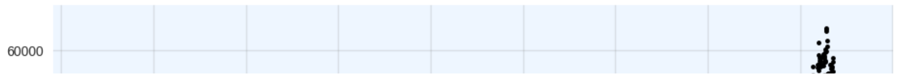
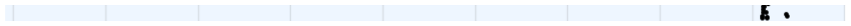
```
#Prophet Model
```

```
import pandas as pd
from prophet import Prophet

# 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])

# Convert the 'Date' column to a datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Rename the 'Date' and 'Close' columns to 'ds' and 'y', respectively
df = df.rename(columns={'Date': 'ds', 'Close': 'y'})

# Create and fit the model
model = Prophet()
model.fit(df)

# Generate future dates for prediction
future_dates = model.make_future_dataframe(periods=30)  # Adjust the number of future data points as needed

# Make predictions
forecast = model.predict(future_dates)

# Plot the forecasted values
model.plot(forecast)

```

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to overrid
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/ubj8kjxp.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/im0sk2uv.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/p
17:19:10 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:19:12 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```

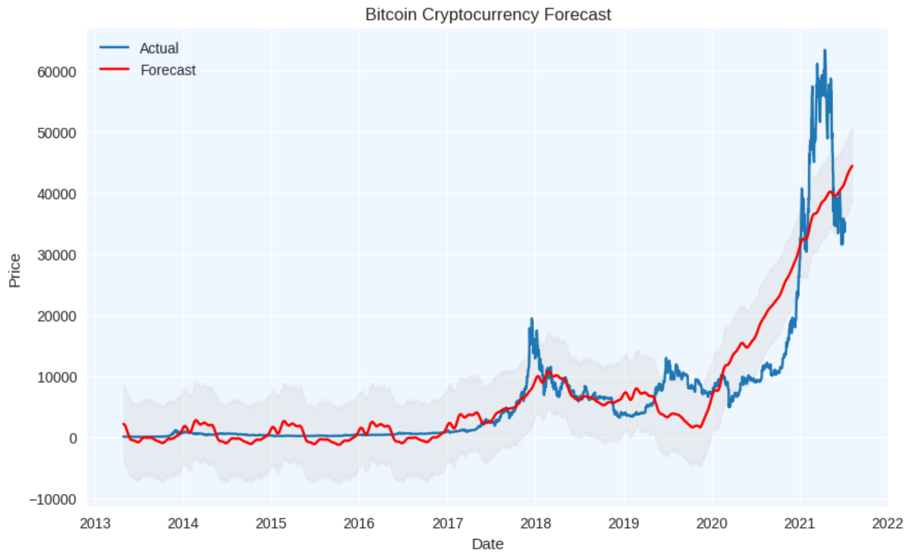


```
#Better representation of Prophet Model
```

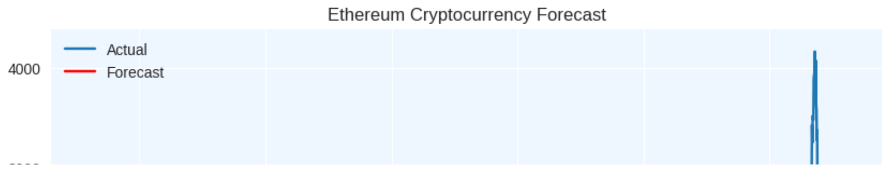


```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# 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")

# Create a dictionary of coins and their respective dataframes
coins = {
    'Bitcoin': btc_df,
    'Ethereum': eth_df,
    'XRP': xrp_df,
    'Litecoin': ltc_df,
    'USDCoin': usdc_df
}

# Set the color palette and background color
#colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']
plt.style.use('seaborn')
plt.rcParams['axes.facecolor'] = '#eef6ff'

# Iterate over the coins dictionary
for coin, df in coins.items():
    # Convert the 'Date' column to a datetime format
    df['Date'] = pd.to_datetime(df['Date'])

    # Rename the 'Date' and 'Close' columns to 'ds' and 'y', respectively
    df = df.rename(columns={'Date': 'ds', 'Close': 'y'})

    # Create and fit the model
    model = Prophet()
    model.fit(df)

    # Generate future dates for prediction
    future_dates = model.make_future_dataframe(periods=30)  # Adjust the number of future data points as needed

    # Make predictions
    forecast = model.predict(future_dates)

    # Plot the forecasted values
    fig = plt.figure(figsize=(10, 6))
    ax = fig.add_subplot(111)
    ax.plot(df['ds'], df['y'], color=colors[0], label='Actual')
    ax.plot(forecast['ds'], forecast['yhat'], color='red', label='Forecast')
    ax.fill_between(forecast['ds'], forecast['yhat_lower'], forecast['yhat_upper'], color='lightgray', alpha=0.3)

    # Customize the plot
    plt.title(f'{coin} Cryptocurrency Forecast')
    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.legend()
    plt.show()

```

```
<ipython-input-34-5725250f138f>:23: MatplotlibDeprecationWarning: The seaborn styles shipp
  plt.style.use('seaborn')
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to overr
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/wrzltda4.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/y7jtkvuv.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model
17:19:15 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:19:15 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```



```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to overr
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/bkv6jfbv.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/xpyxaoqb.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model
17:19:17 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:19:18 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```



```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# 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")

# Create a dictionary of coins and their respective dataframes
coins = {
    'Bitcoin': btc_df,
    'Ethereum': eth_df,
    'XRP': xrp_df,
    'Litecoin': ltc_df,
    'USDCoin': usdc_df
}

# Set the color palette and background color
colors = ['#1f77b4']
plt.rcParams['axes.facecolor'] = '#eef6ff'

# Create a 3x2 grid of subplots
fig, axes = plt.subplots(3, 2, figsize=(12, 12))

```

```
# Iterate over the coins dictionary and plot the forecast in each subplot
for i, (coin, df) in enumerate(coins.items()):
    row = i // 2
    col = i % 2
    ax = axes[row, col]

    # Convert the 'Date' column to a datetime format
    df['Date'] = pd.to_datetime(df['Date'])

    # Rename the 'Date' and 'Close' columns to 'ds' and 'y', respectively
    df = df.rename(columns={'Date': 'ds', 'Close': 'y'})

    # Create and fit the model
    model = Prophet()
    model.fit(df)

    # Generate future dates for prediction
    future_dates = model.make_future_dataframe(periods=30)  # Adjust the number of future data points as needed

    # Make predictions
    forecast = model.predict(future_dates)

    # Plot the forecasted values
    ax.plot(df['ds'], df['y'], color=colors[0], label='Actual')
    ax.plot(forecast['ds'], forecast['yhat'], color= 'red', linestyle='dashed', label='Forecast')
    ax.fill_between(forecast['ds'], forecast['yhat_lower'], forecast['yhat_upper'], color=colors[0], alpha=0.3)

    # Customize the subplot
    ax.set_title(coin)
    ax.set_xlabel('Date')
    ax.set_ylabel('Price')
    ax.legend()

# Adjust the spacing between subplots
fig.tight_layout()

# Show the plot
plt.show()

```

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to overr
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/cvrjm5pf.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/2wu64z51.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model
17:19:27 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:19:28 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to overr
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/ps6iludh.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/i1p2gvg2.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model
17:19:29 - cmdstanpy - INFO - Chain [1] start processing
```

```
#to predict future values- 365 days
```

```
INFO:cmdstanpy:Chain [1] done processing
```

```python
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# 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")

# Create a dictionary of coins and their respective dataframes
coins = {
    'Bitcoin': btc_df,
    'Ethereum': eth_df,
    'XRP': xrp_df,
    'Litecoin': ltc_df,
    'USDCoin': usdc_df
}

# Set the color palette and background color
colors = ['#1f77b4']
plt.rcParams['axes.facecolor'] = '#eef6ff'

# Create a 3x2 grid of subplots
fig, axes = plt.subplots(3, 2, figsize=(12, 12))

# Iterate over the coins dictionary and plot the forecast in each subplot
for i, (coin, df) in enumerate(coins.items()):
    row = i // 2
    col = i % 2
    ax = axes[row, col]

    # Convert the 'Date' column to a datetime format
    df['Date'] = pd.to_datetime(df['Date'])

    # Rename the 'Date' and 'Close' columns to 'ds' and 'y', respectively
    df = df.rename(columns={'Date': 'ds', 'Close': 'y'})

    # Create and fit the model
    model = Prophet()
    model.fit(df)

    # Generate future dates for prediction
    future_dates = model.make_future_dataframe(periods=365)  # Adjust the number of future data points as needed

    # Make predictions
    forecast = model.predict(future_dates)

    # Plot the forecasted values
    ax.plot(df['ds'], df['y'], color=colors[0], label='Actual')
    ax.plot(forecast['ds'], forecast['yhat'], color='red', linestyle='dashed', label='Forecast')
    ax.fill_between(forecast['ds'], forecast['yhat_lower'], forecast['yhat_upper'], color=colors[0], alpha=0.3)

    # Customize the subplot
    ax.set_title(coin)
    ax.set_xlabel('Date')
    ax.set_ylabel('Price')
    ax.legend()

# Adjust the spacing between subplots
fig.tight_layout()

# Show the plot
plt.show()

```

```
#Evaluation of Model
```

```
import pandas as pd
from prophet import Prophet
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

# 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")

```

```
# Create a dictionary of coins and their respective dataframes
coins = {
    'Bitcoin': btc_df,
    'Ethereum': eth_df,
    'XRP': xrp_df,
    'Litecoin': ltc_df,
    'USDCoin': usdc_df
}

# Set the color palette and background color
colors = ['#1f77b4']
plt.rcParams['axes.facecolor'] = '#eef6ff'

# Create a 3x2 grid of subplots
fig, axes = plt.subplots(3, 2, figsize=(12, 12))

# Iterate over the coins dictionary and plot the forecast in each subplot
for i, (coin, df) in enumerate(coins.items()):
    row = i // 2
    col = i % 2
    ax = axes[row, col]

    # Convert the 'Date' column to a datetime format
    df['Date'] = pd.to_datetime(df['Date'])

    # Rename the 'Date' and 'Close' columns to 'ds' and 'y', respectively
    df = df.rename(columns={'Date': 'ds', 'Close': 'y'})

    # Create and fit the model
    model = Prophet(daily_seasonality=False)  # Disable daily seasonality
    model.fit(df)

    # Generate future dates for prediction
    future_dates = model.make_future_dataframe(periods=30)  # Adjust the number of future data points as needed

    # Make predictions
    forecast = model.predict(future_dates)

    # Evaluate the model
    df_eval = forecast.set_index('ds')[['yhat']].join(df.set_index('ds')['y']).reset_index()
    df_eval.fillna(df_eval.mean(), inplace=True)  # Fill missing values with mean
    mse = ((df_eval['yhat'] - df_eval['y']) ** 2).mean()
    rmse = mse ** 0.5
    mae = (df_eval['yhat'] - df_eval['y']).abs().mean()
    r2 = r2_score(df_eval['y'], df_eval['yhat'])

    # Plot the forecasted values
    ax.plot(df['ds'], df['y'], color=colors[0], label='Actual')
    ax.plot(forecast['ds'], forecast['yhat'], color='red', linestyle='dashed', label='Forecast')
    ax.fill_between(forecast['ds'], forecast['yhat_lower'], forecast['yhat_upper'], color=colors[0], alpha=0.3)

    # Customize the subplot
    ax.set_title(coin)
    ax.set_xlabel('Date')
    ax.set_ylabel('Price')
    ax.legend()

    # Display evaluation metrics
    textstr = f"MSE: {mse:.2f}\nRMSE: {rmse:.2f}\nMAE: {mae:.2f}\nR2 Score: {r2:.2f}"
    ax.text(0.05, 0.95, textstr, transform=ax.transAxes, fontsize=10, verticalalignment='top')

# Adjust the spacing between subplots
fig.tight_layout()

# Show the plot
plt.show()

```

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/ltccs77g.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/_2n5i67r.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model
17:31:57 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:31:58 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<ipython-input-40-5b2b048c9bd2>:53: FutureWarning: DataFrame.mean and DataFrame.median wit
  df_eval.fillna(df_eval.mean(), inplace=True)  # Fill missing values with mean
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/pauqku5i.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/wppbkog3.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model
17:31:59 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:32:00 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<ipython-input-40-5b2b048c9bd2>:53: FutureWarning: DataFrame.mean and DataFrame.median wit
  df_eval.fillna(df_eval.mean(), inplace=True)  # Fill missing values with mean
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/8mz72u5s.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/pr9wtrlx.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model
17:32:01 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:32:02 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<ipython-input-40-5b2b048c9bd2>:53: FutureWarning: DataFrame.mean and DataFrame.median wit
  df_eval.fillna(df_eval.mean(), inplace=True)  # Fill missing values with mean
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/hg5i3j8r.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/w086wz7z.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model
17:32:03 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:32:04 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<ipython-input-40-5b2b048c9bd2>:53: FutureWarning: DataFrame.mean and DataFrame.median wit
  df_eval.fillna(df_eval.mean(), inplace=True)  # Fill missing values with mean
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/xsl1hdvm.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/tkeqyvou.json
DEBUG:cmdstanpy:idx 0
```

```python
import pandas as pd
from prophet import Prophet
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# 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")

# Create a dictionary of coins and their respective dataframes
coins = {
    'Bitcoin': btc_df,
    'Ethereum': eth_df,
    'XRP': xrp_df,
    'Litecoin': ltc_df,
    'USDCoin': usdc_df
}

# Initialize an empty DataFrame to store the evaluation metrics
metrics_df = pd.DataFrame(columns=['MSE', 'RMSE', 'MAE', 'R2 Score'])

# Iterate over the coins dictionary and evaluate the model for each coin
for coin, df in coins.items():
    # Convert the 'Date' column to a datetime format
    df['Date'] = pd.to_datetime(df['Date'])

    # Rename the 'Date' and 'Close' columns to 'ds' and 'y', respectively
    df = df.rename(columns={'Date': 'ds', 'Close': 'y'})

    # Create and fit the model
    model = Prophet()
    model.fit(df)

    # Generate future dates for prediction
    future_dates = model.make_future_dataframe(periods=30)  # Adjust the number of future data points as needed

    # Make predictions
    forecast = model.predict(future_dates)

    # Evaluate the model
    df_eval = forecast[['ds', 'yhat']].merge(df[['ds', 'y']], on='ds', how='inner').dropna()
    mse = mean_squared_error(df_eval['y'], df_eval['yhat'])
    rmse = mse ** 0.5
    mae = mean_absolute_error(df_eval['y'], df_eval['yhat'])
    r2 = r2_score(df_eval['y'], df_eval['yhat'])

    # Add the evaluation metrics to the DataFrame
    metrics_df.loc[coin] = [mse, rmse, mae, r2]

# Display the table of evaluation metrics
print(metrics_df)

```

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/8r76dxb2.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/by9i89ri.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=79782',
17:44:51 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:44:51 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/m8haq3z1.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/bph5miy0.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=49837',
17:44:52 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:44:53 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/dtnp1tzq.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/9di9l0u2.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=32572',
17:44:54 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:44:56 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/b5ab6at8.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/roscsy03.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=71320',
17:44:58 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:44:59 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/yeaeu9pc.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/vz72c5og.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=59313',
17:45:00 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:45:01 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
                   MSE         RMSE          MAE  R2 Score
Bitcoin   2.275776e+07  4770.509099  2895.874572  0.821655
Ethereum  3.165409e+04   177.915951   108.106014  0.912347
XRP       3.865966e-02     0.196621     0.110671  0.662744
Litecoin  8.686634e+02    29.473096    19.794538  0.782727
USDCoin   1.746138e-05     0.004179     0.002638  0.624399
```

Here is the analysis for each coin based on the evaluation metrics:

1. **Bitcoin:**
   - MSE: 2.275776e+07
   - RMSE: 4770.509099
   - MAE: 2895.874572
   - R2 Score: 0.821655

   Analysis: The model achieves a relatively high R2 score of 0.821655, indicating that it captures a significant portion of the variance in Bitcoin's price. However, the MSE and RMSE values are quite large, suggesting that there are still considerable prediction errors. The MAE is also relatively high, indicating that the average difference between the predicted and actual values is significant.
2. **Ethereum:**
   - MSE: 3.165409e+04
   - RMSE: 177.915951
   - MAE: 108.106014
   - R2 Score: 0.912347

   Analysis: The model performs well for Ethereum, as indicated by the high R2 score of 0.912347. The MSE, RMSE, and MAE values are relatively small, suggesting that the model's predictions are generally accurate and close to the actual values.
3. **XRP:**
   - MSE: 3.865966e-02
   - RMSE: 0.196621
   - MAE: 0.110671
   - R2 Score: 0.662744

   Analysis: The model achieves a moderate R2 score of 0.662744 for XRP, indicating that it captures a reasonable portion of the variance in XRP's price. The MSE, RMSE, and MAE values are quite small, suggesting that the model's predictions are relatively accurate for XRP.
4. **Litecoin:**
   - MSE: 8.686634e+02
   - RMSE: 29.473096
   - MAE: 19.794538
   - R2 Score: 0.782727

   Analysis: The model performs reasonably well for Litecoin, as indicated by the R2 score of 0.782727. The MSE, RMSE, and MAE values are relatively small, suggesting that the model's predictions are generally accurate and close to the actual values.
5. **USDCoin:**
   - MSE: 1.746138e-05
   - RMSE: 0.004179
   - MAE: 0.002638
   - R2 Score: 0.624399

   Analysis: The model achieves a moderate R2 score of 0.624399 for USDCoin, indicating that it captures a reasonable portion of the variance in USDCoin's price. The MSE, RMSE, and MAE values are very small, suggesting that the model's predictions are quite accurate for USDCoin.

Overall, the model performs well for most coins, with high R2 scores and relatively small error metrics (MSE, RMSE, MAE). However, there is room for improvement, especially for Bitcoin, where the errors are relatively larger. Further analysis and refinement of the model may be required to improve its performance for Bitcoin.

```
#For each refinement step applied to each coin separately:
```

```
import pandas as pd
from prophet import Prophet

# 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")
```

```

# Create a dictionary of coins and their respective dataframes
coins = {
    'Bitcoin': btc_df,
    'Ethereum': eth_df,
    'XRP': xrp_df,
    'Litecoin': ltc_df,
    'USDCoin': usdc_df
}

# Iterate over the coins dictionary and refine the model for each coin
for coin, df in coins.items():
    # Feature Engineering
    # Example: Include additional features like Volume and Market Cap
    df['Volume'] = df['Volume'].fillna(0)
    df['MarketCap'] = df['Close'] * df['Volume']

    # Model Hyperparameter Tuning
    model = Prophet(changepoint_prior_scale=0.05, seasonality_prior_scale=10)
    # Update the hyperparameters according to the desired configuration

    # Data Quality and Preprocessing
    # Example: Handle missing values and outliers
    df = df.fillna(method='ffill')
    df = df.dropna()
    # You can add additional preprocessing steps here if required

    # Print the refined model for each coin
    print(f"Refined Model for {coin}: {model}")

```

```
Refined Model for Bitcoin: <prophet.forecaster.Prophet object at 0x7fdcabc8bb50>
Refined Model for Ethereum: <prophet.forecaster.Prophet object at 0x7fdcaa5fc220>
Refined Model for XRP: <prophet.forecaster.Prophet object at 0x7fdcaa5fecb0>
Refined Model for Litecoin: <prophet.forecaster.Prophet object at 0x7fdcaa5fc220>
Refined Model for USDCoin: <prophet.forecaster.Prophet object at 0x7fdcaa5fead0>
```

```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# 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")

# Create a dictionary of coins and their respective dataframes
coins = {
    'Bitcoin': btc_df,
    'Ethereum': eth_df,
    'XRP': xrp_df,
    'Litecoin': ltc_df,
    'USDCoin': usdc_df
}

# Iterate over the coins dictionary and refine the model for each coin
for coin, df in coins.items():
    # Rename the columns to match the expected format
    df = df.rename(columns={'Date': 'ds', 'Close': 'y'})

    # Feature Engineering
    # Example: Include additional features like Volume and Market Cap
    df['Volume'] = df['Volume'].fillna(0)
    df['MarketCap'] = df['y'] * df['Volume']

    # Model Hyperparameter Tuning
    model = Prophet(changepoint_prior_scale=0.05, seasonality_prior_scale=10)
    # Update the hyperparameters according to the desired configuration

    # Data Quality and Preprocessing
    # Example: Handle missing values and outliers
    df = df.fillna(method='ffill')
    df = df.dropna()
    # You can add additional preprocessing steps here if required

    # Fit the model
    model.fit(df)

    # Generate future dates for prediction
    future_dates = model.make_future_dataframe(periods=30)  # Adjust the number of future data points as needed

    # Make predictions
    forecast = model.predict(future_dates)

    # Plot the forecasted values
    model.plot(forecast)
    plt.title(f"Forecasted Prices for {coin}")
    plt.xlabel("Date")
    plt.ylabel("Price")
    plt.show()

```

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to overr
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/wggjgbsj.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpzeuacr7o/ybcjztqj.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model
18:20:59 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
18:20:59 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```