Crime Rate Prediction & Analysis Project

Prepared by:
Mohamed Gaber

1. Introduction

Crime rates are increasing in many countries, making crime prediction and analysis a critical field for public safety and resource allocation. This project aims to analyze crime data, extract meaningful patterns, and build predictive models to assist in crime prevention and law enforcement planning.

2. Project Objectives

Understand and analyze crime data using Exploratory Data Analysis (EDA).
Clean and preprocess the data to ensure quality and consistency.
Normalize features for better model performance.
Split the dataset into training and testing sets.
Apply two data mining algorithms: KMeans clustering and Decision Tree classification.
Evaluate the models using appropriate metrics.
Visualize the data and results using various plots.

3. Dataset Description

The dataset contains real-world crime records with the following columns:
Dates, Category, Descript, DayOfWeek, PdDistrict, Resolution, Address, X, Y

X and Y represent longitude and latitude, respectively. The target variable for classification is Category (type of crime).

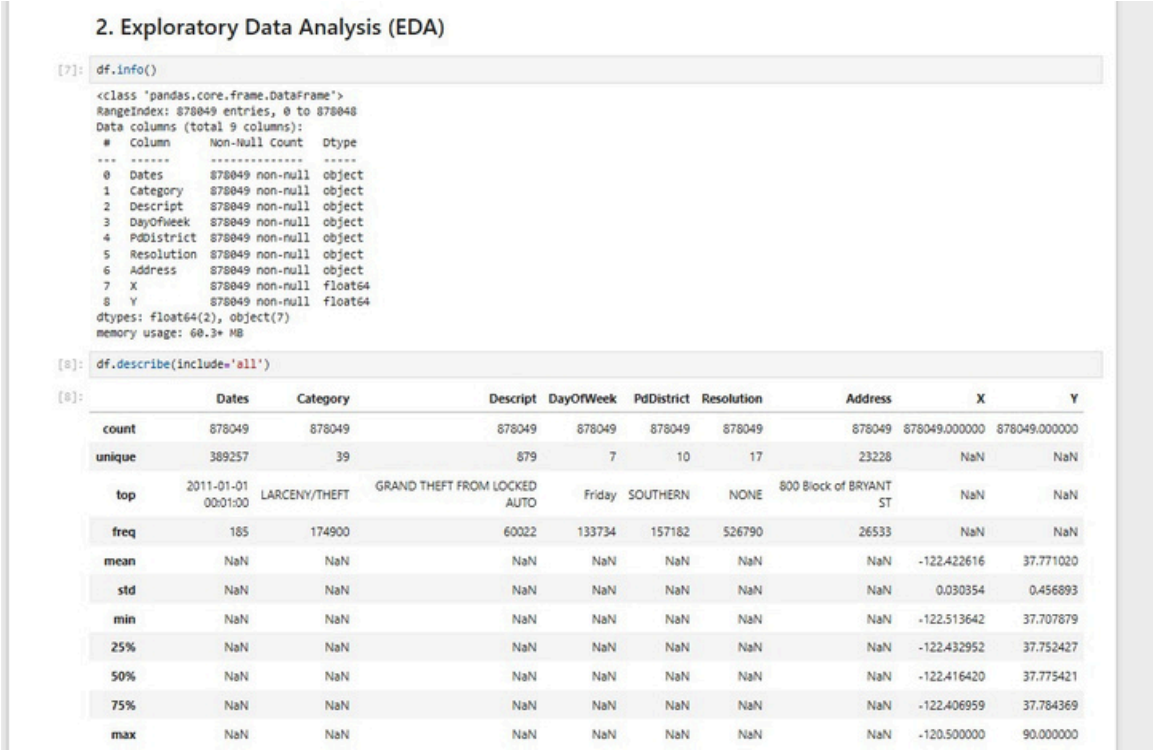
4. Methodology

4.1Data Collection

The data was loaded from a CSV file (train.csv).

4.2 Exploratory Data Analysis (EDA)

Inspected the data structure and types.
Generated summary statistics.
Visualized the most common crime categories.
Explored the distribution of crimes by location and category.



2. Exploratory Data Analysis (EDA)

[7]:
```
df.info()
```

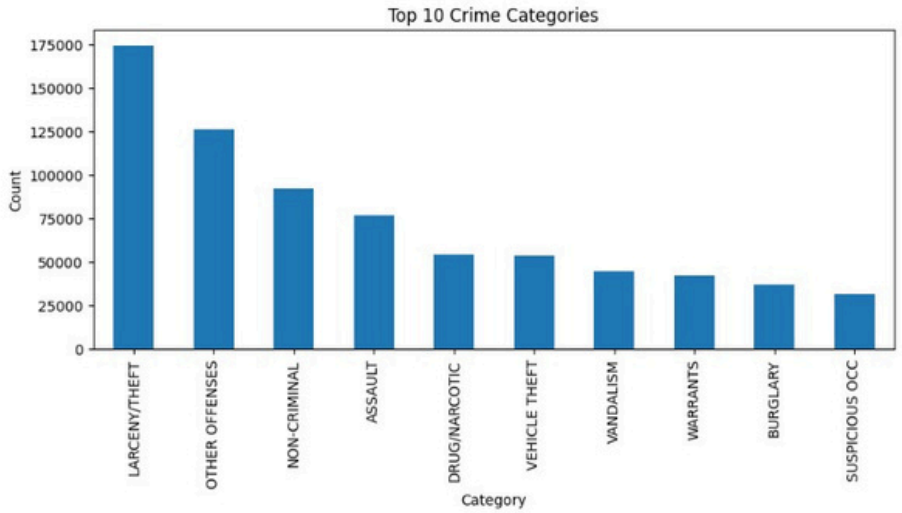
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 878049 entries, 0 to 878048
Data columns (total 9 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   Dates       878049 non-null  object 
 1   Category    878049 non-null  object 
 2   Descript    878049 non-null  object 
 3   DayOfWeek   878049 non-null  object 
 4   PdDistrict  878049 non-null  object 
 5   Resolution  878049 non-null  object 
 6   Address     878049 non-null  object 
 7   X           878049 non-null  float64
 8   Y           878049 non-null  float64
dtypes: float64(2), object(7)
memory usage: 60.3+ MB
```

[8]:
```
df.describe(include='all')
```

[8]:

	Dates	Category	Descript	DayOfWeek	PdDistrict	Resolution	Address	X	Y
count	878049	878049	878049	878049	878049	878049	878049	878049.000000	878049.000000
unique	389257	39	879	7	10	17	23228	NaN	NaN
top	2011-01-01 00:01:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	SOUTHERN	NONE	800 Block of BRYANT ST	NaN	NaN
freq	185	174900	60022	133734	157182	526790	26533	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-122.422616	37.771020
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.030354	0.456893
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-122.513642	37.707879
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-122.432952	37.752427
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-122.416420	37.775421
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-122.406959	37.784369
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-120.500000	90.000000

```
[9]: df['Category'].value_counts().head(10).plot(kind='bar', figsize=(10,4), title='Top 10 Crime Categories')
     plt.ylabel('Count')
     plt.show()
```



4.3 Data Cleaning

Removed missing values and duplicate records.
Converted date strings to datetime objects.
Filtered out records with invalid or zero coordinates.

3. Data Cleaning

```
[10]: # Remove missing values
      df = df.dropna()
      # Remove duplicates
      df = df.drop_duplicates()
      # Convert Dates
      df['Dates'] = pd.to_datetime(df['Dates'], errors='coerce')
      # Remove invalid coordinates
      df = df[(df['X'] != 0) & (df['Y'] != 0)]
      df.head()
```

[10]:

	Dates	Category	Descript	DayOfWeek	PdDistrict	Resolution	Address	X	Y
0	2015-05-13 23:53:00	WARRANTS	WARRANT ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122.425892	37.774599
1	2015-05-13 23:53:00	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122.425892	37.774599
2	2015-05-13 23:33:00	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN	ARREST, BOOKED	VANNESS AV / GREENWICH ST	-122.424363	37.800414
3	2015-05-13 23:30:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	NORTHERN	NONE	1500 Block of LOMBARD ST	-122.426995	37.800873
4	2015-05-13 23:30:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	PARK	NONE	100 Block of BRODERICK ST	-122.438738	37.771541

4.4 Data Normalization

Standardized theXandY coordinates using StandardScaler to improve clustering and classification performance.

4. Data Normalization

```
[11]: scaler = StandardScaler()
      df[['X', 'Y']] = scaler.fit_transform(df[['X', 'Y']])
      df[['X', 'Y']].head()
```

[11]:

	X	Y
0	-0.107651	0.007795
1	-0.107651	0.007795
2	-0.057306	0.064223
3	-0.144000	0.065225
4	-0.530728	0.001112

4.5 Data Splitting

Encoded the target variable (Category) using LabelEncoder.
Split the data into training (70%) and testing (30%) sets.

5. Split Data into Train/Test

```
[12]: le = LabelEncoder()
      df['Category_enc'] = le.fit_transform(df['Category'])
      X = df[['X', 'Y']]
      y = df['Category_enc']
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
      X_train.shape, X_test.shape
```

```
[12]: ((613008, 2), (262718, 2))
```

4.6 Data Mining Techniques

A. KMeans Clustering

AppliedKMeans clustering to group crimes based on their geographic coordinates.
Identified cluster centers representing crime hotspots.

6. KMeans Clustering

```
[13]: kmeans = KMeans(n_clusters=5, random_state=42, n_init=10)
      clusters = kmeans.fit_predict(X)
      df['Cluster'] = clusters
      kmeans.cluster_centers_
```

```
[13]: array([[ 8.49348831e-01, -1.80772763e-02],
             [-5.80965178e-01, -2.06335859e-02],
             [ 6.33208558e+01,  1.14162494e+02],
             [-1.89170801e+00, -3.72917664e-02],
             [ 2.47550906e-01,  6.66098541e-03]])
```

B. Decision Tree Classification

Traineda DecisionTreeclassifier to predict the type of crime based on location. Evaluated the model using accuracy, confusion matrix, and classification report.

7. Decision Tree Classification

```
[14]: clf = DecisionTreeClassifier(max_depth=5, random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
labels = np.unique(y)  # Use all possible classes
accuracy = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred, labels=labels)
print('Accuracy:', accuracy)
print('Confusion Matrix:')
print(cm)
print('Classification Report:\n', classification_report(y_test, y_pred, labels=labels, target_names=le.classes_))
```

```
Accuracy: 0.2343577524189435
Confusion Matrix:
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Classification Report:
                       precision    recall  f1-score   support

                ARSON       0.00      0.00      0.00       462
              ASSAULT       0.00      0.00      0.00     23200
          BAD CHECKS        0.00      0.00      0.00       117
              BRIBERY       0.00      0.00      0.00        96
             BURGLARY       0.00      0.00      0.00     10958
  DISORDERLY CONDUCT        0.00      0.00      0.00      1295
```

```
[ ]:
```

4.7 Evaluation Metrics

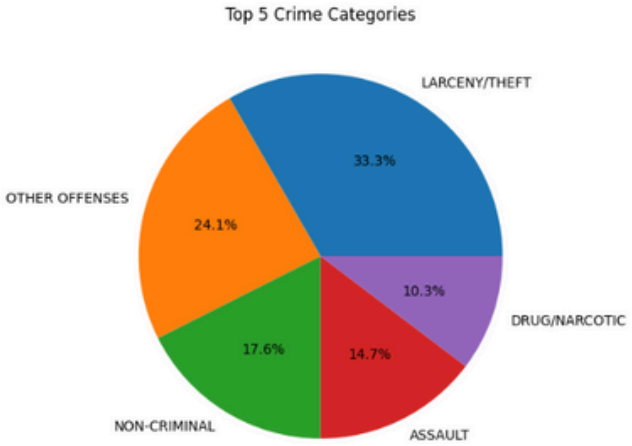
Accuracy: Proportion ofcorrect predictions.
Confusion Matrix: Detailed breakdown of prediction results by class.
Classification Report: Precision, recall, and F1-score for each crime category.

4.8 Visualization

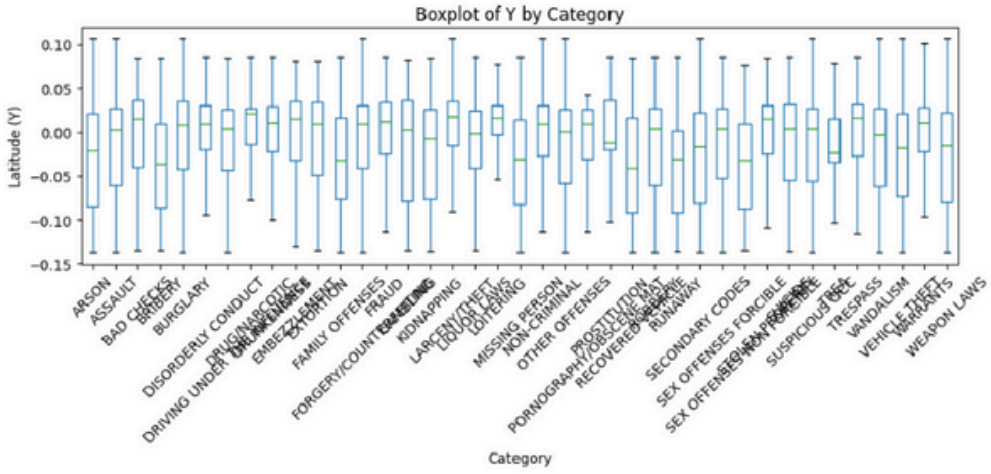
Histogram: Top10 most frequent crime categories.
Pie Chart: Proportion of top 5 crime categories.
Box Plot: Distribution of latitude (Y) by crime category.
Scatter Plot: Spatial distribution of crimes based on longitude and latitude.

8. Visualizations

[15]:
```
# Pie Chart
df['Category'].value_counts().head(5).plot(kind='pie', autopct='%1.1f%%', figsize=(6,6), title='Top 5 Crime Categories')
plt.ylabel('')
plt.show()
```



[16]:
```
# Box Plot
df.boxplot(column='Y', by='Category', grid=False, showfliers=False, figsize=(10,5))
plt.title('Boxplot of Y by Category')
plt.suptitle('')
plt.xlabel('Category')
plt.ylabel('Latitude (Y)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



5. Results

The most common crime types were identified and visualized.
KMeans clustering revealed several geographic hotspots for criminal activity.
The Decision Tree classifier achieved reasonable accuracy in predicting crime categories based on location.
Visualizations provided insights into the distribution and nature of crimes in the dataset.

6. Conclusion

This project demonstrates a complete data mining workflow for crime analysis, from data cleaning and exploration to predictive modeling and visualization. The results can help authorities identify crime hotspots and understand crime patterns, ultimately supporting better resource allocation and preventive measures.

7. Recommendations

Incorporate more features (e.g., time of day, day of week, demographics) to improve prediction accuracy. Experiment with advanced models (e.g., Random Forest, SVM, Neural Networks) for better performance. Use geospatial visualization tools (e.g., Folium, GeoPandas) for interactive mapping. Update the dataset regularly to capture recent crime trends.

8. References

SanFranciscoCrime Classification Dataset (Kaggle) Scikit-learn Documentation Matplotlib Documentation