

IME 611A Financial Engineering, IIT Kanpur

Course Project A

Akash Sonowal (20114003); Sk Raju (20114021)

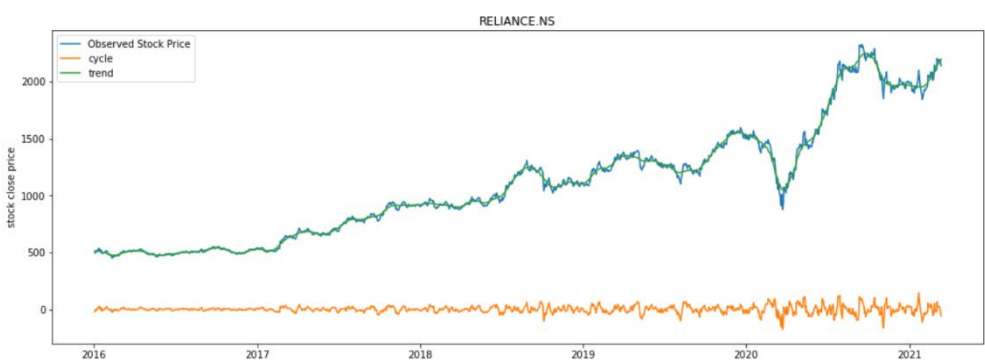
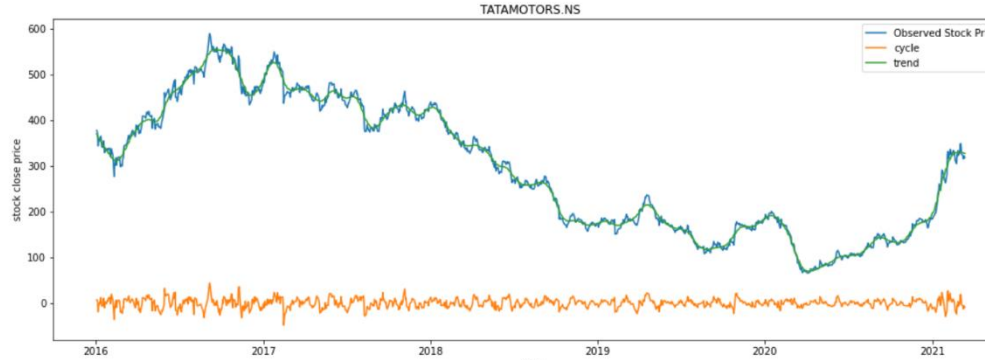
Group 15 (MTech IME, 1st year)

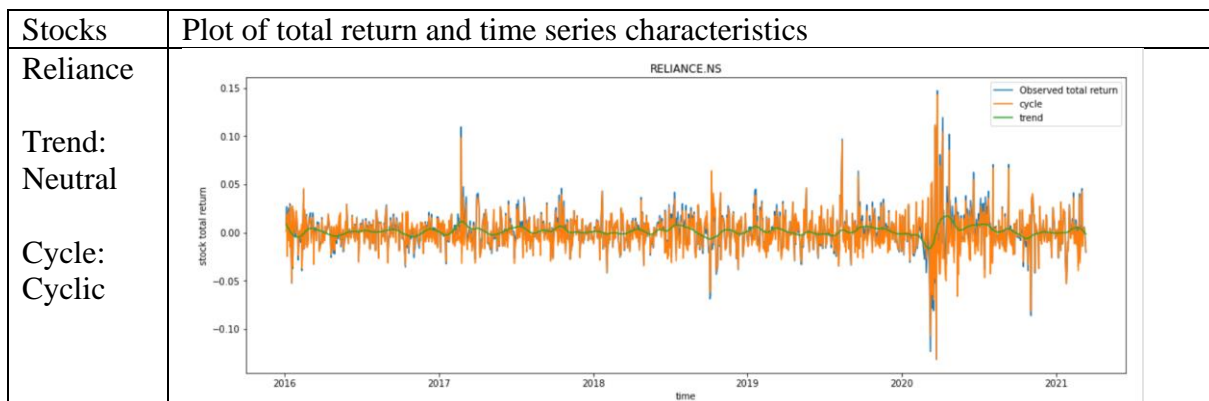
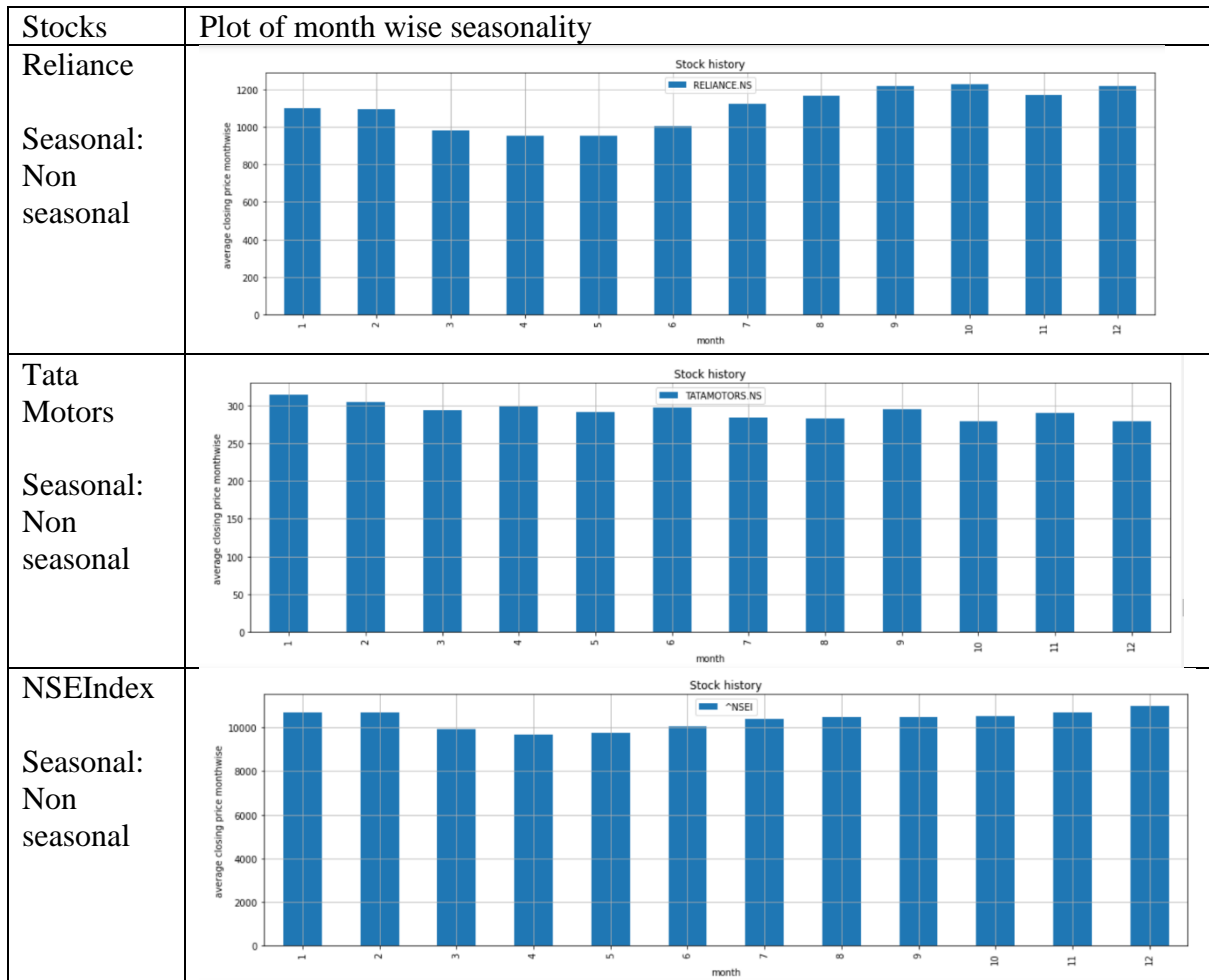
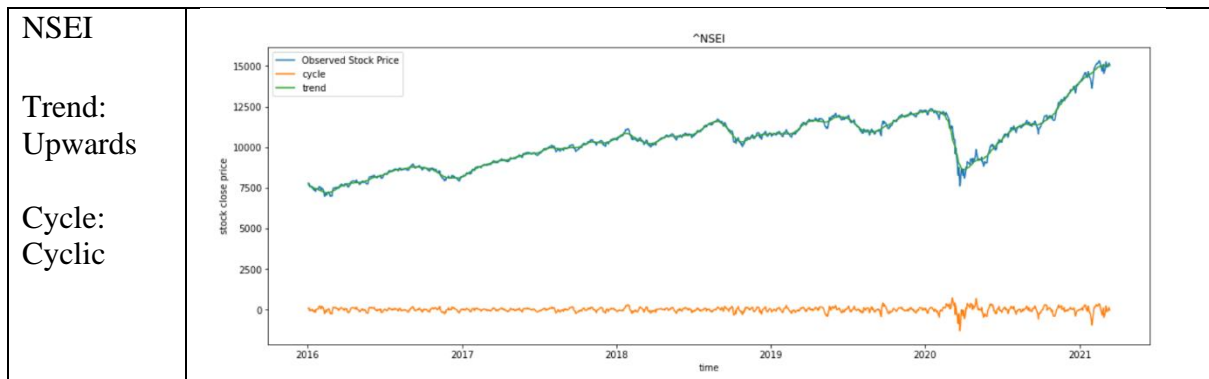
Task 1: Plot the daily stock price of each of these companies. Present the time-series characteristics of these prices. Construct total return and log total return series. Plot the return series and present their time-series characteristics.

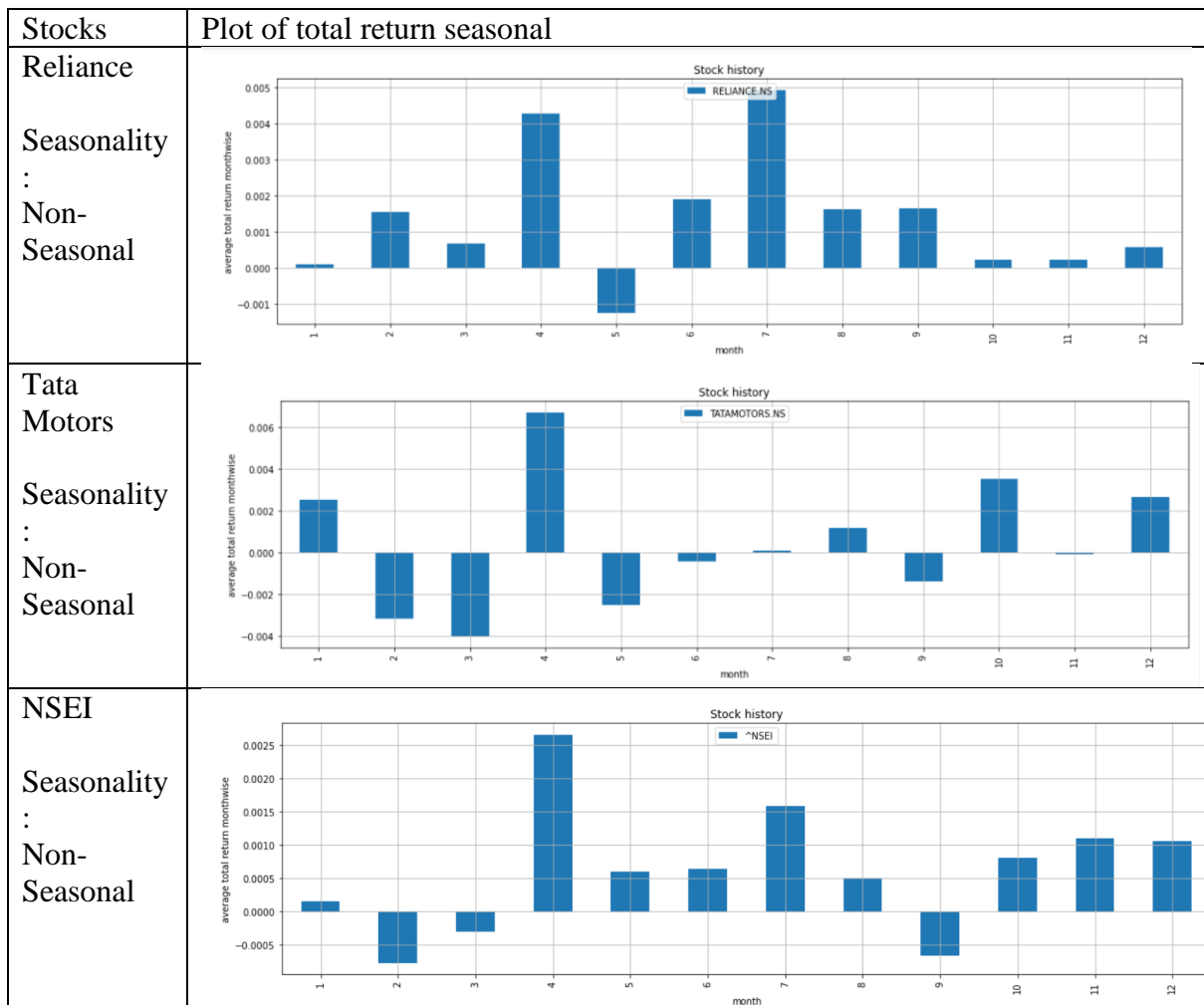
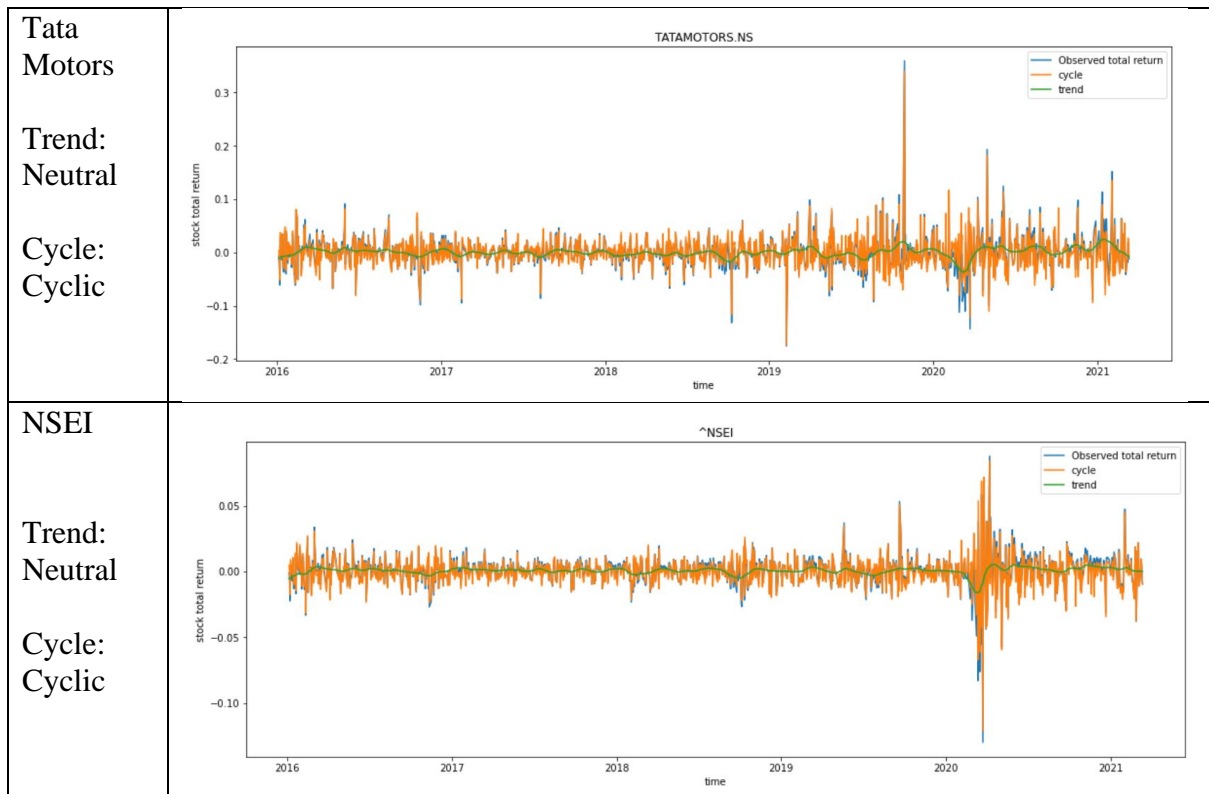
We have plotted the daily time series of the Nifty 50 index and each of the 100 stocks in Nifty 100. We observed the time series characteristics in terms of trend, cycle and seasonality.

- **Trend:** The increasing or decreasing value in the series.
- **Seasonality:** The repeating short-term cycle in the series.
- **Cycle:** Exhibition of rises and falls that are not of the fixed period.

In the Jupyter Notebook, we have visualised all the stocks and the NIFTY50 index. However, due to limited space and better representation, we have picked stocks that exhibit different volatility characteristics to cover the kind of stocks available in the market.

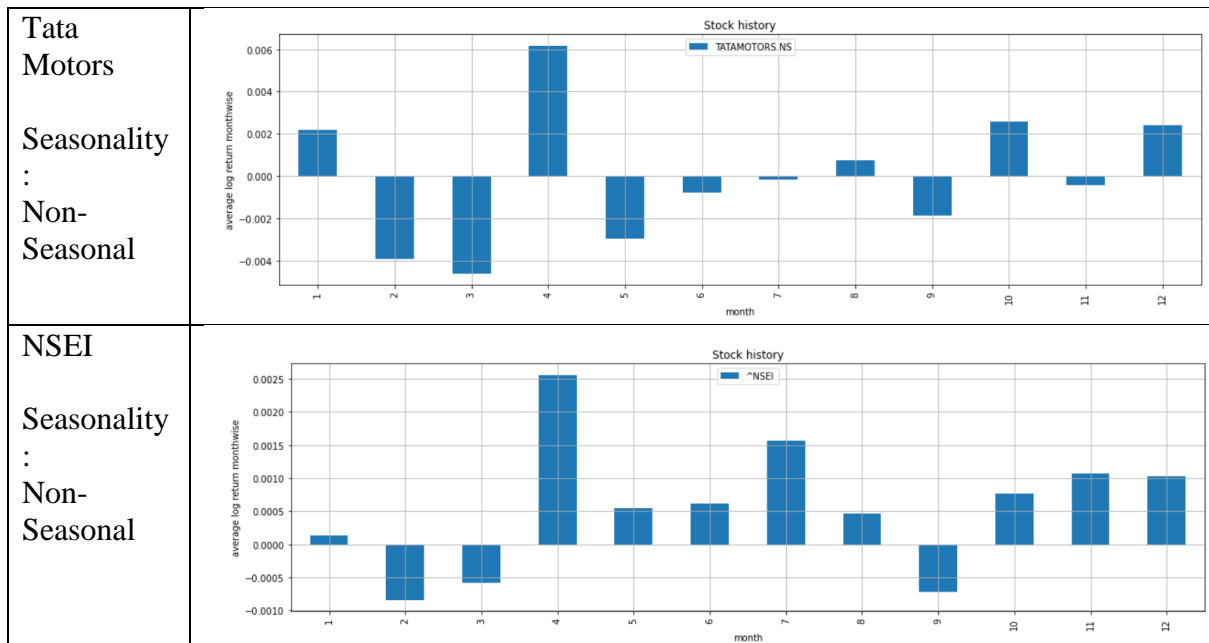
Stocks	Plot of Daily Stock Price & time-series characteristics
Reliance Trend: Upwards Cycle: Cyclic	
Tata Motors Trend: Downwards Cycle: Cyclic	





Stocks	Plot of log total return and time series characteristics
Reliance Trend: Neutral Cycle: Cyclic	
Tata Motors Trend: Neutral Cycle: Cyclic	
NSEI Trend: Neutral Cycle: Cyclic	

Stocks	Plot of log total return seasonal
Reliance Seasonality : Non-Seasonal	



Task 2: Check the distribution of these returns. Present test statistics whether the returns are following a normal distribution. If not, then fit other distribution types which fit the data better.

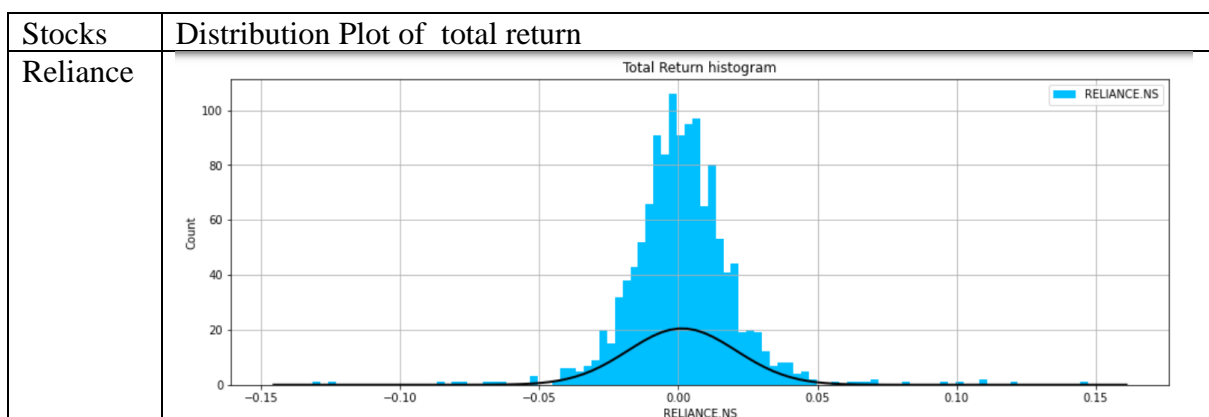
Total returns and log returns were calculated using the following formulas:

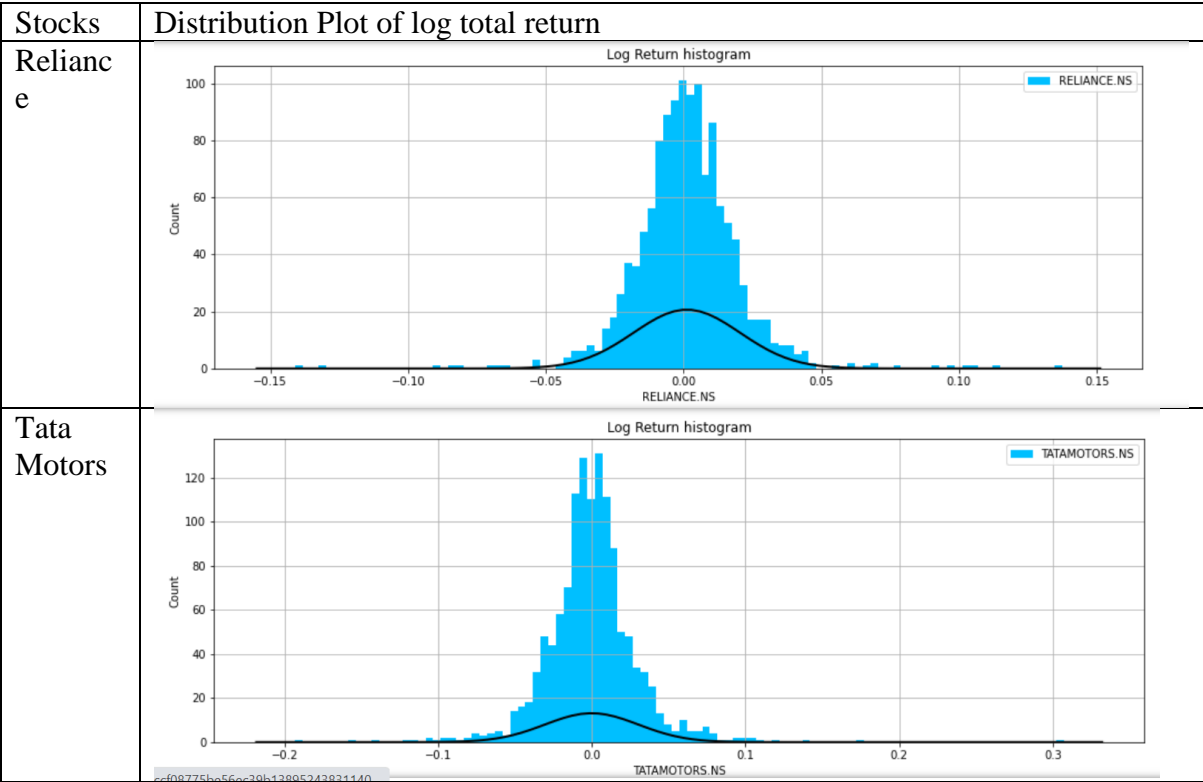
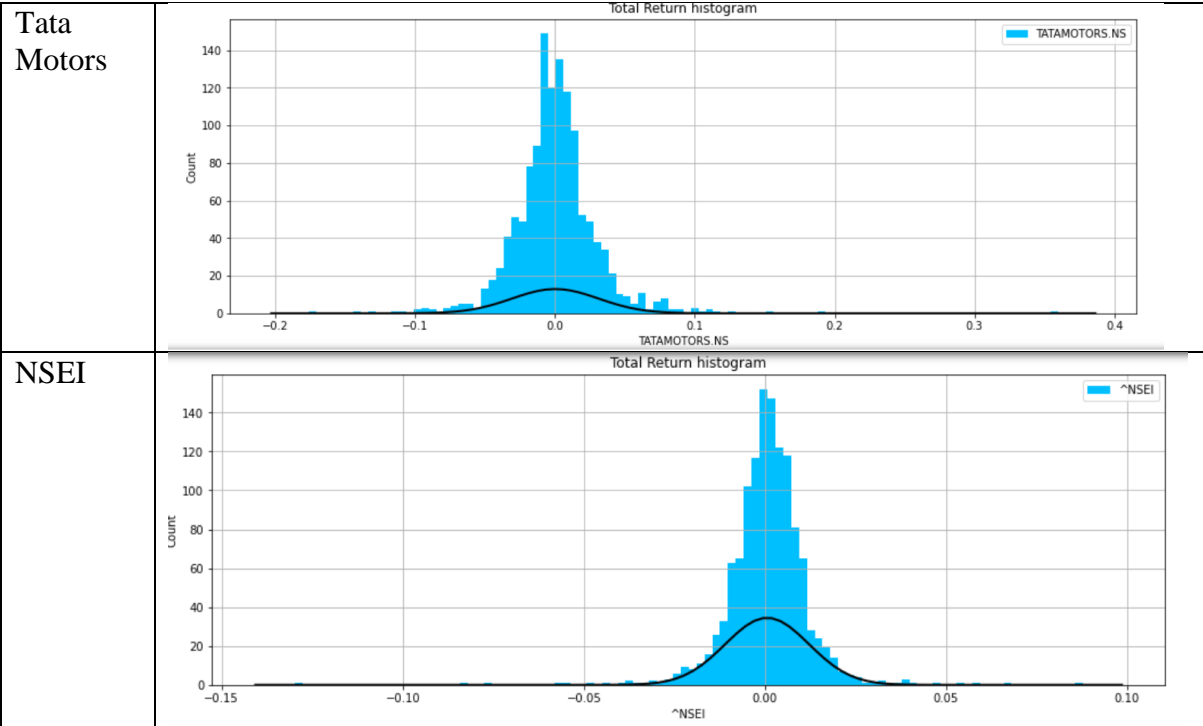
$$\text{Total Return} = \frac{P(t)}{P(t-1)} - 1$$

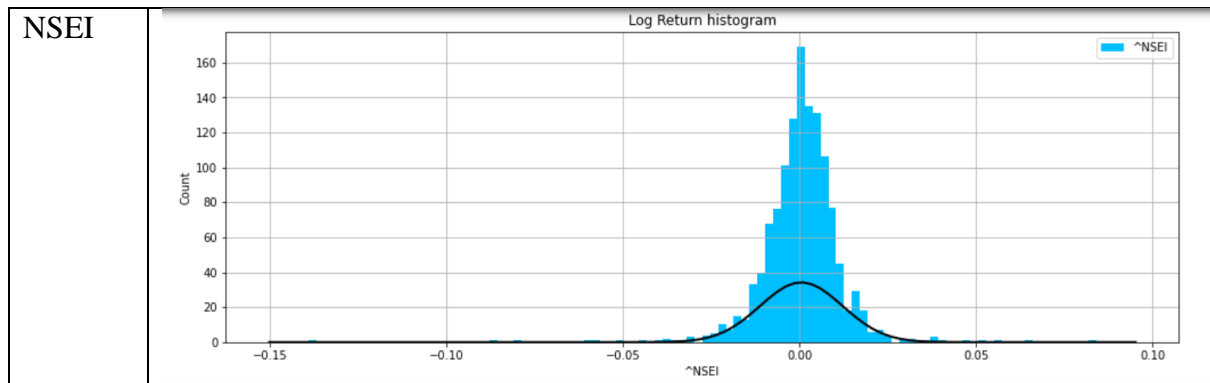
$$\text{Log Return} = \ln\left(\frac{P(t)}{P(t-1)}\right)$$

where $P(t)$ is the stock price at the time t

To check our return distribution, we have plotted histograms and plotted a normal distribution curve to justify our claim. Moreover, we have also done Hypothesis testing to reinforce our claims further. In our results, we found no returns to follow the normal distribution.

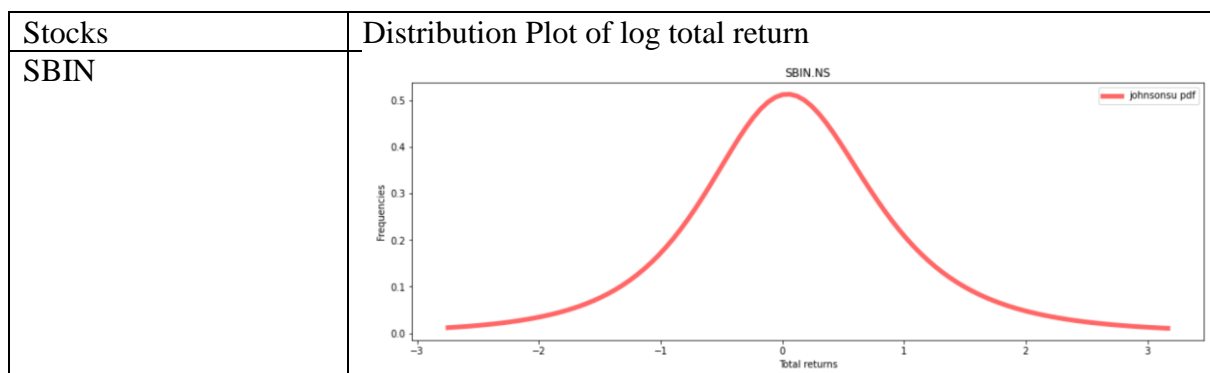


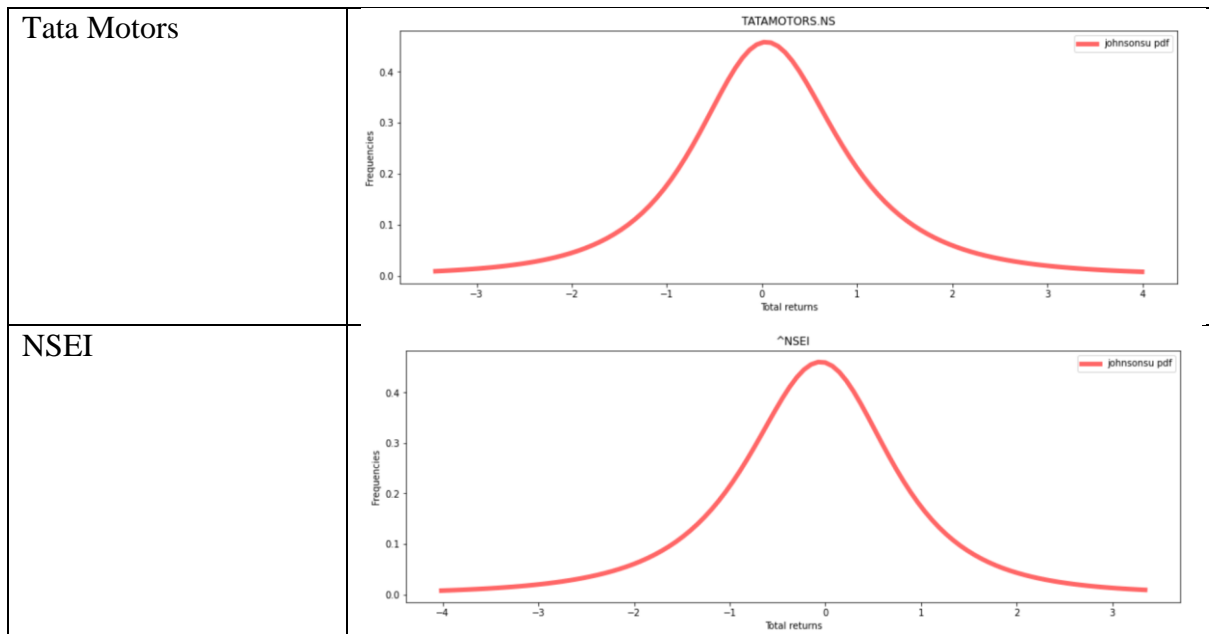




We have done a hypothesis test against 85 distributions available in the python stats library to identify the actual distribution. We have observed from the statistics that “johnsonsu” best fits our data with a p-value of around 0.94 and above for the four stocks (TATAMOTORS.NS', 'SBIN.NS', 'SIEMENS.NS', 'HINDUNILVR.NS') and also for ^NSEI.

Stocks	Distribution Plot of log total return
TATAMOTOR.NS	nct: statistic=0.011373797278837472, pvalue=0.995939263809214 johnsonsu: statistic=0.01259300156698917, pvalue=0.9860423141072306 johnsonsu: statistic=0.01428523823942529, pvalue=0.9537918484420916 johnsonsu: statistic=0.014438873309252531, pvalue=0.949584168446714 nct: statistic=0.01500932965162749, pvalue=0.9320229141018543 t: statistic=0.015512254498664335, pvalue=0.9140524833800687 nct: statistic=0.01556807897174306, pvalue=0.9119183137268368 hypsecant: statistic=0.015808407819124293, pvalue=0.9024222856298189 gennorm: statistic=0.018195725112582772, pvalue=0.7853038519843655 loglaplace: statistic=0.02459894616272637, pvalue=0.4166394112595646 recipinvgauss: statistic=nan, pvalue=nan johnsonsu: statistic=0.012405693759622194, pvalue=0.988168735883823 t: statistic=0.015514603040540753, pvalue=0.9139632496783142 t: statistic=0.016575054087927588, pvalue=0.8689712553850848 nct: statistic=0.01769497308091955, pvalue=0.8126996071546447 tukeylambda: statistic=0.01784252516431928, pvalue=0.8047487325315015 fisk: statistic=0.018609407008113976, pvalue=0.7618863639805111
SBIN.NS	johnsonsu: statistic=0.012987477965564875, pvalue=0.9807041549476978 tukeylambda: statistic=0.013161190439440196, pvalue=0.977959618267108 johnsonsu: statistic=0.013876172149878407, pvalue=0.963917730021452 nct: statistic=0.015327113958971017, pvalue=0.9209331832050034 t: statistic=0.015572716402489029, pvalue=0.9117397974975385 nct: statistic=0.017554920855783918, pvalue=0.8201437089048312 t: statistic=0.018045534106629846, pvalue=0.7936399277719212 hypsecant: statistic=0.019118131811017625, pvalue=0.7323356689673453 dweibull: statistic=0.019725259196977984, pvalue=0.6963117059616124 loglaplace: statistic=0.020221158253142635, pvalue=0.6665403710661546 laplace: statistic=0.020259103878842222, pvalue=0.664255959099985 dgamma: statistic=0.020435166846686836, pvalue=0.653651882565832 gennorm: statistic=0.021053587476922764, pvalue=0.6164456568304364 gennorm: statistic=0.021609645123744525, pvalue=0.5832484641826343





Task 3: Assuming the data to be following a normal distribution, estimate the returns, variances, standard deviation, variance-covariance matrix.

Assuming a normal distribution of our data, we have calculated the annual mean and standard deviation of returns for all the stocks using the mean daily returns.

Also, we have calculated the covariance-variance matrix, which is a pairwise NxN matrix (N=total no of stocks)

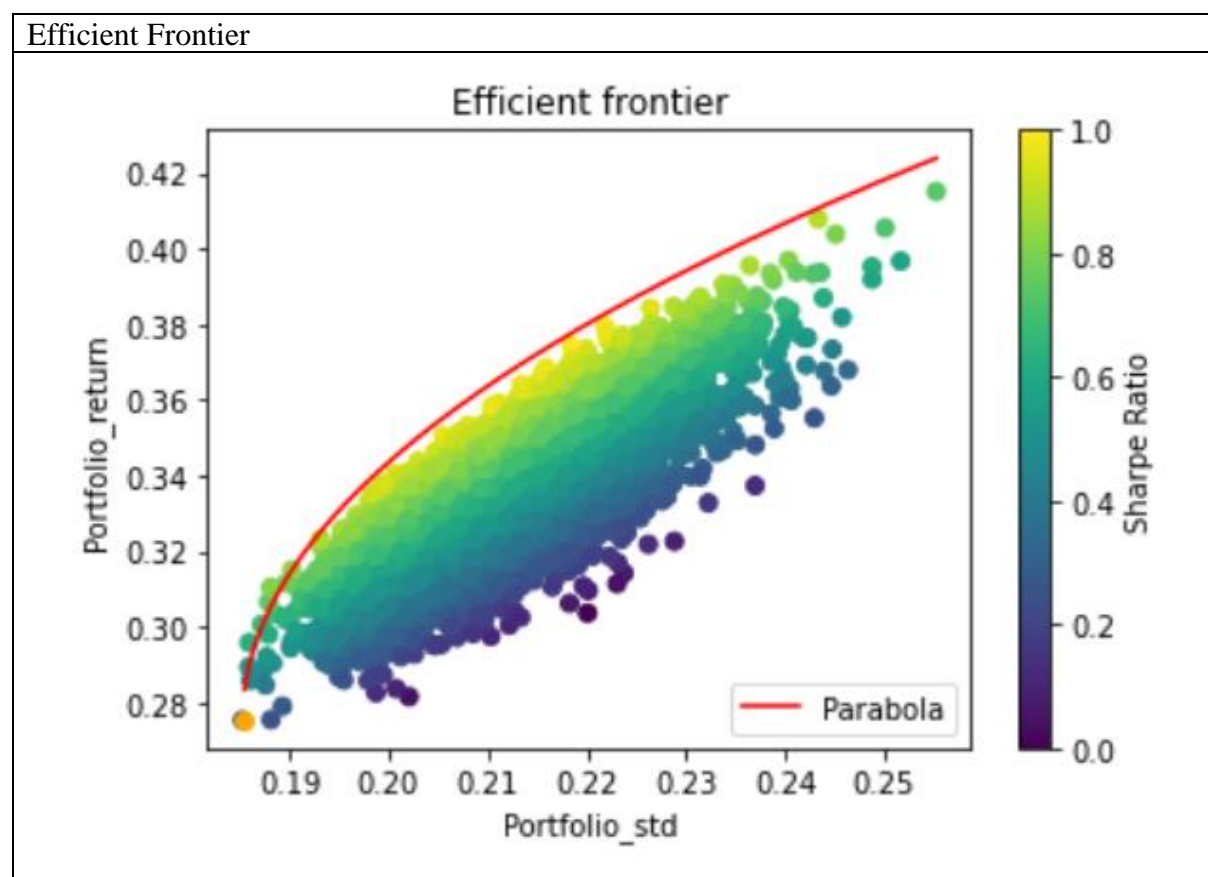
Analysis Assuming Normal Return							
	ABBOTINDIA.NS	ACC.NS	ADANIENT.NS	ADANIPTS.NS	ADANITRANS.NS	ALKEM.NS	AMBUJACEM.NS
mean	0.213027	0.097790	0.717851	0.272800	0.718897	0.154937	0.116335
std	0.265004	0.288383	0.518047	0.366397	0.505339	0.274448	0.301510
var	0.070227	0.083165	0.268373	0.134246	0.255368	0.075322	0.090908
68% confidence_return_lower_range	-0.051977	-0.190593	0.199803	-0.093597	0.213558	-0.119511	-0.185174
68% confidence_return_upper_range	0.478031	0.386173	1.235898	0.639197	1.224237	0.429386	0.417845
95% confidence_return_lower_range	-0.316981	-0.478976	-0.318244	-0.459993	-0.291781	-0.393960	-0.486684
95% confidence_return_upper_range	0.743035	0.674557	1.753946	1.005593	1.729576	0.703834	0.719355

7 rows × 90 columns

Task 4: Construct a Markowitz portfolio and plot the efficient frontier. Prove that the portfolio frontier will have a parabolic structure.

For 15 stocks in our portfolio, we have used stocks with a higher Sharpe ratio (Note: The ideal approach for selecting stocks should be based on covariance and correlation, however considering the size of our portfolio, it was challenging to pick 15 stocks based on covariance matrix as it allowed for only higher covariance)

For calculating our expected portfolio return, we have used annual log return only. Then We plotted the Markowitz portfolio with random weights assigned to stocks simulated for 7000 iterations to manifest a better parabolic shape. Also, to confirm our parabolic shape, we have used a parabola equation with the vertex of minimum risk.



Task 5: Use a risk-free rate of 5% and plot the Security Market Line (SML).
Demonstrate a few underpriced and overpriced securities.

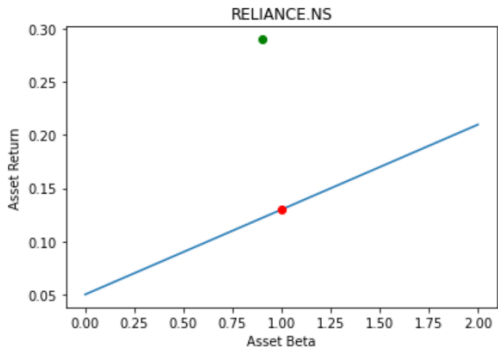
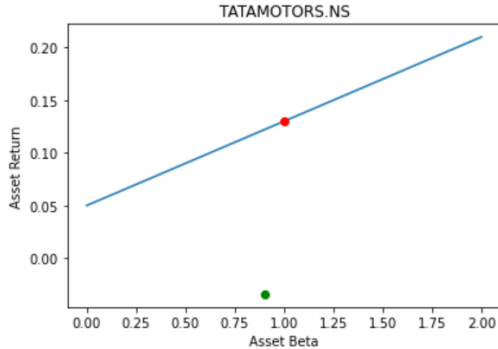
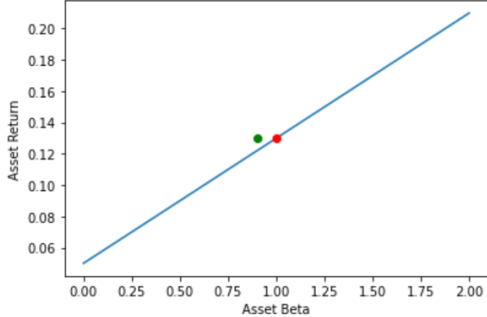
Firstly in our analysis, we have considered our market as the portfolio of all the assets (90 stocks taken in our data) for comparing our stocks with a market. Then we have constructed the Security Market Line (SML) with a 5% risk-free rate by varying β and using the equation $r_i - r_f = \beta_i (r_m - r_f)$, where $\beta = \frac{Cov(r_i, r_m)}{Var(r_m)}$

We then calculated β for each stock which then is used to find the annual log-return of each stock. Finally, after plotting, we found inferences of overpriced stocks (if the point lies above the SML) and underpriced stocks (if it lies below the SML).

Stocks	Security Market Line , Overpriced/Underpriced
Reliance Overpriced	
Tata Motors Underpriced	
NSEI Slightly Underpriced	

Task 6: Read about at least one other approach to estimate variance-covariance matrix and implement the same.

In Task 6, all steps were similar to step 5 except the fact that here we have constructed the market to be the Nifty50 Index.

Stocks	Security Market Line, Overpriced/Underpriced
Reliance Overpriced	
Tata Motors UnderPriced	
NSEI Slightly Overpriced	

Task 7: For the actual return distribution that you have observed, identify the suitable return, and risk measure and highlight how those should be considered in the portfolio weight identification.

We equally distributed the stocks in our portfolio by assigning the same weights to all stocks as 1/15 for 15 stocks (while assuring the sum of weights is 1) and calculated the portfolio return and risk as 0.34 and 0.21, respectively. Then we have distributed the weight as per the

minimum risk criteria of the Markowitz portfolio and found the portfolio return and risk as 0.29 and 0.19, respectively. We have also determined the weights that were assigned for it.

As per our risk appetite, we have chosen a suitable return of 0.36 for which the risk is 0.22 and the Sharpe ratio of 1.61 and also we have determined the weights assigned to our 15 stocks.

```
In [70]: print("Portfolio return with equal distribution of weights: ",portfolio_return(equal_weights))
Portfolio return with equal distribution of weights: 0.3371836491289451

In [71]: print("Portfolio risk with equal distribution of weights: ",portfolio_std(equal_weights))
Portfolio risk with equal distribution of weights: 0.2073820671679175

In [73]: print("Portfolio weights with Markowitz distribution of weights: ",minstd_weights)
Portfolio weights with Markowitz distribution of weights: [0.04259231 0.01337702 0.02581186 0.06516317 0.07151987 0.0410850
6 0.10075082 0.14245872 0.10125561 0.012727 0.12864063 0.14542208
0.04081824 0.01156372 0.0568139 ]

In [74]: print("Portfolio return with Markowitz distribution of weights: ",portfolio_return(minstd_weights))
Portfolio return with Markowitz distribution of weights: 0.29218315435954356

In [75]: print("Portfolio risk with Markowitz distribution of weights: ",portfolio_std(minstd_weights))
Portfolio risk with Markowitz distribution of weights: 0.18607276706618506

In [76]: def find_nearest(array, value):
array = np.asarray(array)
idx = (np.abs(array - value)).argmin()
return array[idx]

In [77]: suitable_return = 0.36

In [79]: print("Risk Measure for suitable return: ",stds[returns.index(find_nearest(returns, suitable_return))])
print("Sharpe ratio for suitable return: ",sr[returns.index(find_nearest(returns, suitable_return))])
print("Weight identification for suitable return:",w[returns.index(find_nearest(returns, suitable_return))])
Risk Measure for suitable return: 0.22388666500193496
Sharpe ratio for suitable return: 1.6079572704026925
Weight identification for suitable return: [0.07280643 0.12123402 0.09268193 0.04004229 0.08172975 0.05660044
0.05712444 0.09680633 0.02424327 0.04417644 0.03209159 0.04696417
0.00941297 0.11648174 0.10760418]
```