Portfolio Optimization-Based Stock Prediction Using Long-Short Term Memory Network in Quantitative Trading

Authors: Van-Dai Ta, Chuan-Ming Liu, Direselign Addis Tadesse

Journal: Applied Sciences, Received: 1 December 2019; Accepted: 4 January 2020; Published: 7 January 2020

Project Report Submitted by

Shreyas Kowshik (17MA20039)
Yaksh Patel (17ME3FP01)

in partial fulfillment of the requirements for subject Optimization Methods in Finance as Term Project

Under guidance of

Prof. Geetanjali Panda
Department of Mathematics, IIT Kharagpur

November, 2021

METHODOLOGY

The following was the overall approach :

1. The data was first split into training-validation-testing sets. The size of the split was a hyperparameter and was varied between 1%-10% of the total data size across iterations.
2. Based on the split, predictive models were fit on the training data. The following three models were fit :
    a. LSTM
    b. Linear Regression
    c. Support Vector Regression
3. After fitting these models, based on the forecasted prices, the top 4 stocks having the best average returns on predictions were selected to form the portfolio. Portfolio allocation was then performed on each of these.
4. The hypothesis to be tested was the following : If one uses future forecasts in the process of portfolio allocation and selects assets based on these forecasts, can one get an overall better return, risk-adjusted return in future. To thus test this, Markowitz models were built over two cases :
    a. Using the training-validation and predicted values using the forecasting models
    b. Using the training-validation data only. This case accounts for the situation when one does not any forecast for the future
5. Markowitz models optimize the portfolio allocation scheme by solving a Mean-Variance Optimization problem. To further do an ablation study, equal-weighted portfolios were constructed with equal weights allocated for each asset. These were then compared with the Markowitz model weights.
6. Based on the weights obtained, metrics of returns, risks and sharpe-ratios were computed on the test data. These were then compared and plotted as shown in the plots below.

Data Preprocessing :

Data was standardized by subtracting mean and dividing by standard deviation of the training set. This is to aid in training and fitting of the prediction models.

Training of Models :

LSTM based approach: LSTM is a deep learning based neural network architecture that is used for sequential decision making problems. It has a complicated mathematical structure designed in order to avoid vanishing and exploding gradient

problems([1]). In our setup, we used a window-size of 5 timesteps to look at, in order to predict one time step into the future. One of the requirements for deep-learning models like a LSTM to do well is the presence of large amounts of data. The more the data available for training, the better these models do. Moreover, these models have a drawback of being non-explainable in their decisions. In other words, there is no standard way to statistically test if these models are necessarily going wrong during prediction phases. Though more sophisticated approaches combining bayesian statistics and neural networks([2]) have recently come up, using these was out of scope of this work. This was also not considered in the original paper.

LR and SVR: Linear Regression and Support Vector Regression were applied with a window-size of 5 for forecasting one time step into the future. These models have the advantage of being statistically interpretable, in the sense that any mistakes made by these models can be mathematically analysed. This property is of utmost importance in finance since there is a lot of money at stake during the trading period and it is extremely significant to know if one can trust the predictions made.

Portfolio Construction :

A total of 10 different portfolios were constructed varying the forecasted period from 1%-10% of the total data points. For instance if the total dataset has 250 timestamps, the forecasting horizon is varied from 3 days to 25 days. The top 4 performing stocks were selected from each and portfolio allocations were done.

For training the LSTM models, we used early-stopping using the validation set to prevent overfitting of the models to the training data. A hidden layer size of 200 was used for the LSTM model. The models were trained for 3000 epochs or iterations. This process was quite time consuming since LSTM is a deep-learning based approach which requires a lot of computation to run. Since we did not have any servers/GPU-compute to speed things up, we ran everything locally, which took a lot of time for the training of the LSTM models.

For the Markowitz model, we assume no constraint on return. Short selling was allowed and the only constraint on the weights was that they must sum to one.

EXPERIMENTS

Dataset - 1

Markowitz Portfolio

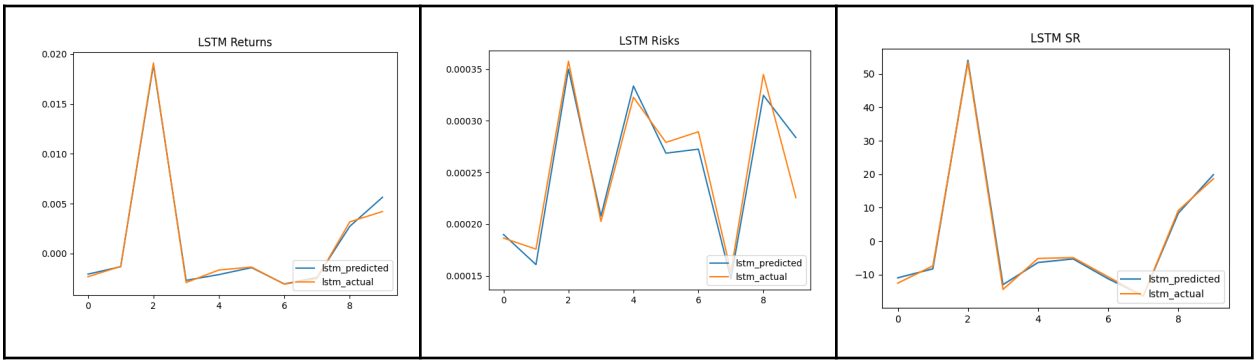


Fig 1. LSTM

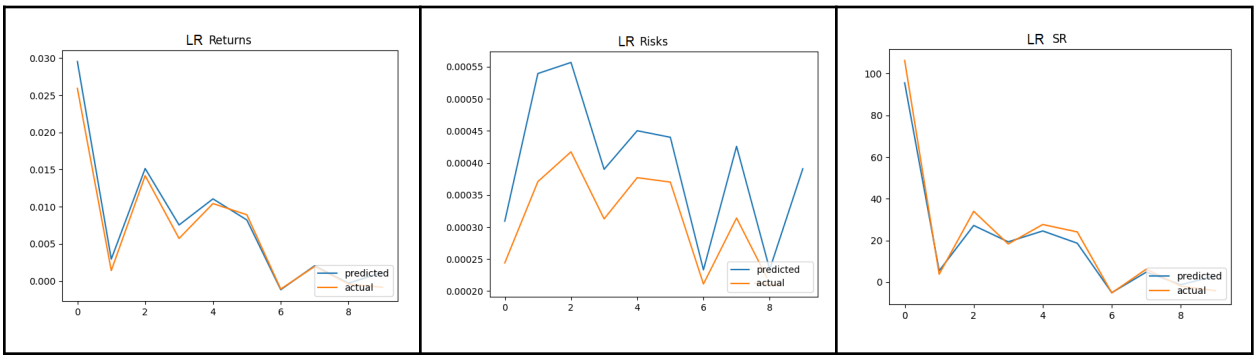


Fig 2. LR

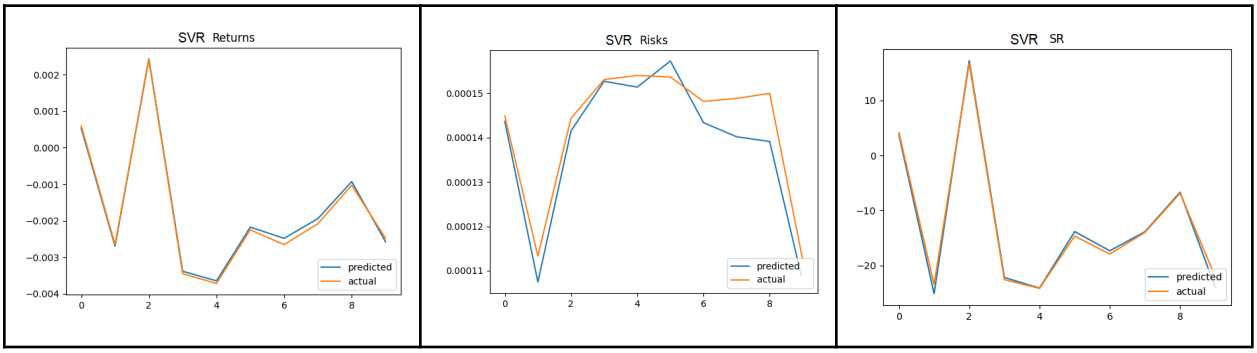


Fig 3. SVR

Equal-Weighted Portfolio

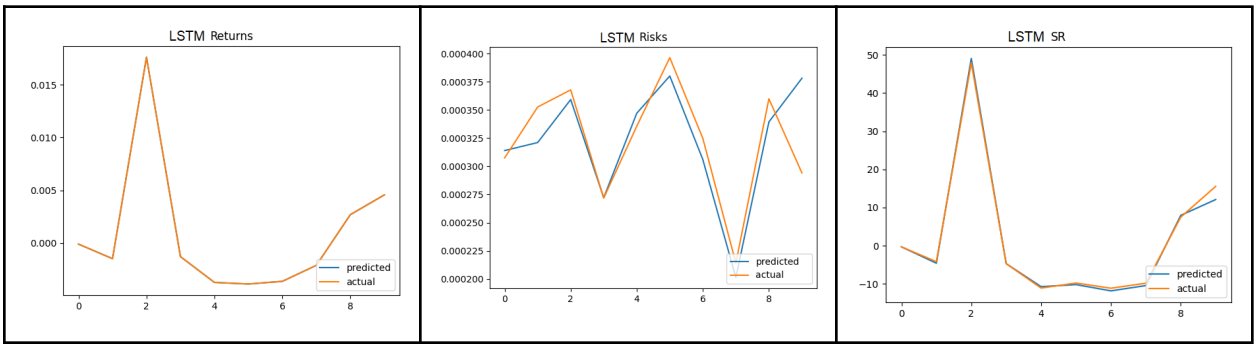


Fig 4. LSTM

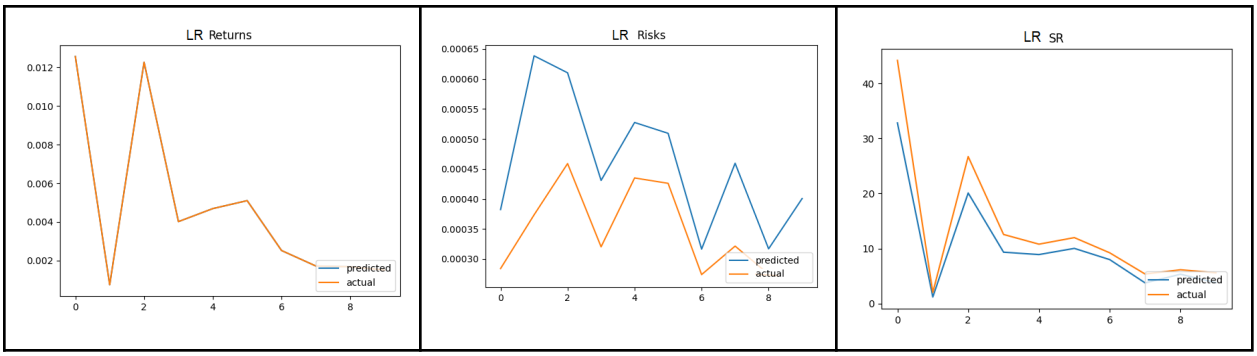


Fig 5. LR

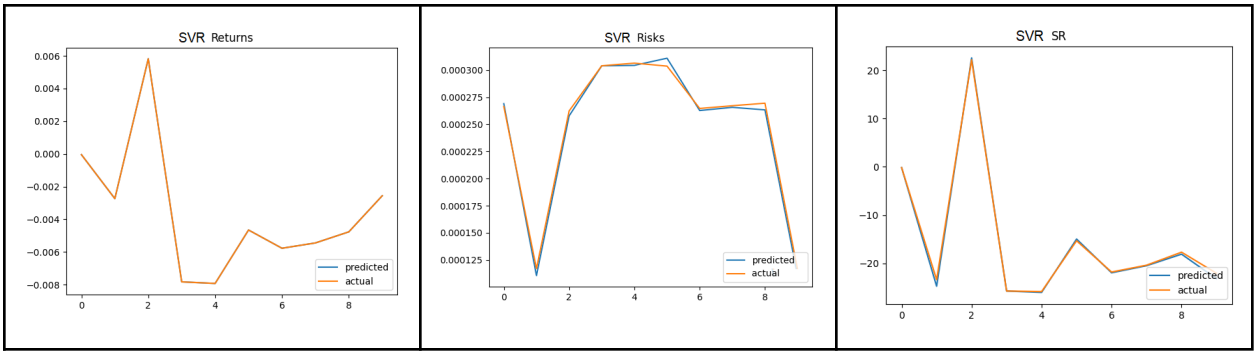
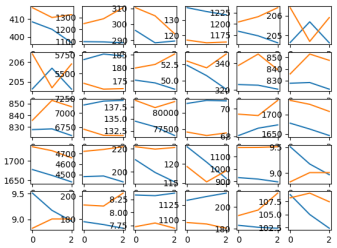
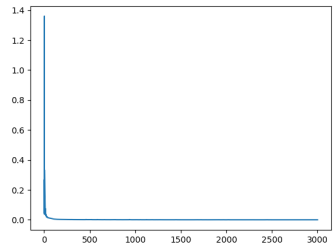


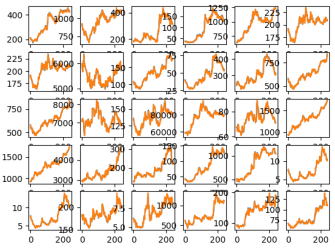
Fig 6. SVR



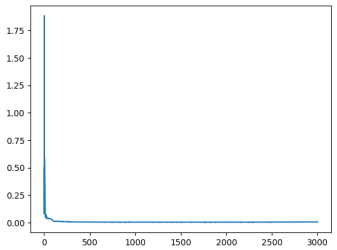
Forecast on Test data



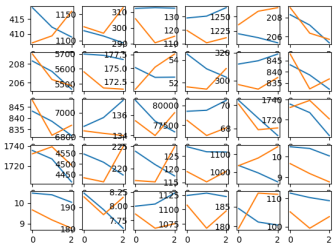
Training Loss



Forecasts on training set

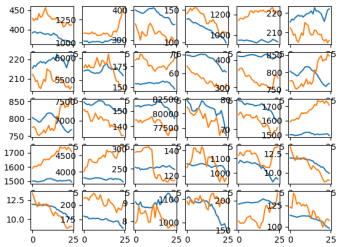


Validation loss

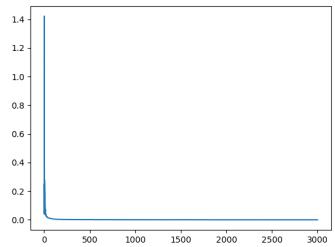


Forecasts on validation set

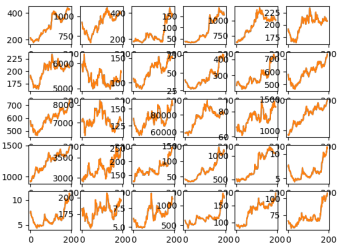
Fig 7. LSTM P1 (1%) Plots



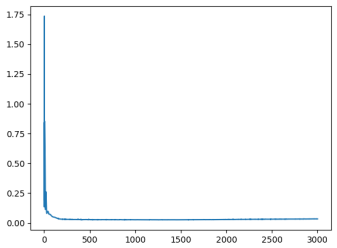
Forecast on Test data



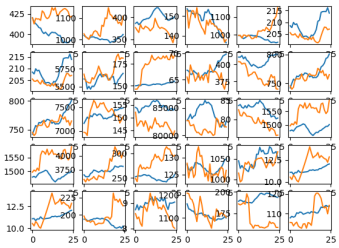
Training Loss



Forecasts on training set

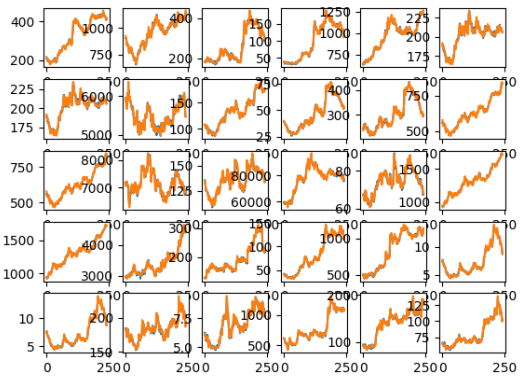


Validation loss

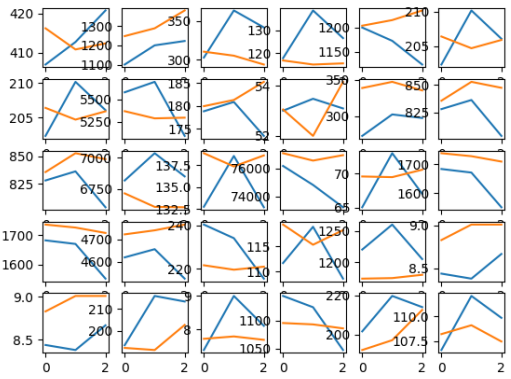


Forecasts on validation set

Fig 8. LSTM P10 (10%) Plots

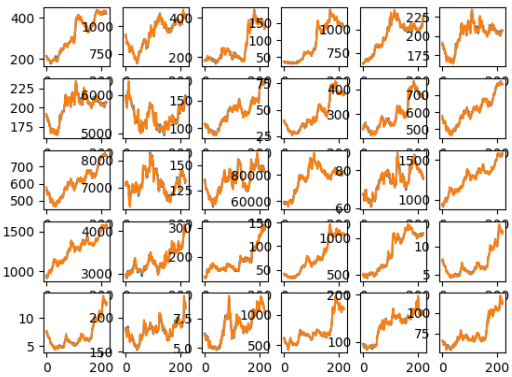


Forecast on training set

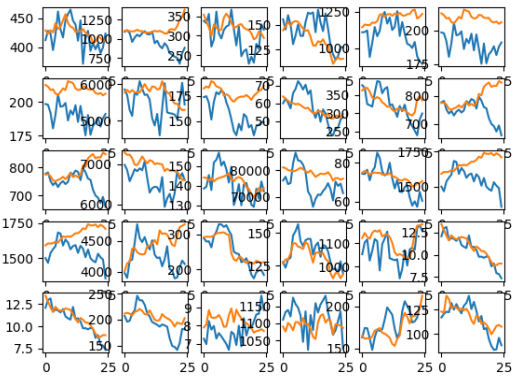


Forecast on test set

Fig 9. LR P1(1%) Plots

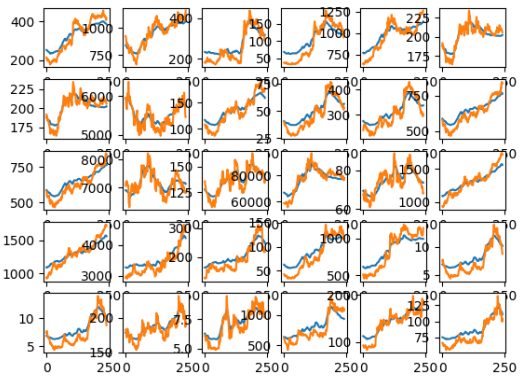


Forecast on training set

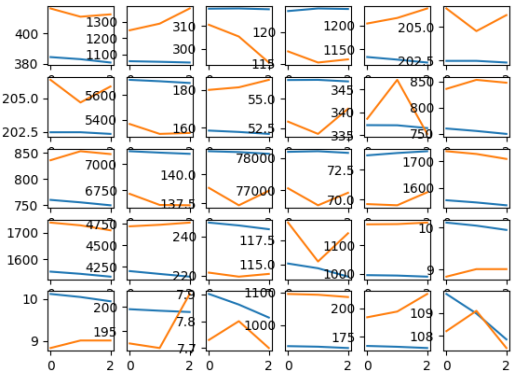


Forecast on test set

Fig 10. LR P10(10%) Plots

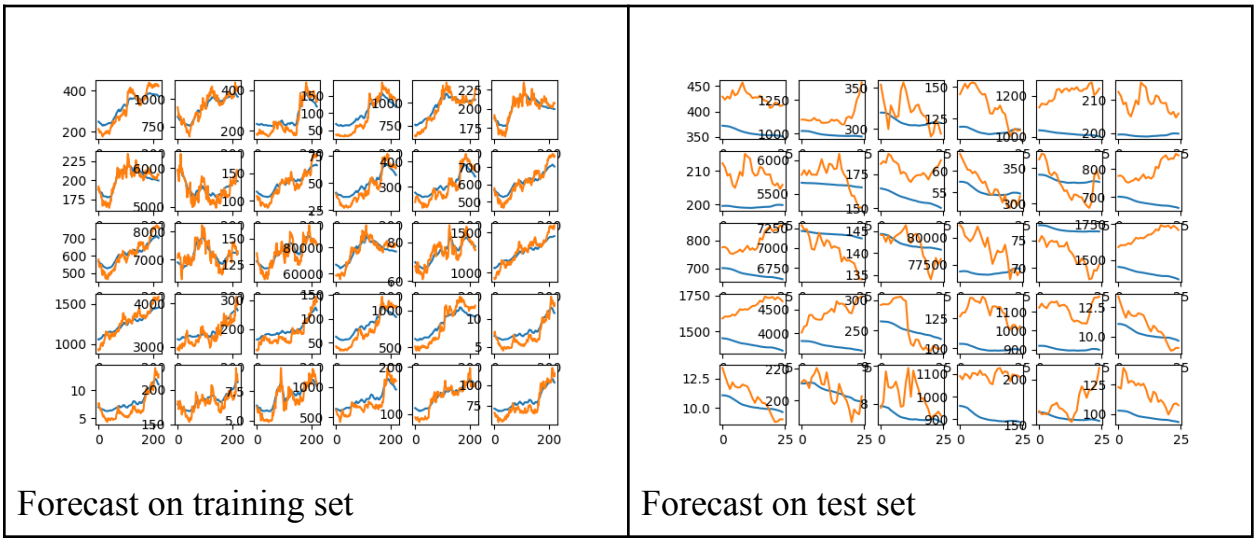


Forecast on training set



Forecast on test set

Fig 11. SVR P1(1%) Plots



Forecast on training set | Forecast on test set

Fig 12. SVR P10(10%) Plots

Dataset - 2

Markowitz Model

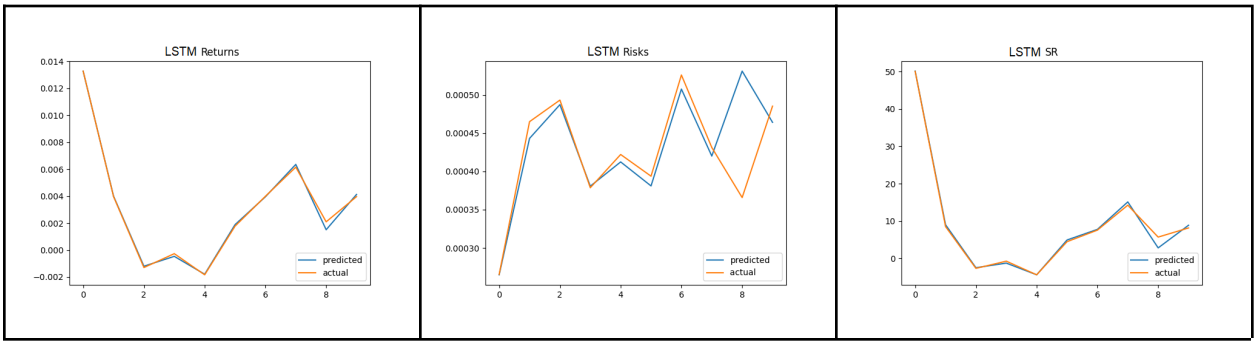


Fig 13. LSTM

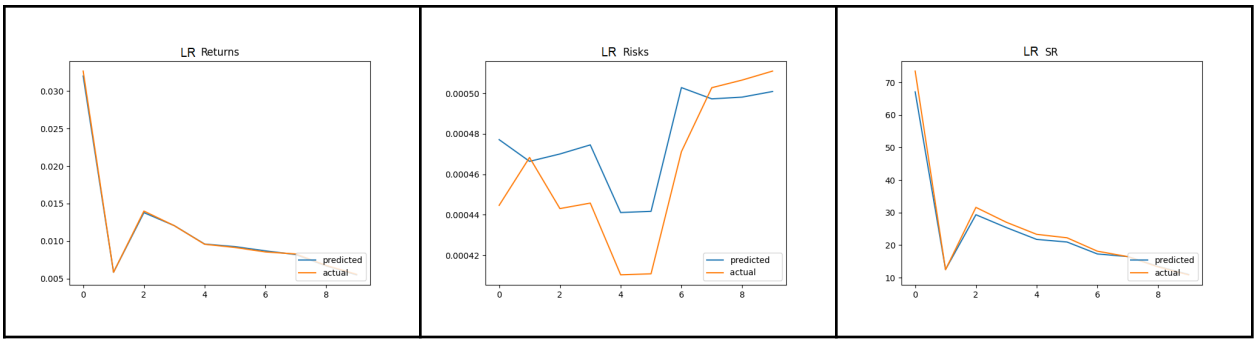


Fig 14. LR

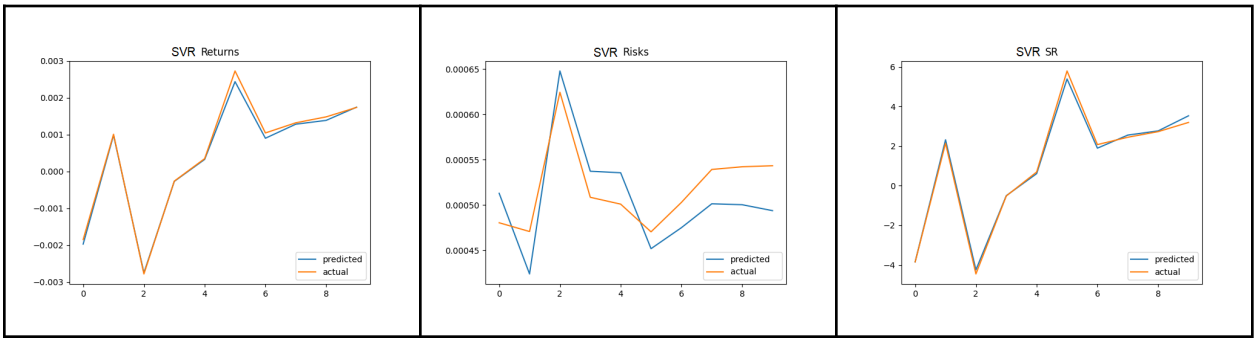


Fig 15. SVR

Equal-Weighted Portfolio

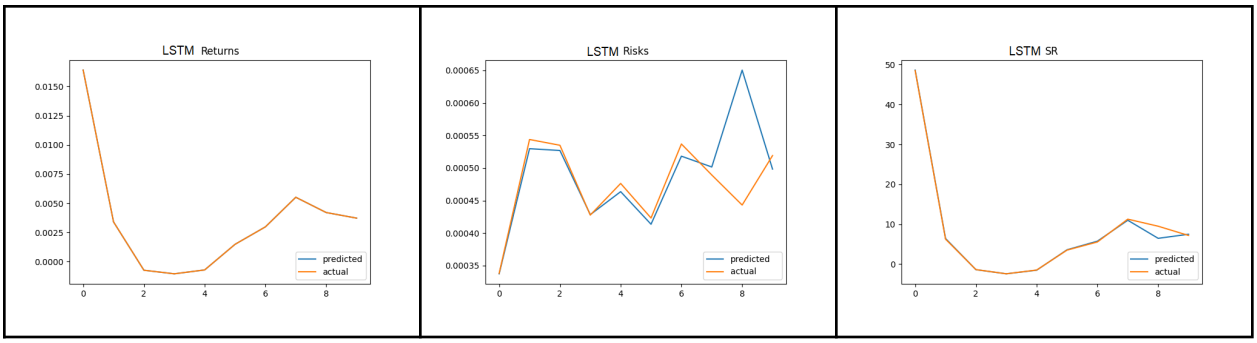


Fig 16. LSTM

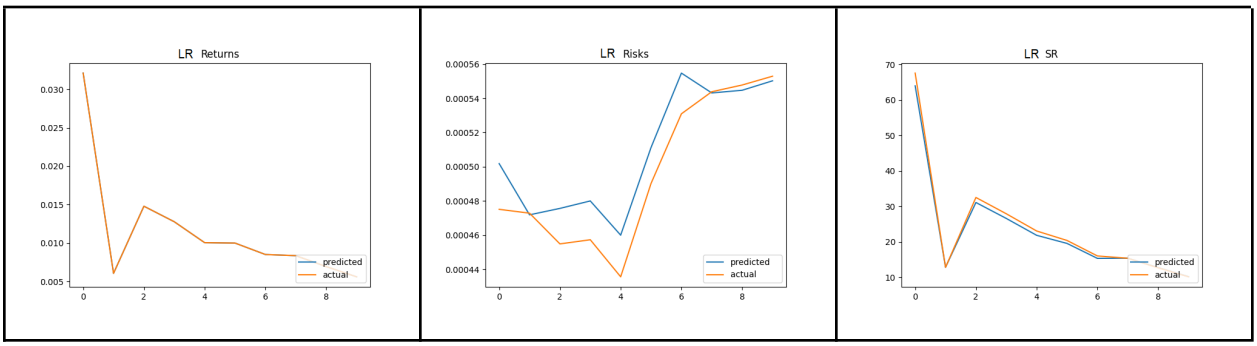


Fig 17. LR

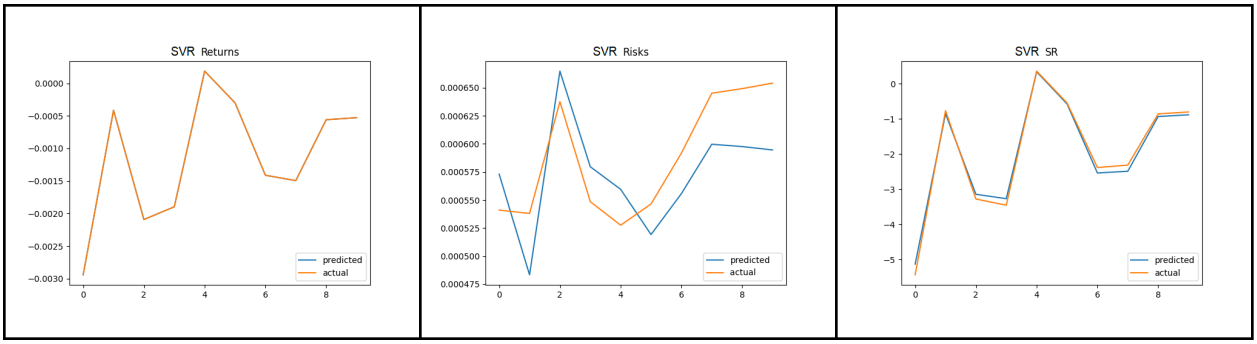
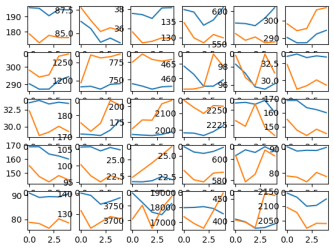
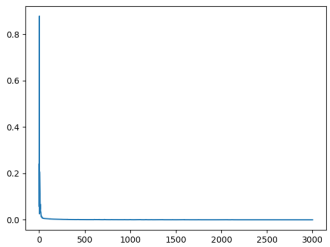


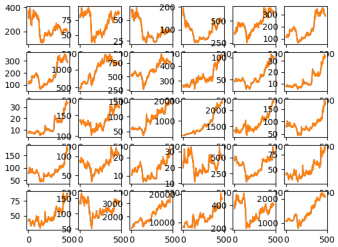
Fig 18. SVR



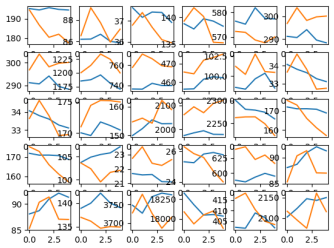
Forecast on Test data



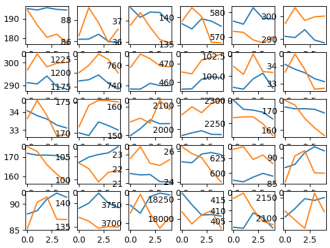
Training Loss



Forecasts on training set

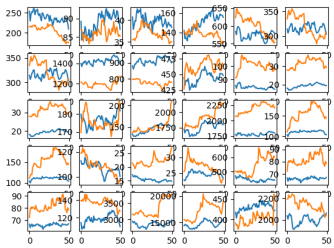


Validation loss

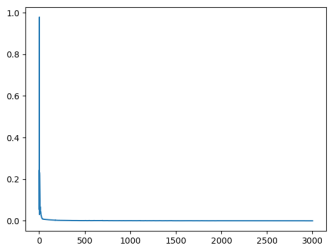


Forecasts on validation set

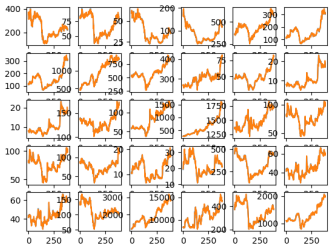
Fig 19. LSTM P1 (1%) Plots



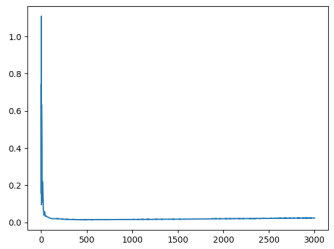
Forecast on Test data



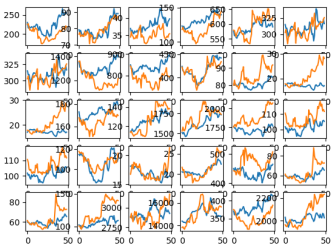
Training Loss



Forecasts on training set

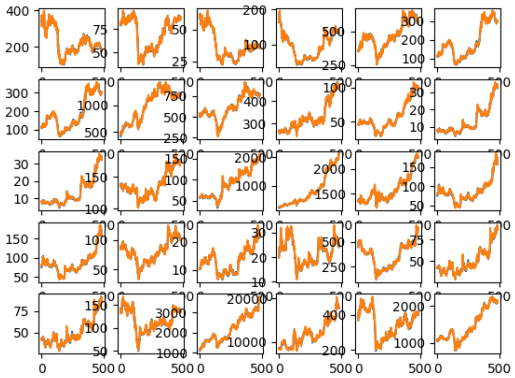


Validation loss

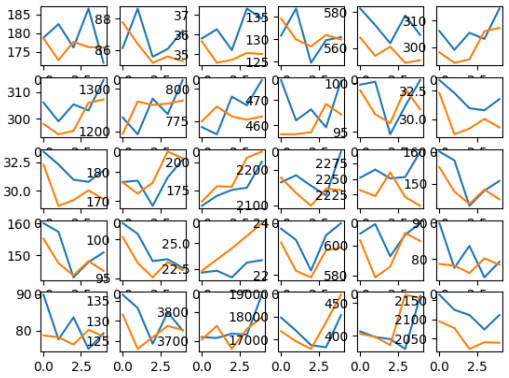


Forecasts on validation set

Fig 20. LSTM P10 (10%) Plots

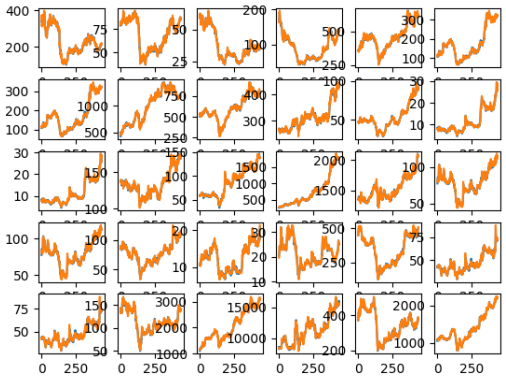


Forecast on training set

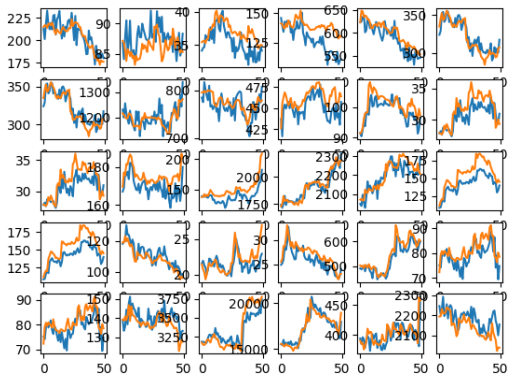


Forecast on test set

Fig 21. LR P1(1%) Plots

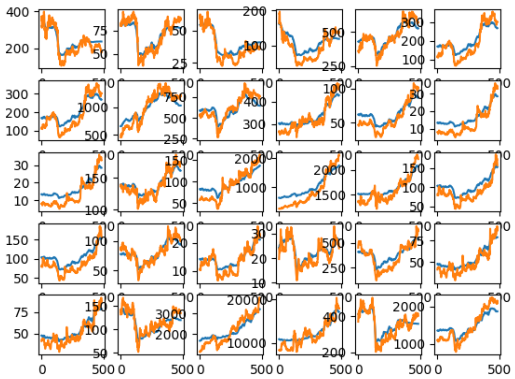


Forecast on training set

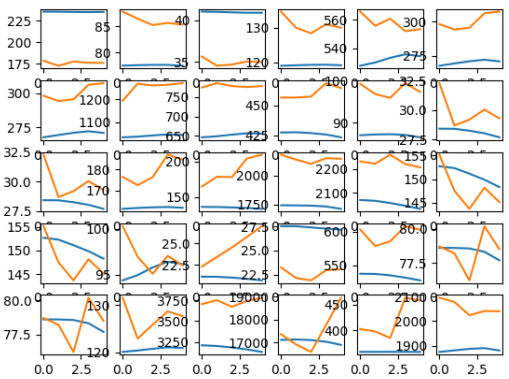


Forecast on test set

Fig 22. LR P10(10%) Plots

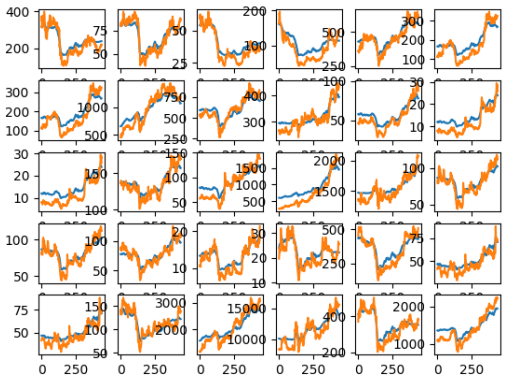


Forecast on training set

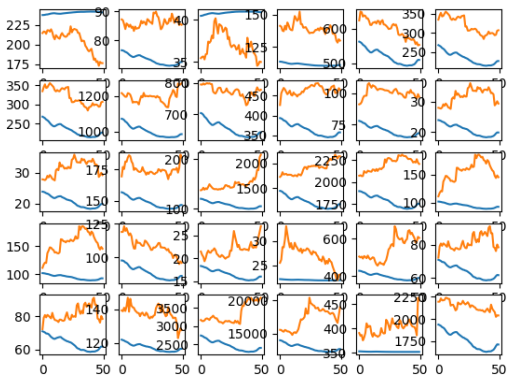


Forecast on test set

Fig 23. SVR P1(1%) Plots



Forecast on training set



Forecast on test set

Fig 24. SVR P10(10%) Plots

CONCLUSION

- **Forecasts**: Looking at the predictions for the P1 and P10 portfolios, we can see that all models fit the training data pretty well. LSTM and LR fit it almost perfectly while SVR is not that accurate there. On the test sets, the forecasts of LR are the best, followed by LSTM and SVR. This can be explained by the size of the dataset that we used. LSTMs need a lot of data to capture the patterns otherwise leading to an overparameterized model that can lead to overfitting. This is what we believe has been the cause of the poorer performance of LSTMs compared to LR here. Linear Regression on the other hand is a statistical fitting procedure that is well parameterized for the given prediction problem, as is also evident in the results. These predictions clearly do have a strong correlation with the test-set portfolio evaluation metrics as we see. We believe by increasing the amount of data for training, the LSTM model should eventually outperform all other models.

- **Performance**: For LR and LSTM, we obtain better Sharpe Ratios on a wide array of portfolios (P1-P10). Returns in general were better for both LSTM and LR compared to the model that does not use predictions. On some portfolios, however, using the predictions does not lead to an improvement, which can be attributed to the forecasting errors of the individual models.

- **Picked Stock Properties**: Stocks and eventually the portfolio constructed using the LSTM models was more risk-averse, we observed. Linear Regression on the other hand is aggressive in terms of the returns but also leads to higher risks of the final constructed portfolio.

- **SVR Performance**: SVR based portfolios in general did not do that well. These can be attributed to the poor predictive performance of these models, leading to incorrect estimations of the stock behaviors. This was also observed in the paper and our results are in accordance with them.

- **Scalability of LSTM**: One noteworthy thing to observe is that the LSTM based forecasting models scales with more number of samples as well as more data. For instance, if we get more data from twitter sentiments, market news, how a particular industry is performing or other economic indicators, we can improve the performance of the model by the capture of correlations between these quantities and the stock performances. These extra data points can easily be

incorporated into the LSTM training-evaluation setup, which is highly desirable in real world trading scenarios.

- **LSTM is compute intensive**: LSTM is a compute intensive model. In our experiments, we ran 10*2=20 different LSTMs for 3000 iterations which took a lot of time to train locally. With more samples, this time will also increase. Thus one may need GPU-compute/servers to speed-up this process. Moreover GPU-compute also reduces inference time of these models. In a real world setup, one may thus need to invest in this infrastructure for better performance.

CODE

Our programs were divided into multiple files, each being quite lengthy. Thus, instead of pasting code in the report we have attached a folder with the submission containing all code files. Also, to maintain the flow of the report we thought that this would be in the best interest for the us and evaluator too.

Following are the descriptions of code files:

- **construct_eq.py**: Construct equal-weighted portfolio from the chosen stocks of a model
- **construct_portfolio.py**: Construct markowitz model portfolio from the chosen stocks of a model
- **lr_asset_selection.py**: Fit linear regression model to data and pick top 4 performing stocks in terms of returns. Also plots all the metrics and predictions.
- **lstm_asset_selection.py**: Fit LSTM model to data and pick top 4 performing stocks in terms of returns. Also plots all the metrics and predictions.
- **svr_asset_selection.py**: Fit support vector regression model to data and pick top 4 performing stocks in terms of returns. Also plots all the metrics and predictions.
- **run_lr_train.sh**: Bash script to run multiple experiments sequentially for training Linear Regression models on 1%-10% of the data.
- **run_lstm_train.sh**: Bash script to run multiple experiments sequentially for training LSTM models on 1%-10% of the data.
- **run_svr_train.sh**: Bash script to run multiple experiments sequentially for training Support Vector Regression models on 1%-10% of the data.

REFERENCES

[1] LSTM : https://www.bioinf.jku.at/publications/older/2604.pdf
[2] Bayesian LSTM : https://arxiv.org/pdf/1712.08773.pdf