Stock Market prediction using Deep Learning and Machine Learning methods and Portfolio Optimization

Leena Dighole

Rutgers University

15 Dec 2020

Abstract

The stock prediction is an essential key for the active portfolio management. Various methods are used to improve the prediction accuracy in order to construct the portfolios which will give more excess returns (value added) compared to market. The Active management assumes that the market is not perfectly efficient and focus on returns and risks compared to benchmark.

For stock predictions, 2 different methods are used in this paper. Long Short Term Recurrent Neural Network (LSTM RNN) is one of the deep learning methods used to predict the stock returns. Support Vector Regressor (SVR) is one of the machine learning technique used to predict time series data is also used in this project. The performances of the predictions of these models are compared and found that LSTM RNN network outperforms SVR which is inline with our literature review as Neural network tries to capture data knowledge more accurately than a simple SVR.

The portfolio assets are selected based upon market capitalization of the assets. The Mean Variance Optimization (MVO) is used to improve the performances of the portfolio. SP500 index is used as the benchmark. The active returns portfolio optimization technique is used to maximize the risk adjusted annual active return. The main objective of the paper is to analyze the performances of LSTM RNN and SVR methods using MVO portfolio optimization.

Keywords: LSTM RNN, SVR, portfolio optimization

---

1. Introduction

Forecasting stock prices is a challenging part of the Active portfolio management. The number of various parameters affects the stock values and its difficult to anticipate them in a single model. Various machine learning and deep learning algorithms are used and their prediction accuracies are compared in order to prefer the proper model according to available computational costs and processing data.

Portfolio is the collection of different assets to be used for investment. Roughly portfolio management can be called as the decision making process to maximize the profit over certain time period over selected assets. Traditionally it was done using one's experience, judgement and qualitative analysis. The Quantitative analysis includes trading strategy design using quantitative methods. The main objective of quantitative trading strategies is to select assets and the investment allocation weights in order to maximize the excess returns with minimal risks. Stock prediction is one of the important information for a portfolio manager to construct a more optimal portfolio with high return and low risk tradeoff.

Stock predictions are done using various machine learning algorithms which have strong mathematical and statistical properties. The main aim of machine learning or deep learning methods are to extract knowledge from given data. For time series data like stock market historical prices, these models try to learn the pattern of the series and predictions are done based upon the trained models. Thumb rule says that more the data is available to train these different models, more accurate model will predict the stock prices. These model training comes with the underlying constraint of the computational cost and time. The different models often faces the tradeoff between computational cost with available resources verses the time for sufficient prediction accuracy. Various researches are done in this applications and these are used as the basis of the literature review for this paper.

This paper focuses on predicting stock returns using LSTM RNN and SVR models. The performance analysis is done based upon training and testing errors. SP500 index is used as the benchmark for this paper. The top 30 stocks are selected using their market capitalization as a selection criteria. The daily data from yahoo finance from 2013 to 2020 is used for the analysis.

The analysis is divided into 2 parts:

- Predict the return of assets using LSTM RNN and SVR models
- Optimize the asset allocation using MVO optimization

The analysis is done using Daily as well as monthly returns and their corresponding performances are briefly discussed in section 5.

The literature review for the motivation of selecting LSTM RNN and SVR as a predicting models is briefly discussed into the following literature survey section 2. The section 3 describes the methodologies of LSTM and SVR models briefly. Section 4 discusses data selection and the algorithm used in this paper. Section 5 consists an empirical performance analysis section. Section 6 consists of summary about the paper followed by appropriate references used for this project. LSTM RNN model predicts stock returns more accurately compared to SVR model. Python is used for building ,training different ML models and verifying their performances.

2. Literature Review

Recently many researchers implemented different algorithms to predict stock returns. Hao [1] in his research paper showed the usage of SVR to predict stock returns. Yilin [2] in his research paper compared the performances of 3 machine learning and 3 deep learning algorithms. He used LSTM RNN, Convolutional Neural Network (CNN), Deep Multilayer Perceptrons (DMLP) as deep learning algorithms and Random Forest (RF) , Support Vector Regressor(SVR) and ARIMA models as machine learning algorithms. He used different optimization techniques like MVF(Mean Variance Forecasting) and Omega with Forecasting model (OF). His findings includes that in 3 deep learning models, LSTM RNN outperforms DMLP and CNN. Amongst his 3 machine learning model performance comparison, Random Forest outperforms SVR and ARIMA models. When all 6 methods are compared, surprisingly the Random Forest outperforms all deep learning methods. The ascending order as per the performances of these models is : RF,LSTM RNN, DMLP, SVR, ARIMA, CNN. CNN performs better in image processing applications but worst in time series predictions. The RF models captures the idea of constructing decision trees by randomly bootstrapping the data and randomly selecting the features. This randomness avoids the

problem of overfitting leading to getting better predictions. LSTM RNN model captures the patterns of time series data using short term memory which increases its prediction accuracy. The 2 models selected in this paper were based upon the intuition of comparing one deep learning and machine learning algorithm. We choose LSTM RNN and SVR models for predicting the stock returns.

LSTM is the special architecture of RNN which remembers the values of either long or short time durations[3]. Jia [4] investigated the effectiveness of usage of LSTM RNN for stock predictions. Tingwei [5] showed the performances of LSTM networks and SVR using SP500 as a benchmark. The models are compared using RMSE, MAE, MAPE, AMAPE errors supporting the fact that LSTM has the lowest error. Van in his research paper supports the same observations [7] about outperformance of LSTM RNN over SVR and Linear Regression. This literature survey motivated to implement LSTM RNN and SVR models for stock predictions and analyze their performances.

3. Methodology

The 2 models and their working is described below along with the concepts of portfolio optimization. The models are:

- LSTM RNN model
- SVR Model
- Portfolio Optimization

3.1 LSTM RNN model

Long Short Term Memory Recurrent Neural Network is a special type of RNN models. The important features of memory extension makes it more suitable for time series predictions. Each neuron in LSTM has a memory cell that links current data with the previous ones.

LSTM has 3 gates and they are Input gate(It), forget gate(Ft), output gate (Ot). Writing a certain neural network into memory involves inputting it to memory cell. Forget get allows us to decide whether previous data is important to be involved for prediction at current state or not. LSTM RNN is used to overcome the limitations of RNN to retain the long term information

property. These memory cells are present in each hidden layer. This paper uses ADAM optimizer for training the NN. Adam is better than stochastic gradient descent algorithms and its widely used for stock predictions supported by this paper references.

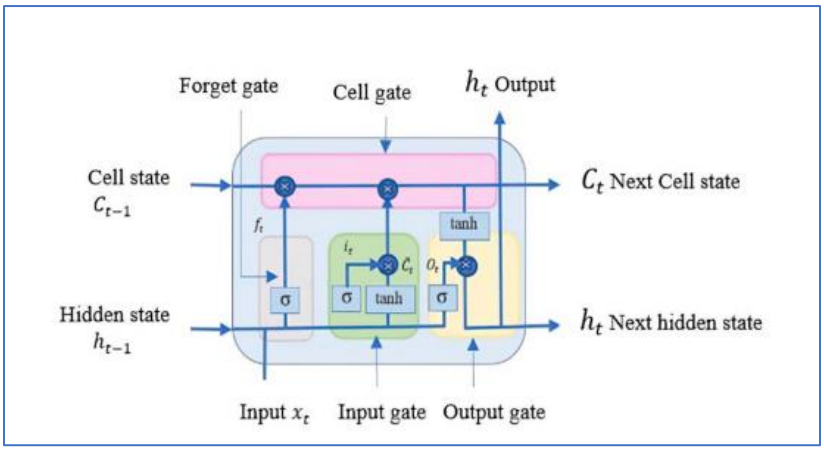

Figure 1: LSTM structure at time step t

Memory cell makes decisions about which information to be stored. It also decides when to allow reading, forgetting and writing into the memory. The activation functions are used which computes values between 0 and 1. Commonly used activation functions are sigmoid, tanh and Relu functions.

The activation functions of LSTM RNN networks for a memory cell are:

$$i_t = \sigma_1(W_i x_t + U_i c_{t-1} + b_i)$$

$$f_t = \sigma_1(W_f x_t + U_f c_{t-1} + b_f)$$

$$o_t = \sigma_1(W_o x_t + U_o c_{t-1} + b_o)$$

$$c_t = f_t \bullet c_{t-1} + i_t \bullet \sigma_1(W_c x_t + b_c)$$

$$h_t = o_t \bullet \sigma_2(c_t)$$

Figure 2: LSTM memory cell equations

Where, x_t is the input vector at time t

h_t is the output vector at time t

c_t is the memory cell state at time t

i_t is the input gate vector at time t

f_t is the forget gate vector at time t

o_t is output gate vector at time t

W and U are weight matrices and b are biases

Sigma represents activation functions

Various parameters for selecting LSTM RNN networks can be viewed in below picture:

Parameters of LSTM neural network.

Parameter	Value
Hidden nodes	5,10,15,20,25,30
Hidden layers	1,2,3, ...,10
Learning rate	0.0001,0.001,0.01,0.1
Patient	0,5,10
Batch size	50,100,200
Dropout rate	0.1, 0.2, ...,0.5
Recurrent dropout rate	0.1, 0.2, ...,0.5
Loss function	Mean absolute error
Optimizer	SGD, RMSprop, Adam

Figure 3: LSTM Parameters

LSTM RNN model Algorithm:

We used 3 hidden layers and one output layer in our paper.

Keywords:

- Batch = batch size of training data used for Stochastic Gradient descent optimization
- Epoch = no of times the model will be passed through the given training data set

LSTM Layers:

1. First LSTM model Layer:

- Number of nodes = 50

- We want return sequences =True to be able to add next layer
- Input shape consists of no of time steps and no of features

2.Second LSTM layer:

- Number of nodes = 50
- We want return sequences= False as we want to use 2 LSTM layer model in this project

3. We will add certain dense layers after LSTM in our RNN model

- Third layer will be a dense layer with 25 nodes
- Forth layer will be one dense layer

The LSTM Model summary shows the total parameters used in the models.

Using the daily returns and a rolling window size of 250 days(approx. 1 year) and 4 hidden layers of LSTM with batch size =10 and epoch =5, the output is shown in below figure. Total 31000 parameters are tunned while predicting the stock price of one asset in LSTM RNN model.

The batch size and no of epochs are selected based upon the managing the tradeoff between accuracy and computational time.

```
1
Epoch 1/5
136/136 [==============================] - 30s 221ms/step - loss: 0.0112 - val_loss: 0.3052
Epoch 2/5
136/136 [==============================] - 27s 201ms/step - loss: 0.0042 - val_loss: 0.3023
Epoch 3/5
136/136 [==============================] - 29s 214ms/step - loss: 0.0043 - val_loss: 0.3160
Epoch 4/5
136/136 [==============================] - 27s 197ms/step - loss: 0.0041 - val_loss: 0.3305
Epoch 5/5
136/136 [==============================] - 31s 231ms/step - loss: 0.0042 - val_loss: 0.2941
Model: "sequential_96"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_191 (LSTM)              (None, 250, 50)           10400
_________________________________________________________________
lstm_192 (LSTM)              (None, 50)                20200
_________________________________________________________________
dense_190 (Dense)            (None, 25)                1275
_________________________________________________________________
dense_191 (Dense)            (None, 1)                 26
=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
```

Figure 4: LSTM Model Summary for batch size =10 and epoch =5

3.2 SVR Model

Support Vector Regressor is a machine learning algorithm which is widely used in stock market predictions. Vapnik's Structural risk minimization principle is used in SVR to solve different regression problems. Statistical learning theory is the origin of SVR model. This is applied to regulate generalization as well as discover the appropriate tradeoff between empirical risk and model complexity.

In this paper we used radial basis kernel function of SVR which is given as,

$K(x, x') = \exp(-\text{gamma} * ||x - x'||^2)$ where gamma >0

Where,

high C value aims at classifying all training examples correctly

gamma value represents how much influence a single training example has on the model parameters.

Parameters of SVR.

Parameter	Value
C	$2^0, 2^1, \ldots, 2^5$
γ	$2^{-5}, 2^{-4}, \ldots, 2^0$

Figure 5: SVR Parameters

SVR Model Construction:

- Kernel fn = 'rbf'
- Regularization parameter = C
- Gamma value

With reference from paper of Yilin[2] and some trials we specified values of SVR to be,

Gamma = 0.1 (10% weightage for each sample influence)

C = 1000 (managing tradeoff between computational time and accuracy)

3.3 Portfolio Optimization

Portfolio construction involves an attempt to construct an efficient portfolio which maximizes the expected return with a minimized risk. A portfolio manager always want to construct an optimal portfolio by managing the tradeoff between expected returns and underlying risk. Aim is to get higher value added returns or maximum expected active returns over minimal active risk. The paper uses SP500 as the benchmark. The top 30 assets by market capitalization over the period of 2013 to 2020 are used to construct an optimal portfolio. The equally weighted assets is our benchmark portfolio. Using the predictions performed by LSTM RNN and SVR models, we are predicting the expected returns of each asset. Using these predicted returns, we want to maximize our risk adjusted annual active return using an objective function.

The basic assumptions for constructing a portfolio is that market is not perfectly efficient. If the market is assumed to be efficient then its prices captures all information and beating market seems to be impossible. Lets discuss the objective function of maximizing active returns with some constraints.

The objective function Formula:

=>max(active returns - active risk penalty)

=> max(wA'*alpha-(lambda/2*wA'*sigma*wA)

Lets asssume,

wA =wP-wB

sigma = cov(alpha)

lambda = risk aversion parameter

wA =active weights

wP = portfolio weights

wb= benchmark portfolio weights(here they are assumed to be equally weighted)

Constraints:

1. Beta neutrality wA'*beta = 0

2. Full investment wA'1 = 0

3. Tracking Error = sqrt(wA'*Sigma*wA) <= 0.03 (3%)

Where, beta = cov(Ra,Rm)/var(Rm)

Beta is the ratio of covariance of asset return and market return with the variance of the market. Higher the beta of a stock, more risk is associated with it as it fluctuates more compared to market. But more excess return is associated with it. The beta neutral condition aims to construct market risk free portfolios. This constraint enables to give less weightage to the assets with beta greater than 1 and vice versa. Full investment constraint states that the addition of all active weights of assets should be 1. Tracking error constraint makes sure that the portfolio variance should not exceed certain limits above benchmark portfolio. The active risk is bounded while constructing an optimized portfolio.

Note that our benchmark portfolio is the equally weighted assets portfolio

wB = [1/30,1/30....,1/30]... for assets = 30

Also,

Benchmark return (Rb) = alpha*wB

Portfolio return(Rp) = alpha*wP

Active return(Ra) = Rp-Rb

We will construct our portfolio using above constraints and maximizing risk adjusted active return using predicted alpha returns. Once we have our active weights, we can calculate below parameters.

Performance analysis:

- IR = active return/ active risk

  = Ra/TE

- TE = sqrt(wA'*Sigma*wA)
- IC = IR/sqrt(BR)

Note that IR = IC *(sqrt(BR))

If we are making monthly portfolios, then in one year we could make 12 bets. The breadth value for portfolio with 30 stocks is calculated as BR = 12*30 =360

- Portfolio variance (var(P))= wP'*Sigma*wP
- Benchmark variance(var(B)) = wB'*Sigma*wB
- Sharpe ratio = Rp/sqrt(var(P))

Using the maximum risk adjusted annual active return, a portfolio manager can be able to construct an optimized portfolio by assigning proper weights to the assets.

We can also use other optimization functions and try to construct portfolios. Another approaches may involve maximizing IR, maximizing Sharpe ratios, minimizing volatility etc.

4. Data

The Stock return predictions needs historical data and a good prediction model. The SP500 is used as the benchmark for our asset selections. The top 30 assets are selected based upon their market capitalization to construct an optimized portfolio.

Dataset span: 01/01/2013 to 12/15/2020

Using these historical prices of data, they are converted into simple daily returns and log returns. The log returns are used in this paper.

The top 5 historical assets prices graph is shown below:

Figure 6 : Top 5 assets historical closing price plot

The corresponding simple daily returns plot is shown below:

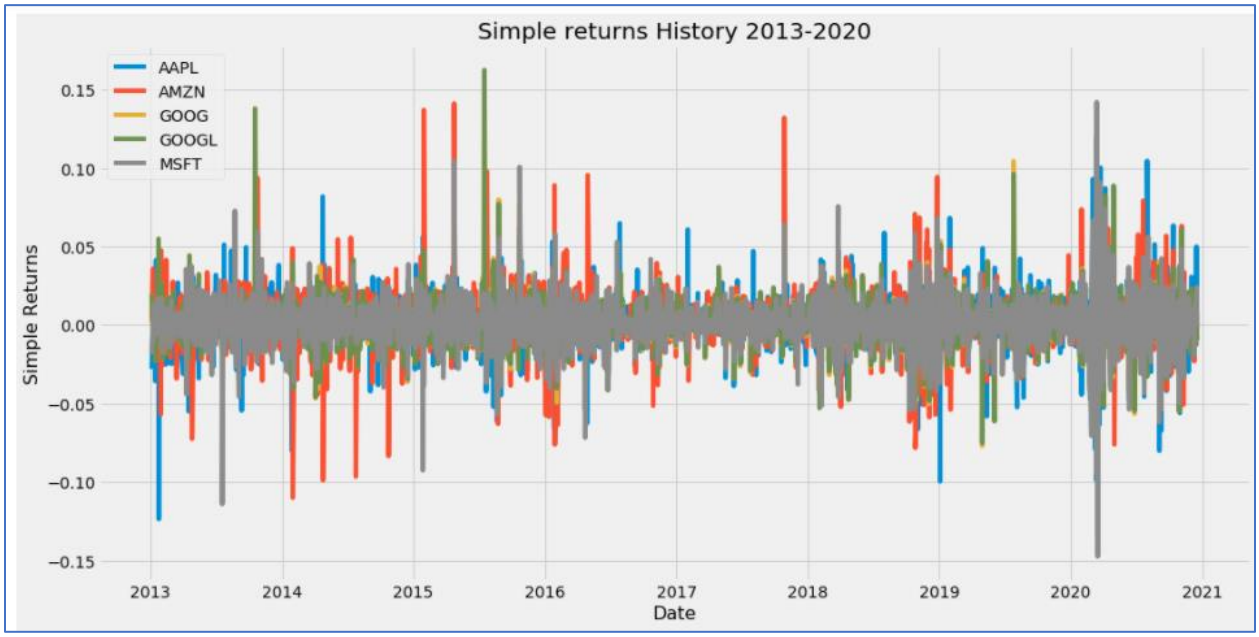

Figure 7 : Top 5 assets historical Simple returns plot

The log returns are calculated and due to good statistical properties of log returns, we used them in our paper for stock predictions.

The log normal returns log(1+ri) are assumed to be normally distributed, where ri is the simple return

Properties:

- log(1+rt) = log(Pt)-log(pt-1)
- If returns are very small r<<1, log(1+r) ~ r ....r<<<1
- The simple additive property of logs can be used to avoid complex calculations for compound returns.

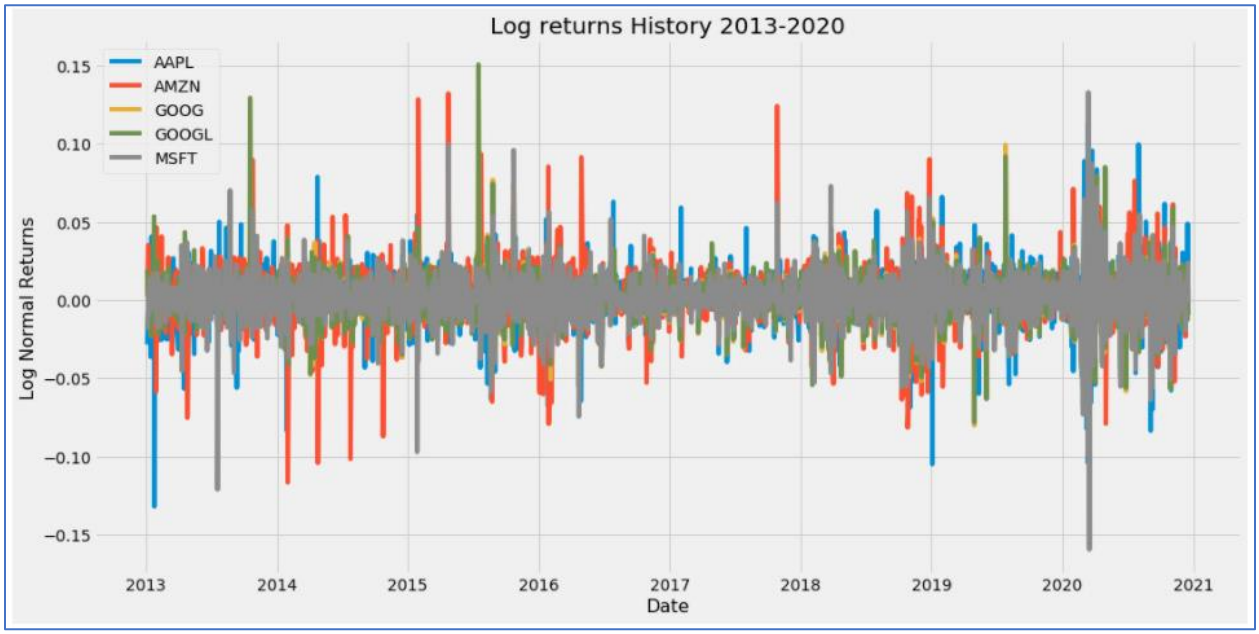

Figure 8 : Top 5 assets historical Simple returns plot

Data Preprocessing:

The log returns from historical prices needs preprocessing before used to make any predictions. The historical dataset is directly downloaded in python using yahoo finance package for the defined span. The LSTM RNN and SVR models needs to be trained and then used to perform predictions.

Algorithm:

1. Take user input for rolling window size of the data
2. Divide the 80% data into training and 20% testing dataset

3. Scale the data using minmax scaler(0,1)
4. Train the models using scaled training dataset
5. Perform predictions on scaled test data set
6. Inverse scale the predictions to get the predicted log returns

The scaling of features is performed which includes robustness to very small deviations of features. The scaling enables us to preserve zero entries in sparse data. Feature scaling represents the significance of the features in the process of selecting most significant features. LSTM model space requires 3-D data which are:

- Samples
- Time steps
- Features

The time steps refers to how many steps in time we want the backpropagation algorithm to use when calculating gradients for weight updates during training phase of model. The 3-D LSTM scaled data dimensions with 30 stocks and 250 rolling window size on daily returns is shown in below figure:

```
The x_train dimensions are:
 (1354, 250, 30)
The y_train dimensions are:
 (1354, 30)
The x_test dimensions are:
 (401, 250, 30)
The y_test dimensions are:
 (401, 30)
```

Figure 9 : LSTM 3D data dimensions

X_train and y_train has 1354 entries due to 80% split of the original data.

250 is the timestep representing last one year's daily data is used in predicting next day's return

30 represents the total no of stocks/ assets in the portfolio.

The data preprocessing is an essential key to make a model compatible training and testing dataset.

Scaled Predicted values are converted into log predicted returns using inverse minmax scaling.

5. Empirical session and Results

5.1 Performance analysis of LSTM RNN and SVR model:

This part compares the performance parameters of the 2 models and describes assertion to the fact that LSTM RNN model outperforms SVR which is inline with our Literature survey.

Specifications:

- Benchmark = SP500
- Models: LSTM RNN, SVR
- Historical dataset = 1/1/2013 to 12/15/2020
- Daily log returns and Rolling window size = 250 (approx. 1 year)
- Monthly log returns and Rolling window size = 36 (approx. 3 year)
- Parameters = RMSE, MAE

The Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are used for the comparison of prediction accuracy of both models.

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(r_t - \hat{r}_t)^2$$

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|r_t - \hat{r}_t|$$

Figure 10: RMSE and MAE equations

Where, N = Total number of observations

r_t = Actual return

r'_t =Predicted return

Case 1: Daily returns

Performance Table:(Assets = 30 stocks and rolling window = 252(approx. 1 year previous data))

Parameter	LSTM RNN	SVR
RMSE	0.0238	0.0266
MAE	0.0155	0.0182

Table 1 : RMSE and MAE for daily returns predictions

Case 2: Monthly returns

Performance Table:(Assets = 30 stocks and rolling window = 36(approx. 3 years previous data))

Parameter	LSTM RNN	SVR
RMSE	0.0795	0. 0858
MAE	0.0610	0.0655

Table 2 : RMSE and MAE for Monthly returns predictions

Thus we can say that LSTM RNN model gives more accurate predictions than SVR which is an expected result inline with our Literature review. As the number of training dataset increases, lesser will be the RMSE and MAE error values of the models. We compared the Daily returns and Monthly returns errors and can be observed that the model prediction accuracy increases as the no of training dataset increases.

5.2 Portfolio Optimization Performance Analysis

Specifications:

- Benchmark portfolio = Equally weighted portfolio
- Optimization criteria = Max risk adjusted annual active return, Max Sharpe ratio, Min Volatility

Note that the optimization criteria of max risk adjusted annual active return was not successfully able to code in python. Here are some observations made using full investment constraints and max sharpe ratio and min volatility constraints. Lets assume,

- Equally weighted = Benchmark portfolio
- Optimization 1 = Max Sharpe ratio optimization
- Optimization 2 = Min volatility optimization

Case 1: Daily returns

Performance Table:(Assets = 30 stocks and rolling window = 252(approx. 1 year previous data))

Parameter	Equally weighted	Optimization 1	Optimization 2
Expected Annual Return	20%	35.8%	35.8%
Annual Volatility(Risk)	17%	24.4%	24.4%
Sharpe Ratio	1.17	1.39	1.39

Table 3 : Portfolio optimization parameters for daily returns

The Asset allocation for different optimization is different. For Max sharpe ratio optimization and min volatility optimization, weights allocated to 30 stocks are shown in below figures,

```
2)MVO Optimization 1 :Max Sharpe Ratio
The weights after optimizing according to max sharpe ratios are:
 OrderedDict([('AAPL', 0.00218), ('ADBE', 0.08583), ('AMZN', 0.13412), ('BAC', 0.0), ('CMCSA', 0.0), ('CRM', 0.0),
 ('DIS', 0.0), ('FB', 0.0), ('GOOG', 0.0), ('GOOGL', 0.0), ('HD', 0.0), ('INTC', 0.0), ('JNJ', 0.0), ('JPM', 0.0),
 ('KO', 0.0), ('MA', 0.0), ('MRK', 0.0), ('MSFT', 0.0314), ('NFLX', 0.18379), ('NKE', 0.10604), ('NVDA', 0.23931),
 ('PEP', 0.0), ('PFE', 0.0), ('PG', 0.0), ('PYPL', 0.0), ('T', 0.0), ('UNH', 0.21733), ('V', 0.0), ('VZ', 0.0), ('WMT', 0.0)])
Expected annual return: 35.8%
Annual volatility: 24.4%
Sharpe Ratio: 1.39
```

Figure 11: Asset allocation weights for daily returns using Max Sharpe ratio optimization

```
3)MVO Optimization 2 :Min Volatility Criteria
The weights after optimizing according to min volatility are:
 OrderedDict([('AAPL', 0.0), ('ADBE', 0.0), ('AMZN', 0.03754), ('BAC', 0.0), ('CMCSA', 0.0), ('CRM', 0.0), ('DIS', 0.01095),
 ('FB', 0.00836), ('GOOG', 0.0), ('GOOGL', 0.0), ('HD', 0.0), ('INTC', 0.0), ('JNJ', 0.12193), ('JPM', 0.0), ('KO', 0.15968)
 , ('MA', 0.0), ('MRK', 0.05165), ('MSFT', 0.0), ('NFLX', 0.00793), ('NKE', 0.01204), ('NVDA', 0.0), ('PEP', 0.0),
 ('PFE', 0.06249), ('PG', 0.09131), ('PYPL', 0.02819), ('T', 0.0), ('UNH', 0.0), ('V', 0.0), ('VZ', 0.2395), ('WMT', 0.16843)])
Expected annual return: 35.8%
Annual volatility: 24.4%
Sharpe Ratio: 1.39
```

Figure 12: Asset allocation weights for daily returns using Min Volatility optimization

Results:

- For different optimizations even if the sharpe ratio is same and same annual volatility, the individual weights of the 30 assets are different in different optimizing methods.
- The 2 optimized portfolios have higher Expected returns over potentially low risk compared to simply equally weighted portfolio. Sharpe ratio of optimized portfolio (1.39) are higher than the equally weighted portfolio (1.17)

Case 2: Monthly returns

Performance Table:(Assets = 30 stocks and rolling window = 36(approx. 3 years previous data))

Parameter	Equally weighted	Optimization 1	Optimization 2
Expected Annual Return	19%	26.6%	26.6%
Annual Volatility(Risk)	13%	13%	13%
Sharpe Ratio	1.46	1.9	1.9

Table 4 : Portfolio optimization parameters for monthly returns

The Asset allocation for different optimization is different. For Max sharpe ratio optimization and min volatility optimization, weights allocated to 30 stocks are shown in below figures,

```
2)MVO Optimization 1 :Max Sharpe Ratio
The weights after optimizing according to max sharpe ratios are:
 OrderedDict([('AAPL', 0.0), ('ADBE', 0.0962), ('AMZN', 0.0), ('BAC', 0.0), ('CMCSA', 0.0), ('CRM', 0.0), ('DIS', 0.0),
 ('FB', 0.0), ('GOOG', 0.0), ('GOOGL', 0.0), ('HD', 0.0), ('INTC', 0.0), ('JNJ', 0.0), ('JPM', 0.0), ('KO', 0.0),
 ('MA', 0.0), ('MRK', 0.0), ('MSFT', 0.18127), ('NFLX', 0.07774), ('NKE', 0.10993), ('NVDA', 0.08869), ('PEP', 0.0),
 ('PFE', 0.0), ('PG', 0.15905), ('PYPL', 0.0), ('T', 0.0), ('UNH', 0.28713), ('V', 0.0), ('VZ', 0.0), ('WMT', 0.0)])
Expected annual return: 26.6%
Annual volatility: 13.0%
Sharpe Ratio: 1.90
```

Figure 13: Asset allocation weights for monthly returns using Max Sharpe ratio optimization

```
3)MVO Optimization 2 :Min Volatility Criteria
The weights after optimizing according to min volatility are:
 OrderedDict([('AAPL', 0.0), ('ADBE', 0.0), ('AMZN', 0.0), ('BAC', 0.0), ('CMCSA', 0.0), ('CRM', 0.0), ('DIS', 0.0),
 ('FB', 0.0), ('GOOG', 0.01002), ('GOOGL', 0.0), ('HD', 0.0), ('INTC', 0.0584), ('JNJ', 0.0), ('JPM', 0.0), ('KO', 0.01784),
 ('MA', 0.0), ('MRK', 0.11245), ('MSFT', 0.0), ('NFLX', 0.07664), ('NKE', 0.04783), ('NVDA', 0.0), ('PEP', 0.05284),
 ('PFE', 0.0), ('PG', 0.27246), ('PYPL', 0.0), ('T', 0.10878), ('UNH', 0.12355), ('V', 0.0), ('VZ', 0.0), ('WMT', 0.11919)])
Expected annual return: 26.6%
Annual volatility: 13.0%
Sharpe Ratio: 1.90
```

Figure 14 : Asset allocation weights for monthly returns using Min Volatility optimization

Results:

- For different optimizations even if the sharpe ratio is same and same annual volatility, the individual weights of the 30 assets are different in different optimizing methods.
- The 2 optimized portfolios have higher Expected returns over potentially low risk compared to simply equally weighted portfolio. Sharpe ratio of optimized portfolio (1.46) are higher than the equally weighted portfolio (1.9)

The LSTM RNN outperforms SVR predictions and this is strongly supported by the findings of the references. The optimization improves the sharpe ratio over the equally weighted benchmark portfolio. It allocates weights which gives maximum expected returns with comparatively low risk

6. Summary and Future Scope

6.1 Algorithmic Approach Summary

The Algorithm used in constructing the portfolios using LSTM RNN model and MVO optimization technique is discussed briefly below:

1. Select the no of assets (n)to be used in constructing portfolio.
2. The top n assets are selected using market capitalization.
3. Download the historical dataset for n assets.
4. Normalization of historical dataset - converting into simple returns/ log returns (good statistical properties)
5. User decides to use daily or monthly returns for portfolio construction
6. Split the dataset into 80% train and 20% test dataset
7. Select the rolling window size and scale the data features using minmax scaler in range [0,1]
8. Construct and train LSTM RNN model
9. Construct and train SVR model
10. Predict the returns using Test dataset
11. Analysis of the performance of the model using RMSE, MAE
12. Construct an equally weighted benchmark portfolio and calculate its Expected return, volatility and Sharpe ratio
13. Optimizing portfolio using active weights constraint
14. Optimize portfolio using Sharpe ratio (alternative optimization)
15. Optimize portfolio using min volatility (alternative optimization)
16. Compare the results of equally weighted portfolio and optimized portfolios
17. Repeat the steps 1-15 using different no of assets in portfolio
18. Analyze the performances

We can try different approaches described in future scope of the project to check the performances of different approaches!

6.2 Future Scope

- We can use LSTM model with encoder-decoder to perform its performance. The motivation behind using encoder-decoder is taking all the features of inputs and encode them into latent space to reduce the dimensionality which is also called as non-linear PCA and then decode it to check the original features. We can train our model using Adam optimizer and mean squared loss error function to be able to predict future stock prices more accurately.
- LSTM model using dropout layers. The dropout percentage (probability value between range [0,1]) randomly drops the input data at each node. The random dropping avoids problem of overfitting the data into LSTM n/w giving better performance
- Using Multiple LSTM layers. The intuition is training the model with multiple LSTM layers should improve the performance of the model. We need to consider the hardware capabilities and computational time to try out different no of LSTM layers of our model.
- Varying the input parameters of the LSTM model. We can vary the number of assets in our portfolio (top 5/10/30/50 stocks by Market capitalization) and also the rolling window size of the predictions (past 30/60/90 days data). We can check the performances of these variation and try to improve the performance of the LSTM RNN model.
- Different alpha models for prediction. In this project we used LSTM RNN model for predicting future log returns of our portfolio. We can potentially try different Deep learning methods such as Convolutional neural networks (CNN), Deep Multilayer Perceptron (DMLP), Simple RNN models to check their performances. Also we can use different Machine Learning methods such as Logistic regression, PCA, Random Forest, XGBoost and even Linear regression and check their respective performances.
- Optimizing Portfolio. In this project we used the Mean Variance Optimizing technique with 2 different approaches Maximizing Sharpe Ratio and Active weights optimization within 3% TE constraint. We can use different optimizing techniques such as complex optimization, MSAD (Mean Absolute Deviation) optimization, Optimizing for getting highest IR, Mean Variance Forecasting (MVF) optimization etc.

7. Conclusion

The project's motivation was to analyze the LSTM RNN model and SVR model predictions for constructing different portfolios using different Optimization techniques. The Active Portfolio management just using any DL/ML techniques doesn't necessarily gives us the exact estimations for the investments because market is so complex to be able to analyze using only one method. But here we tried to analyze the approach in an attempt to construct an efficient portfolio with higher returns as lower risks. LSTM RNN gives better predictions over SVR but needs a long time to train the model. The Sharpe ratio of optimized portfolios are greater than the equally weighted portfolios. The references and this paper's results are inline with the performance analysis of models.

REFERENCES

[1] Hao, C. Y., Wang, J. Q., & Xu, W. (2013). Prediction-based portfolio selection model using support vector machines. In Proceedings of sixth international conference on business intelligence and financial engineering (pp. 567–571).

[2] Yilin Ma, Ruizhu Han , Weizhong Wang.(2020). Portfolio optimization with return prediction using deep learning and machine learning.

[3] Jain L C, Medsker L R. Recurrent Neural Networks: Design and Applications. Joint Conference on Neural Networks. 1999:1537 – 1541

[4] Jia H. Investigation Into The Effectiveness Of Long Short Term Memory Networks For Stock Price Prediction. 2016.

[5] Tingwei Gao, Yueting Chai, Yi Liu.(2020) Applying Long Short Term Memory Neural Networks for Predicting Stock Closing Price.

[6] Fischer, T.; Krauss, C. Deep learning with long short-term memory networks for financial market predictions. Eur. J. Oper. Res. 2018, 270, 654–669.

[7] Van-Dai Ta , CHUAN-MING Liu and Direselign Addis Tadesse,2020. Portfolio Optimization-Based Stock Prediction Using Long-Short Term Memory Network in Quantitative Trading.

[8] S. Selvin, R. Vinayakumar, E.-A. Gopalakrishnan, V. K. Menon, and K.-P. Soman, "Stock Price Prediction Using LSTM, RNN and CNNSliding Window Model," in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI 2017), 2017, pp. 1643–1647.