**Requirements – Data Science – Python/ML Software Developer**

1. **Python Programming Skill [Must]**

* Strong understanding of Python programming fundamentals and basic data science modules – NumPy, Pandas, and SciKit-Learn
* Exposure to data visualization libraries such as Matplotlib and Seaborn
* Experience using Keras/Tensorflow/PyTorch
* Experience in performance optimization

1. **Understanding of Machine Learning Concepts [Desirable]**

* Data pre-processing and normalization
* Feature selection and dimensionality reduction
* Unsupervised learning – parameter optimization; strengths and weaknesses of various algorithms
* Supervised learning – parameter optimization; strengths and weakness of various algorithms

1. **Knowledge of Deep Learning Algorithms [Optional]**

* Conceptual understanding of convolutional neural networks and/or recurrent neural networks
* Conceptual understanding of representation learning, *e.g.* word/token/sentence embeddings
* Exposure to image analysis algorithms
* Exposure to NLP algorithms, *e.g.* single-sentence sentiment analysis (VADER, TextBlob, *etc.*), aspect-based sentiment analysis, and document summarization

**Python Programming Assignment (Phase – 1)**

**Pre-requisites**

Developers should have Python libraries pre-installed on their laptops – NumPy, Pandas, Matplotlib, Seaborn, ScikitLearn, Keras/TensorFlow/PyTorch

**Evaluation criteria**

* Use of appropriate Python libraries
* Execution time and memory requirement of the final code

1. **Generalized Hurst Exponent with exponential smoothing:** Implement the algorithm and generate the output using the four input data sets (GHE\_1.txt, …, GHE\_4.txt)

**Formulae**

**[1]**

, where S is a constant term  **[2]**

**Algorithm to be implemented**

* 1. Read the input data set (*e.g.,* GHE\_1.txt) and obtain a vector **S**, *i.e.* an array of size 1 × 1000
  2. Set θ = 100 and q = 2
  3. For T\_max = 1, …, 19 [both inclusive]
     1. Set T = 1000 – T\_max
     2. Estimate , where
     3. Estimate using equation (1) for t = 0, 1, …, (T – 1) [both inclusive]
     4. For t = 0, …, (T-1) [both inclusive]
        1. Obtain Nt = absolute value for S[t + T\_max] – S[t]
        2. Obtain Dt = absolute value of S[t]
        3. Obtain and for q = 2
  4. For J = 5, …, 19 [both inclusive]
     1. Select the first-J values of K and T\_max. Denote this set of values as and TJ, respectively
     2. Perform linear regression using the formula , where y values are **ln(KJ)** values, x values are **ln(TJ)** values, and **ln** denotes the natural logarithm
     3. Obtain **b** *i.e.,* slope of the linear regression line
  5. Obtain b\_avg *i.e.,* average value for **b**
  6. Return the value b\_avg/q

1. **Non-Parametric Moving Block Bootstrap**

**Input data**: Tab-separated file comprising daily stock market prices for 100 assets (Input\_Data\_Non\_Param\_BootStrap.txt)

**Expected output**

* An output matrix of size 1000 × 100, where 1000 denotes the number of bootstraps and 100 denotes the number of assets
* Symbols for top-5 assets
* Boxplots for the top-5 assets
* Execution time

**Algorithm to be implemented**

* 1. Read the input file and generate a matrix **S** of size 100 × 1500. The first row of the input file corresponds to the header and the first column contains the symbols
  2. Use random.seed(5000) and randint
  3. Randomly generate B = 16 blocks of 30 contiguous days from **S**. That is, generate B submatrices of size 100 × 30 such that the start of each block is selected at random with replacement
  4. For each block b (where b = 1, 2, …, B) estimate the vector (*i.e.*, 1 × 100 array) of asset-wise returns as follows –
     1. For each of the 100 assets, calculate log\_return = log(S\_last) – log(S\_first), where S\_last and S\_first denote the asset’s stock price on the first and last day, respectively, in block b
  5. Concatenate the **B** blocks to create a matrix of size B × 100 (B blocks and 100 log-returns per block)
  6. Obtain the column sums and generate the matrix **M** = 1 × 100. Next, exponentiate the matrix and subtract one from each element. In other words, for each element m\_i (where i = 1,2,…100) of **M**, obtain exp(m\_i) – 1. Denote the new matrix as **M\_REV**
  7. Repeat steps 2 – 5 1000 times and obtain the final matrix **M\_Total** by concatenating the **M\_REV** matrices so that **M\_Total** has size 1000 × 100
  8. Use **M\_Total** to estimate the mean value per asset and identify the top-5 assets. Next, generate the boxplots of returns for each of the top-5 assets