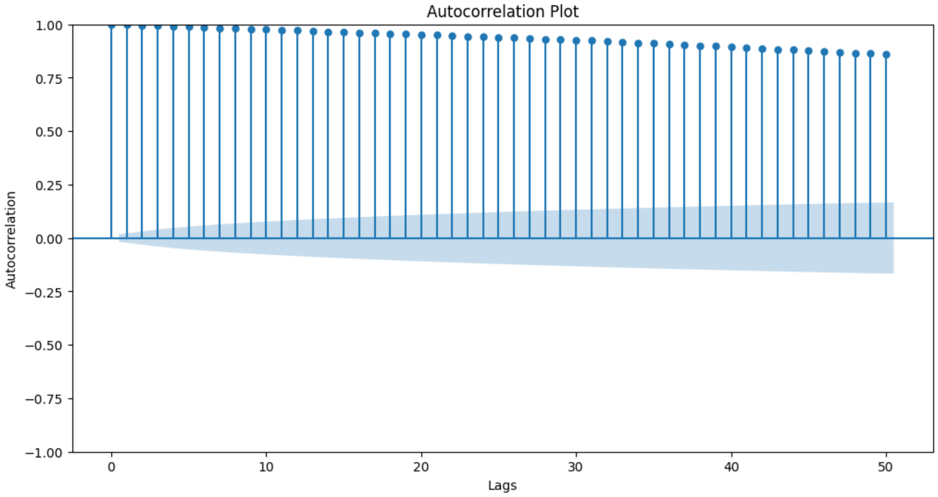
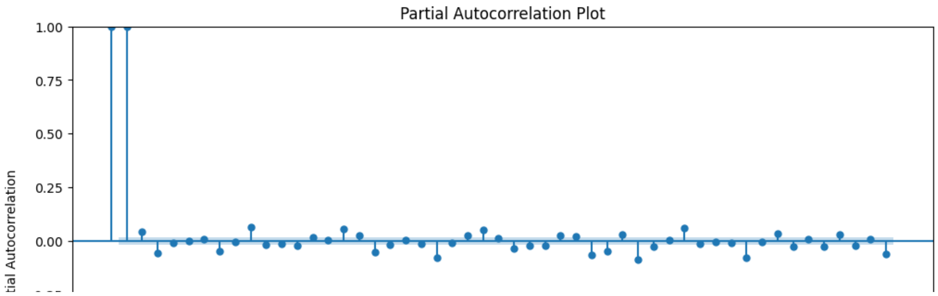
```
import pandas as pd

# Read the individual CSV files
btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")

# Concatenate the datasets vertically
df = pd.concat([btc_df, eth_df, xrp_df, ltc_df, usdc_df])

# Save the merged dataset to a new CSV file
df.to_csv("cryptocurrency.csv", index=False)
data = df
df = data.copy()
```

```
#!pip install statsmodels
#!pip install scikit-learn
#!pip install pandas

```

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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Load the dataset
df = pd.read_csv("cryptocurrency.csv")

# Preprocess the data
# Assuming you have a column named "Date" containing the timestamps
df["Date"] = pd.to_datetime(df["Date"])
df.set_index("Date", inplace=True)

# Plot the autocorrelation and partial autocorrelation plots
fig, ax = plt.subplots(figsize=(12, 6))
plot_acf(df["Close"], ax=ax, lags=50)
plt.xlabel("Lags")
plt.ylabel("Autocorrelation")
plt.title("Autocorrelation Plot")

fig, ax = plt.subplots(figsize=(12, 6))
plot_pacf(df["Close"], ax=ax, lags=50)
plt.xlabel("Lags")
plt.ylabel("Partial Autocorrelation")
plt.title("Partial Autocorrelation Plot")

plt.show()

```

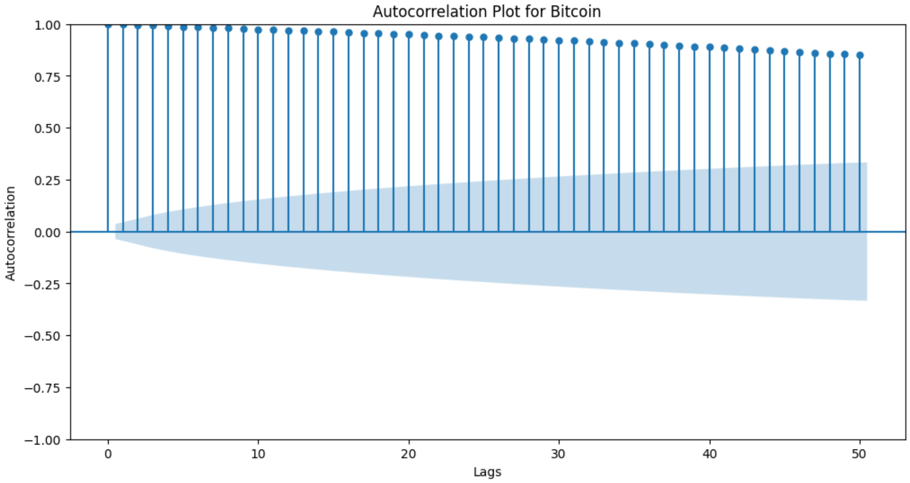


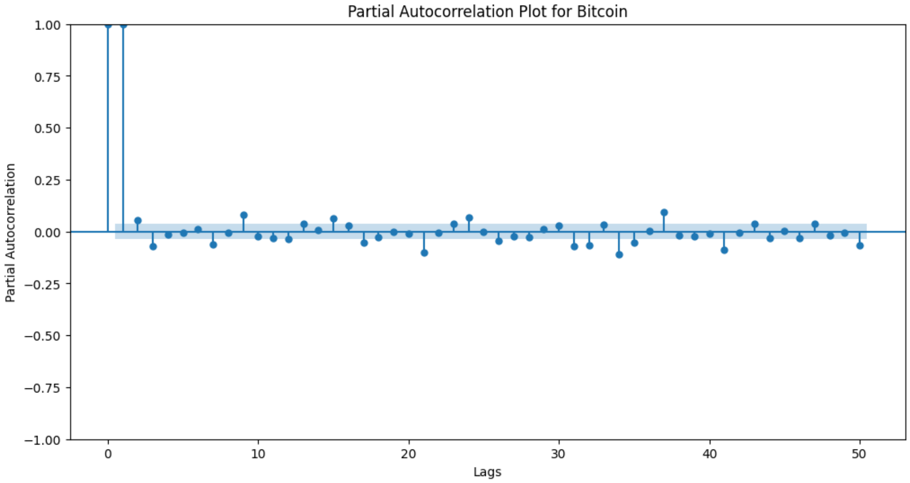


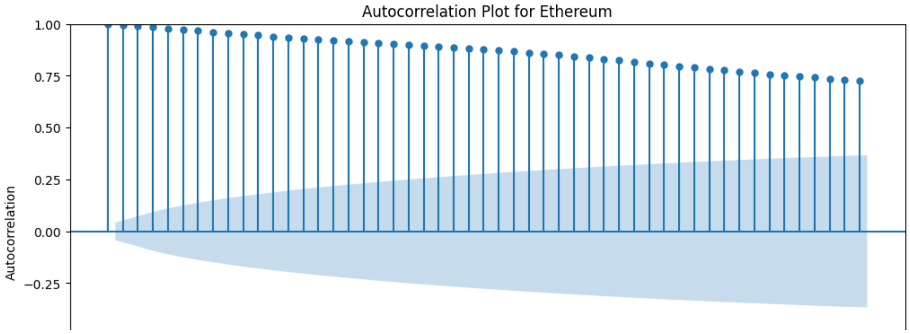
```
#For each Coin
```

```
#As Function adn Class
```

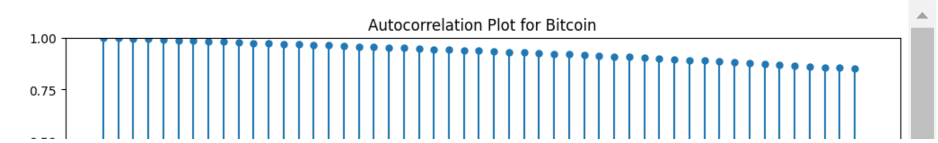
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

def find_arima_order(data, coin):
    # Preprocess the data
    data["Date"] = pd.to_datetime(data["Date"])
    data.set_index("Date", inplace=True)

    # Plot the autocorrelation and partial autocorrelation plots
    fig, ax = plt.subplots(figsize=(12, 6))
    plot_acf(data["Close"], ax=ax, lags=50)
    plt.xlabel("Lags")
    plt.ylabel("Autocorrelation")
    plt.title(f"Autocorrelation Plot for {coin}")

    fig, ax = plt.subplots(figsize=(12, 6))
    plot_pacf(data["Close"], ax=ax, lags=50)
    plt.xlabel("Lags")
    plt.ylabel("Partial Autocorrelation")
    plt.title(f"Partial Autocorrelation Plot for {coin}")

    plt.show()

# Read the individual CSV files
btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")

coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']

# Call the function for each coin
for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
    find_arima_order(coin_df, coin)

```





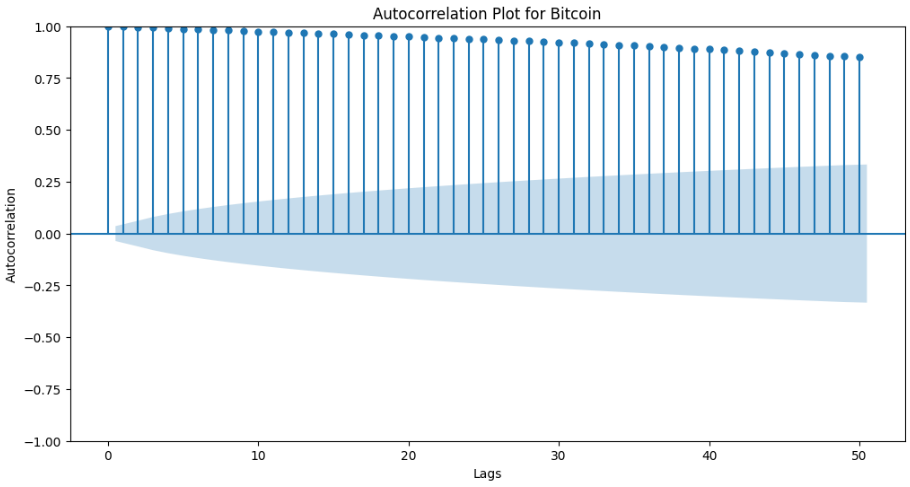


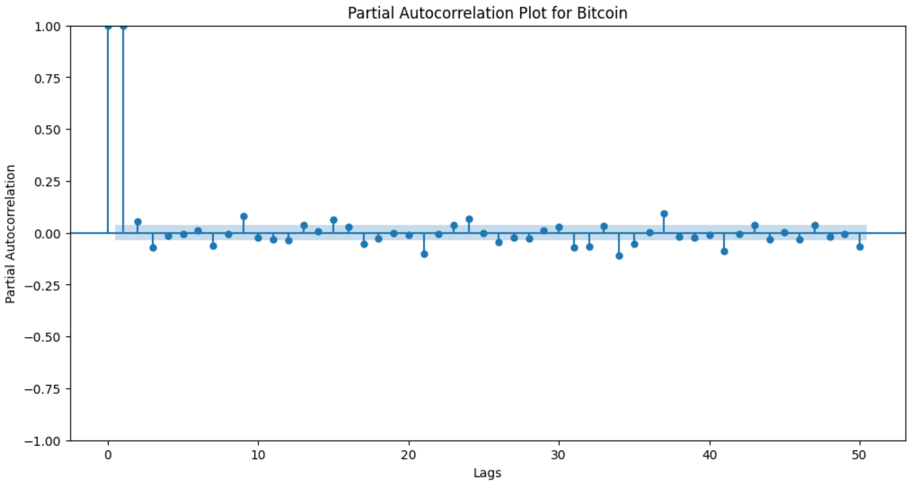
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

def fit_arima_model(data, coin):
    # Preprocess the data
    data["Date"] = pd.to_datetime(data["Date"])
    data.set_index("Date", inplace=True)

    # Plot the autocorrelation and partial autocorrelation plots
    fig, ax = plt.subplots(figsize=(12, 6))
    plot_acf(data["Close"], ax=ax, lags=50)
    plt.xlabel("Lags")
    plt.ylabel("Autocorrelation")
    plt.title(f"Autocorrelation Plot for {coin}")

    fig, ax = plt.subplots(figsize=(12, 6))
    plot_pacf(data["Close"], ax=ax, lags=50)
    plt.xlabel("Lags")
    plt.ylabel("Partial Autocorrelation")
    plt.title(f"Partial Autocorrelation Plot for {coin}")

    plt.show()

    # Determine the order for the ARIMA model
    # Update these values based on the insights from the plots
    p = 2  # Autoregressive order
    d = 0  # Integrated order
    q = 1  # Moving average order

    # Fit the ARIMA model
    model = ARIMA(data["Close"], order=(p, d, q))
    fitted_model = model.fit()

    # Print the model summary
    print(f"ARIMA Model Summary for {coin}:")
    print(fitted_model.summary())

# Read the individual CSV files
btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")

coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']

# Fit the ARIMA model for each coin
for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
    fit_arima_model(coin_df, coin)

```

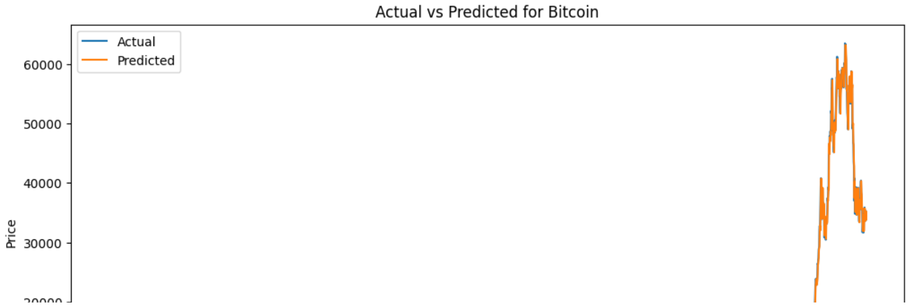


```
#Actual Vs Predicted
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

def fit_arima_model(data, coin):
    # Preprocess the data
    data["Date"] = pd.to_datetime(data["Date"])
    data.set_index("Date", inplace=True)

    # Plot the autocorrelation and partial autocorrelation plots
    fig, ax = plt.subplots(figsize=(12, 6))
    plot_acf(data["Close"], ax=ax, lags=50)
    plt.xlabel("Lags")
    plt.ylabel("Autocorrelation")
    plt.title(f"Autocorrelation Plot for {coin}")

    fig, ax = plt.subplots(figsize=(12, 6))
    plot_pacf(data["Close"], ax=ax, lags=50)
    plt.xlabel("Lags")
    plt.ylabel("Partial Autocorrelation")
    plt.title(f"Partial Autocorrelation Plot for {coin}")

    plt.show()

    # Determine the order for the ARIMA model
    # Update these values based on the insights from the plots
    p = 2  # Autoregressive order
    d = 0  # Integrated order
    q = 1  # Moving average order

    # Fit the ARIMA model
    model = ARIMA(data["Close"], order=(p, d, q))
    fitted_model = model.fit()

    # Generate predictions
    predictions = fitted_model.predict(start=data.index[0], end=data.index[-1])

    # Plot actual vs predicted values
    plt.figure(figsize=(12, 6))
    plt.plot(data.index, data["Close"], label="Actual")
    plt.plot(data.index, predictions, label="Predicted")
    plt.xlabel("Date")
    plt.ylabel("Price")
    plt.title(f"Actual vs Predicted for {coin}")
    plt.legend()
    plt.show()

    # Print the model summary
    print(f"ARIMA Model Summary for {coin}:")
    print(fitted_model.summary())

# Read the individual CSV files
btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")

coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']

# Fit the ARIMA model for each coin
for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
    fit_arima_model(coin_df, coin)

```



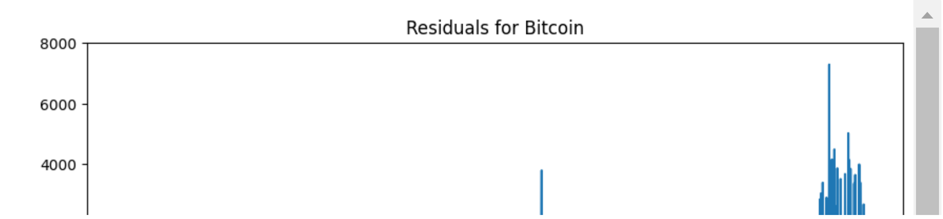




```
#In addition to the actual vs predicted values plot, there are several other analysis techniques you can apply to evaluate the ARIMA model
```

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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

def residual_analysis(data, coin):
    # Fit the ARIMA model
    model = ARIMA(data["Close"], order=(p, d, q))
    fitted_model = model.fit()

    # Get the residuals
    residuals = fitted_model.resid

    # Plot the residuals
    plt.figure(figsize=(10, 6))
    plt.plot(residuals)
    plt.title(f"Residuals for {coin}")
    plt.xlabel("Time")
    plt.ylabel("Residuals")
    plt.show()

    # Summary statistics of residuals
    print(f"Summary Statistics of Residuals for {coin}:")
    print(residuals.describe())

    # Autocorrelation plot of residuals
    fig, ax = plt.subplots(figsize=(10, 6))
    plot_acf(residuals, ax=ax, lags=50)
    plt.xlabel("Lags")
    plt.ylabel("Autocorrelation")
    plt.title(f"Autocorrelation Plot of Residuals for {coin}")
    plt.show()

# Read the individual CSV files
btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")

coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']

# Define the ARIMA order (p, d, q)
p = 2
d = 0
q = 1

# Perform residual analysis for each coin
for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
    residual_analysis(coin_df, coin)

```



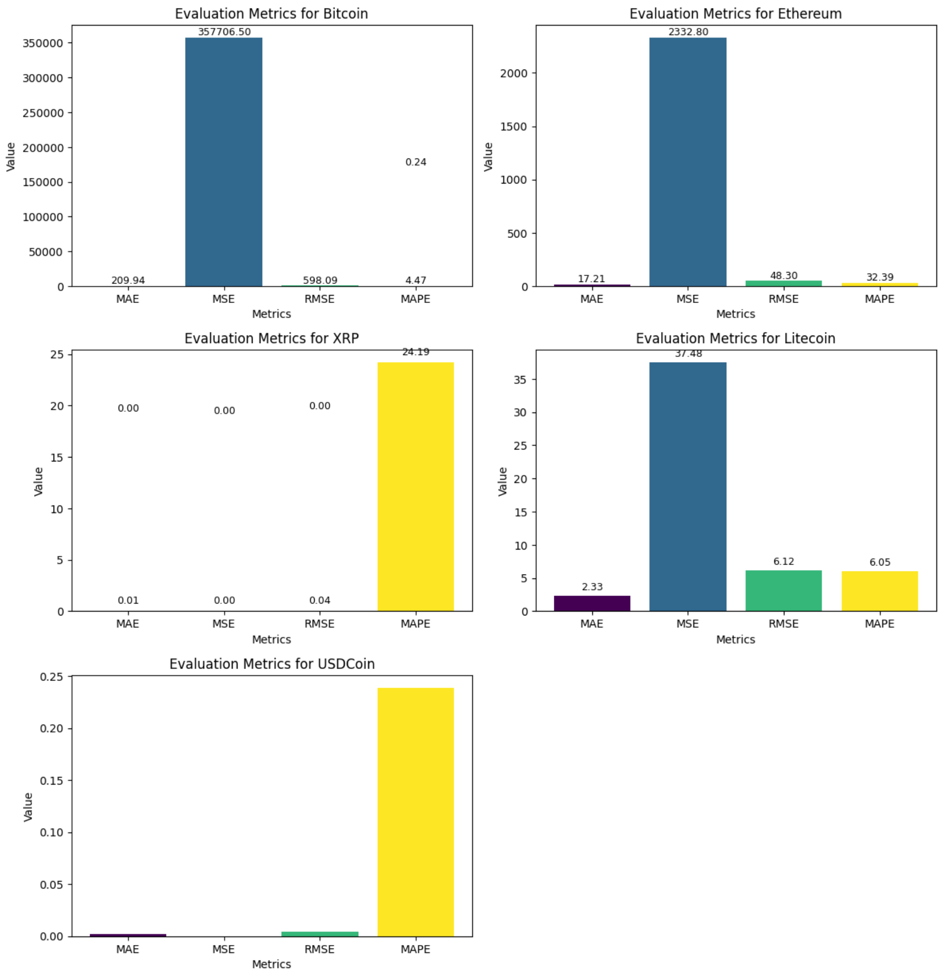
```
#Model Evaluation Metrics: Calculate various evaluation metrics to quantify the performance of the ARIMA model. Some commonly used metrics
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error

def fit_arima_model(data, coin):
    # Preprocess the data
    data["Date"] = pd.to_datetime(data["Date"])
    data.set_index("Date", inplace=True)

    # Determine the order for the ARIMA model
    # Update these values based on your analysis
    p = 2  # Autoregressive order
    d = 0  # Integrated order
    q = 1  # Moving average order

    # Fit the ARIMA model
    model = ARIMA(data["Close"], order=(p, d, q))
    fitted_model = model.fit()

    # Generate predictions
    predictions = fitted_model.predict(start=data.index[0], end=data.index[-1])

    # Calculate evaluation metrics
    mae = mean_absolute_error(data["Close"], predictions)
    mse = mean_squared_error(data["Close"], predictions)
    rmse = np.sqrt(mse)
    mape = np.mean(np.abs((data["Close"] - predictions) / data["Close"])) * 100

    # Print the evaluation metrics for the current coin
    print("Evaluation Metrics for", coin)
    print("Mean Absolute Error (MAE):", mae)
    print("Mean Squared Error (MSE):", mse)
    print("Root Mean Squared Error (RMSE):", rmse)
    print("Mean Absolute Percentage Error (MAPE):", mape)
    print("----------------------------------------")

# Read the individual CSV files
btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")

coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']

# Fit the ARIMA model for each coin
for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
    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
---------------------------------------
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error

def fit_arima_model(data, coin, row, col):
    # Preprocess the data
    data["Date"] = pd.to_datetime(data["Date"])
    data.set_index("Date", inplace=True)

    # Determine the order for the ARIMA model
    # Update these values based on your analysis
    p = 2  # Autoregressive order
    d = 0  # Integrated order
    q = 1  # Moving average order

    # Fit the ARIMA model
    model = ARIMA(data["Close"], order=(p, d, q))
    fitted_model = model.fit()

    # Generate predictions
    predictions = fitted_model.predict(start=data.index[0], end=data.index[-1])

    # Calculate evaluation metrics
    mae = mean_absolute_error(data["Close"], predictions)
    mse = mean_squared_error(data["Close"], predictions)
    rmse = np.sqrt(mse)
    mape = np.mean(np.abs((data["Close"] - predictions) / data["Close"])) * 100

    # Create bar plots for evaluation metrics
    metrics = ["MAE", "MSE", "RMSE", "MAPE"]
    values = [mae, mse, rmse, mape]
    colors = plt.cm.viridis(np.linspace(0, 1, len(metrics)))  # Colormap for colors

    plt.subplot(3, 2, row*2 + col + 1)
    bars = plt.bar(metrics, values, color=colors)
    plt.xlabel("Metrics")
    plt.ylabel("Value")
    plt.title(f"Evaluation Metrics for {coin}")

    # Label the levels
    for i, (bar, value) in enumerate(zip(bars, values)):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.5,
                 f"{value:.2f}", ha='center', va='bottom', color='black', fontsize=9)

    plt.tight_layout()

# Read the individual CSV files
btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")

coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']

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

# Fit the ARIMA model for each coin and create subplots
for i, (coin, coin_df) in enumerate(zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df])):
    fit_arima_model(coin_df, coin, i // 2, i % 2)

plt.show()

```



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.

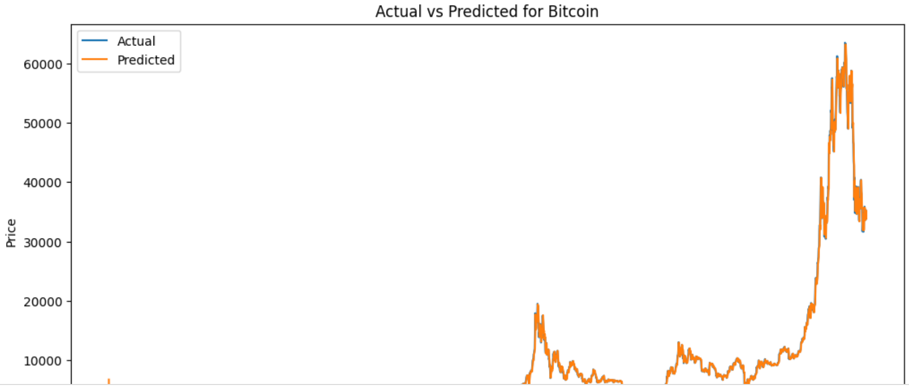
Overall, the performance of the ARIMA model varies across different coins. XRP and USDCoin show the best performance, with low error metrics, indicating accurate predictions. Bitcoin and Ethereum have higher error metrics, suggesting that the model's predictions for these coins may have more significant 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.

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

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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error, mean_squared_error

def fit_arima_model(data, coin):
    try:
        # Preprocess the data
        data["Date"] = pd.to_datetime(data["Date"])
        data.set_index("Date", inplace=True)

        # Fit the ARIMA model
        model = ARIMA(data["Close"], order=(2, 0, 1))
        fitted_model = model.fit()

        # Generate predictions
        predictions = fitted_model.predict(start=data.index[0], end=data.index[-1])

        # Calculate evaluation metrics
        mae = mean_absolute_error(data["Close"], predictions)
        mse = mean_squared_error(data["Close"], predictions)
        rmse = np.sqrt(mse)
        mape = np.mean(np.abs((data["Close"] - predictions) / data["Close"])) * 100

        # Print evaluation metrics
        print(f"Evaluation Metrics for {coin}:")
        print(f"Mean Absolute Error (MAE): {mae}")
        print(f"Mean Squared Error (MSE): {mse}")
        print(f"Root Mean Squared Error (RMSE): {rmse}")
        print(f"Mean Absolute Percentage Error (MAPE): {mape}")
        print("----------------------------------------")

        # Plot actual vs predicted values
        plt.figure(figsize=(12, 6))
        plt.plot(data.index, data["Close"], label="Actual")
        plt.plot(data.index, predictions, label="Predicted")
        plt.xlabel("Date")
        plt.ylabel("Price")
        plt.title(f"Actual vs Predicted for {coin}")
        plt.legend()
        plt.show()

        # Print the model summary
        print(f"ARIMA Model Summary for {coin}:")
        print(fitted_model.summary())

    except Exception as e:
        print(f"Error occurred while fitting the ARIMA model for {coin}:")
        print(str(e))

# Read the individual CSV files
btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")

coins = ['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USDCoin']

# Fit the ARIMA model for each coin
for coin, coin_df in zip(coins, [btc_df, eth_df, xrp_df, ltc_df, usdc_df]):
    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
----------------------------------------
```


Actual vs Predicted for Bitcoin
Actual
Predicted
Price