PROJECT REPORT

CRYPTOCURRENCY DATA ANALYSIS USING PYTHON

Name:
Student ID:

Introduction

This Python application is a cryptocurrency data analysis tool designed to help users fetch and analyse historical price data for any given cryptocurrency trading pair label (e.g., BTC/USDT, ETH/BTC). The application utilizes the **ccxt** library, a powerful tool for accessing market data from various cryptocurrency exchanges like Binance. Additionally, it leverages **pandas** for efficient data manipulation and storage, **numpy** for performing numerical computations such as calculating volatility, and **matplotlib** for creating insightful visualizations of price trends. By combining these libraries, the application allows users to retrieve, store, analyse, and visualize historical market data for their chosen cryptocurrency pairs, providing a streamlined solution for crypto enthusiasts and traders to evaluate market trends and price volatility effectively.

Data Collection and Preprocessing

Import necessary libraries

```python
import ccxt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

- Import **ccxt** library to fetch and download cryptocurrency data.
- Import **pandas** library for data manipulation and analysis.
- Import **numpy** library for numerical operations.
- Import **matplotlib** library for creating visualizations like plots and charts.

Fetching Cryptocurrency Data

```python
def fetch_crypto_data(symbol, exchange_name, timeframe='1d', limit=30):
    # Fetch historical cryptocurrency price data and save to CSV.
    exchange = getattr(ccxt, exchange_name)()
    bars = exchange.fetch_ohlcv(symbol, timeframe=timeframe, limit=limit)
    data = pd.DataFrame(bars, columns=['timestamp', 'open', 'high', 'low', 'close', 'volume'])
    data['timestamp'] = pd.to_datetime(data['timestamp'], unit='ms')
    data.to_csv(f'{symbol.replace("/", "_")}_data.csv', index=False)
    print(f"Data saved to {symbol.replace('/', '_')}_data.csv")
    return data
```

In above code snippet, there's a function as fetch_crypto_data. It has following arguments.

- **symbol** - The cryptocurrency pair you want to analyze (e.g., "BTC/USDT").
- **exchange_name -** The name of the cryptocurrency exchange (e.g., "binance").
- **timeframe**: The time interval for each data point (default is "1d" = 1 day)
- **limit:** The number of data points to fetch (default is 30)

In the first line of the function, there is a variable as exchange which dynamically creates an instance of the specified cryptocurrency exchange (e.g., Binance) using the ccxt library. Variable bar fetches historical price data (OHLCV: Open, High, Low, Close, Volume) for the given cryptocurrency pair and time interval from the exchange.

Variable data converts the fetched data into a Pandas DataFrame and assigns column names like timestamp, open, high, low, close, and volume to the respective data. In the next line converts the timestamp column from milliseconds (since Unix epoch) to a human-readable date and time format.

After that saves the DataFrame to a CSV file, replacing / in the symbol (e.g., "BTC/USDT") with _ for a valid file name (e.g., "BTC_USDT_data.csv"), and excludes row numbers from the file. After saving csv file, next line prints a confirmation message indicating that the data has been successfully saved to the specified CSV file. After that it returns the DataFrame containing the fetched cryptocurrency data so it can be used later in the program.

Load CSV File

```
def load_data_from_csv(file_name):
    # Load cryptocurrency data from CSV file.
    data = pd.read_csv(file_name)
    data['timestamp'] = pd.to_datetime(data['timestamp'])
    return data
```

In above code snippet, defines a function named **load_data_from_csv** that takes one parameter, **file_name**, which is the name of the CSV file to be loaded. In the first line of the function, variable data reads the CSV file specified by **file_name** into a Pandas DataFrame, allowing for easy manipulation and analysis of the data.

In the next line, it converts the timestamp column in the DataFrame into a datetime format for easier handling of date and time-related operations. Finally the function returns the DataFrame containing the data loaded from the CSV file, now with the timestamp column formatted as datetime objects.

Analysis and Findings

Calculate Volatility

```
def calculate_volatility(prices):
    # Calculate volatility as the standard deviation of returns using numpy.
    returns = np.diff(prices) / prices[:-1]
    volatility = np.std(returns)
    return volatility
```

In the provided image, it defines a function named **calculate_volatility** that takes one parameter, prices, which is a list or array of price values. In the first line, it calculates the percentage returns for consecutive prices by dividing the difference between each price and its previous value (**np.diff(prices)**) by the previous price (**prices[:-1]**).

After that variable **volatility** computes the standard deviation of the calculated returns using **np.std**, which serves as a measure of price variability or volatility. Finally, the function returns the calculated volatility value, representing the statistical measure of the degree to which prices vary over time.

Calculate Volatility

```
def visualize_trends(data, symbol):
    # Visualize price trends with dates on the x-axis and smaller fonts.
    plt.figure(figsize=(12, 6))
    plt.plot(data['timestamp'], data['close'], label=f'{symbol} Price', color='blue')
    plt.title(f'{symbol} Price Trend', fontsize=12)
    plt.xlabel('Date', fontsize=10)
    plt.ylabel('Price', fontsize=10)
    plt.xticks(fontsize=8, rotation=45)
    plt.yticks(fontsize=8)
    plt.legend(fontsize=10)
    plt.grid()
    plt.tight_layout()
    plt.show()
```

In the provided code, there is a function named **visualize_trends** and it has following arguments

- **data -** A DataFrame containing price data
- **symbol -** The cryptocurrency trading pair (ex - "BTC/USDT")

This function is all about creating a clear and easy-to-read line chart that shows how the price of a cryptocurrency has changed over time. It starts by setting up a figure with a specific size to ensure the chart looks good. Then, it plots the closing prices of the cryptocurrency against their dates, with a blue line and a label to make it clear what the chart represents. The title of the chart shows the cryptocurrency pair, and labels for the x-axis (dates) and y-axis (prices) make it easy to understand the data.

To keep things neat, it adjusts the size of the text for the axis labels and tick marks, rotating the date labels slightly so they don't overlap. A legend is added to explain the line, and a grid is included to make it easier to follow the trend. Finally, it ensures everything is spaced nicely and displays the chart for the user to see the price trend.

Conclusion

After all the process is done, the application provides a histogram of price trend for a given cryptocurrency pair.

Sample Output