Date	7th November 2024
Team ID	SWTID1727321147
Project Name	Time Series Analysis For Bitcoin Price Prediction using Prophet
Minimum Marks	10 Marks

Data Exploration and Preprocessing Template

This phase identifies data sources, assesses quality issues like missing values and outliers, and implements resolution plans to ensure accurate and reliable time series analysis.

Section Description

1. **Data Overview**
   - The dataset consists of historical Bitcoin prices obtained from Yahoo Finance. It includes features such as date, open, high, low, close prices, volume, and adjusted close prices.
   - Data spans several years to capture both short-term fluctuations and long-term trends.
2. **Univariate Analysis**
   - Explore individual features like Bitcoin closing prices and volume.
   - Plot line charts to observe the general trend of prices over time.
3. **Bivariate Analysis**
   - Analyze the relationship between two features, such as Bitcoin prices and volume, using scatter plots and correlation analysis.
4. **Multivariate Analysis**
   - Investigate relationships between multiple features (e.g., open, high, low, and close prices) to detect co-movements in Bitcoin prices.
5. **Outliers and Anomalies**
   - Detect extreme spikes or drops in price data using statistical methods like IQR or Z-score.
   - Handle anomalies by smoothing the data (e.g., using moving averages) to reduce noise for the Prophet model.

Data Preprocessing Code Screenshots

1. **Loading Data**
   - Code for loading the historical Bitcoin price data from Yahoo Finance using `yfinance` or a CSV file.
   -
2. **Handling Missing Data**

```
# Download Bitcoin historical data from Yahoo Finance
bitcoin_data = yf.download('BTC-USD', start='2017-01-01', end='2024-11-01')

# Prepare data for Prophet
df_prophet = bitcoin_data[['Close']].reset_index()
df_prophet = df_prophet.rename(columns={'Date': 'ds', 'Close': 'y'})
```

   - Code to handle missing values in the dataset, such as forward-filling or interpolation.
   -

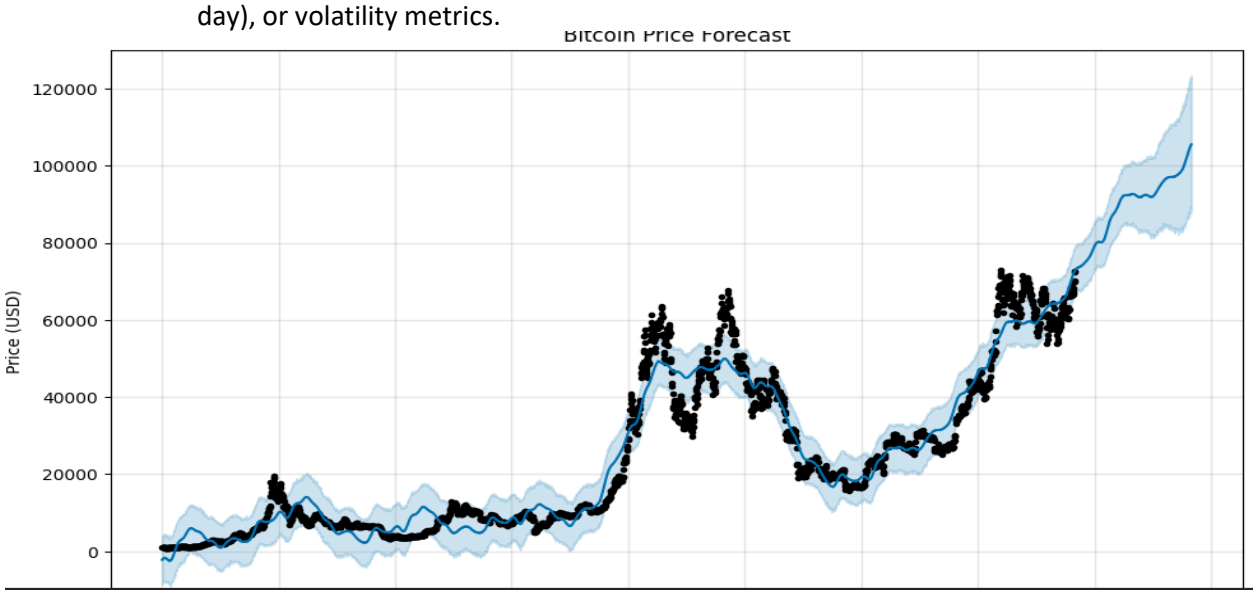
```
# Ensure 'ds' is in datetime format
df_prophet['ds'] = df_prophet['ds'].dt.date
df_prophet['y'] = df_prophet['y'].astype(int)
df_prophet.columns = range(df_prophet.shape[1])
```

3. **Data Transformation**
   - Apply log transformations or scaling if needed to normalize the data for better model performance.
   -

```
# code to convert tabular array into string array
dateArray= [str(item) for item in df_prophet[0]]
yearArray= [str(item) for item in df_prophet[1]]

#
new_data = {'ds':dateArray , 'y': yearArray}
```

4. **Feature Engineering**
   - Generate new features such as daily returns, moving averages (e.g., 50-day and 200-day), or volatility metrics.



5. **Save Processed Data**
    - Code for saving the preprocessed dataset as a CSV file for future model training.

```
# Plot components of the forecast (trend, seasonality, etc.)
fig2 = model.plot_components(forecast)
plt.show()

# Print some predictions
print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail())
```

    -
6.

```
[********************100%**********************]  1 of 1 completed
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1we4cvmv/k_3ayxk1.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1we4cvmv/630fdnkd.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=14858', 'data', 'file=/tmp/tmp1we4c
10:55:59 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:56:02 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```