School of Mechanical & Manufacturing Engineering, SMME

National University of Science and Technology, NUST

FUNDAMENTALS OF PROGRAMMING 2

Name: Khansa Malik

Reg. No: 453832

Crime Data Analysis and Prediction

Introduction:

This lab report details the analysis and prediction of crime data in Pakistan using various data science techniques. The dataset used contains crime records from different regions of Pakistan.

Data Used:

C:\Users\Khansa Malik\Downloads\pakistan_crimes.csv

Key Concepts:

- **Libraries Used**:
  - `pandas`: For data manipulation and analysis.
  - `numpy`: For numerical operations.
  - `matplotlib.pyplot`: For data visualization.
  - `sklearn.cluster.KMeans`: For clustering analysis.
  - `sklearn.ensemble.IsolationForest`: For anomaly detection.
- **Methods and Functions**:
  - **Data Loading**: `pd.read_csv()` to load the dataset.
  - **Data Cleaning**: Handling missing values using `dropna()`.
  - **Exploratory Data Analysis (EDA)**: Grouping and plotting data to understand trends.
  - **Crime Rate Calculation**: Calculating crime rates per 100,000 people.
  - **Visualization**: Using `plt.plot()`, `plt.bar()`, and other plotting functions to visualize data.
  - **Correlation Analysis**: Using `corr()` to find correlations between regions.
  - **Probability Model**: Calculating probabilities of different crime types in different regions.

- **Clustering Analysis**: Using K-means clustering to group regions based on crime counts.
- **Anomaly Detection**: Using Isolation Forest to detect anomalies in crime data.

Data Loading and Cleaning:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.ensemble import IsolationForest

# Load the dataset
df = pd.read_csv(r'C:\Users\Khansa Malik\Downloads\pakistan_crimes.csv')

# Display the first few rows of the dataset
print(df.head())

# Check for the correct column names
print("Column names in the dataset:", df.columns)

# Convert 'Year' column to datetime format (using only the year)
df['Year'] = pd.to_datetime(df['Year'], format='%Y')

# Handle missing values
df = df.dropna()
```

Exploratory Data Analysis (EDA):

- **Number of Crimes per Year**

```
crimes_per_year =
df.groupby(df['Year'].dt.year).sum(numeric_only=True).drop(columns=['
_id'])

plt.figure(figsize=(10, 6))
plt.plot(crimes_per_year.index, crimes_per_year.sum(axis=1),
marker='o')
plt.title('Number of Crimes per Year')
plt.xlabel('Year')
plt.ylabel('Number of Crimes')
plt.grid(True)
plt.show()
```

- **Crime Rate per 100,000 People**

```
regions = ['Punjab', 'Sindh', 'KP', 'Balochistan',
'Islamabad','Railways', 'G.B', 'AJK']
population_data = {
'Punjab': 110012442,
'Sindh': 47850000,
'KP': 3998876,
'Balochistan': 12300000,
'Islamabad': 1014825,
'Railways': 1000000,
'G.B': 1492924,
'AJK': 4045366
}

crime_rate = {}
for region in regions:
total_crimes = df[region].sum()
population = population_data[region]
crime_rate[region] = (total_crimes / population) * 100000

plt.figure(figsize=(12, 6))
plt.bar(crime_rate.keys(), crime_rate.values())
plt.title('Crime Rate per 100,000 People')
plt.xlabel('Region')
plt.ylabel('Crime Rate')
plt.grid(True)
plt.show()
```

- **Distribution of Crime Types**

```
crime_type_distribution = df['Offence'].value_counts()

plt.figure(figsize=(12, 6))
crime_type_distribution.plot(kind='bar')
plt.title('Distribution of Crime Types')
plt.xlabel('Crime Type')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.grid(True)
plt.show()
```

- **Yearly Comparison of Crime Types**

```
yearly_crime_types = df.groupby([df['Year'].dt.year,
'Offence']).sum(numeric_only=True).drop(columns=['_id']).unstack(fill
_value=0)

plt.figure(figsize=(12, 8))
```

```
yearly_crime_types.plot(kind='bar', stacked=True, figsize=(12, 8))
plt.title('Yearly Comparison of Crime Types')
plt.xlabel('Year')
plt.ylabel('Number of Crimes')
plt.legend(title='Crime Type')
plt.grid(True)
plt.tight_layout()
plt.show()
```

- **Trends of Specific Crime Types in Different Regions**

```
crime_type_to_analyze = input("Enter the crime you want to analyze:
")
region_trends = df[df['Offence'] ==
crime_type_to_analyze].groupby([df['Year'].dt.year])[regions].sum()

plt.figure(figsize=(12, 8))
for region in regions:
plt.plot(region_trends.index, region_trends[region], marker='o',
label=region)
plt.title(f'Trends of {crime_type_to_analyze} in Different Regions')
plt.xlabel('Year')
plt.ylabel('Number of Crimes')
plt.legend(title='Region')
plt.grid(True)
plt.tight_layout()
plt.show()
```

- **Percentage Change in Crime Rates Over Time**

```
crime_rate_change = crimes_per_year.pct_change().dropna() * 100

plt.figure(figsize=(12, 6))
plt.plot(crime_rate_change.index, crime_rate_change.sum(axis=1),
marker='o')
plt.title('Percentage Change in Crime Rates Over Time')
plt.xlabel('Year')
plt.ylabel('Percentage Change')
plt.grid(True)
plt.tight_layout()
plt.show()
```

- **Total Crime Severity Score Over the Years**

```
severity_scores = {
'Murder': 10,
'Attempt to Murder': 8,
'Kidnapping /Abduction': 7,
```

```
'Dacoity': 6,
'Robbery': 5,
}
df['Severity_Score'] = df['Offence'].map(severity_scores)
severity_per_year =
df.groupby(df['Year'].dt.year)['Severity_Score'].sum()

plt.figure(figsize=(12, 6))
plt.plot(severity_per_year.index, severity_per_year, marker='o')
plt.title('Total Crime Severity Score Over the Years')
plt.xlabel('Year')
plt.ylabel('Total Severity Score')
plt.grid(True)
plt.tight_layout()
plt.show()
```

- **Correlation Analysis**

```
correlation_matrix = df[regions].corr()
plt.figure(figsize=(10, 8))
plt.imshow(correlation_matrix, cmap='coolwarm', interpolation='none')
plt.colorbar()
plt.xticks(range(len(correlation_matrix.columns)),
correlation_matrix.columns, rotation=90
plt.yticks(range(len(correlation_matrix.columns)),correlation_matrix.
columns)
plt.title('Correlation Matrix')
plt.show()
```

Probability Model for Crime Prediction

Calculate Probabilities

```
crime_types = df['Offence'].unique()

probabilities = {}
for region in regions:
    region_data = df[region]
    total_crimes_in_region = region_data.sum()
    probabilities[region] = {}
    for crime_type in crime_types:
        crime_count = df[df['Offence'] == crime_type][region].sum()
        probabilities[region][crime_type] = crime_count /
total_crimes_in_region

print("Probabilities of Different Types of Crimes in Different Regions:")
for region, crime_probs in probabilities.items():
    print(f"Region: {region}")
```

```
    for crime_type, prob in crime_probs.items():
        print(f"  Crime Type: {crime_type}, Probability: {prob:.4f}")
```

Predict Crime Likelihood

```
def predict_crime(region, crime_type):
    if region in probabilities and crime_type in probabilities[region]:
        return probabilities[region][crime_type]
    else:
        return 0.0
```

Clustering Analysis using K-means

```
crime_data = df[regions].sum().values.reshape(-1, 1)

kmeans = KMeans(n_clusters=3, random_state=0).fit(crime_data)
labels = kmeans.labels_

plt.figure(figsize=(10, 6))
plt.scatter(regions, crime_data, c=labels, cmap='viridis')
plt.title('K-means Clustering of Regions Based on Total Crime Counts')
plt.xlabel('Region')
plt.ylabel('Total Crime Counts')
plt.grid(True)
plt.show()
```

Anomaly Detection using Isolation Forest

```
total_crimes_per_year =
df.groupby(df['Year'].dt.year).sum(numeric_only=True).drop(columns=['_id'])
.sum(axis=1).values.reshape(-1, 1)

iso_forest = IsolationForest(contamination=0.1, random_state=0)
anomalies = iso_forest.fit_predict(total_crimes_per_year)

plt.figure(figsize=(10, 6))
plt.plot(df['Year'].dt.year.unique(), total_crimes_per_year, marker='o',
label='Total Crimes')
plt.scatter(df['Year'].dt.year.unique()[anomalies == -1],
total_crimes_per_year[anomalies == -1], color='red', label='Anomalies')
plt.title('Anomaly Detection in Total Crimes Per Year')
plt.xlabel('Year')
plt.ylabel('Total Crimes')
plt.legend()
plt.grid(True)
plt.show()
```
