Data Collection and Preprocessing Phase

Date	20 Nov 2024
Team ID	739720
Project Title	Time Series for Bitcoin Price Prediction using Prophet
Maximum Marks	6 Marks

Preprocessing Template

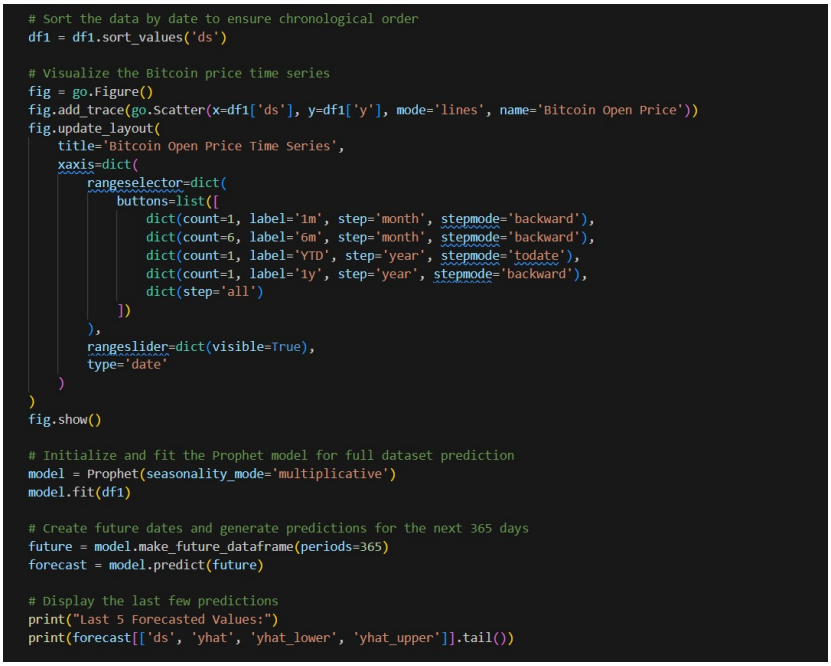
The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Section	**Description**
Data Overview	Provide an overview of the Bitcoin dataset(e.g., data range, frequency,columns like ds and y).ensure the dataset has ds (data) and y (value) columns, as required by Prophet.
Missing Value Handling	Identify and fill any missing dates or values(e.g., using forward-fill or interpolation).
Resampling	Resample data to daily or weekly frequency if needed,especially if there are irregular intervals.
Outlier Removal	Detect and handle outliers in price data that might distort the forecasting model.
Feature Engineering	Add features like lag values, rolling averages, or external regressors(volume, sentiment, etc.).
Train-Test Split	Split the dataset into training and testing parts while maintaining chronological order.

Data Preprocessing Code Screenshots	
Loading Data & Handling Missing Data	
Creating Prophet Data & Prophet Forecast	

```python
# Import necessary libraries
import pandas as pd
import plotly.graph_objects as go
from prophet import Prophet
from prophet.plot import plot_plotly, plot_components_plotly
from datetime import datetime, timedelta
from sklearn.metrics import mean_absolute_error
import warnings

# Suppress warnings and format floats for readability
warnings.filterwarnings('ignore')
pd.options.display.float_format = '${:,.2f}'.format

# Load the dataset from the CSV file
df = pd.read_csv('C:/Users/abhinaya/OneDrive/Desktop/BitcoinTimeSeriesFbProphet-main/BTC-USD_Historical_Data_2016_2024.csv', parse_dates=['Date'])

# Select and rename columns for Prophet
df1 = df[['Date', 'Open']].rename(columns={'Date': 'ds', 'Open': 'y'})

# Ensure 'ds' is in datetime format and timezone-naive
df1['ds'] = pd.to_datetime(df1['ds'])
if df1['ds'].dt.tz is not None:
    df1['ds'] = df1['ds'].dt.tz_localize(None)

# Sort the data by date to ensure chronological order
df1 = df1.sort_values('ds')
```

```python
# Predict the next day's price
next_day = (datetime.today() + timedelta(days=1)).strftime('%Y-%m-%d')
try:
    predicted_value = forecast.loc[forecast['ds'] == next_day, 'yhat'].iloc[0]
    print(f"Predicted Bitcoin Open Price for {next_day}: ${predicted_value:,.2f}")
except IndexError:
    print(f"No prediction found for {next_day}")

# Visualize the forecast and its components
plot_plotly(model, forecast).show()
plot_components_plotly(model, forecast).show()

# Evaluate the model using a train-test split
train = df1[df1['ds'] < '2023-01-01']
test = df1[df1['ds'] >= '2023-01-01']

# Create a new Prophet model instance for evaluation
eval_model = Prophet(seasonality_mode='multiplicative')
eval_model.fit(train)

# Make predictions on the test period
future_test = eval_model.make_future_dataframe(periods=len(test))
forecast_test = eval_model.predict(future_test)

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(test['y'], forecast_test['yhat'][-len(test):])
print(f"Mean Absolute Error: ${mae:,.2f}")
```

Visualisation	

```python
# Sort the data by date to ensure chronological order
df1 = df1.sort_values('ds')

# Visualize the Bitcoin price time series
fig = go.Figure()
fig.add_trace(go.Scatter(x=df1['ds'], y=df1['y'], mode='lines', name='Bitcoin Open Price'))
fig.update_layout(
    title='Bitcoin Open Price Time Series',
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(count=1, label='1m', step='month', stepmode='backward'),
                dict(count=6, label='6m', step='month', stepmode='backward'),
                dict(count=1, label='YTD', step='year', stepmode='todate'),
                dict(count=1, label='1y', step='year', stepmode='backward'),
                dict(step='all')
            ])
        ),
        rangeslider=dict(visible=True),
        type='date'
    )
)
fig.show()

# Initialize and fit the Prophet model for full dataset prediction
model = Prophet(seasonality_mode='multiplicative')
model.fit(df1)

# Create future dates and generate predictions for the next 365 days
future = model.make_future_dataframe(periods=365)
forecast = model.predict(future)

# Display the last few predictions
print("Last 5 Forecasted Values:")
print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail())
```