x) = F_{m-1}(x) + \eta . h_m(x)$$

Where, $F_m(x)$ is the current model,$F_{m-1}(x)$ is the previous model, η is learning rate and $h_m(x)$ is a newly added weak learner.

   Characteristics:
   - Captures non-linear patterns effectively.
   - Performs well on complex datasets.

   Pros:
   - High accuracy and adaptability.
   - Robust to class imbalance.

   Cons:
   - Computationally demanding.
   - Overfitting can occur without careful tuning

3. **Naive Bayes:**

Mathematical Formula:

$$P(C|X) = \frac{P(X|C)P(C)}{P(CX)}$$

Where $P(C|X)$ -> Posterior probability, $P(X|C)$ → likelihood, $P(C)$ → prior, and $P(X)$ is the evidence

Characteristics:
- Performs well with categorical features.
- Simplifies computations due to independence assumption.

Pros:
- Extremely fast and efficient.
- Effective on small datasets.

Cons:
- Assumes feature independence, which is rarely true in real-world scenarios.

4. **Neural Networks:**

Mathematical Formula:

$$y = f(W_2.f(W_1.X + b_1) + b_2)$$

Where, W_1 and W_2 are weights, b_1 and b_2 are biases, *f* is the activation function, and *y* is the output.

Characteristics:
- Highly flexible for learning non-linear relationships.
- Scalable to large datasets.

Pros:
- Can model intricate data patterns.
- Adaptable to a variety of tasks.

Cons:
- Resource-intensive.
- Risk of overfitting if not properly regularized.

Provide brief details of the NN model you have used.

1. **Architecture:**
   - **Input Layer:** Matches the number of features in the dataset.
   - **Two Hidden Layers:** Fully connected layers with ReLU activation. Dropout (40%) applied for regularization.
   - **Output Layer:** Outputs probabilities for each class using the Softmax activation function.

2. **Training Details:**
   - **Loss Function:** Cross-Entropy Loss for multi-class classification.
   - **Optimizer:** Adam Optimizer
   - **Learning rate:** 0.01
   - **Batch Size:** 32
   - **Epochs:** 20

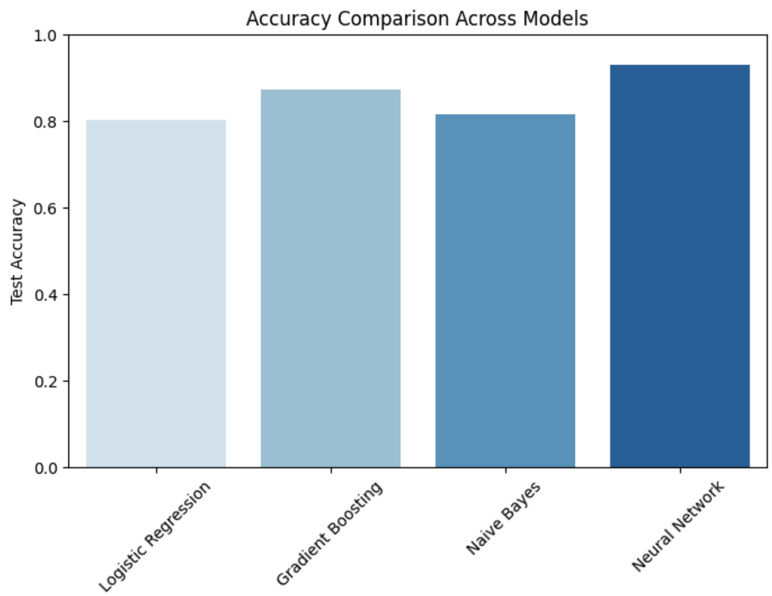
Provide your loss value and accuracy for all 4 methods (3 ML models & 1 NN).

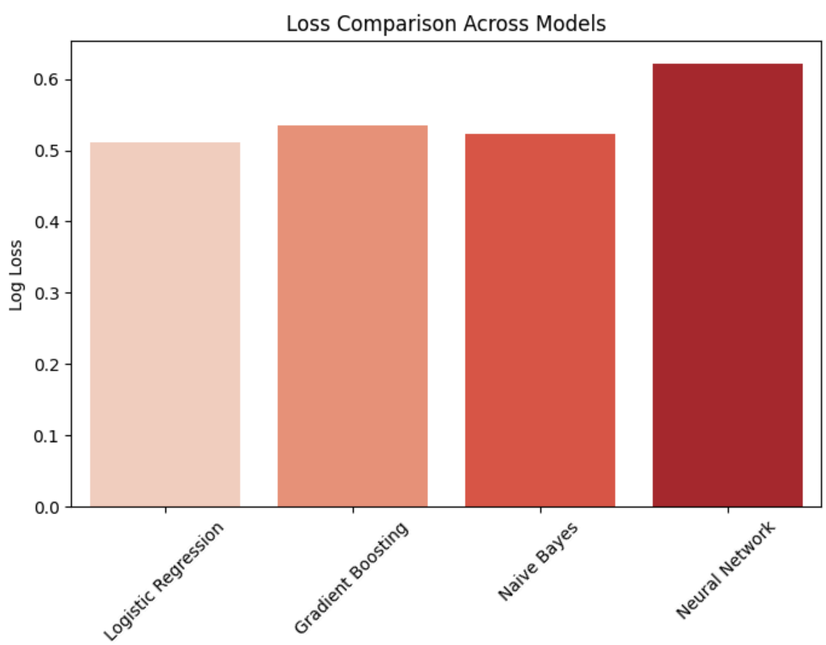
```
Model: Logistic Regression
  - Test Accuracy: 0.8043
  - Test Loss: 0.5107
  - Training Time: 0.85 seconds

Model: Gradient Boosting
  - Test Accuracy: 0.8744
  - Test Loss: 0.5341
  - Training Time: 13.72 seconds

Model: Naive Bayes
  - Test Accuracy: 0.8161
  - Test Loss: 0.5231
  - Training Time: 0.02 seconds
```

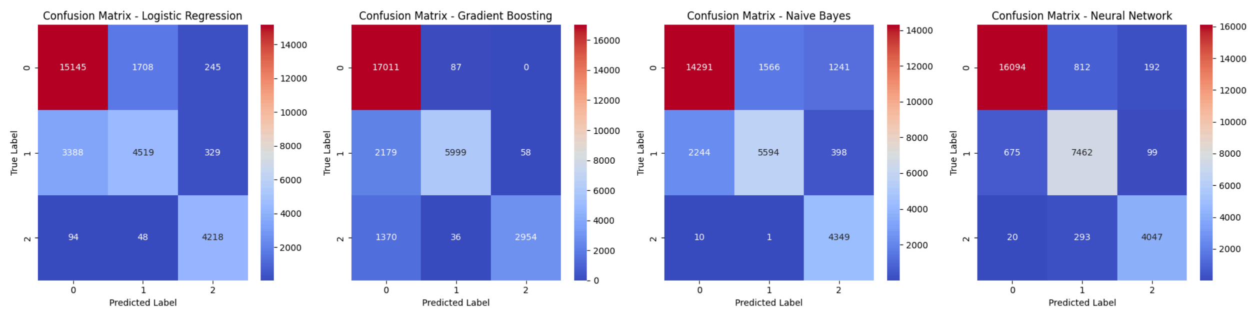
Neural Network:

```
Test Accuracy: 0.9296
Test Loss: 0.6218
```

<u>Show the plot comparing the predictions vs the actual test data for all methods used. Analyze the results. You can consider accuracy/time/loss as some of the metrics to compare the methods</u>







Overall Observation:

From the accuracy plot, we can observe that the Neural Network outperformed other models with a test accuracy of approximately 93%, closely followed by Gradient Boosting at 87%. Logistic Regression and Naive Bayes lagged slightly with accuracies of 80% and 81%, respectively.

The log loss comparison revealed that Naive Bayes achieved the lowest log loss at 0.5231, indicating its strong confidence in predictions. Gradient Boosting also performed well with a log loss of 0.5341. However, Logistic Regression had a log loss of 0.5107, slightly better than the Neural Network, which had the highest log loss of 0.6218. This suggests some uncertainty in the Neural Network's predictions despite achieving the best accuracy.

Examining the confusion matrices, Neural Network and Gradient Boosting were better at handling class imbalances, correctly predicting most instances of each class. Logistic Regression and Naive Bayes showed higher misclassification rates, especially in minority classes, which could affect their reliability in imbalanced datasets.

Overall, the Neural Network emerged as the best-performing model in terms of accuracy, making it suitable for applications prioritizing prediction quality. However, if lower log loss is more critical, Naive Bayes or Gradient Boosting may be preferred due to their more confident predictions.