CS229 INNOVATION LAB REPORT, IIT PATNA

Stock Portfolio Optimization using LSTM and MPT

By- Shivanshu Sanjeev(1901CS56) and Saurav Sonu(1901MM30)
Under the supervision of Prof. Somanath Tripathy, IIT Patna

Introduction

Stock markets have existed for centuries.Since its inception, stock markets have served many purposes, the most important being to provide companies with a source to raise capital for investment and expansion.For the individual investor, the stock market provides a way to invest your income to earn a share of the companies' profits.And for a risk averse investor the most important thing is to manage a portfolio.
Portfolio management refers to managing an individual's investments in the form of bonds, shares, cash, mutual funds etc so that he earns the maximum profits within the stipulated time frame.Portfolio management **minimizes the risks** involved in investing and also increases the chance of making profits.

So, the goal of this project is to build an **optimized portfolio for investment** in the stock market where the investor will provide the names of certain number stocks and the model will return the best possible combination of shares amongst the top-performing stocks which will be first predicted by the **LSTM model** and then the selected stocks will be optimized using **Markowitz Portfolio Theory**.

Existing Work And Our Innovation

There are numerous classical methods to build a portfolio like Dow Jones theory, Random Walk theory, Formula theory and then the most accurate one is **Markowitz Portfolio theory.**

But since Markowitz portfolio theory considers covariance and correlation between two stocks as a measure of risk, it overlooks the possibility of unwanted and unexpected changes in stock prices such as downside risks.

So to back up the decision of what stocks we should invest in, we first predict the behaviour of stocks with the help of **LSTM**, by doing this we have decreased the risk of choosing the unwanted stocks even further as MPT alone, only decides on the basis previous performance but LSTM will feed in the the predicted future data to the MPT model which makes it more accurate and less risky for investors.

- **How LSTM is Helping make a better stock choice for a higher return in lesser time**

  As accurate as MPT is for portfolio optimization, its dependency on covariance and correlation sometimes tend to overlook the sudden changes in the market price also called downside risk.

  Two portfolios that have the same level of variance and returns are considered equally desirable under modern portfolio theory. One portfolio may have that variance because of frequent small losses. In contrast, the other could have that variance because of rare spectacular declines. Even though the two stocks behaviour might show satisfactory correlation according to MPT, it might result in overall loss in a shorter span of time.

  What LSTM will do is predict such a sudden change and will discard the risky stock well before passing it into the MPT model and not only it will satisfy the diversity of MPT but also takes care of sudden decline in stock value making the model more accurate and useful for investment in a shorter span of time and on a higher frequency.

THEORY

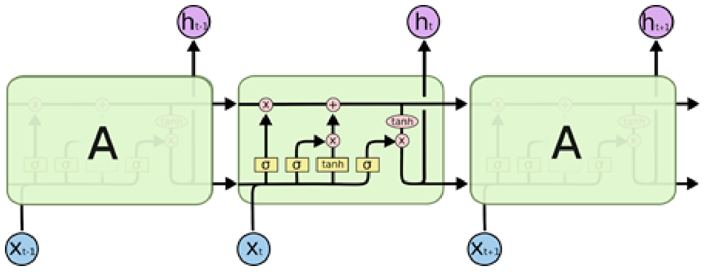
❖ LSTM

The Long short-term memory (LSTM) is a special type of Recurrent Neural Networks (RNN) architecture that is mainly used in deep learning. LSTM uses a set of feedback connections to process sequences of data. This architecture is known for its efficiency in making predictions, processing, and classifying large-scale time-series data despite the presence of some lags between events. It was named long short-term memory because its cell unit has the ability to forget a part of previously stored data and can, at the same time, memorize additional new pieces of information.

An LSTM unit encapsulates the following elements:

- Cell: represents the memory part of the LSTM that monitors the dependencies between different elements constituting the input sequence.
- Input gate: regulates the information flowing into the cell.
- Output gate: regulates the information flowing out of the cell.
- Forget gate: remember the different values over a time interval.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$



❖ **Adam Optimizer**

The type of optimizer used can greatly affect how fast the algorithm converges to the minimum value. Here we have chosen to use Adam optimizer. The Adam optimizer combines the perks of two other optimizers: ADAgrad and RMSprop.

The stochastic gradient descent update for RMS prop is given by:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t$$

where $E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$

Adam(Adaptive Movement Estimation) is another method that computes the adaptive learning rates for each parameter based on its past gradients.

$$v_t = \beta v_{t-1} + (1-\beta)g_t^2$$

$$m_t = \beta m_{t-1} + (1-\beta)g_t$$

where

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

So, the final update is given by $$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

The minimization problem is given by:

$$\underset{f \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} V(f(x_i), y_i) + \gamma ||f||_K^2$$

Another important aspect of training the model is making sure the weights do not get too large, hence, overfit. For this purpose, we chose to do regularization.

❖ MPT

Modern portfolio theory (MPT) is a theory on how risk-averse investors can construct portfolios to maximize expected return based on a given level of market risk. Harry Markowitz pioneered this theory in his paper “Portfolio Selection”.

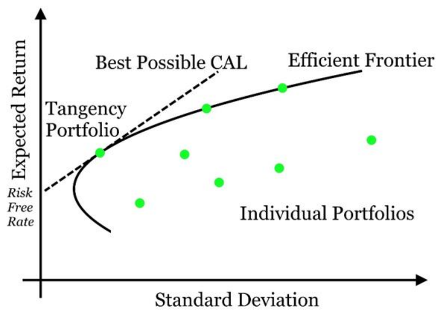
Modern portfolio theory argues that an investment's risk and return characteristics should not be viewed alone, but should be evaluated by how the investment affects the overall portfolio's risk and return. MPT shows that an investor can construct a portfolio of multiple assets that will maximize returns for a given level of risk.

Likewise, given a desired level of expected return, an investor can construct a portfolio with the lowest possible risk. Based on statistical measures such as variance and correlation, an individual investment's performance is less important than how it impacts the entire portfolio. MPT assumes that investors are risk-averse, meaning they prefer a less risky portfolio to a riskier one for a given level of return.

❖ Efficient Frontier Curve and Max Sharpe Ratio

The Efficient Frontier is a common phrase in Modern Finance since the inception of Modern Portfolio Theory in 1952 by Harry Markowitz.

The efficient frontier, a cornerstone of modern portfolio theory, shows the set of portfolios that provide the highest level of return for the lowest level of risk. When a portfolio falls to the right of the efficient frontier, they possess greater risk relative to their return. Conversely, when a portfolio falls beneath the slope of the efficient frontier, they offer a lower level of return relative to risk.



Owning a portfolio with the Maximum Sharpe Ratio is the dream of every investor ("or more likely should be")!!!

Sharpe Ratio - It is the average return earned in excess of the risk-free rate per unit of volatility or total risk. Volatility is a measure of the price fluctuations of an asset or portfolio.

So what is the dream portfolio and how can you make one?

According to Markowitz's theory, there is an optimal portfolio that could be designed with a perfect balance between risk and return.

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

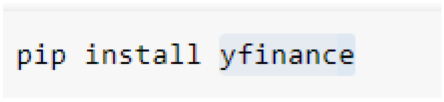
R_p = Return of portfolio

R_f = Risk-Free rate

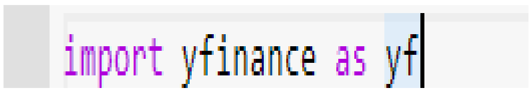
σ_p = Standard deviation of portfolio's excess return

IMPLEMENTATION

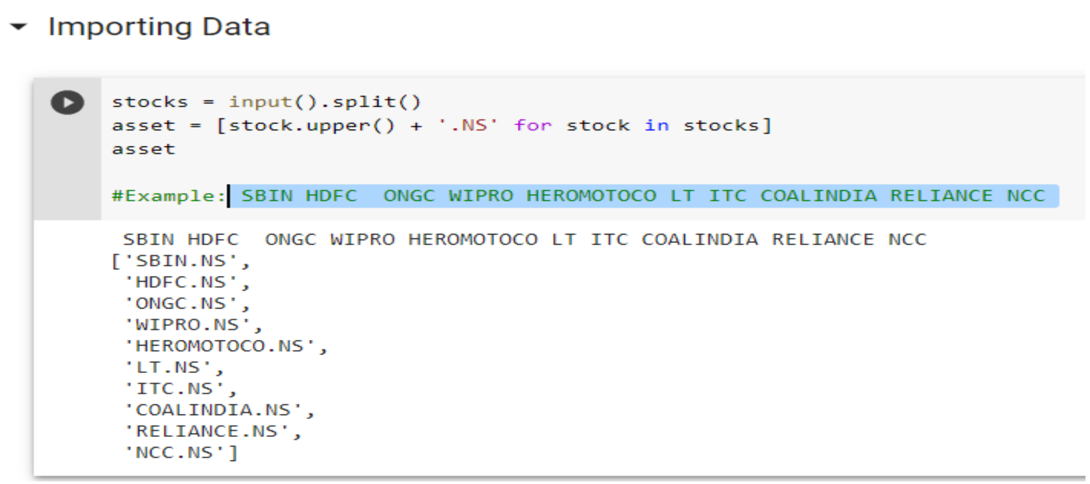
1. First our Goal is to fetch the current stock data using Yahoo Finance API.The fetched data will keep on changing as the market changes because it always gives present day stock values and hence it was chosen instead of reading data from CSV files.



```
pip install yfinance
```



```
import yfinance as yf
```

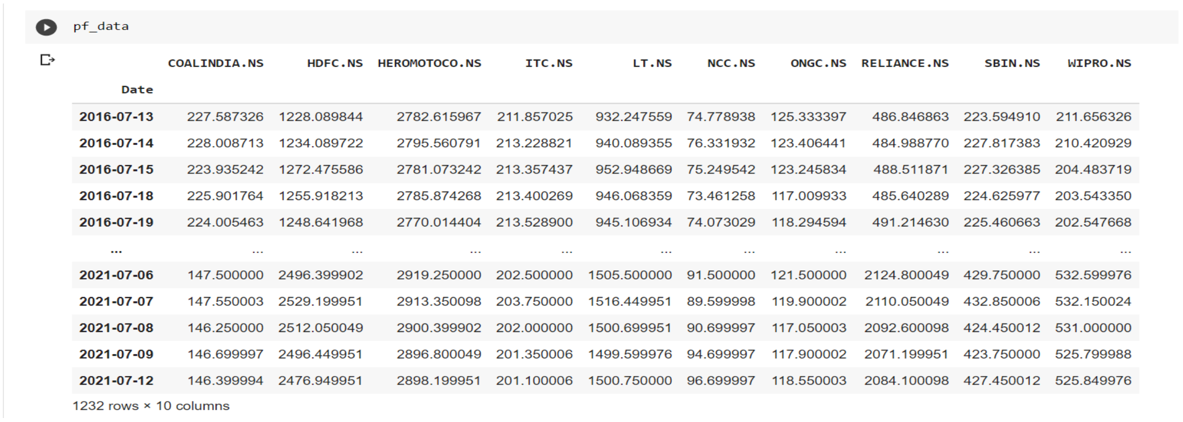


Importing Data

```
stocks = input().split()
asset = [stock.upper() + '.NS' for stock in stocks]
asset

#Example: SBIN HDFC  ONGC WIPRO HEROMOTOCO LT ITC COALINDIA RELIANCE NCC
```

```
 SBIN HDFC  ONGC WIPRO HEROMOTOCO LT ITC COALINDIA RELIANCE NCC
['SBIN.NS',
 'HDFC.NS',
 'ONGC.NS',
 'WIPRO.NS',
 'HEROMOTOCO.NS',
 'LT.NS',
 'ITC.NS',
 'COALINDIA.NS',
 'RELIANCE.NS',
 'NCC.NS']
```

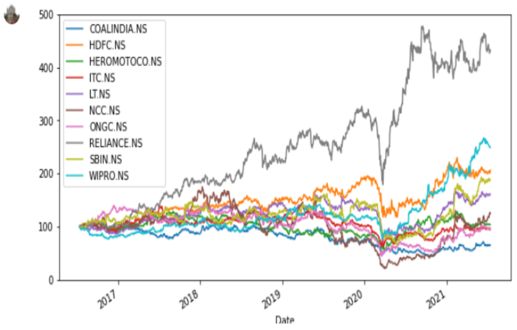
2. The Imported data is Now Plotted, processed and regularised for using it efficiently in a single LSTM trainer. Also Train Data and Test data is stored separately.

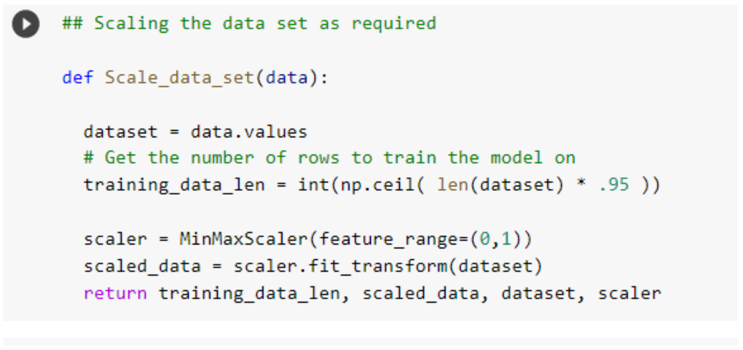


```
pf_data
```

Date	COALINDIA.NS	HDFC.NS	HEROMOTOCO.NS	ITC.NS	LT.NS	NCC.NS	ONGC.NS	RELIANCE.NS	SBIN.NS	WIPRO.NS
2016-07-13	227.587326	1228.089844	2782.615967	211.857025	932.247559	74.778938	125.333397	486.846863	223.594910	211.656326
2016-07-14	228.008713	1234.089722	2795.560791	213.228821	940.089355	76.331932	123.406441	484.988770	227.817383	210.420929
2016-07-15	223.935242	1272.475586	2781.073242	213.357437	952.948669	75.249542	123.245834	488.511871	227.326385	204.483719
2016-07-18	225.901764	1255.918213	2785.874268	213.400269	946.068359	73.461258	117.009933	485.640289	224.625977	203.543350
2016-07-19	224.005463	1248.641968	2770.014404	213.528900	945.106934	74.073029	118.294594	491.214630	225.460663	202.547668
...	...	...	...	...	...	...	...	...	...	...
2021-07-06	147.500000	2496.399902	2919.250000	202.500000	1505.500000	91.500000	121.500000	2124.800049	429.750000	532.599976
2021-07-07	147.550003	2529.199951	2913.350098	203.750000	1516.449951	89.599998	119.900002	2110.050049	432.850006	532.150024
2021-07-08	146.250000	2512.050049	2900.399902	202.000000	1500.699951	90.699997	117.050003	2092.600098	424.450012	531.000000
2021-07-09	146.699997	2496.449951	2896.800049	201.350006	1499.599976	94.699997	117.900002	2071.199951	423.750000	525.799988
2021-07-12	146.399994	2476.949951	2898.199951	201.100006	1500.750000	96.699997	118.550003	2084.100098	427.450012	525.849976

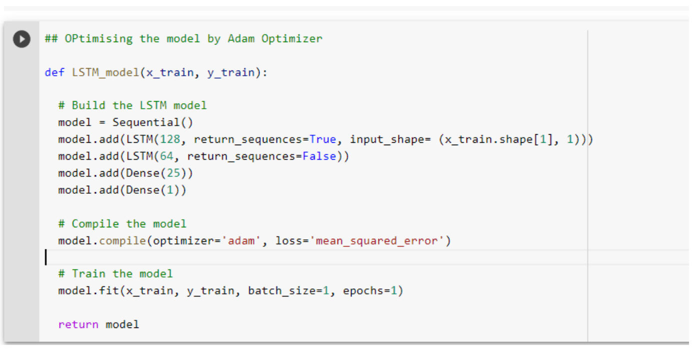
1232 rows × 10 columns





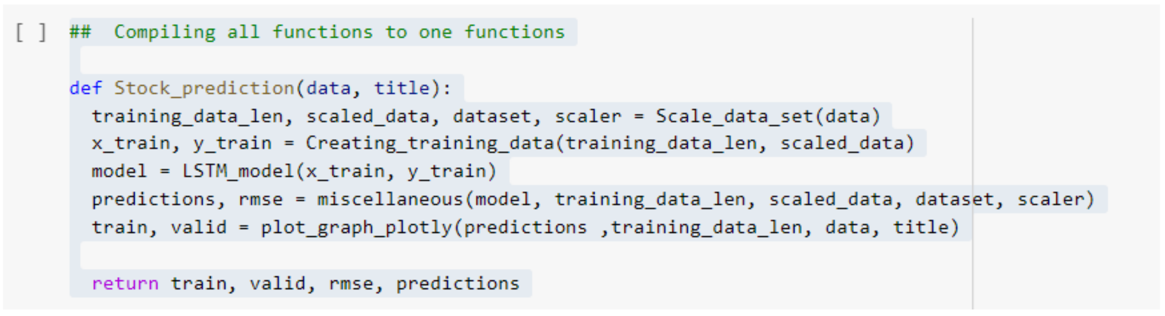
```
## Scaling the data set as required

def Scale_data_set(data):

  dataset = data.values
  # Get the number of rows to train the model on
  training_data_len = int(np.ceil( len(dataset) * .95 ))

  scaler = MinMaxScaler(feature_range=(0,1))
  scaled_data = scaler.fit_transform(dataset)
  return training_data_len, scaled_data, dataset, scaler
```

3. To improve the speed of the model we will first optimise it with the help of Adam optimiser. Without it just to run all the epochs for a single company stock it takes more time than all the stock predicted at once using Adam Optimiser.



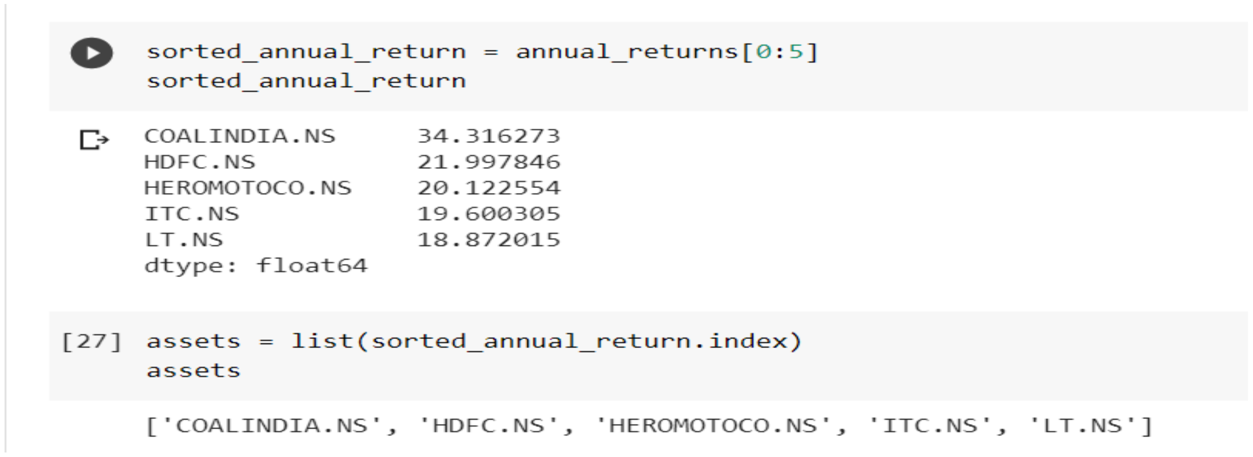
```
## OPtimising the model by Adam Optimizer

def LSTM_model(x_train, y_train):

  # Build the LSTM model
  model = Sequential()
  model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
  model.add(LSTM(64, return_sequences=False))
  model.add(Dense(25))
  model.add(Dense(1))

  # Compile the model
  model.compile(optimizer='adam', loss='mean_squared_error')

  # Train the model
  model.fit(x_train, y_train, batch_size=1, epochs=1)

  return model
```

4. Now the Data is ready to train and predict the values which will be compared by test data later on. Graphs plotted and rms values are also calculated.



```
[ ]  ##  Compiling all functions to one functions

     def Stock_prediction(data, title):
       training_data_len, scaled_data, dataset, scaler = Scale_data_set(data)
       x_train, y_train = Creating_training_data(training_data_len, scaled_data)
       model = LSTM_model(x_train, y_train)
       predictions, rmse = miscellaneous(model, training_data_len, scaled_data, dataset, scaler)
       train, valid = plot_graph_plotly(predictions ,training_data_len, data, title)

       return train, valid, rmse, predictions
```

5. After the prediction is made the predicted data give a certain return value for each stock share. We then took the Moving **Average** of daily return of stocks for next six months and then selected the top 5 best performing stocks .



```
sorted_annual_return = annual_returns[0:5]
sorted_annual_return
```

```
COALINDIA.NS      34.316273
HDFC.NS           21.997846
HEROMOTOCO.NS     20.122554
ITC.NS            19.600305
LT.NS             18.872015
dtype: float64
```

```
[27] assets = list(sorted_annual_return.index)
     assets
```

```
['COALINDIA.NS', 'HDFC.NS', 'HEROMOTOCO.NS', 'ITC.NS', 'LT.NS']
```

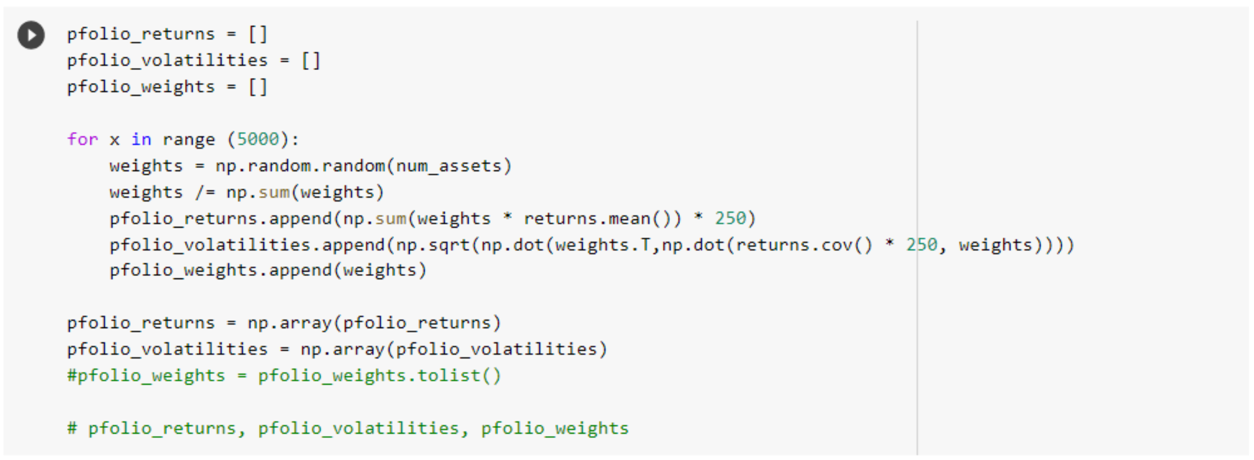
6. Now that we have successfully selected the appropriate stocks to invest in we will proceed on building an Optimum Portfolio . First we will calculate the variance of each stock, and then **covariance** and leading to **correlation** between each pair of stocks.

$$\operatorname{cov}(X,Y) = \operatorname{E}(XY) - \operatorname{E}(X)\operatorname{E}(Y)$$

$$\sigma_P^2 = \operatorname{E}\left[(R_P - \operatorname{E}(R_P))^2\right] = \operatorname{E}\left[\left(\left(\sum_i w_i R_i\right) - \left(\sum_i w_i \operatorname{E}(R_i)\right)\right)^2\right] =$$

$$= \operatorname{E}\left[\left(\sum_i w_i (R_i - \operatorname{E}(R_i))\right)^2\right] = \operatorname{E}\left[\sum_i \sum_j w_i w_j (R_i - \operatorname{E}(R_i))(R_j - \operatorname{E}(R_j))\right] =$$

$$= \sum_i \sum_j w_i w_j \operatorname{E}\left[(R_i - \operatorname{E}(R_i))(R_j - \operatorname{E}(R_j))\right] = \sum_i \sum_j w_i w_j \operatorname{cov}(R_i, R_j)$$



```
pfolio_returns = []
pfolio_volatilities = []
pfolio_weights = []

for x in range (5000):
    weights = np.random.random(num_assets)
    weights /= np.sum(weights)
    pfolio_returns.append(np.sum(weights * returns.mean()) * 250)
    pfolio_volatilities.append(np.sqrt(np.dot(weights.T,np.dot(returns.cov() * 250, weights))))
    pfolio_weights.append(weights)

pfolio_returns = np.array(pfolio_returns)
pfolio_volatilities = np.array(pfolio_volatilities)
#pfolio_weights = pfolio_weights.tolist()

# pfolio_returns, pfolio_volatilities, pfolio_weights
```

And then we plotted the correlation matrix of each stock.

7. With 5000 randomized weights we now plotted the graph of return versus volatility and visualized the efficient frontier . And then we finally calculated the maximum return , minimum risk and maximum sharpe ratio portfolios. Hence we finally receive our desired percentage of investment in each stocks.

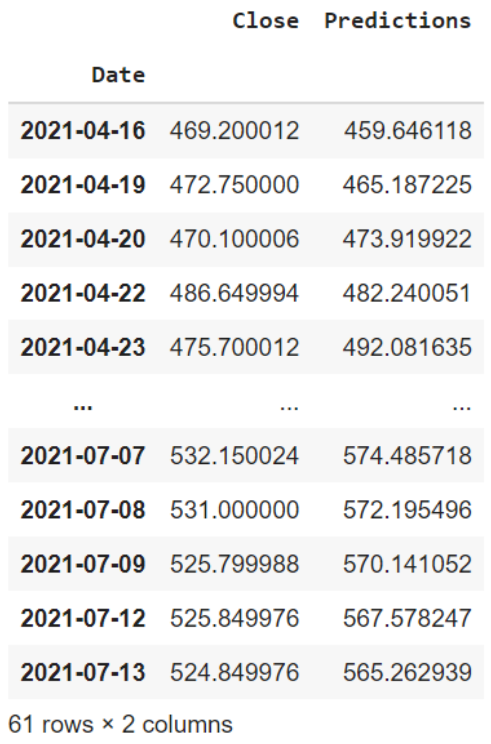
$$S = \frac{\mathrm{E}(R - R_f)}{\sigma}$$

Calculating sharpe ratio

```
#Max Sharpe Ratio
MaxSharpeRatio = go.Scatter(
    name='Maximium Return',
    mode='markers',
    x=[max_rvolatility],
    y=[max_return],
    marker=dict(color='red',size=14,line=dict(width=3, color='black'))
)

#Min Vol
MinVol = go.Scatter(
    name='Mininium Risk',
    mode='markers',
    x=[min_volatility],
    y=[min_vreturns],
    marker=dict(color='green',size=14,line=dict(width=3, color='black'))
)

#Random portfolio
EF_curve = go.Scatter(
    name='Random Portfolios',
    mode='markers',
    x= pfolio_volatilities,
    y= pfolio_returns,
    marker=dict(color='blue',size=5)
)

data = [MaxSharpeRatio, MinVol, EF_curve]
```

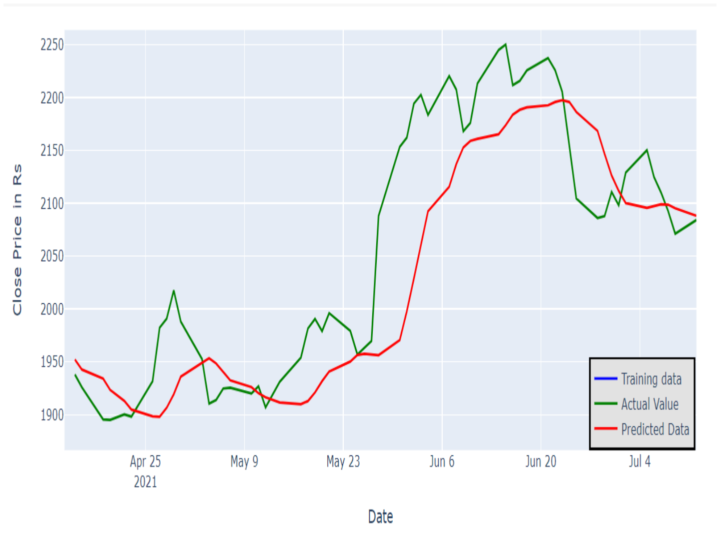
Results And Observation

- For LSTM the testing RMSE values we found were as low as 2.646, 3.524 and 3.895 for ONGC, NCC and ITC stocks respectively which is pretty accurate to predict future values of stock. Given aside are the closing actual and predicted values of Wipro Stock given by the LSTM model.

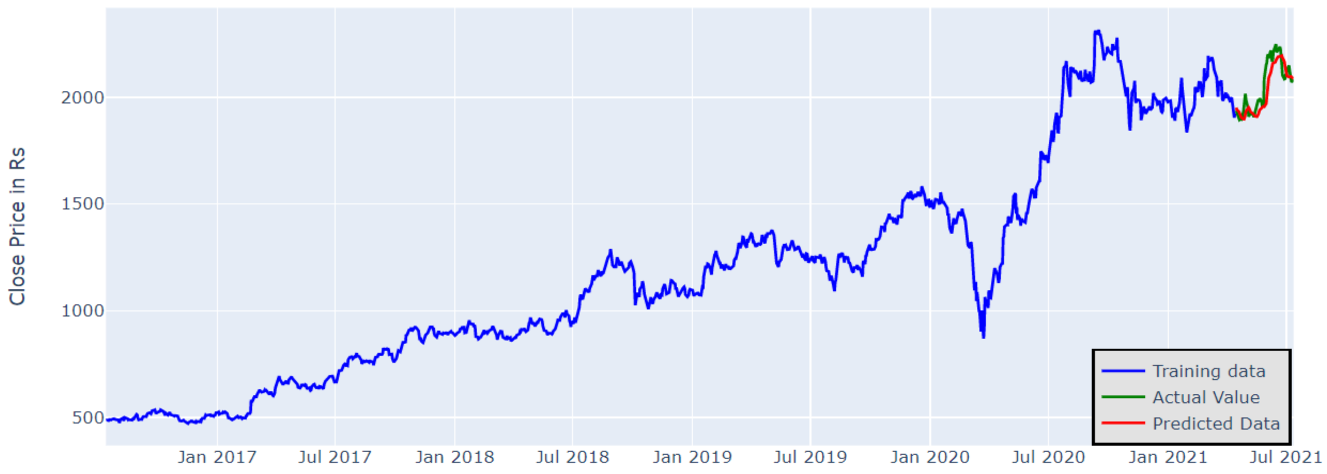


	Close	Predictions
Date		
2021-04-16	469.200012	459.646118
2021-04-19	472.750000	465.187225
2021-04-20	470.100006	473.919922
2021-04-22	486.649994	482.240051
2021-04-23	475.700012	492.081635
...	...	...
2021-07-07	532.150024	574.485718
2021-07-08	531.000000	572.195496
2021-07-09	525.799988	570.141052
2021-07-12	525.849976	567.578247
2021-07-13	524.849976	565.262939

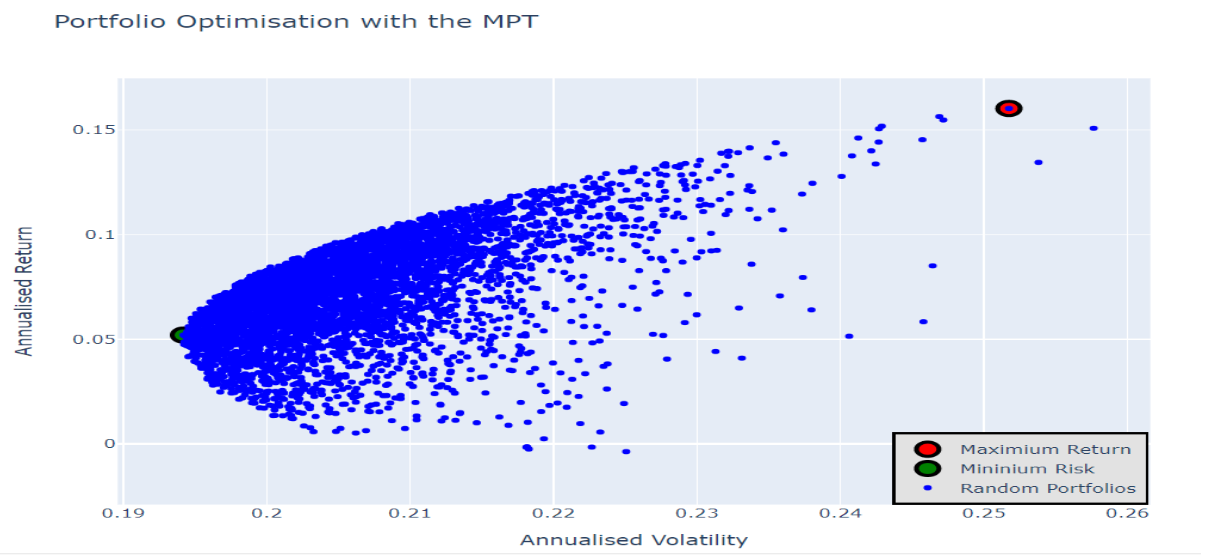
61 rows × 2 columns

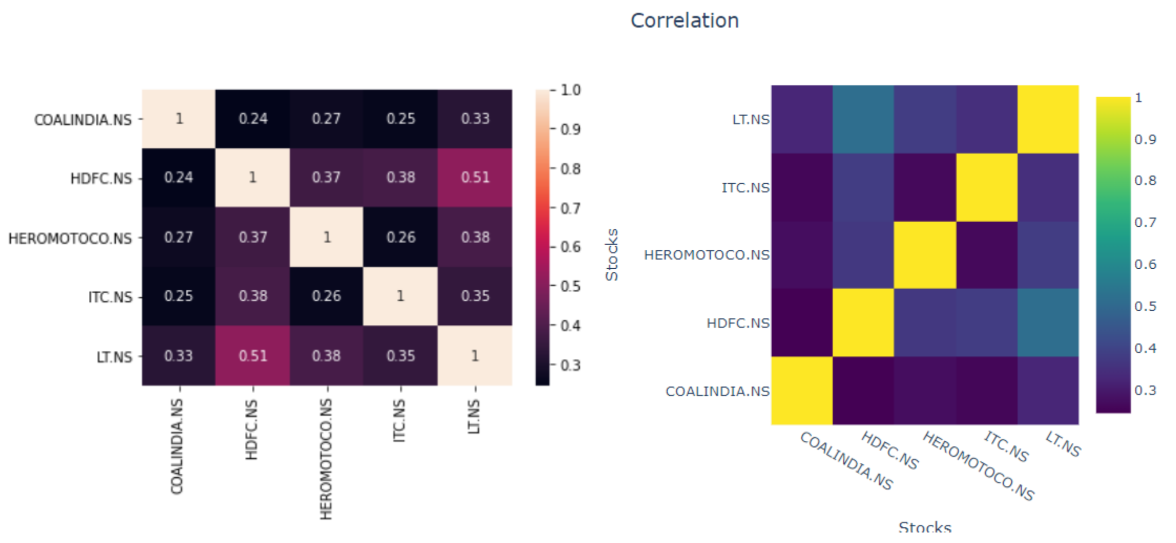


Zomm

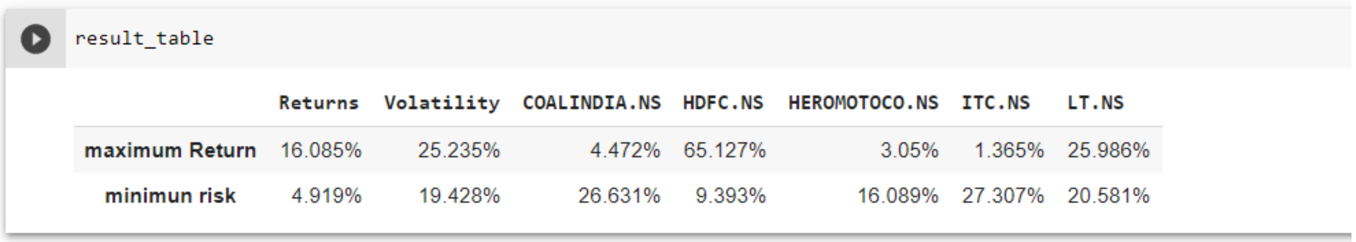


- For MPT below is the correlation matrix and efficient frontier graph observed. By the table below, we can get the max return of 16.085% with a volatility of 25.235% by investing the highest 65.127% in HDFC.It also shows that we can get the return of 4.919% at the min risk possible by investing the highest 26.631% in Coal India.





	COALINDIA.NS	HDFC.NS	HEROMOTOCO.NS	ITC.NS	LT.NS
COALINDIA.NS	1	0.24	0.27	0.25	0.33
HDFC.NS	0.24	1	0.37	0.38	0.51
HEROMOTOCO.NS	0.27	0.37	1	0.26	0.38
ITC.NS	0.25	0.38	0.26	1	0.35
LT.NS	0.33	0.51	0.38	0.35	1



```
result_table
```

	Returns	Volatility	COALINDIA.NS	HDFC.NS	HEROMOTOCO.NS	ITC.NS	LT.NS
maximum Return	16.085%	25.235%	4.472%	65.127%	3.05%	1.365%	25.986%
minimun risk	4.919%	19.428%	26.631%	9.393%	16.089%	27.307%	20.581%

Hence we successfully built a risk averse portfolio and maximum sharpe ratio portfolio.The portfolio we get is Dynamic and can be used for high frequency tradings.

CONCLUSION

Through this project, we learnt how to apply deep learning techniques such as LSTM and Linear Regression Models, how to use keras-tensorflow library, how to collect and preprocess given data, how to analyze model's performance and optimise LSTM Network and to ensure increase in positive results. We built a model to accurately predict the future closing price of a given stock, using the Long Short Term Memory Neural Net algorithm.

And also using the predicted stock prices from the LSTM, we were able to make an optimized portfolio which can help investors in making risk-averse decisions using Markowitz Portfolio theory .

References

Long Short Term Memory (LSTM) - Recurrent Neural Networks | Coursera

Modern Portfolio Theory Explained! - YouTube

Modern Portfolio Theory (MPT) (investopedia.com)

LSTM Networks | A Detailed Explanation | Towards Data Science