Course Code : CBS3007 Course: Data Mining and Analytics

Alan Thomas
21BBS0115

Aim: To analyze and forecast crime incidents using Autoregressive (AR) and Autoregressive Integrated Moving Average (ARIMA) time series techniques. The goal is to prepare a detailed real-time dataset for a specific location, evaluate temporal crime patterns, and predict future incidents based on historical data.

Packages Used: pandas, numpy, matplotlib.pyplot, statsmodels

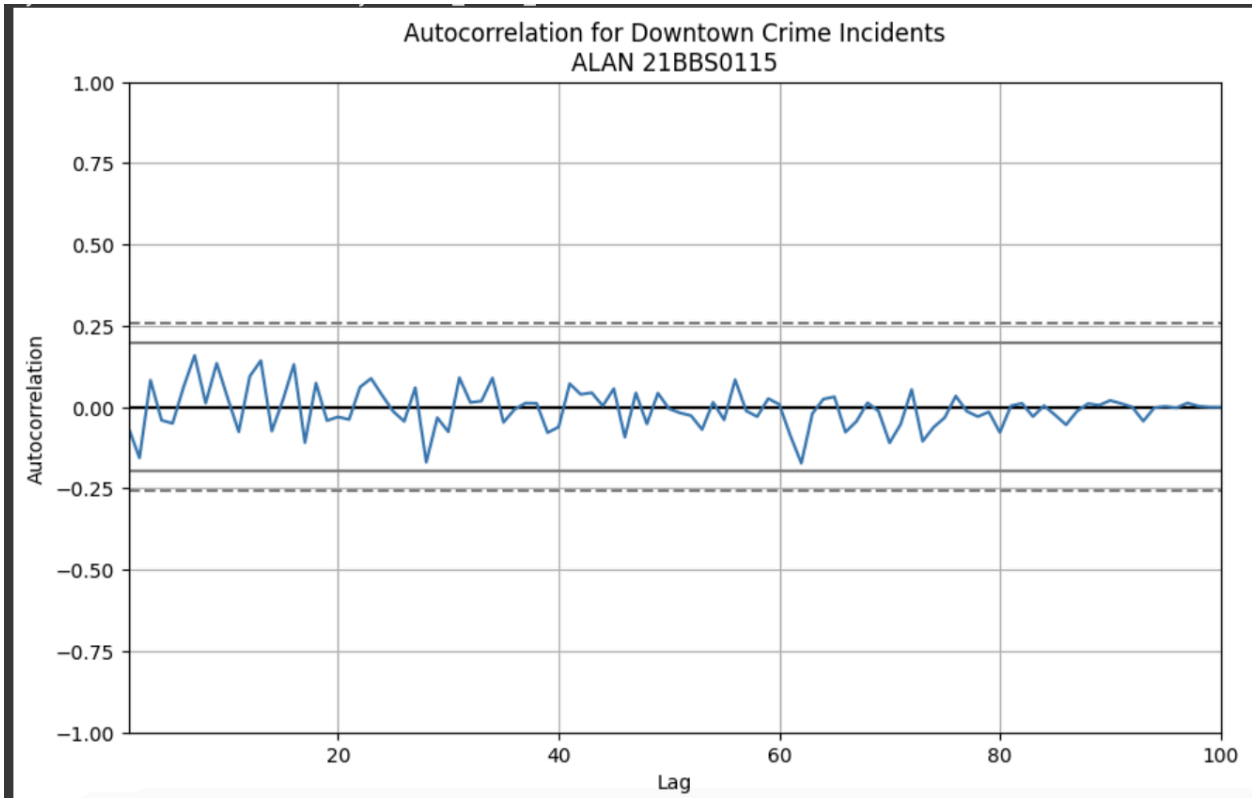
CODE 👍

```
import pandas as pd
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
from pandas.plotting import autocorrelation_plot
```

```
np.random.seed(42)
date_range = pd.date_range(start="2023-01-01", periods=100)
location_list = [
   "Downtown", "Suburb", "Industrial Area", "Residential Zone", "City
Center",
   "Eastside", "Westside", "Uptown", "Rural Outskirts", "Metro Hub"
][:5]
```

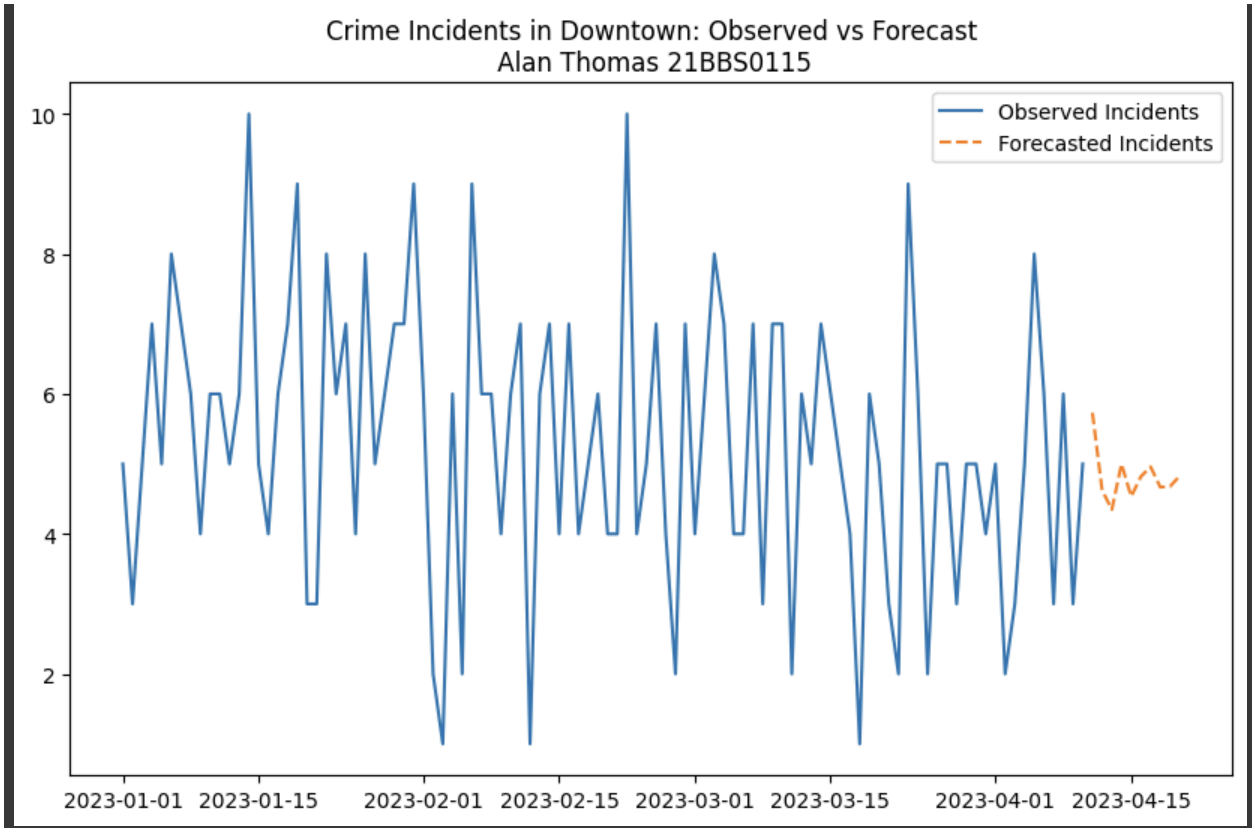
```
crime_categories = ["Assault", "Burglary", "Theft", "Vandalism", "Drug
Offense"]

synthetic_records = []
for day in date_range:
   for area in location_list:
       incidents = np.random.poisson(lam=5)
       for _ in range(incidents):
           synthetic_records.append({
               "Date": day,
               "Location": area,
               "Crime_Type": np.random.choice(crime_categories),
               "Severity": np.random.choice(["Low", "Medium", "High"],
p=[0.5, 0.3, 0.2])
           })

crime_df = pd.DataFrame(synthetic_records)

aggregated_data = crime_df.groupby(['Date',
'Location']).size().reset_index(name='Incident_Count')

crime_df.to_csv("synthetic_crime_dataset.csv", index=False)
print("Synthetic dataset saved as 'synthetic_crime_dataset.csv'.")

downtown_crime = aggregated_data[aggregated_data['Location'] ==
'Downtown']
downtown_crime.set_index('Date', inplace=True)

downtown_series =
downtown_crime['Incident_Count'].reindex(pd.date_range(start=downtown_crim
e.index.min(),

end=downtown_crime.index.max()),
                                                    fill_value=0)

plt.figure(figsize=(10, 6))
autocorrelation_plot(downtown_series)
plt.title("Autocorrelation for Downtown Crime Incidents\nALAN 21BBS0115")
plt.show()
```

```
model = ARIMA(downtown_series, order=(5, 1, 0))
model_fit = model.fit()

print("ARIMA Model Summary:\n")
print(model_fit.summary())

future_forecast = model_fit.forecast(steps=10)
print("\nForecast for the next 10 days:\n", future_forecast)

plt.figure(figsize=(10, 6))
plt.plot(downtown_series, label="Observed Incidents")
plt.plot(future_forecast, label="Forecasted Incidents", linestyle="--")
plt.title("Crime Incidents in Downtown: Observed vs Forecast\n Alan Thomas
21BBS0115")
plt.legend()
plt.show()

downtown_series.to_csv("downtown_crime_series.csv", index=True)
```



ARIMA model Implementation

```
ARIMA Model Summary:

                               SARIMAX Results
==============================================================================
Dep. Variable:         Incident_Count   No. Observations:                  100
Model:                 ARIMA(5, 1, 0)   Log Likelihood                -216.117
Date:                Sun, 17 Nov 2024   AIC                            444.233
Time:                        17:52:12   BIC                            459.804
Sample:                  01-01-2023   HQIC                           450.533
                       - 04-10-2023
Covariance Type:                 opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.8643      0.101     -8.568      0.000      -1.062      -0.667
ar.L2         -0.8301      0.131     -6.325      0.000      -1.087      -0.573
ar.L3         -0.5598      0.152     -3.692      0.000      -0.857      -0.263
ar.L4         -0.3952      0.115     -3.447      0.001      -0.620      -0.170
ar.L5         -0.2311      0.095     -2.423      0.015      -0.418      -0.044
sigma2         4.5484      0.634      7.178      0.000       3.306       5.790
===================================================================================
Ljung-Box (L1) (Q):                   0.52   Jarque-Bera (JB):                 0.30
Prob(Q):                              0.47   Prob(JB):                         0.86
Heteroskedasticity (H):               0.76   Skew:                            -0.09
Prob(H) (two-sided):                  0.44   Kurtosis:                         3.20
===================================================================================
```



Q2

Aim: To visualize and analyze various types of moving averages (including Simple, Exponential, Weighted, and Hull Moving Averages), along with Autoregressive Moving Average (ARMA) and Moving Average Crossover techniques, using a crime dataset to identify temporal trends and patterns.

Packages Used: pandas, numpy, matplotlib.pyplot, statsmodels

CODE 👍
#used the same created DataSet

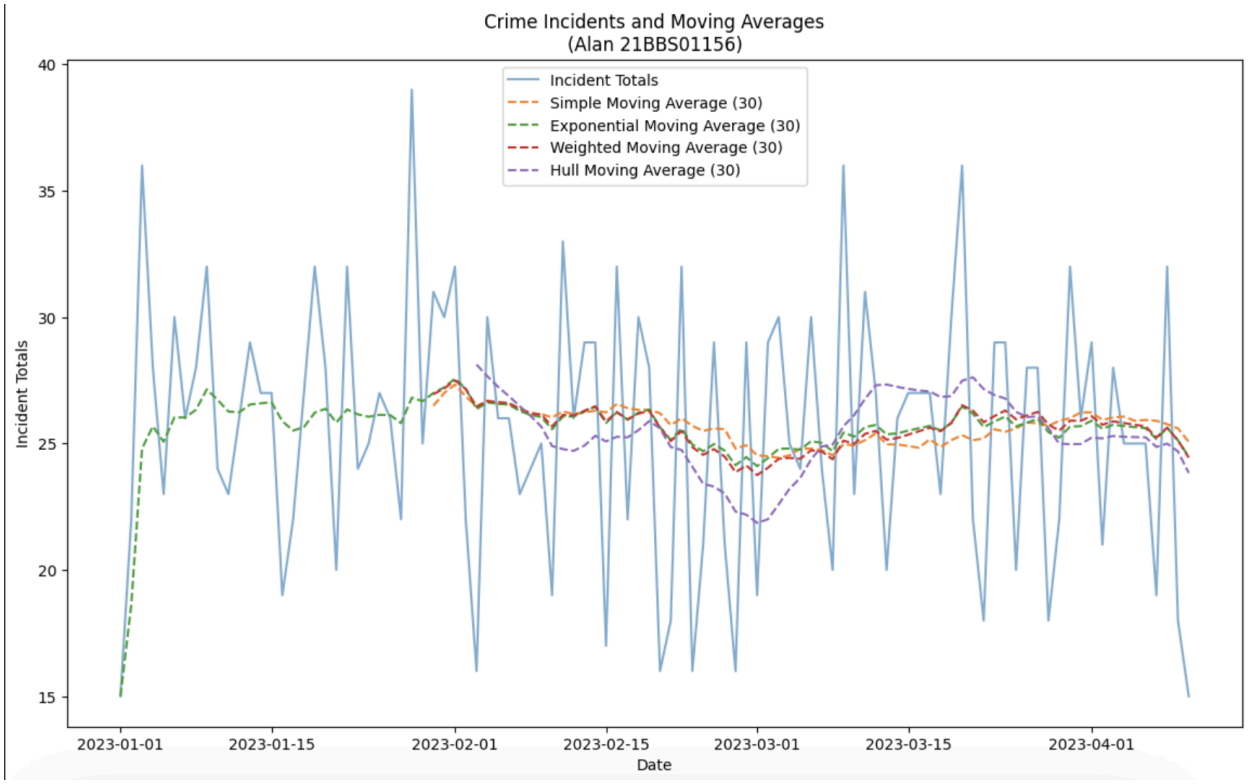
```
data_file = "synthetic_crime_dataset.csv"
crime_df = pd.read_csv(data_file)
```

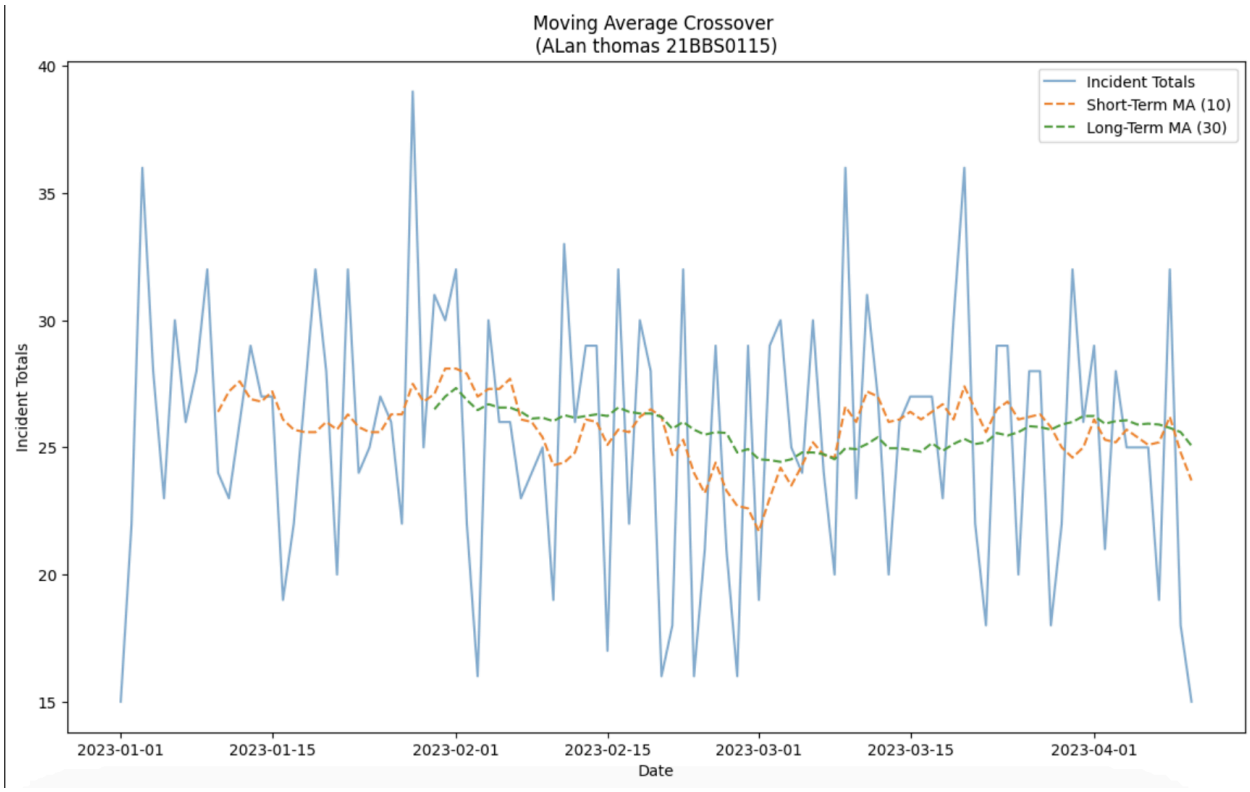
```
crime_df['Date'] = pd.to_datetime(crime_df['Date'])
```

```
daily_crime =
crime_df.groupby('Date').size().reset_index(name='Incident_Totals')
daily_crime.set_index('Date', inplace=True)

def compute_weighted_ma(series, window_size):
   weights = np.arange(1, window_size + 1)
   return series.rolling(window_size).apply(lambda values: np.dot(values,
weights) / weights.sum(), raw=True)

def compute_hull_ma(series, window_size):
   half_window = window_size // 2
   sqrt_window = int(np.sqrt(window_size))
   wma_half = compute_weighted_ma(series, half_window)
   wma_full = compute_weighted_ma(series, window_size)
   return compute_weighted_ma(2 * wma_half - wma_full, sqrt_window)

daily_crime['SMA_30'] =
daily_crime['Incident_Totals'].rolling(window=30).mean()
daily_crime['EMA_30'] = daily_crime['Incident_Totals'].ewm(span=30).mean()
daily_crime['WMA_30'] =
compute_weighted_ma(daily_crime['Incident_Totals'], 30)
daily_crime['HMA_30'] = compute_hull_ma(daily_crime['Incident_Totals'],
30)

daily_crime['Short_MA'] =
daily_crime['Incident_Totals'].rolling(window=10).mean()
daily_crime['Long_MA'] =
daily_crime['Incident_Totals'].rolling(window=30).mean()

plt.figure(figsize=(14, 8))
plt.plot(daily_crime['Incident_Totals'], label="Incident Totals",
alpha=0.6)
plt.plot(daily_crime['SMA_30'], label="Simple Moving Average (30)",
linestyle='--')
plt.plot(daily_crime['EMA_30'], label="Exponential Moving Average (30)",
linestyle='--')
plt.plot(daily_crime['WMA_30'], label="Weighted Moving Average (30)",
linestyle='--')
plt.plot(daily_crime['HMA_30'], label="Hull Moving Average (30)",
linestyle='--')
```

```
plt.title("Crime Incidents and Moving Averages\n(Alan 21BBS01156)")
plt.xlabel("Date")
plt.ylabel("Incident Totals")
plt.legend()
plt.show()

plt.figure(figsize=(14, 8))
plt.plot(daily_crime['Incident_Totals'], label="Incident Totals",
alpha=0.6)
plt.plot(daily_crime['Short_MA'], label="Short-Term MA (10)",
linestyle='--')
plt.plot(daily_crime['Long_MA'], label="Long-Term MA (30)",
linestyle='--')
plt.title("Moving Average Crossover \n(ALan thomas 21BBS0115)")
plt.xlabel("Date")
plt.ylabel("Incident Totals")
plt.legend()
plt.show()

output_file = "crime_analysis_with_moving_averages.csv"
daily_crime.to_csv(output_file, index=True)
```

Crossovers


Crime Incidents and Moving Averages
(Alan 21BBS01156)
Incident Totals
Simple Moving Average (30)
Exponential Moving Average (30)
Weighted Moving Average (30)
Hull Moving Average (30)
Incident Totals
Date
40
35
30
25
20
15
2023-01-01
2023-01-15
2023-02-01
2023-02-15
2023-03-01
2023-03-15
2023-04-01


Moving Average Crossover
(ALan thomas 21BBS0115)
Incident Totals
Short-Term MA (10)
Long-Term MA (30)
Incident Totals
Date
40
35
30
25
20
15
2023-01-01
2023-01-15
2023-02-01
2023-02-15
2023-03-01
2023-03-15
2023-04-01

Github repo :
https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/THEORY%20DA