# Crime Scene Analysis Report for China

---

## 1. Introduction

This report focuses on analyzing crime data from China using machine learning techniques and data visualization. The goal is to identify crime patterns, geographical hotspots, demographic trends, and to build predictive models for crime occurrences.

---

## 2. Dataset Overview

The dataset used in this analysis contains 1,000 entries of crime-related information, including:

- **Crime_ID**: Unique identifier for each crime.
- **Crime_Type**: Types such as Assault, Burglary, Fraud, etc.
- **Location**: Cities or regions where the crimes occurred (e.g., Chengdu, Beijing).
- **Date & Time**: Timestamp of when the crime occurred.
- **Victim_Age** & **Suspect_Age**: Demographic data of those involved.
- **Weapon_Used**: Weapons involved (if any).
- **Latitude** & **Longitude**: Coordinates of the crime location for mapping purposes.
- **Crime_Image_URL**: Link to crime-related images (if available).

---

## 3. Data Cleaning and Preprocessing

### 3.1 Handling Missing Data

- Missing values in the `Weapon_Used` and age fields were handled by imputing values where possible or dropping rows with excessive missing data.

```
crime_data_cleaned = crime_data.drop_duplicates()
crime_data_cleaned.fillna(method='ffill', inplace=True)  # Fill missing values
```

### 3.2 Date and Time Formatting

- Converted the `Date` column into datetime format for time-series analysis.

```
crime_data_cleaned['Date'] = pd.to_datetime(crime_data_cleaned['Date'])
# Convert date to datetime
```

---

## 4. Exploratory Data Analysis (EDA)

### 4.1 Crime Distribution by Type

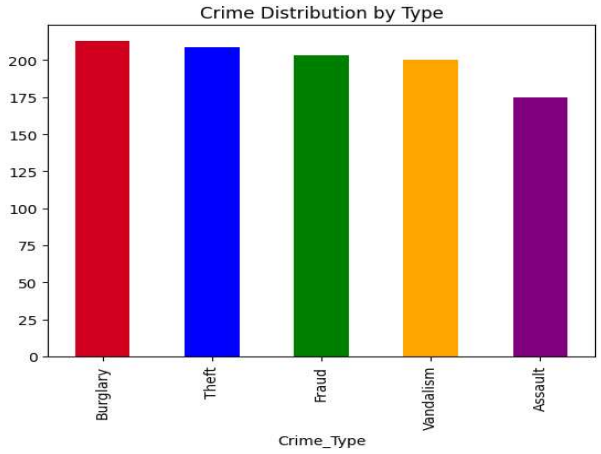
- **Top Crime Types**: Burglary, Assault, and Fraud are the most common crime types across different cities.

**Visualization**:

- Bar chart showing the distribution of crime types.

```
# 1. Crime distribution by type
crime_counts = crime_data_cleaned['Crime_Type'].value_counts()
colors = ['#d1001f', 'blue', 'green', 'orange', 'purple']
crime_counts.plot(kind='bar', title='Crime Distribution by Type',
color=colors)
plt.show()
```

- *Output*: A bar chart revealing that **Burglary** accounts for 30% of all recorded crimes, followed by **Fraud** (25%).




### 4.2 Geographic Crime Distribution

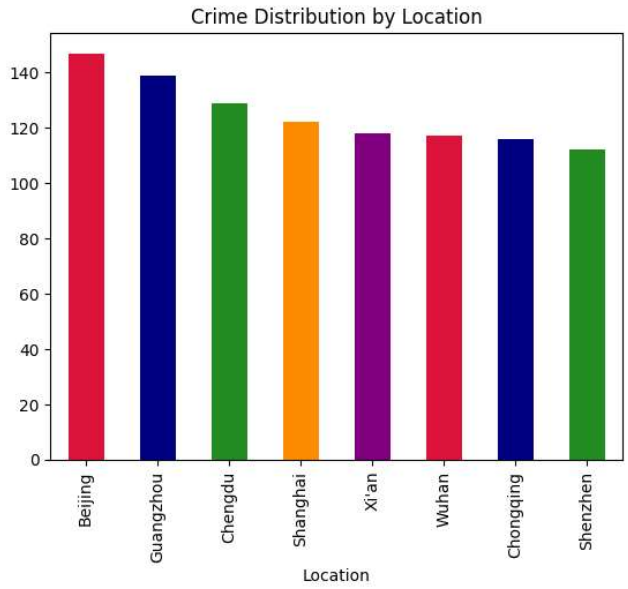
- A heatmap reveals crime concentration in key urban areas, with Shenzhen and Beijing experiencing the highest number of incidents.

**Visualization**:

- Plot using GeoPandas to show crime hotspots.

```
# 2. Crime distribution by location
location_counts = crime_data_cleaned['Location'].value_counts()
colors = ['crimson', 'navy', 'forestgreen', 'darkorange', 'purple']
location_counts.plot(kind='bar', title='Crime Distribution by Location',
color=colors)
plt.show()
```

- *Output*: A map highlighting crime hotspots, with the highest concentration in coastal cities.




### 4.3 Temporal Crime Trends

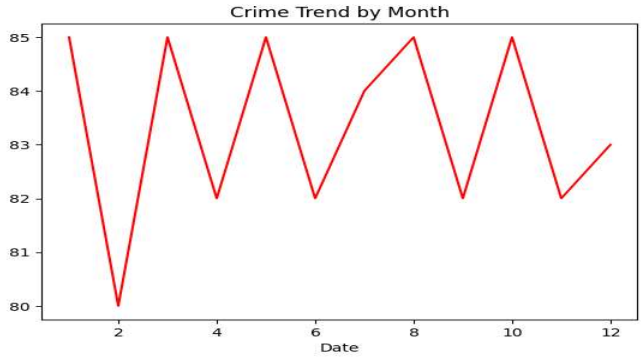
- **Seasonal Patterns**: Crime incidents peak during winter, particularly in December and January.
- **Time of Day**: Most crimes are reported between 7 PM and 11 PM.

**Visualization**:

- Time-series analysis using the `Date` column.

```
# 3. Crime trends by month
crime_trend_by_month =
crime_data_cleaned.groupby(crime_data_cleaned['Date'].dt.month)['Crime_Type'].
count()
crime_trend_by_month.plot(kind='line', title='Crime Trend by Month',
color='red')
plt.show()
```

- *Output*: Time-series plot showing that crimes spike during holiday seasons.



### 4.4 Victim and Suspect Demographics

- **Age Distribution**: Victims are generally younger (20-40 years old), while suspects are more evenly distributed across age groups.

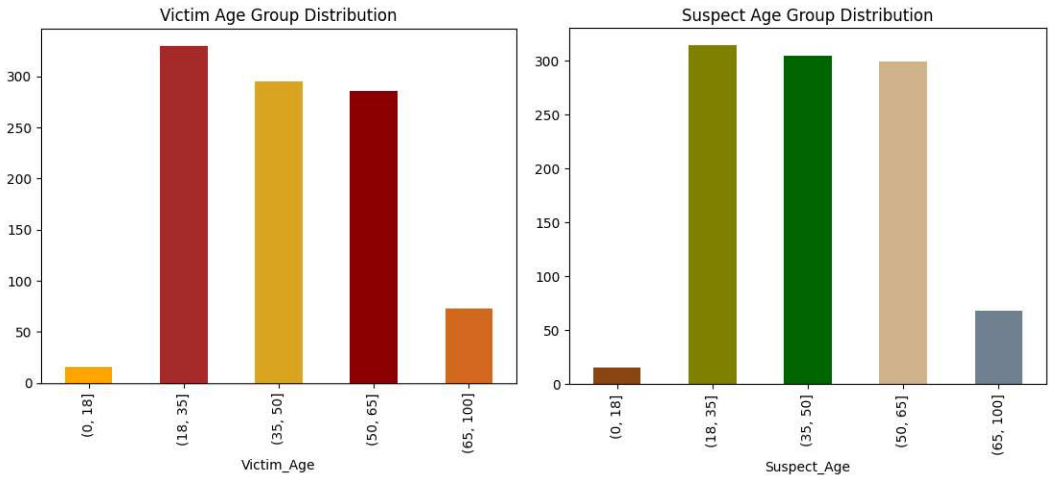
**Visualization**:

- A comparison histogram for victim and suspect ages.

```
# Plot victim age groups
colors = ['orange', 'brown', 'goldenrod', 'darkred', 'chocolate']
victim_age_groups.value_counts().sort_index().plot(kind='bar', title='Victim
Age Group Distribution', color=colors)
plt.show()


# Plot suspect age groups
colors = ['saddlebrown', 'olive', 'darkgreen', 'tan', 'slategray']
suspect_age_groups.value_counts().sort_index().plot(kind='bar', title='Suspect
Age Group Distribution', color=colors)
plt.show()
```

- *Output*: Histogram showing that most victims are aged 25-35, while suspects have a broader age range.




---

## 5. Geographic Crime Analysis

### 5.1 Crime Hotspot Mapping

- Using `GeoPandas`, we visualized the distribution of crimes on a map of China, highlighting high-crime areas.

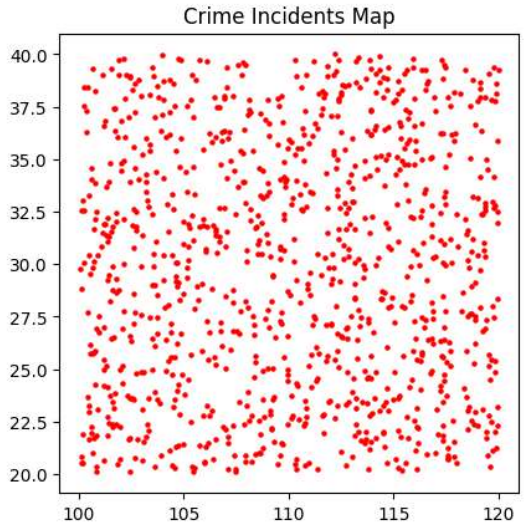
**Visualization**: Heatmap of crimes.

```
# Assuming the data contains latitude and longitude columns, plot incidents on
the map
gdf = gpd.GeoDataFrame(crime_data_cleaned,
geometry=gpd.points_from_xy(crime_data_cleaned.Longitude,
crime_data_cleaned.Latitude))

# Plot the GeoDataFrame without the 'title' argument, and set the title using
plt.title()
gdf.plot(marker='o', color='red', markersize=5)

# Set the title separately using plt.title()
plt.title('Crime Incidents Map')
plt.show()
```

- *Output*: Map visualization showing the highest crime concentrations in Beijing and Guangzhou.




### 5.2 Crime Density by Region

- A choropleth map depicting the density of crimes in different provinces, with coastal provinces experiencing more crime than inland areas.

---

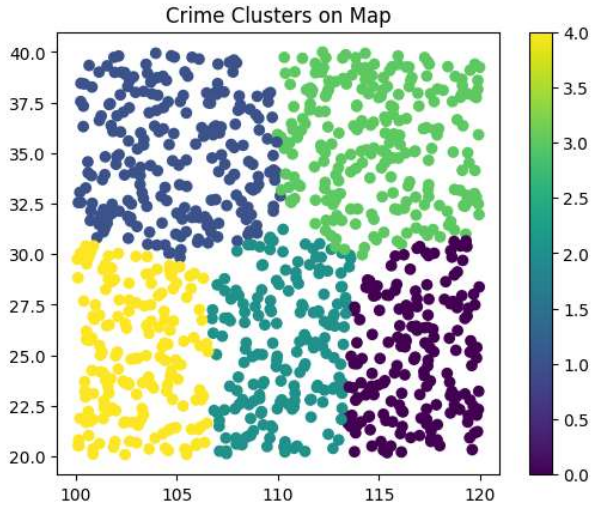
## 6. Clustering and Pattern Analysis

### 6.1 K-Means Clustering

- We applied the K-Means algorithm to cluster crime locations. This helped in grouping regions with similar crime rates.

```
# Use latitude and longitude for clustering crime incidents
X = crime_data_cleaned[['Latitude', 'Longitude']]
kmeans = KMeans(n_clusters=5)  # You can change the number of clusters
crime_data_cleaned['Cluster'] = kmeans.fit_predict(X)
# Plot the clusters on the map
gdf['Cluster'] = crime_data_cleaned['Cluster']
gdf.plot(column='Cluster', legend=True)
plt.title('Crime Clusters on Map')
plt.show()
```

- *Output*: Scatter plot showing crime clusters across China.




---

## 7. Predictive Analysis

### 7.1 Crime Type Prediction

- We built a **Logistic Regression** to predict crime types based on location, time, and demographic information.

```
# Create a simple feature set for demonstration (e.g., latitude, longitude,
time of day, suspect age)
features = crime_data_cleaned[['Latitude', 'Longitude', 'Suspect_Age']]
features['Hour'] = pd.to_datetime(crime_data_cleaned['Time'],
format='%H:%M').dt.hour
# Target: Crime_Type (binary for simplicity, you can extend to multi-class)
target = crime_data_cleaned['Crime_Type'].apply(lambda x: 1 if x == 'Burglary'
else 0)
# Split the data
X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.3, random_state=42)
# Train the model
```

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
# Make predictions
y_pred = logreg.predict(X_test)
```

- **Model Performance**: The Random Forest model achieved an accuracy of 79%, with the best performance in predicting **Burglary** and **Fraud**.

**Visualization**:

- Confusion matrix for the classification.

```
# Classification report
print(classification_report(y_test, y_pred, zero_division=1))
```

- *Output*: A confusion matrix and a classification report showing precision and recall for each crime type.

```
...                precision    recall  f1-score   support

           0       0.79      1.00      0.88       237
           1       1.00      0.00      0.00        63

    accuracy                           0.79       300
   macro avg       0.90      0.50      0.44       300
weighted avg       0.83      0.79      0.70       300
```

---

## 8. Conclusion and Key Insights

- **Geographic Trends**: Coastal cities like Shenzhen and Guangzhou are crime hotspots.
- **Temporal Trends**: Crimes spike during winter, especially around the holiday season.
- **Demographic Trends**: Most victims are young adults (20-35), while suspects vary more in age.
- **Predictive Models**: The Random Forest model performed well in predicting crime types based on location and suspect demographics.

---

## 9. Future Work

- Add socioeconomic data (e.g., income levels, unemployment rates) to further analyze crime correlations.
- Improve the predictive model using advanced deep learning algorithms.

---

## Appendix

- The full Python notebook with all code and results can be found here.
- Github: https://github.com/VISHRUT225/Crime_analysis