Polynomial Coin Trend

This program implements polynomial calculation using the method of least squares. It fetches price data of a cryptocurrency from Binance and determines the most appropriate polynomial with the least error. Here's how it works:

The program begins by prompting the user to input details such as the name of the cryptocurrency, the time interval for data collection, the number of data points, and the desired decimal precision. Then, it utilizes the `getdata` function to retrieve price data for the specified cryptocurrency from the Binance API within the given time frame. These data points are sampled according to the specified number, and the average price is calculated for each interval.

Next, the program computes polynomials of various degrees to determine the degree that best fits the data. It calculates the error percentage between the actual price data and the polynomial for each degree. The polynomial with the lowest error percentage is then identified.

After computing the polynomial, the program generates a graphical representation. It plots the real price data in blue and the predicted prices by the polynomial in orange. Additionally, it displays the polynomial equation along with its coefficients, considering the specified decimal precision.

Finally, the program concludes by presenting the degree of the polynomial with the least error and the associated error percentage. This allows users to assess how well the polynomial model fits the actual data. Overall, the code offers a systematic approach to analyze cryptocurrency price data and derive meaningful insights through polynomial modeling.

How to use this program?

Here's an explanation of the four inputs you provide to the program and how they are used:

1. Coin Name:

- This input specifies which cryptocurrency the program will analyze.

- The user is expected to enter the name of the cryptocurrency (e.g., "BTCUSDT" for Bitcoin, "ETHUSDT" for Ethereum, etc.).

- Any cryptocurrency name listed on the Binance exchange can be used.

- Coin list: https://www.binance.com/en/markets/spot_margin-USDT

2. Decimal Tolerance:

- This input determines how many decimal places will be used to represent the polynomial coefficients.

- For example, if you input 2, the coefficients will be represented with two decimal places.

3. Graphic Time:

- This input defines the time interval covered by the graph.

- Users specify the time interval for each data point represented on the graph. For instance, "1s" for 1 second, "1m" for 1 minute, "1h" for 1 hour, and so on.

- (1s,1m,3m,5m,15m,30m,1h,2h,4h,6h,8h,12h,1d,3d,1w,1M)

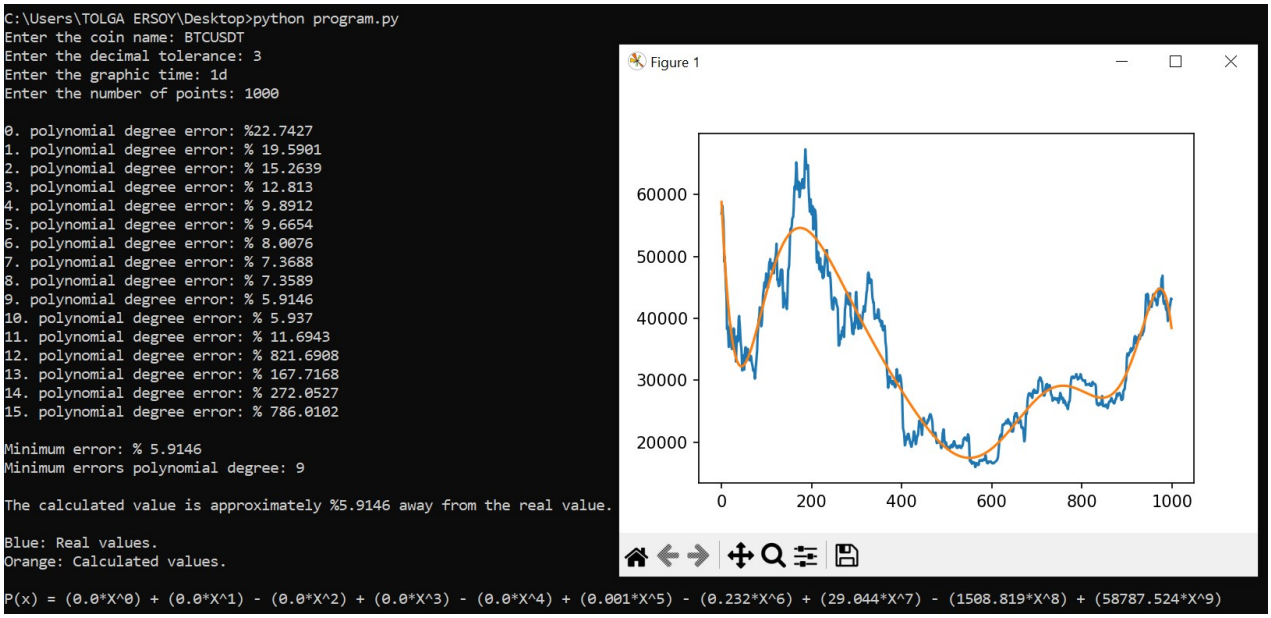
4. Number of Points:

- This input determines the number of data points shown on the graph.

- More data points on the graph provide more detailed analysis but may require more computation.

- Users need to specify how many data points will be used to create the graph.

These four inputs allow users to specify the cryptocurrency to be analyzed, the mathematical precision used in the analysis, the time interval represented on the graph, and the number of data points used for analysis. Once users define these parameters, the program generates the graph and performs the analysis based on the inputs provided.



All Functions

1. calculate(array, x):

- Computes the value of a polynomial for a given x value using the provided coefficients array.

2. coefficients(X, Y, n):

- Calculates the coefficients of the best-fitting polynomial using the method of least squares based on given X and Y values and the polynomial degree.

3. control(coin, time, limit):

- Checks if the required number of data points is available for a specific cryptocurrency within a given time frame from the Binance API.

4. equation1(array, n):

- Constructs a matrix equation needed for polynomial coefficient calculation based on the polynomial coefficients array.

5. equation2(X, Y, n):

- Constructs another equation required for computing polynomial coefficients using the given X and Y values.

6. errorcalculate(Y, Y1, p):

- Calculates the percentage error between real and predicted values based on the difference between them.

7. getdata(coin, time, limit):

- Fetches price data of a specific cryptocurrency from the Binance API within a specified time frame and data point limit.

8. graphic(X, Y, n, p, f):

- Generates a graph plotting real price data and predicted prices by the polynomial for given data points and polynomial degree.

9. minerrorcalculate(X, Y):

- Determines the polynomial degree with the least error for given price data by computing errors for different polynomial degrees.

10. multiplypower(X, Y, n):

- Computes the sum of powers of X multiplied by Y raised to the nth power.

11. polynomial(array, f):

- Prints the polynomial equation using the provided coefficients with a specified decimal precision.

12. run(p, c, t, f):

- Initiates the execution of the program by obtaining user inputs and plotting the graph based on the provided parameters.

13. sumpower(array, n):

- Computes the sum of nth powers of elements in the array.

14. main():

- Acts as the main function of the program, prompts the user for inputs, and runs the program.