In [640…

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Load MinWage variable

In [641…

```
data = pd.read_csv("./minimum wage.csv")
data['Effective Date'] = pd.to_datetime(data['Effective Date'], format='%(
data = data.sort_values(by='Effective Date')

data['Minimum Wage'] = data['Minimum Wage'].replace('[\$,]', '', regex=Tru

monthly_range = pd.date_range(start='2003-01-01', end='2024-09-01', freq=
monthly_df = pd.DataFrame({'date': monthly_range})

data = data.set_index('Effective Date')
monthly_df = monthly_df.merge(data[['Minimum Wage']], how='left', left_on=

monthly_df['MinWage'] = monthly_df['Minimum Wage'].ffill().bfill()

monthly_df.loc[monthly_df['date'] < '2011-05-01', 'MinWage'] = 8.00

monthly_df = monthly_df.drop(columns=['Minimum Wage'])

monthly_df.head(1000)
```

Out[641…

	date	MinWage
0	2003-01-01	8.00
1	2003-02-01	8.00
2	2003-03-01	8.00
3	2003-04-01	8.00
4	2003-05-01	8.00
...	...	...
256	2024-05-01	16.75
257	2024-06-01	17.40
258	2024-07-01	17.40
259	2024-08-01	17.40
260	2024-09-01	17.40

261 rows × 2 columns

load crime_number variable

In [642…

```
crime_data = pd.read_csv("./crimedata_csv_AllNeighbourhoods_AllYears/crim

crime_data['date'] = pd.to_datetime(crime_data[['YEAR', 'MONTH']].assign(

monthly_crime_counts = crime_data.groupby('date').size().reset_index(name

monthly_df = monthly_df.merge(monthly_crime_counts, how='left', on='date'

monthly_df['crime_number'] = monthly_df['crime_number'].fillna(0)

monthly_df.head(1000)
```

Out[642…

```
           date  MinWage  crime_number
  0  2003-01-01     8.00          4926
  1  2003-02-01     8.00          4148
  2  2003-03-01     8.00          4550
  3  2003-04-01     8.00          4759
  4  2003-05-01     8.00          5297
...         ...      ...           ...
256  2024-05-01    16.75          3007
257  2024-06-01    17.40          2810
258  2024-07-01    17.40          3053
259  2024-08-01    17.40          2977
260  2024-09-01    17.40          2566

261 rows × 3 columns
```

In [643…

```
cpi_data = pd.read_csv("./1810000601-eng (1).csv", header=0, index_col=0)

cpi_data = cpi_data.T

cpi_data = cpi_data.rename(columns={"All-items 8": "CPI"})

cpi_data = cpi_data.reset_index().rename(columns={"index": "date"})
cpi_data['date'] = pd.to_datetime(cpi_data['date'], format='%b-%y')

cpi_data = cpi_data[['date', 'CPI']]

cpi_data.index.name = "index"

monthly_df = monthly_df.merge(cpi_data, how='left', on='date')

monthly_df.head(1000)
```

Out[643…

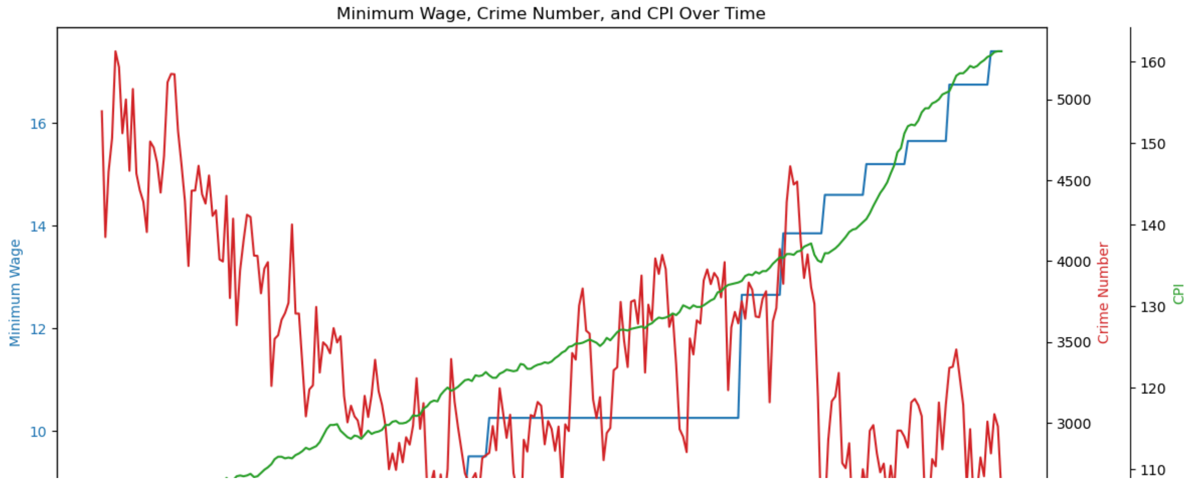
```
         date  MinWage  crime_number    CPI
0  2003-01-01     8.00          4926  102.4
1  2003-02-01     8.00          4148  102.9
```

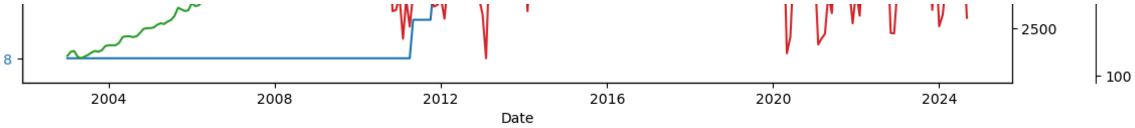
1	2003-02-01	8.00	4148	102.9
2	2003-03-01	8.00	4550	103.0
3	2003-04-01	8.00	4759	102.3
4	2003-05-01	8.00	5297	102.1
...	...	...	...	...
256	2024-05-01	16.75	3007	160.6
257	2024-06-01	17.40	2810	160.8
258	2024-07-01	17.40	3053	161.2
259	2024-08-01	17.40	2977	161.3
260	2024-09-01	17.40	2566	161.3

261 rows × 4 columns

In [644…

```
fig, ax1 = plt.subplots(figsize=(12, 6))

color = 'tab:blue'
ax1.set_xlabel('Date')
ax1.set_ylabel('Minimum Wage', color=color)
ax1.plot(monthly_df['date'], monthly_df['MinWage'], color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()
color = 'tab:red'

ax2.set_ylabel('Crime Number', color=color)
ax2.plot(monthly_df['date'], monthly_df['crime_number'], color=color)

ax3 = ax1.twinx()
color = 'tab:green'
ax3.spines['right'].set_position(('outward', 60))
ax3.set_ylabel('CPI', color=color)
ax3.plot(monthly_df['date'], monthly_df['CPI'], color=color)

fig.tight_layout()
plt.title('Minimum Wage, Crime Number, and CPI Over Time')
plt.show()
```





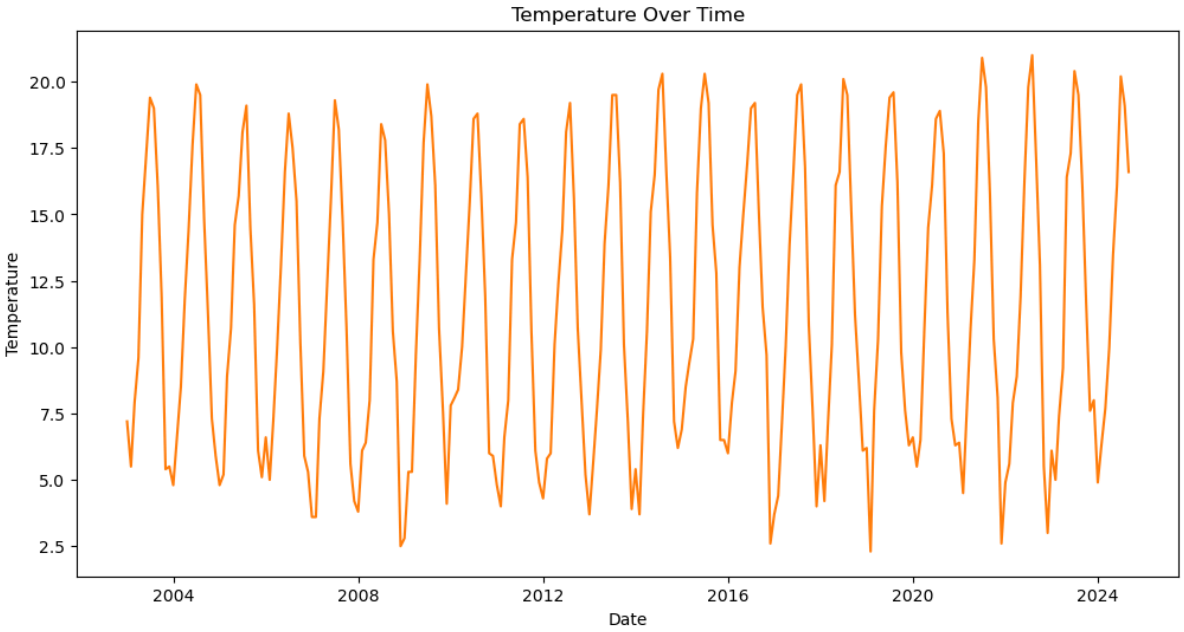
Load temperature variable

In [645…

```
temperature_data = pd.read_csv("./temperature.csv", header=None, names=["

temperature_data['date'] = pd.to_datetime(temperature_data['date'], forma

merged_df = pd.merge(monthly_df, temperature_data, how='left', on='date')
```

In [646…

```
plt.figure(figsize=(12, 6))
plt.plot(merged_df['date'], merged_df['temperature'], color='tab:orange')
plt.title('Temperature Over Time')
plt.xlabel('Date')
plt.ylabel('Temperature')
plt.show()
```



Loaded police officer ratio and weighted clearance rate

In [647…

```
police_data = pd.read_csv("./unemployment_police.csv")

police_data['Date'] = pd.to_datetime(police_data['Date'])

police_data.rename(columns={'Date': 'date'}, inplace=True)

merged_df = pd.merge(merged_df, police_data, how='left', on='date')

merged_df.head(6)
```

Out[647…

	date	MinWage	crime_number	CPI	temperature	Police officers per 100,000 population	Weighted clearance rate	Uner
0	2003-01-01	8.0	4926	102.4	7.2	204.2	22.74	
1	2003-02-01	8.0	4148	102.9	5.5	204.2	22.74	
2	2003-03-01	8.0	4550	103.0	7.9	204.2	22.74	
3	2003-04-01	8.0	4759	102.3	9.6	204.2	22.74	
4	2003-05-01	8.0	5297	102.1	15.0	204.2	22.74	
5	2003-06-01	8.0	5199	102.3	17.2	204.2	22.74	

In [648…

```
# print the average of crime number, minimum wage, CPI, temperature, unem
print("Average of crime number: ", merged_df['crime_number'].mean())
print("Average of minimum wage: ", merged_df['MinWage'].mean())
print("Average of CPI: ", merged_df['CPI'].mean())
print("Average of temperature: ", merged_df['temperature'].mean())
```

```
Average of crime number:  3434.015325670498
Average of minimum wage:  10.696360153256707
Average of CPI:  125.5800766283525
Average of temperature:  11.418773946360155
```

In [649…

```
import pandas as pd

def generate_data_description(df):
    stats = df.describe(include='all').transpose()

    stats['unique_values'] = df.nunique()

    stats = stats[['count', 'unique_values', 'mean', 'std', 'min', '25%',

    stats.index.name = 'column_name'

    return stats

data_description = generate_data_description(merged_df)

data_description
```

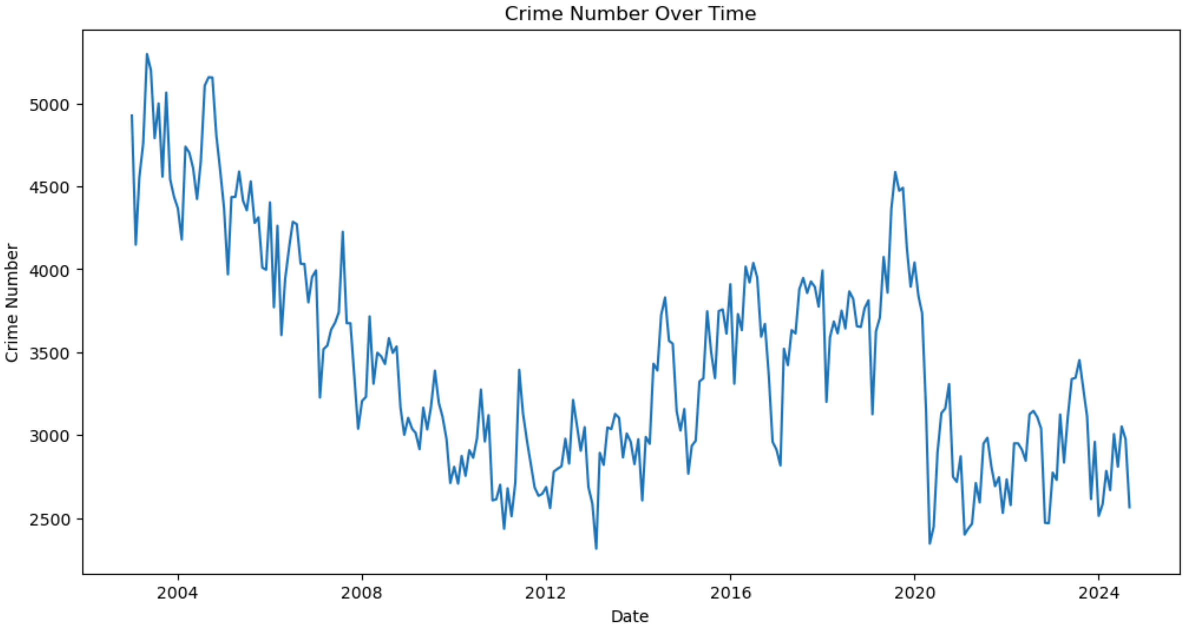
```
/var/folders/qx/5b_gxq5s5fqc9hdq8fcjn60h0000gn/T/ipykernel_15041/251178911
1.py:4: FutureWarning: Treating datetime data as categorical rather than nu
meric in `.describe` is deprecated and will be removed in a future version
of pandas. Specify `datetime_is_numeric=True` to silence this warning and a
dopt the future behavior now.
  stats = df.describe(include='all').transpose()
```

Out[649…

	count	unique_values	mean	std	min	25%	50
column_name							
date	261	261	NaN	NaN	NaN	NaN	N
MinWage	261.0	11	10.69636	2.960168	8.0	8.0	10
crime_number	261.0	245	3434.015326	672.739325	2316.0	2913.0	331
CPI	261.0	206	125.580077	15.707729	102.1	113.7	12
temperature	261.0	136	11.418774	5.351577	2.3	6.5	1
Police officers per 100,000 population	261.0	20	204.136782	15.397368	183.8	191.5	19
Weighted clearance rate	261.0	19	26.333678	2.262443	22.74	24.53	25
Vancouver Unemployment Rate	261.0	56	6.164368	1.572723	3.2	4.8	

In [650…

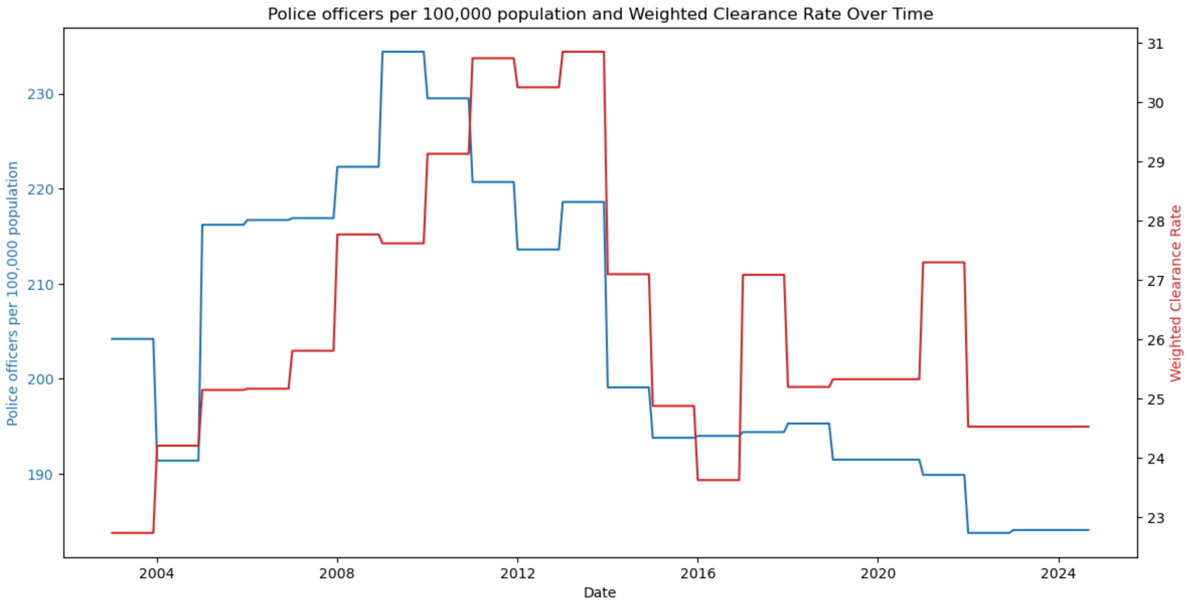
```
# plot the crime number data over time
plt.figure(figsize=(12, 6))
plt.plot(merged_df['date'], merged_df['crime_number'], color='tab:blue')
plt.title('Crime Number Over Time')
plt.xlabel('Date')
plt.ylabel('Crime Number')
plt.show()
```



In [651…

```
# visualize Police officers per 100,000 population and weighted clearance
fig, ax1 = plt.subplots(figsize=(12, 6))

color = 'tab:blue'
ax1.set_xlabel('Date')
ax1.set_ylabel('Police officers per 100,000 population', color=color)
```

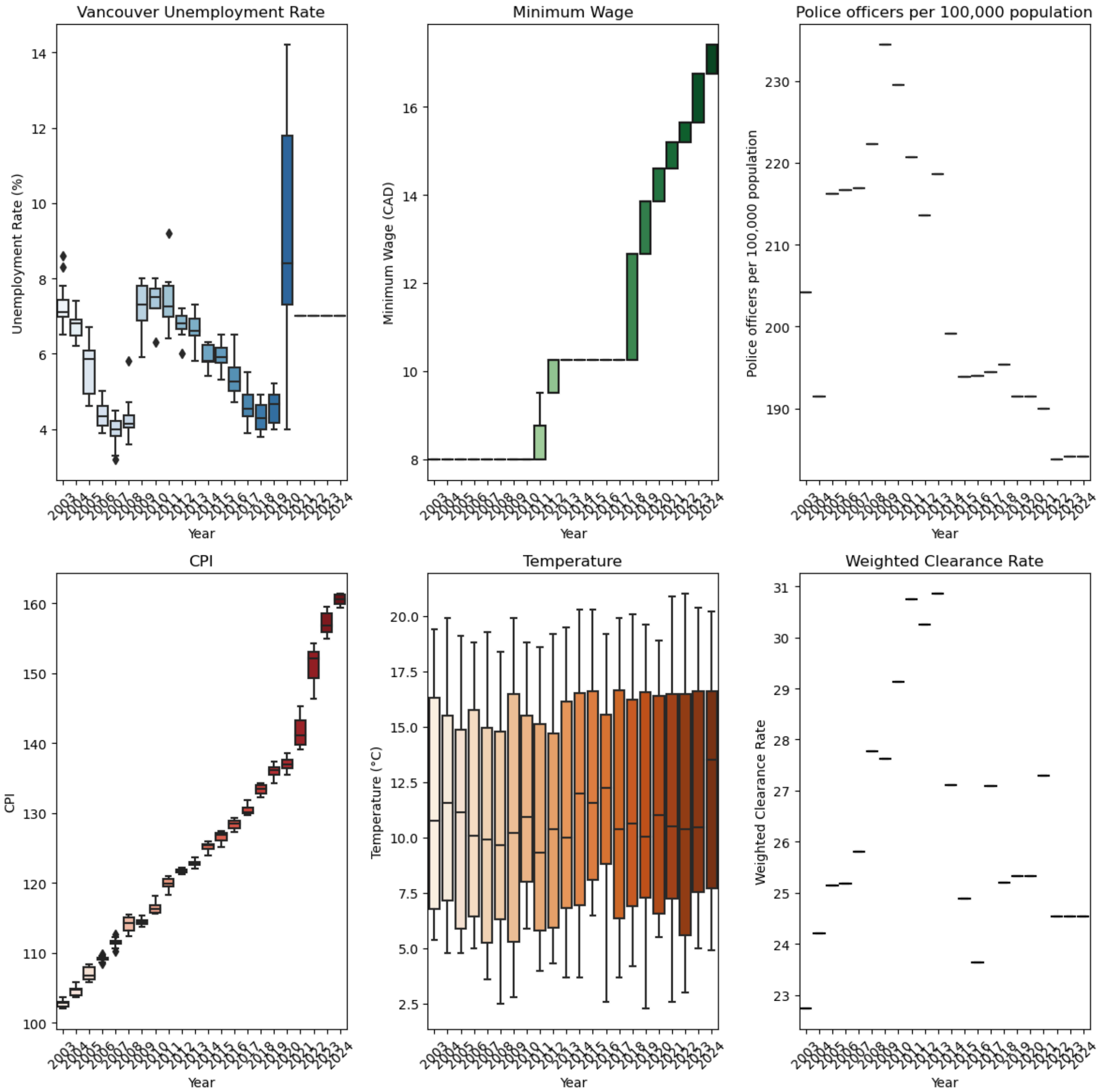
```
ax1.plot(merged_df['date'], merged_df['Police officers per 100,000 popula
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()
color = 'tab:red'

ax2.set_ylabel('Weighted Clearance Rate', color=color)
ax2.plot(merged_df['date'], merged_df['Weighted clearance rate'], color=co

fig.tight_layout()
plt.title('Police officers per 100,000 population and Weighted Clearance R
plt.show()
```



In [652…

```
import matplotlib.pyplot as plt
import seaborn as sns

merged_df['year'] = merged_df['date'].dt.year

fig, ax = plt.subplots(2, 3, figsize=(12, 12))

sns.boxplot(data=merged_df, x='year', y='Vancouver Unemployment Rate', ax=
ax[0, 0].set_title('Vancouver Unemployment Rate')
ax[0, 0].set_xlabel('Year')
ax[0, 0].set_ylabel('Unemployment Rate (%)')
ax[0, 0].tick_params(axis='x', rotation=45)

sns.boxplot(data=merged_df, x='year', y='MinWage', ax=ax[0, 1], palette='(
ax[0, 1].set_title('Minimum Wage')
ax[0, 1].set_xlabel('Year')
ax[0, 1].set_ylabel('Minimum Wage (CAD)')
ax[0, 1].tick_params(axis='x', rotation=45)

sns.boxplot(data=merged_df, x='year', y='CPI', ax=ax[1, 0], palette='Reds
ax[1, 0].set_title('CPI')
ax[1, 0].set_xlabel('Year')
ax[1, 0].set_ylabel('CPI')
ax[1, 0].tick_params(axis='x', rotation=45)

sns.boxplot(data=merged_df, x='year', y='temperature', ax=ax[1, 1], palett
ax[1, 1].set_title('Temperature')
```

```
ax[1, 1].set_xlabel('Year')
ax[1, 1].set_ylabel('Temperature (°C)')
ax[1, 1].tick_params(axis='x', rotation=45)

sns.boxplot(data=merged_df, x='year', y='Police officers per 100,000 popu
ax[0, 2].set_title('Police officers per 100,000 population')
ax[0, 2].set_xlabel('Year')
ax[0, 2].set_ylabel('Police officers per 100,000 population')
ax[0, 2].tick_params(axis='x', rotation=45)

sns.boxplot(data=merged_df, x='year', y='Weighted clearance rate', ax=ax[
ax[1, 2].set_title('Weighted Clearance Rate')
ax[1, 2].set_xlabel('Year')
ax[1, 2].set_ylabel('Weighted Clearance Rate')
ax[1, 2].tick_params(axis='x', rotation=45)

plt.suptitle('')
plt.tight_layout()

# Show the plot
plt.show()
```



In [653

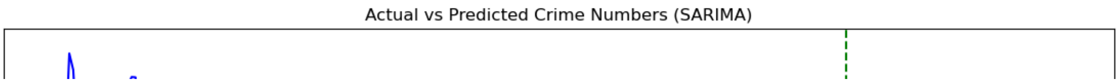
```
merged_df_arima = merged_df.copy()
merged_df_xgb = merged_df.copy()
```

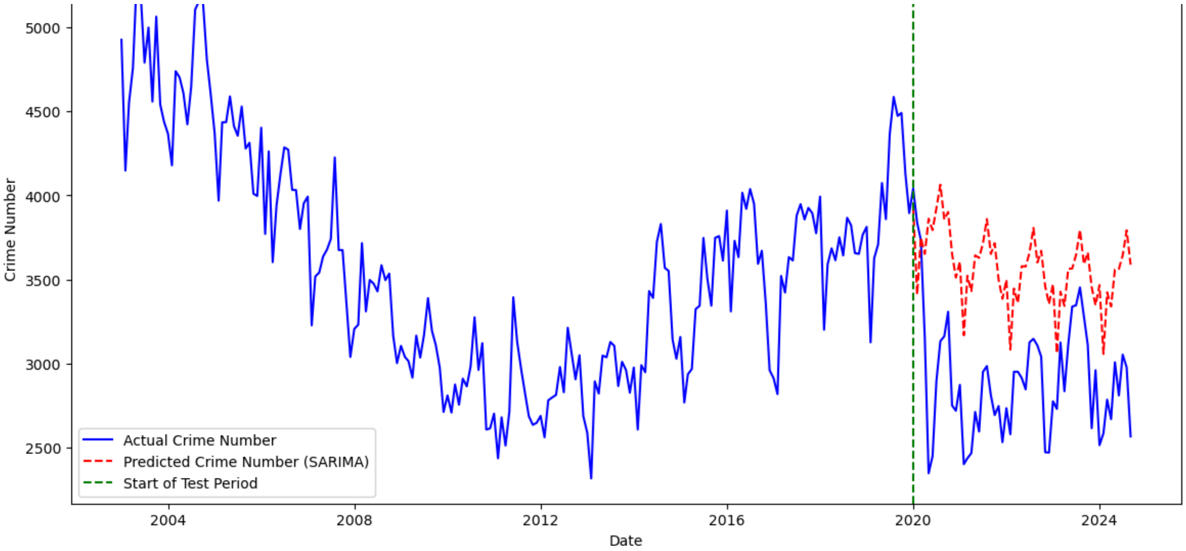
Method 1: SARIMA

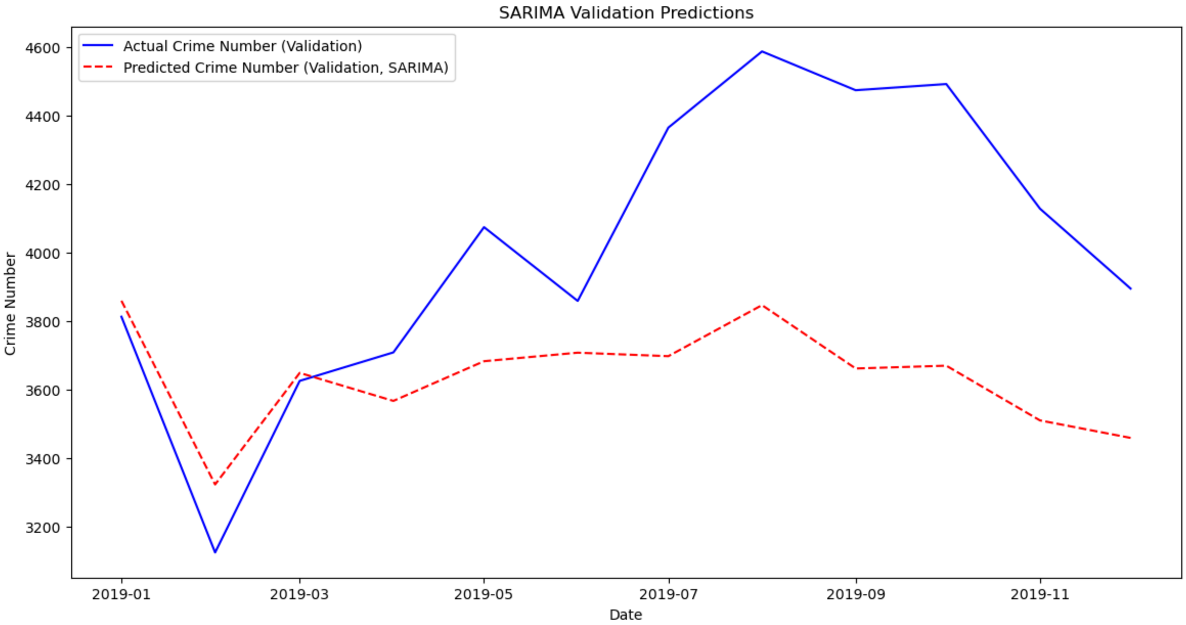
In [654…

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_absolute_percentage_error
import matplotlib.pyplot as plt
import pandas as pd

train_data = merged_df_arima[(merged_df_arima['date'] < '2019-01-01')]
val_data = merged_df_arima[(merged_df_arima['date'] >= '2019-01-01') & (me

crime_series_train = train_data.set_index('date')['crime_number'].asfreq(
crime_series_val = val_data.set_index('date')['crime_number'].asfreq('MS'

sarima_model = SARIMAX(crime_series_train, order=(1, 0, 0), seasonal_orde

forecast_val = sarima_model.predict(start=crime_series_val.index[0], end=

mape_val = mean_absolute_percentage_error(crime_series_val, forecast_val)
print(f"Validation MAPE: {mape_val * 100:.2f}%")

full_train_data = merged_df_arima[merged_df_arima['date'] < '2020-01-01']
sarima_model_full = SARIMAX(full_train_data, order=(1, 0, 0), seasonal_or

forecast_start = '2020-01-01'
forecast_end = merged_df_arima['date'].max()
forecast_test = sarima_model_full.predict(start=forecast_start, end=foreca

forecast_test_df = forecast_test.reset_index()
forecast_test_df.columns = ['date', 'predicted_crime_number_arima']
merged_df_arima = pd.merge(merged_df_arima, forecast_test_df, how='left',

plt.figure(figsize=(14, 7))
plt.plot(merged_df_arima['date'], merged_df_arima['crime_number'], label='
plt.plot(merged_df_arima['date'], merged_df_arima['predicted_crime_number_
plt.axvline(x=pd.to_datetime('2020-01-01'), color='green', linestyle='--'
plt.xlabel("Date")
plt.ylabel("Crime Number")
plt.title("Actual vs Predicted Crime Numbers (SARIMA)")
plt.legend()
plt.show()



plt.figure(figsize=(14, 7))
plt.plot(crime_series_val.index, crime_series_val, label="Actual Crime Nu
plt.plot(crime_series_val.index, forecast_val, label="Predicted Crime Numb
plt.xlabel("Date")
plt.ylabel("Crime Number")
plt.title("SARIMA Validation Predictions")
plt.legend()
plt.show()
```

```
Validation MAPE: 9.95%
```



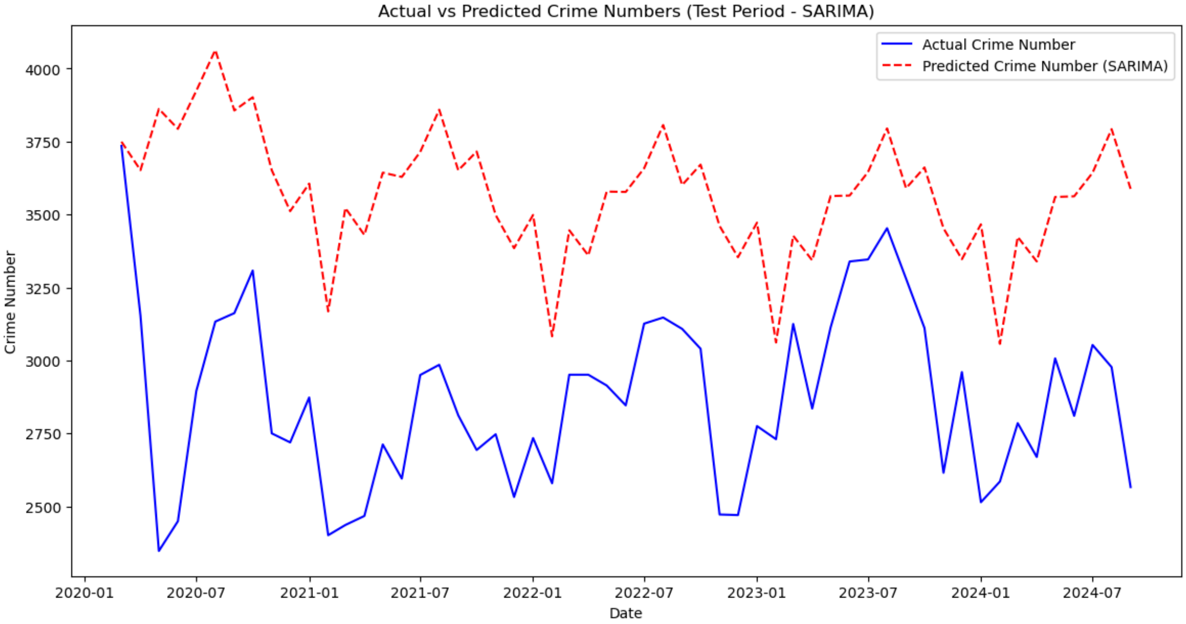




In [655…

```
from sklearn.metrics import mean_absolute_percentage_error

test_data_actual = merged_df_arima.loc[merged_df_arima['date'] >= '2020-0
test_data_predicted = merged_df_arima.loc[merged_df_arima['date'] >= '202

valid_indices = test_data_actual.notna() & test_data_predicted.notna()
test_data_actual = test_data_actual[valid_indices]
test_data_predicted = test_data_predicted[valid_indices]

mape_test = mean_absolute_percentage_error(test_data_actual, test_data_pre
print(f"Test MAPE: {mape_test * 100:.2f}%")

percent_diff = ((test_data_actual - test_data_predicted) / test_data_actu
print(f"Average percent difference after 2020-03-01: {percent_diff:.2f}%"

plt.figure(figsize=(14, 7))
plt.plot(merged_df_arima.loc[merged_df_arima['date'] >= '2020-03-01', 'da
plt.plot(merged_df_arima.loc[merged_df_arima['date'] >= '2020-03-01', 'da
plt.xlabel("Date")
plt.ylabel("Crime Number")
plt.title("Actual vs Predicted Crime Numbers (Test Period - SARIMA)")
plt.legend()
plt.show()
```

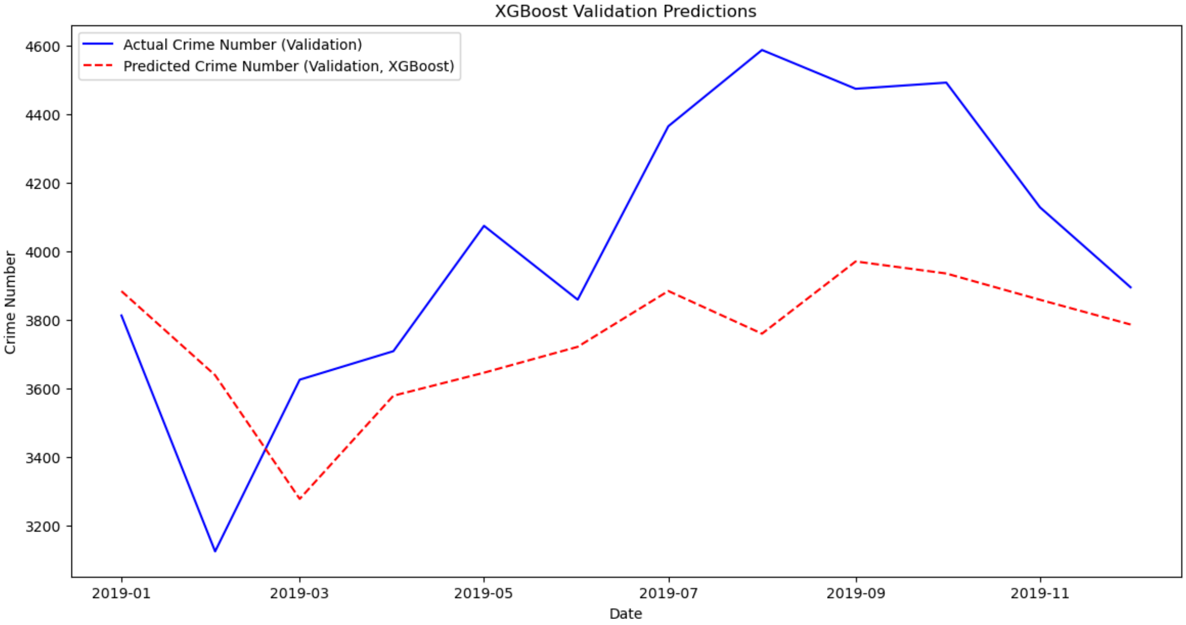
```
Test MAPE: 25.31%
Average percent difference after 2020-03-01: -25.31%
```



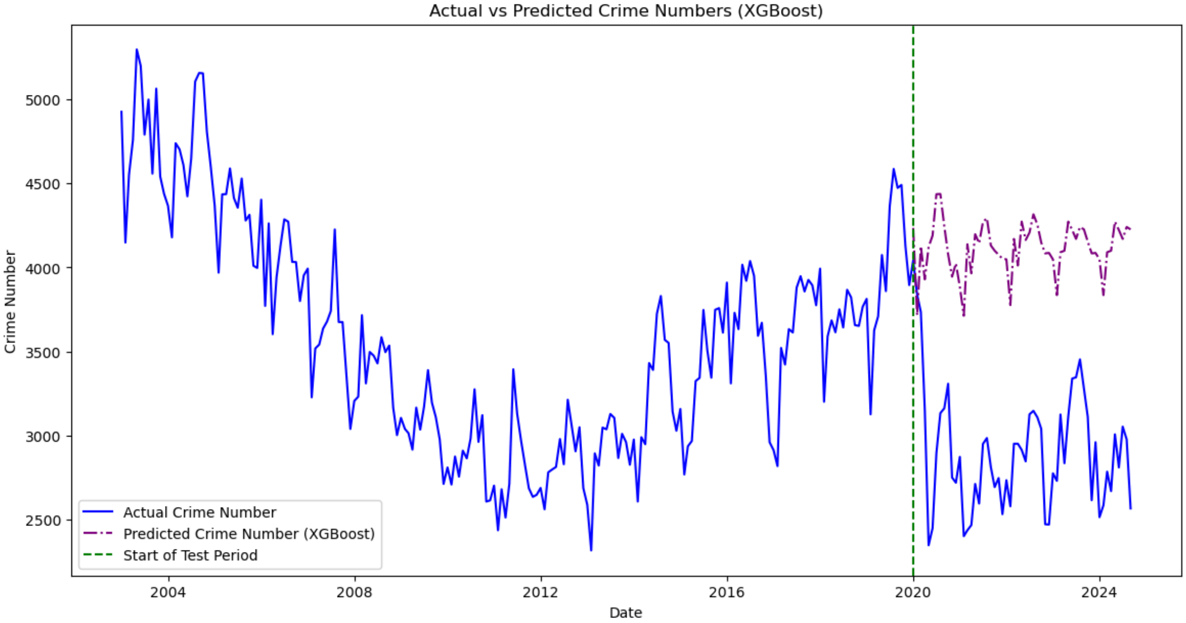
Method 2: XGBoost

In [656…

```
import xgboost as xgb
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_percentage_error

def create_lagged_features(df, target_col, lags):
    df = df.copy()
    for lag in range(1, lags + 1):
        df[f'{target_col}_lag_{lag}'] = df[target_col].shift(lag)
    return df

train_data = merged_df_xgb[merged_df_xgb['date'] < '2019-01-01']
val_data = merged_df_xgb[(merged_df_xgb['date'] >= '2019-01-01') & (merged

train_data = create_lagged_features(train_data, 'crime_number', 3)
val_data = create_lagged_features(val_data, 'crime_number', 3)

val_data.fillna(method='ffill', inplace=True)
val_data.fillna(val_data.mean(numeric_only=True), inplace=True)

train_data['month'] = train_data['date'].dt.month
train_data['year'] = train_data['date'].dt.year
val_data['month'] = val_data['date'].dt.month
val_data['year'] = val_data['date'].dt.year

features = [
    'crime_number_lag_1', 'crime_number_lag_2', 'crime_number_lag_3',
    'month', 'year', 'Police officers per 100,000 population',
    'Weighted clearance rate', 'Vancouver Unemployment Rate',
    'CPI', 'MinWage'
]
X_train = train_data[features]
X_train = train_data[features]
y_train = train_data['crime_number']
X_val = val_data[features]
y_val = val_data['crime_number']

xgb_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=1
xgb_model.fit(X_train, y_train)

val_predictions = xgb_model.predict(X_val)

plt.figure(figsize=(14, 7))

plt.plot(val_data['date'], y_val, label="Actual Crime Number (Validation)"
plt.plot(val_data['date'], val_predictions, label="Predicted Crime Number

plt.xlabel("Date")
plt.ylabel("Crime Number")
plt.title("XGBoost Validation Predictions")
plt.legend()
plt.show()

mape_val = mean_absolute_percentage_error(y_val, val_predictions)
print(f"Validation MAPE: {mape_val * 100:.2f}%")

full_train_data = merged_df_xgb[merged_df_xgb['date'] < '2020-01-01']
full_train_data = create_lagged_features(full_train_data, 'crime_number',
full_train_data['month'] = full_train_data['date'].dt.month
full_train_data['year'] = full_train_data['date'].dt.year

X_full_train = full_train_data[features]
y_full_train = full_train_data['crime_number']


xgb_model.fit(X_full_train, y_full_train)


test_data = merged_df_xgb[merged_df_xgb['date'] >= '2020-01-01'].copy()
test_data = create_lagged_features(test_data, 'crime_number', 3).fillna(me
test_data['month'] = test_data['date'].dt.month
test_data['year'] = test_data['date'].dt.year

test_predictions = []
for i in range(len(test_data)):
    X_test_row = test_data[features].iloc[i].values.reshape(1, -1)
    pred = xgb_model.predict(X_test_row)[0]
    test_predictions.append(pred)

    if i + 1 < len(test_data):
        test_data.loc[test_data.index[i + 1], 'crime_number_lag_1'] = pre
        test_data.loc[test_data.index[i + 1], 'crime_number_lag_2'] = tes
        test_data.loc[test_data.index[i + 1], 'crime_number_lag_3'] = tes

test_data['predicted_crime_number_xgb'] = test_predictions

merged_df_xgb = pd.merge(merged_df_xgb, test_data[['date', 'predicted_cri

plt.figure(figsize=(14, 7))
plt.plot(merged_df_xgb['date'], merged_df_xgb['crime_number'], label="Actu
plt.plot(merged_df_xgb['date'], merged_df_xgb['predicted_crime_number_xgb
plt.axvline(x=pd.to_datetime('2020-01-01'), color='green', linestyle='--'
plt.xlabel("Date")
plt.ylabel("Crime Number")
plt.title("Actual vs Predicted Crime Numbers (XGBoost)")
plt.legend()
plt.show()
```



```
Validation MAPE: 8.94%
```

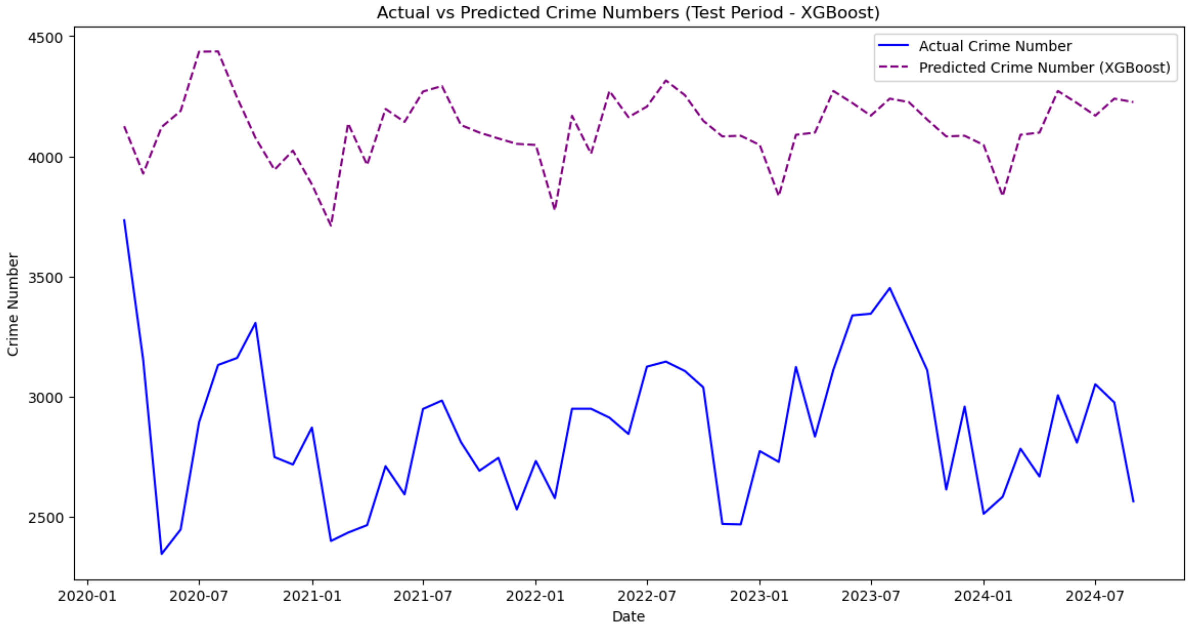


In [657…

```
from sklearn.metrics import mean_absolute_percentage_error


# Filter test data after 2020-03-01 (real test data period)
test_data_actual = merged_df_xgb.loc[merged_df_xgb['date'] >= '2020-03-01
test_data_predicted = merged_df_xgb.loc[merged_df_xgb['date'] >= '2020-03-

# Drop NaN values
valid_indices = test_data_actual.notna() & test_data_predicted.notna()
test_data_actual = test_data_actual[valid_indices]
test_data_predicted = test_data_predicted[valid_indices]

# Calculate MAPE
```

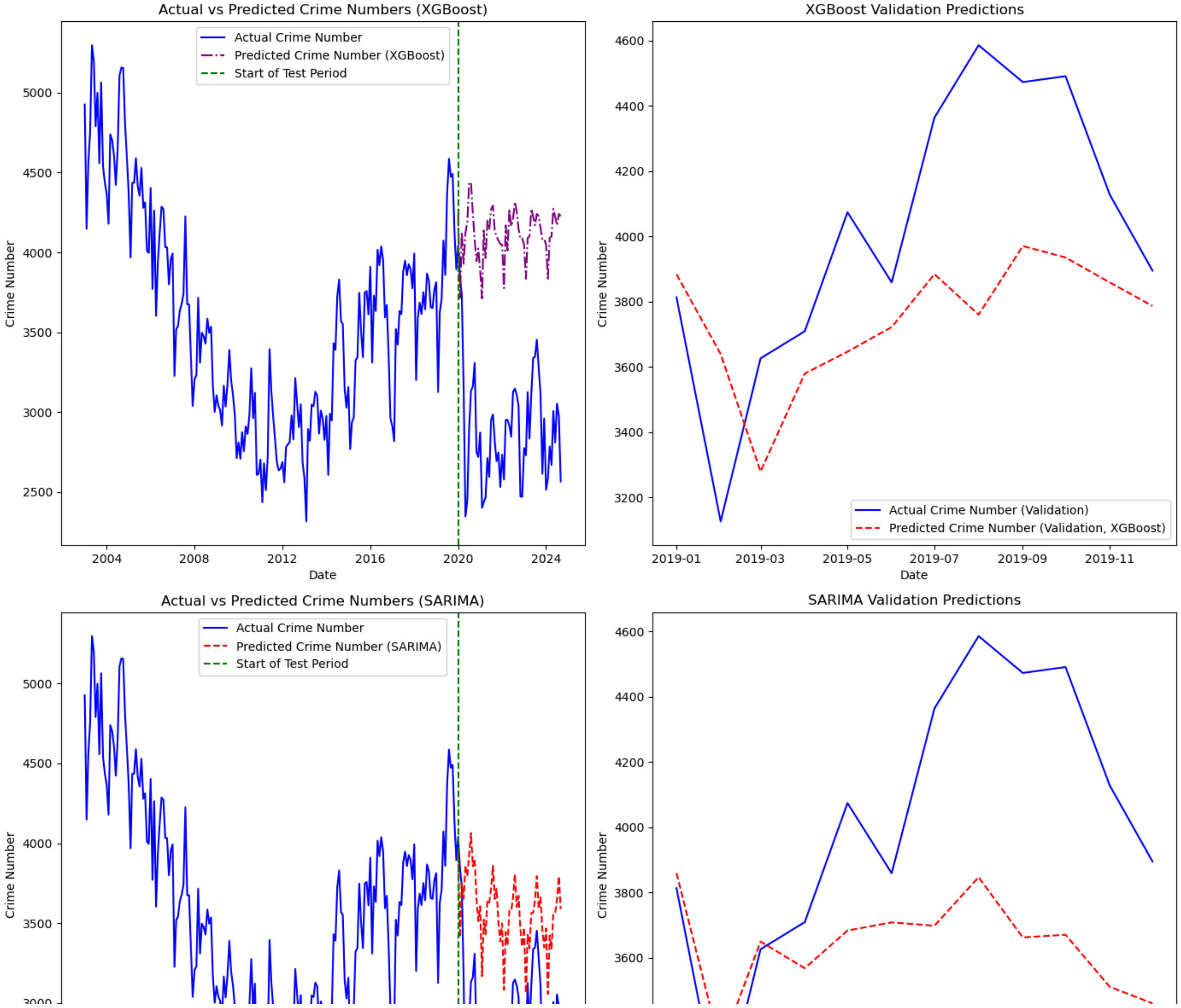
```
mape_test = mean_absolute_percentage_error(test_data_actual, test_data_pre
print(f"Test MAPE: {mape_test * 100:.2f}%")

# Calculate percentage difference
percent_diff = ((test_data_actual - test_data_predicted) / test_data_actua
print(f"Average percent difference after 2020-03-01: {percent_diff:.2f}%"

# Optional: Visualize actual vs predicted values for the test period
plt.figure(figsize=(14, 7))
plt.plot(merged_df_xgb.loc[merged_df_xgb['date'] >= '2020-03-01', 'date']
plt.plot(merged_df_xgb.loc[merged_df_xgb['date'] >= '2020-03-01', 'date']
plt.xlabel("Date")
plt.ylabel("Crime Number")
plt.title("Actual vs Predicted Crime Numbers (Test Period - XGBoost)")
plt.legend()
plt.show()
```

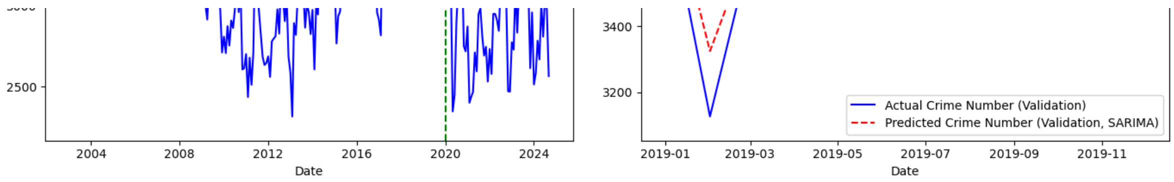
```
Test MAPE: 44.95%
Average percent difference after 2020-03-01: -44.95%
```



In [658…

```
 # put Actual vs Predicted Crime Numbers (XGBoost), XGBoost Validation Pre
# 1 figure with 4 subplots
plt.figure(figsize=(14, 14))

# Actual vs Predicted Crime Numbers (XGBoost)
plt.subplot(2, 2, 1)

plt.plot(merged_df_xgb['date'], merged_df_xgb['crime_number'], label="Actu
plt.plot(merged_df_xgb['date'], merged_df_xgb['predicted_crime_number_xgb
plt.axvline(x=pd.to_datetime('2020-01-01'), color='green', linestyle='--'
plt.xlabel("Date")
plt.ylabel("Crime Number")
plt.title("Actual vs Predicted Crime Numbers (XGBoost)")
plt.legend()

# XGBoost Validation Predictions
plt.subplot(2, 2, 2)

plt.plot(val_data['date'], y_val, label="Actual Crime Number (Validation)"
plt.plot(val_data['date'], val_predictions, label="Predicted Crime Number
plt.xlabel("Date")
plt.ylabel("Crime Number")
plt.title("XGBoost Validation Predictions")
plt.legend()

# Actual vs Predicted Crime Numbers (SARIMA)
plt.subplot(2, 2, 3)

plt.plot(merged_df_arima['date'], merged_df_arima['crime_number'], label='
plt.plot(merged_df_arima['date'], merged_df_arima['predicted_crime_number_
plt.axvline(x=pd.to_datetime('2020-01-01'), color='green', linestyle='--',
plt.xlabel("Date")
plt.ylabel("Crime Number")
plt.title("Actual vs Predicted Crime Numbers (SARIMA)")
plt.legend()

# SARIMA Validation Predictions
plt.subplot(2, 2, 4)

plt.plot(val_data['date'], y_val, label="Actual Crime Number (Validation)"
plt.plot(val_data['date'], forecast_val, label="Predicted Crime Number (Va
plt.xlabel("Date")
plt.ylabel("Crime Number")
plt.title("SARIMA Validation Predictions")
plt.legend()

plt.tight_layout()
plt.show()
```




2500
2004
2008
2012
2016
2020
2024
Date
3400
3200
Actual Crime Number (Validation)
Predicted Crime Number (Validation, SARIMA)
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
Date