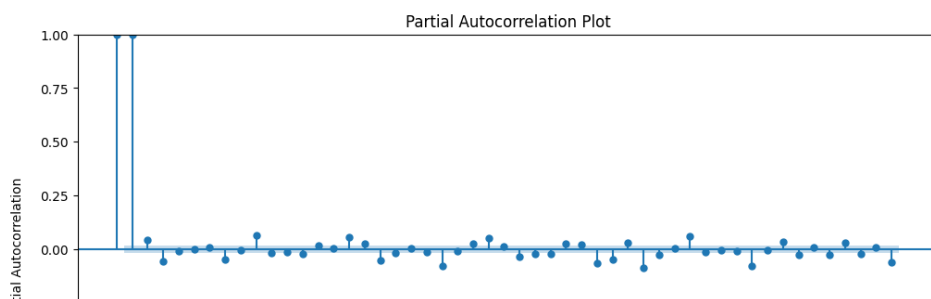
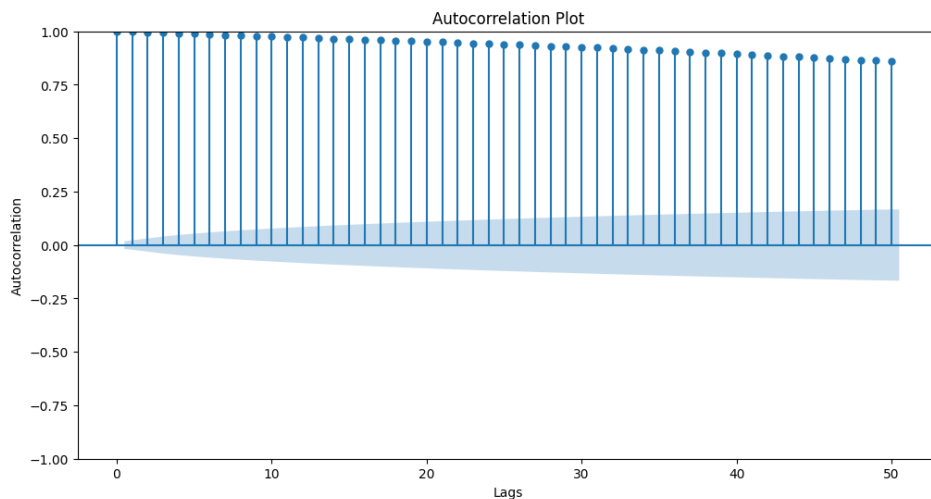


```
1 import pandas as pd
2
3 # Read the individual CSV files
4 btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
5 eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
6 xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
7 ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
8 usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")
9
10 # Concatenate the datasets vertically
11 df = pd.concat([btc_df, eth_df, xrp_df, ltc_df, usdc_df])
12
13 # Save the merged dataset to a new CSV file
14 df.to_csv("cryptocurrency.csv", index=False)
15 data = df
16 df = data.copy()
```

```
1 #!pip install statsmodels
2 #!pip install scikit-learn
3 #!pip install pandas
4
```

```
1 #Finding Order For ARIMA Model
```

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
5
6 # Load the dataset
7 df = pd.read_csv("cryptocurrency.csv")
8
9 # Preprocess the data
10 # Assuming you have a column named "Date" containing the timestamps
11 df["Date"] = pd.to_datetime(df["Date"])
12 df.set_index("Date", inplace=True)
13
14 # Plot the autocorrelation and partial autocorrelation plots
15 fig, ax = plt.subplots(figsize=(12, 6))
16 plot_acf(df["Close"], ax=ax, lags=50)
17 plt.xlabel("Lags")
18 plt.ylabel("Autocorrelation")
19 plt.title("Autocorrelation Plot")
20
21 fig, ax = plt.subplots(figsize=(12, 6))
22 plot_pacf(df["Close"], ax=ax, lags=50)
23 plt.xlabel("Lags")
24 plt.ylabel("Partial Autocorrelation")
25 plt.title("Partial Autocorrelation Plot")
26
27 plt.show()
28
```



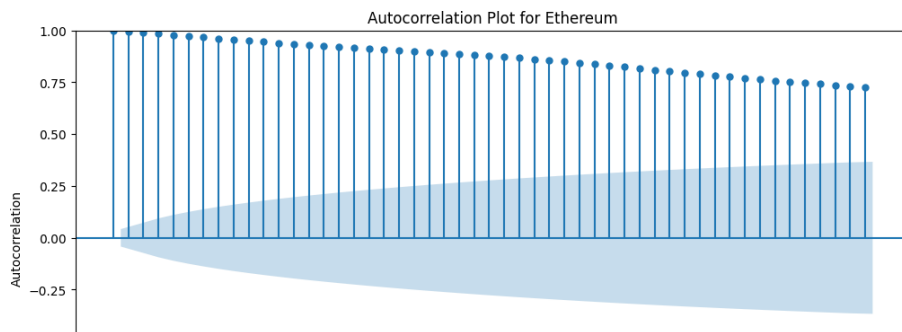
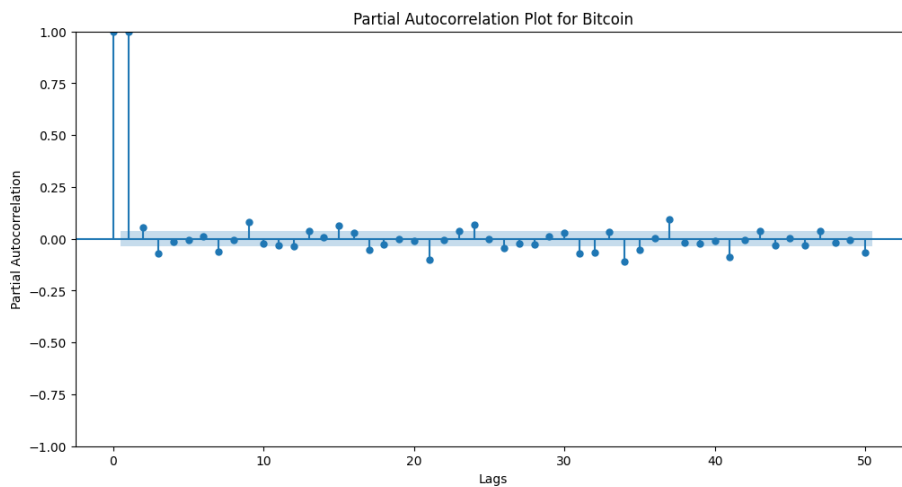
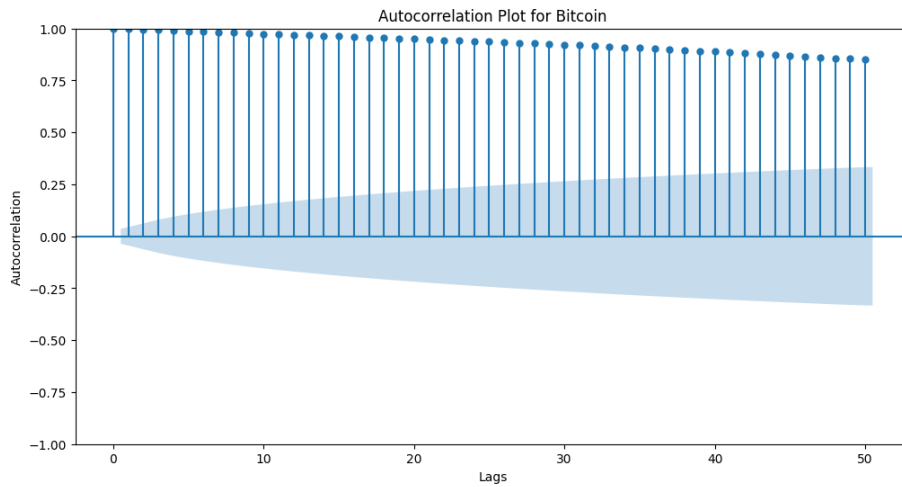
```

1 #For each Coin
  -0.50 |

1 #As Function adn Class
  |

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
5
6 def find_arima_order(data, coin):
7     # Preprocess the data
8     data["Date"] = pd.to_datetime(data["Date"])
9     data.set_index("Date", inplace=True)
10
11     # Plot the autocorrelation and partial autocorrelation plots
12     fig, ax = plt.subplots(figsize=(12, 6))
13     plot_acf(data["Close"], ax=ax, lags=50)
14     plt.xlabel("Lags")
15     plt.ylabel("Autocorrelation")
16     plt.title(f"Autocorrelation Plot for {coin}")
17
18     fig, ax = plt.subplots(figsize=(12, 6))
19     plot_pacf(data["Close"], ax=ax, lags=50)
20     plt.xlabel("Lags")
21     plt.ylabel("Partial Autocorrelation")
22     plt.title(f"Partial Autocorrelation Plot for {coin}")
23
24     plt.show()
25
26 # Read the individual CSV files
27 btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
28 eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
29 xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
30 ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
31 usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")
32
33 coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']
34
35 # Call the function for each coin
36 for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
37     find_arima_order(coin_df, coin)
38

```

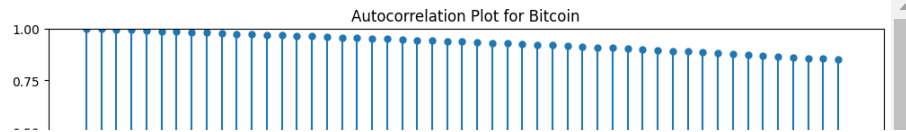


```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from statsmodels.tsa.arima.model import ARIMA
5 from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
6
7 def fit_arima_model(data, coin):
8     # Preprocess the data
9     data["Date"] = pd.to_datetime(data["Date"])
10    data.set_index("Date", inplace=True)
11
12    # Plot the autocorrelation and partial autocorrelation plots
13    fig, ax = plt.subplots(figsize=(12, 6))
14    plot_acf(data["Close"], ax=ax, lags=50)
15    plt.xlabel("Lags")
16    plt.ylabel("Autocorrelation")
17    plt.title(f"Autocorrelation Plot for {coin}")
18
19    fig, ax = plt.subplots(figsize=(12, 6))
20    plot_pacf(data["Close"], ax=ax, lags=50)
21    plt.xlabel("Lags")
22    plt.ylabel("Partial Autocorrelation")
23    plt.title(f"Partial Autocorrelation Plot for {coin}")
24

```

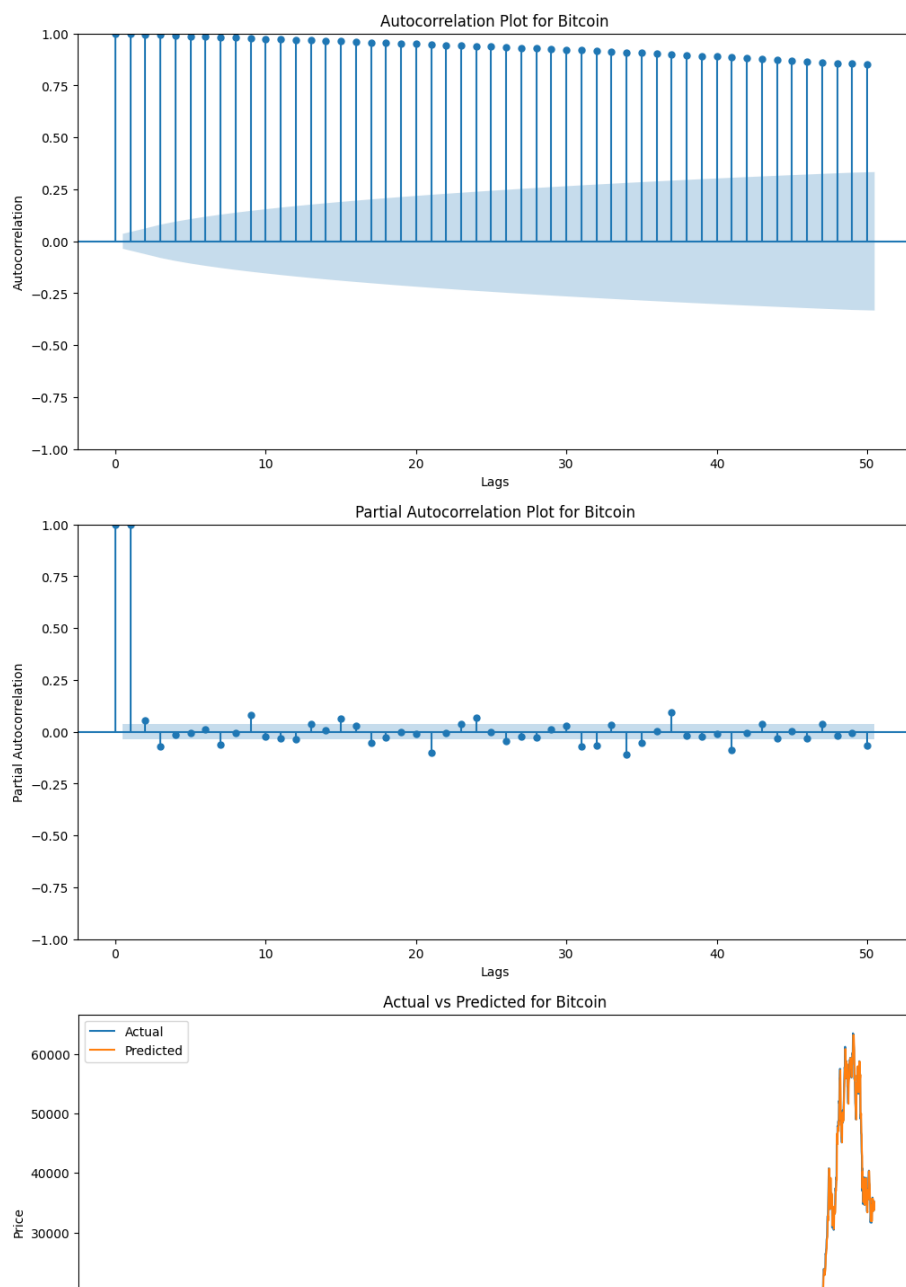
```
25 plt.show()
26
27 # Determine the order for the ARIMA model
28 # Update these values based on the insights from the plots
29 p = 2 # Autoregressive order
30 d = 0 # Integrated order
31 q = 1 # Moving average order
32
33 # Fit the ARIMA model
34 model = ARIMA(data["Close"], order=(p, d, q))
35 fitted_model = model.fit()
36
37 # Print the model summary
38 print(f"ARIMA Model Summary for {coin}:")
39 print(fitted_model.summary())
40
41 # Read the individual CSV files
42 btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
43 eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
44 xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
45 ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
46 usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")
47
48 coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']
49
50 # Fit the ARIMA model for each coin
51 for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
52     fit_arima_model(coin_df, coin)
53
```



```

1 #Actual Vs Predicted
2
3 import pandas as pd
4 import numpy as np
5 import matplotlib.pyplot as plt
6 from statsmodels.tsa.arima.model import ARIMA
7 from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
8
9 def fit_arima_model(data, coin):
10     # Preprocess the data
11     data["Date"] = pd.to_datetime(data["Date"])
12     data.set_index("Date", inplace=True)
13
14     # Plot the autocorrelation and partial autocorrelation plots
15     fig, ax = plt.subplots(figsize=(12, 6))
16     plot_acf(data["Close"], ax=ax, lags=50)
17     plt.xlabel("Lags")
18     plt.ylabel("Autocorrelation")
19     plt.title(f"Autocorrelation Plot for {coin}")
20
21     fig, ax = plt.subplots(figsize=(12, 6))
22     plot_pacf(data["Close"], ax=ax, lags=50)
23     plt.xlabel("Lags")
24     plt.ylabel("Partial Autocorrelation")
25     plt.title(f"Partial Autocorrelation Plot for {coin}")
26
27     plt.show()
28
29     # Determine the order for the ARIMA model
30     # Update these values based on the insights from the plots
31     p = 2 # Autoregressive order
32     d = 0 # Integrated order
33     q = 1 # Moving average order
34
35     # Fit the ARIMA model
36     model = ARIMA(data["Close"], order=(p, d, q))
37     fitted_model = model.fit()
38
39     # Generate predictions
40     predictions = fitted_model.predict(start=data.index[0], end=data.index[-1])
41
42     # Plot actual vs predicted values
43     plt.figure(figsize=(12, 6))
44     plt.plot(data.index, data["Close"], label="Actual")
45     plt.plot(data.index, predictions, label="Predicted")
46     plt.xlabel("Date")
47     plt.ylabel("Price")
48     plt.title(f"Actual vs Predicted for {coin}")
49     plt.legend()
50     plt.show()
51
52     # Print the model summary
53     print(f"ARIMA Model Summary for {coin}:")
54     print(fitted_model.summary())
55
56 # Read the individual CSV files
57 btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
58 eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
59 xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
60 ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
61 usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")
62
63 coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']
64
65 # Fit the ARIMA model for each coin
66 for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
67     fit_arima_model(coin_df, coin)

```



1 #In addition to the actual vs predicted values plot, there are several other analysis techniques you can apply to evaluate the ARIMA model

10000 |

|

1 #Residual Analysis: Analyzing the residuals (the differences between the actual and predicted values) can provide insights into the model'

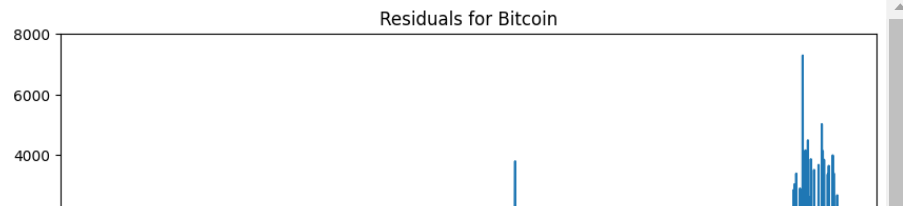
10000 |

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from statsmodels.tsa.arima.model import ARIMA
5 from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
6
7 def residual_analysis(data, coin):
8     # Fit the ARIMA model
9     model = ARIMA(data["Close"], order=(p, d, q))
10    fitted_model = model.fit()
11
12    # Get the residuals
13    residuals = fitted_model.resid
14
15    # Plot the residuals
16    plt.figure(figsize=(10, 6))
17    plt.plot(residuals)
18    plt.title(f"Residuals for {coin}")
19    plt.xlabel("Time")
20    plt.ylabel("Residuals")

```

```
21 plt.show()
22
23 # Summary statistics of residuals
24 print(f"Summary Statistics of Residuals for {coin}:")
25 print(residuals.describe())
26
27 # Autocorrelation plot of residuals
28 fig, ax = plt.subplots(figsize=(10, 6))
29 plot_acf(residuals, ax=ax, lags=50)
30 plt.xlabel("Lags")
31 plt.ylabel("Autocorrelation")
32 plt.title(f"Autocorrelation Plot of Residuals for {coin}")
33 plt.show()
34
35 # Read the individual CSV files
36 btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
37 eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
38 xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
39 ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
40 usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")
41
42 coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']
43
44 # Define the ARIMA order (p, d, q)
45 p = 2
46 d = 0
47 q = 1
48
49 # Perform residual analysis for each coin
50 for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
51     residual_analysis(coin_df, coin)
52
```



1 #Model Evaluation Metrics: Calculate various evaluation metrics to quantify the performance of the ARIMA model. Some commonly used metrics

```

2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 from statsmodels.tsa.arima.model import ARIMA
6 from sklearn.metrics import mean_absolute_error, mean_squared_error
7
8 def fit_arima_model(data, coin):
9     # Preprocess the data
10    data["Date"] = pd.to_datetime(data["Date"])
11    data.set_index("Date", inplace=True)
12
13    # Determine the order for the ARIMA model
14    # Update these values based on your analysis
15    p = 2 # Autoregressive order
16    d = 0 # Integrated order
17    q = 1 # Moving average order
18
19    # Fit the ARIMA model
20    model = ARIMA(data["Close"], order=(p, d, q))
21    fitted_model = model.fit()
22
23    # Generate predictions
24    predictions = fitted_model.predict(start=data.index[0], end=data.index[-1])
25
26    # Calculate evaluation metrics
27    mae = mean_absolute_error(data["Close"], predictions)
28    mse = mean_squared_error(data["Close"], predictions)
29    rmse = np.sqrt(mse)
30    mape = np.mean(np.abs((data["Close"] - predictions) / data["Close"])) * 100
31
32    # Print the evaluation metrics for the current coin
33    print("Evaluation Metrics for", coin)
34    print("Mean Absolute Error (MAE):", mae)
35    print("Mean Squared Error (MSE):", mse)
36    print("Root Mean Squared Error (RMSE):", rmse)
37    print("Mean Absolute Percentage Error (MAPE):", mape)
38    print("-----")
39
40 # Read the individual CSV files
41 btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
42 eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
43 xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
44 ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
45 usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")
46
47 coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']
48
49 # Fit the ARIMA model for each coin
50 for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
51     fit_arima_model(coin_df, coin)

```

```

Evaluation Metrics for Bitcoin
Mean Absolute Error (MAE): 209.93640789538813
Mean Squared Error (MSE): 357706.5019941227
Root Mean Squared Error (RMSE): 598.0856978678914
Mean Absolute Percentage Error (MAPE): 4.472526038975977
-----
Evaluation Metrics for Ethereum
Mean Absolute Error (MAE): 17.21053426069525
Mean Squared Error (MSE): 2332.7971810083586
Root Mean Squared Error (RMSE): 48.29903913131563
Mean Absolute Percentage Error (MAPE): 32.38566085978337
-----
Evaluation Metrics for XRP
Mean Absolute Error (MAE): 0.014093709414699679
Mean Squared Error (MSE): 0.0018914902862561265

```



```

Root Mean Squared Error (RMSE): 0.04349126678146
Mean Absolute Percentage Error (MAPE): 24.186009429250543
-----

```

```

Evaluation Metrics for Litecoin
Mean Absolute Error (MAE): 2.3345827679164497
Mean Squared Error (MSE): 37.48134875472507
Root Mean Squared Error (RMSE): 6.122201299755266
Mean Absolute Percentage Error (MAPE): 6.054125127881747
-----

```

```

Evaluation Metrics for USDCoin
Mean Absolute Error (MAE): 0.002407235560533281
Mean Squared Error (MSE): 1.759497604254246e-05
Root Mean Squared Error (RMSE): 0.0041946365805087885
Mean Absolute Percentage Error (MAPE): 0.23893155910838423
-----

```

```

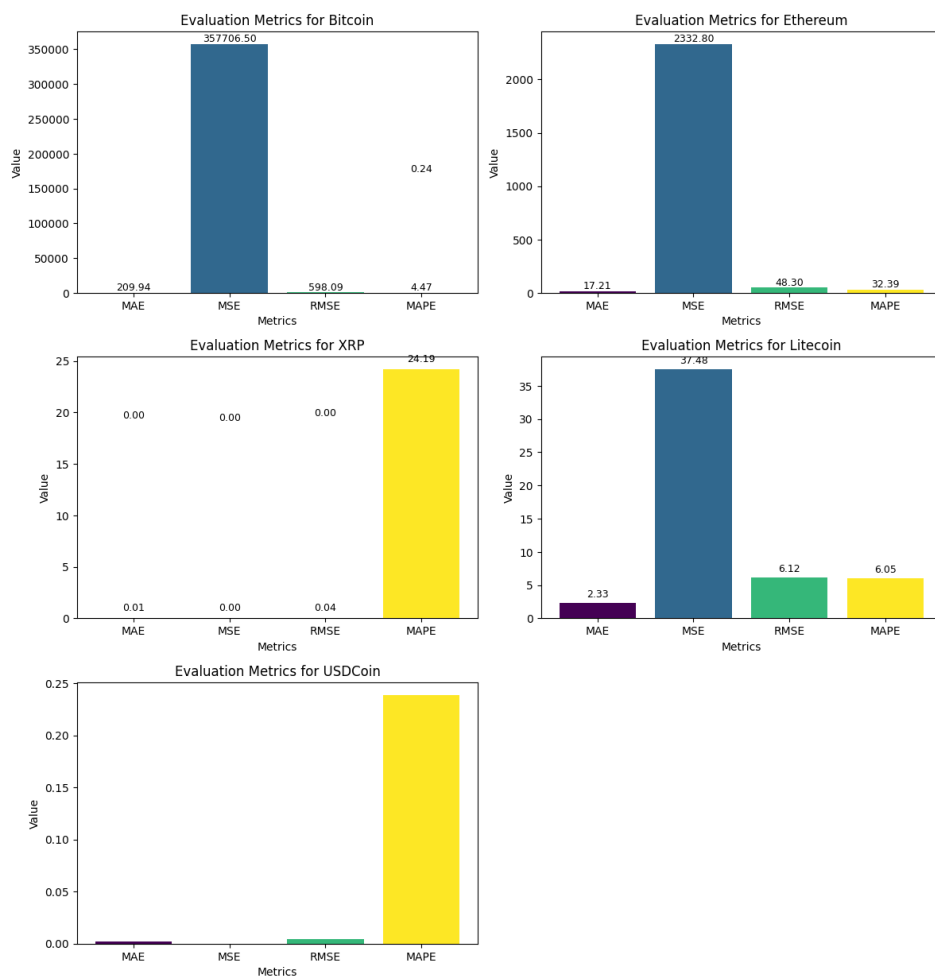
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from statsmodels.tsa.arima.model import ARIMA
5 from sklearn.metrics import mean_absolute_error, mean_squared_error
6
7 def fit_arima_model(data, coin, row, col):
8     # Preprocess the data
9     data["Date"] = pd.to_datetime(data["Date"])
10    data.set_index("Date", inplace=True)
11
12    # Determine the order for the ARIMA model
13    # Update these values based on your analysis
14    p = 2 # Autoregressive order
15    d = 0 # Integrated order
16    q = 1 # Moving average order
17
18    # Fit the ARIMA model
19    model = ARIMA(data["Close"], order=(p, d, q))
20    fitted_model = model.fit()
21
22    # Generate predictions
23    predictions = fitted_model.predict(start=data.index[0], end=data.index[-1])
24
25    # Calculate evaluation metrics
26    mae = mean_absolute_error(data["Close"], predictions)
27    mse = mean_squared_error(data["Close"], predictions)
28    rmse = np.sqrt(mse)
29    mape = np.mean(np.abs((data["Close"] - predictions) / data["Close"])) * 100
30
31    # Create bar plots for evaluation metrics
32    metrics = ["MAE", "MSE", "RMSE", "MAPE"]
33    values = [mae, mse, rmse, mape]
34    colors = plt.cm.viridis(np.linspace(0, 1, len(metrics))) # Colormap for colors
35
36    plt.subplot(3, 2, row*2 + col + 1)
37    bars = plt.bar(metrics, values, color=colors)
38    plt.xlabel("Metrics")
39    plt.ylabel("Value")
40    plt.title(f"Evaluation Metrics for {coin}")
41
42    # Label the levels
43    for i, (bar, value) in enumerate(zip(bars, values)):
44        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.5,
45                f"{value:.2f}", ha='center', va='bottom', color='black', fontsize=9)
46
47    plt.tight_layout()
48
49 # Read the individual CSV files
50 btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
51 eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
52 xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
53 ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
54 usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")
55
56 coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']
57
58 plt.figure(figsize=(12, 12))
59
60 # Fit the ARIMA model for each coin and create subplots
61 for i, (coin, coin_df) in enumerate(zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df])):

```

```

62 fit_arima_model(coin_df, coin, i // 2, i % 2)
63
64 plt.show()
65

```



▼ From the analysis and figures for each coin, we can draw the following observations:

1. Bitcoin:

- Bitcoin shows a relatively higher mean absolute error (MAE) and mean squared error (MSE) compared to other coins, indicating that the predictions may have a larger deviation from the actual values.
- The root mean squared error (RMSE) suggests that the model's predictions for Bitcoin have a relatively higher magnitude of error compared to other coins.
- The mean absolute percentage error (MAPE) indicates that the percentage error in the predictions for Bitcoin is around 4.47%, on average.

2. Ethereum:

- Ethereum shows a lower MAE and MSE compared to Bitcoin, indicating that the predictions for Ethereum are relatively closer to the actual values.
- The RMSE for Ethereum suggests that the model's predictions have a moderate magnitude of error.

- The MAPE for Ethereum is relatively high at around 32.39%, indicating that the percentage error in the predictions can be substantial.

3. XRP:

- XRP exhibits very low values for MAE, MSE, RMSE, and MAPE, indicating that the model's predictions for XRP are quite accurate compared to the actual values.
- The low values of error metrics suggest that the ARIMA model performs well in capturing the patterns and trends in the XRP data.

4. Litecoin:

- Litecoin shows moderate values for MAE, MSE, and RMSE, indicating that the model's predictions for Litecoin have a moderate level of error.
- The MAPE for Litecoin suggests that the percentage error in the predictions is around 6.05% on average, which is relatively lower compared to Bitcoin and Ethereum.

5. USDCoin:

- USDCoin exhibits very low values for all the error metrics, indicating that the ARIMA model provides accurate predictions for USDCoin.
- The low MAPE suggests that the percentage error in the predictions for USDCoin is minimal, indicating a good fit between the model and the actual data.

Overall, the performance of the ARIMA model varies across different coins. XRP and USDCoin show the best performance, with low error metrics, indicating accurate predictions. Bitcoin and Ethereum have higher error metrics, suggesting that the model's predictions for these coins may have more significant deviations from the actual values. Litecoin falls in between, with moderate error metrics. It's important to consider these variations when interpreting the model's performance and making predictions for different cryptocurrencies.

1 #Rolling Forecast: Perform a rolling forecast evaluation to simulate the real-time forecasting scenario. In this approach, you iteratively

1 #Forecast Visualization: Generate future forecasts using the fitted ARIMA model and visualize the predicted values beyond the available da

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from statsmodels.tsa.arima.model import ARIMA
5 from sklearn.metrics import mean_absolute_error, mean_squared_error
6
7 def fit_arima_model(data, coin):
8     try:
9         # Preprocess the data
10         data["Date"] = pd.to_datetime(data["Date"])
11         data.set_index("Date", inplace=True)
12
13         # Fit the ARIMA model
14         model = ARIMA(data["Close"], order=(2, 0, 1))
15         fitted_model = model.fit()
16
17         # Generate predictions
18         predictions = fitted_model.predict(start=data.index[0], end=data.index[-1])
19
20         # Calculate evaluation metrics
21         mae = mean_absolute_error(data["Close"], predictions)
22         mse = mean_squared_error(data["Close"], predictions)
23         rmse = np.sqrt(mse)
24         mape = np.mean(np.abs((data["Close"] - predictions) / data["Close"])) * 100
25
26         # Print evaluation metrics
27         print(f"Evaluation Metrics for {coin}:")
28         print(f"Mean Absolute Error (MAE): {mae}")
29         print(f"Mean Squared Error (MSE): {mse}")
30         print(f"Root Mean Squared Error (RMSE): {rmse}")
31         print(f"Mean Absolute Percentage Error (MAPE): {mape}")
32         print("-----")
33
34         # Plot actual vs predicted values
35         plt.figure(figsize=(12, 6))
36         plt.plot(data.index, data["Close"], label="Actual")
37         plt.plot(data.index, predictions, label="Predicted")
38         plt.xlabel("Date")
39         plt.ylabel("Price")
40         plt.title(f"Actual vs Predicted for {coin}")

```

```
41     plt.legend()
42     plt.show()
43
44     # Print the model summary
45     print(f"ARIMA Model Summary for {coin}:")
46     print(fitted_model.summary())
47
48     except Exception as e:
49         print(f"Error occurred while fitting the ARIMA model for {coin}:")
50         print(str(e))
51
52 # Read the individual CSV files
53 btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
54 eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
55 xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
56 ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
57 usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")
58
59 coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']
60
61 # Fit the ARIMA model for each coin
62 for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
63     fit_arima_model(coin_df, coin)
64
```

Evaluation Metrics for Bitcoin:
Mean Absolute Error (MAE): 209.93640789538813
Mean Squared Error (MSE): 357706.5019941227
Root Mean Squared Error (RMSE): 598.0856978678914
Mean Absolute Percentage Error (MAPE): 4.472526038975977

