Quant Club Coding Round Assignment

Table Of Contents

Assignment-1

ANOMALY DETECTION

Introduction

Anomalies or outliers in a time series data are points which deviate more than a tolerable limit relative to some standard metric characterizing the time series data. There are actually several types among the outliers which fit in the scope of this definition.

In this assignment, we will be building a system to detect **"Additive Outliers".** You don't need to worry what we exactly mean by that right now. You can just consider these to be points which are either too high or too low with respect to the mean of the observed data. In this assignment, you will be coding 2 methods to detect these types of points in a given data series.

1. Bollinger Bands Method

If you have followed our posts on the page, this method should be a cakewalk for you.

A. Make a list mean[i] which has the mean of the past n observations where n is a fixed window which you are free to decide. We recommend you to take a

value of 10. Similarly, make another list sd[i] which has the standard deviation of the past n observations.

B. Now pick a threshold, say 2 (you have to change it and find an appropriate threshold). Now, if the value of the data point at a specific time is greater than or less than mean[i] + 2 * sd[i] or mean[i] - 2*sd[i] respectively we call the point at time i as anomaly time.
C. We append this point's index to our anomaly list. Please note that this method has a limitation wrt the first n points since you can't calculate the mean and sd. Handle this case appropriately.
D. For the above exercise you may use the following formulas for mean and standard deviation. Whether you want to make a function for calculating mean and SD or not is your choice.
E. Find the outliers in the given data and plot it using the utility functions provided.

***Mean of* { x1,x2,..... xn }**

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)$$

Standard deviation of { x1,x2,..... xN }

$$s = \sqrt{\frac{\sum_{i=1}^{N} \left(x_i - \bar{x}\right)^2}{N-1}}$$

2. Slope Method

In this method, instead of using the original data series for anomaly detection, we first calculate a new slope series and then subsequently use that for detecting anomalies.

A. Make a list and call it slope. Calculate the slope at each point by using this formula slope[i] = value[i] - value[i-1] and append it to the list.
B. If you think about it, if a value is an outlier, it will cause a huge spike in the above slope series and hence it will be a piece of cake to detect it in the above difference series.
C. Calculate the mean and sd of the entire slope list. Now, we will model the slope series as a Gaussian Distribution. Which in simple terms mean that any

point which is above mean + 3.5 * sd or below mean - 3.5 * sd can be marked as an outlier with > 99.9% confidence (From Z-Distribution in case you are curious).

D. Find the outliers in the given data and plot it using the utility functions provided.

3. Your Own Innovative Method

This section is optional. We would love to see if you can find any creative ways to detect anomalies in the given data.

Language Preferences

1. Python

If you have used python earlier and are familiar with it then we prefer that you use python. You will find a Colab notebook attached. You are required to clone the given drive folder in your drive and code the solution in the colab notebook.

We have provided you some utility functions to help you code easily and not worry about not-so-important details. These functions are available in the quant_club_test_utils.py file which is automatically imported in the notebook so you need not worry about that either.

The provided utility functions include:

1. make_and_save_data_file() to create a datafile
2. read_csv_with_index() to load the datafile
3. plot_anomaly() to plot the given anomaly list

Please refer to the colab notebook for more details.

You are strictly not allowed to use any library. Failing this will lead to the disqualification of your submission.

2. C++

If you are not familiar with python, you can use C++ as well.

1. For C++ we have provided you with a text file named “testfile.txt” which contains the data series in txt format.
2. Do not change the text files name or contents of the text file and have it in the same folder as your solution file.
3. We have provided you with a function which will read the text file “testfile.txt” and return an array of floats. Use that function to load the testfile.
4. You can implement the above 3 parts by making separate functions in the same cpp file.

5. Please print the anomaly indexes to the console in a properly formatted way.

Note : We prefer you choose Python, but in case you are not familiar with it we are fine if you choose C++ as well. We do not have any bias wrt your language preferences.

Assignment-2

MONTE CARLO SIMULATIONS

Introduction-

Monte Carlo (MC) simulations are models that compile thousands to millions of different possibilities using a predetermined ‘random' (changing) variable to estimate the probability of complicated events. It takes a variable and assigns it a random value from a given range, then uses that variable to conduct calculations. After that, it iterates several times before averaging the results received after each iteration. This average answer offers us a rough estimate of our solution to the problem.

Important Concepts-

Understanding the two important terms or theories- **Volatility** and **Normal Distribution** for this task is very much essential before you move on towards the programming aspect of the assignment.

VOLATILITY-

In layman's terms, Volatility is simply the up-down movement of the stock market, however the real explanation is way much broader concept wise. Volatility is a measure of risk and is estimated by the standard deviation. Given the stock's volatility, we can estimate its price range. More the range of the stock, higher are the chances that volatility will be elevated (higher risk).

NORMAL DISTRIBUTION-

The **Normal distribution**, also known as the **Gaussian distribution**, is a symmetric probability distribution centred on the mean, indicating that data around the mean occur more frequently than data far from it. The normal distribution will show as a **bell curve** on a graph.
The normal distribution is the most common type of distribution assumed in technical stock market analysis and in other types of statistical analyses. The standard normal distribution has two parameters: the mean and the standard deviation. For a normal distribution, 68% of the observations are within +/- one standard deviation of the mean, 95% are within +/- two standard deviations, and 99.7% are within +- three standard deviations. Note that all normal distributions are symmetrical, but all symmetrical distributions are not normal.

Some Formulas and Relations:
A random variable X is said to have a normal distribution with mean μ ($-\infty < \mu < \infty$) and variance $\sigma^2 > 0$ if it has density function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(1/2)[(x-\mu)/\sigma]^2}, \qquad -\infty < x < \infty.$$

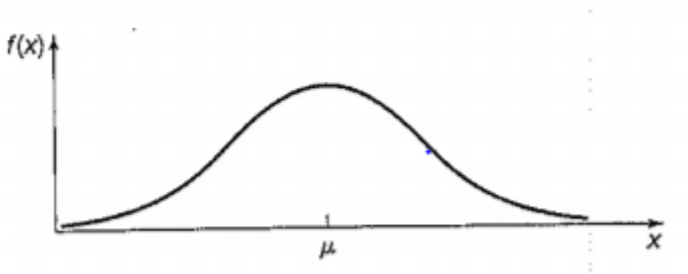
Mean of { x1,x2,..... xn }

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)$$

Standard deviation of { x1,x2,..... xN }

$$s = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N-1}}$$

Standard Deviation (Std) = $(\text{Variance})^{0.5}$



Normal Distribution

These are the basics of Normal Distribution and Volatility and many more things will be needed for you to understand and do the task completely. You are encouraged to find out things on your own which is a very essential skill that we want you to develop. A good starting point would be to look into the relationships between Normal Distribution and Volatility.

Methodology and Steps

1. Four parameters- timesteps, starting_price, volatility, and max_simulations are defined and set to certain values which would remain constant throughout the task and used as inputs.

```
timesteps = 30
starting_price = 600
volatility = 2%
max_simulations = 5000
```

2. Assuming you are at day 0, you need to predict the price of the 30th day starting from the starting_price using Monte Carlo simulations.
3. You need to track the possible movements (or paths) of the price for each simulation upto max_simulations and finally create a two dimensional space of all possible prices (path) encountered.
4. This two dimensional space or matrix of size (max_simulations*(timesteps+1)) needs to be stored and all the different paths simulated are to be plotted.

NOTE - 1. You need to explain all the plots and inferences drawn from the plots.
2. Understanding the above concepts explained is very much vital and necessary for completing the task.

Language Preferences

1. Python

There is a Google Colab Notebook attached, you are required to clone the given drive folder in your drive and bring forth your solution by coding in the provided Colab notebook. (i.e., notebook provided is the solution template)
Important and necessary utility functions are provided by our team to help you to code easily without worrying about the not-so-important details.
The provided utility functions include:

1. plot_paths() to plot all the paths
2. plot_final_dist() to plot the final distribution
3. plot_returns() to plot the expected returns

Please refer to the colab notebook for more details.

You can only use the numpy random module(already imported in the notebook) for sampling from a normal distribution and any other library or modules are not strictly not allowed

You are strictly not allowed to use any library (except for random modules in numpy which are already imported in the notebook). Failing this will lead to the disqualification of your submission.

2. C++

If you are not familiar with python, you can use C++ as well.

1. For C++, you have a pre-structured and constructed template named MonteCarlo.cpp. **Download and code in there**.
2. Parameters and variables are defined in the file, you need to use them to carry out the task.
3. It is advisable that you do not make modifications to the variables, parameters, and their set values.
4. The file contains an empty 2D array of float type named path[max_simulations][timesteps+1], which will store all your predicted paths and possibilities.
5. There is a section provided for you to program, you need to code your solution following all steps discussed above and finally store your simulated paths in the path[max_simulations][timesteps+1] array
6. Finally, you need to convert this 2D array into a text file, for which the code has already been provided to you.
7. **Upload the paths.txt file in the C++ folder** in your drive and **run the C++Monte_carlo_task notebook** for plotting(you don't have to code anything here, just run the cells)

Note : We prefer you choose Python, but in case you are not familiar with it we are fine if you choose C++ as well. We do not have any bias wrt your language preferences.

In case of any doubts related to the assignments, you may contact us at:

7076210444 (Ishan Goel)
9547773307 (Rahul Gorai)