Machine Learning "M2GRA": Forecasting of a financial time-series for S&P500 equity index using machine learning techniques

Youssef LOURAOUI
M2 Risk and Asset Management
Université Paris-Saclay
youssef.louraoui@essec.edu

François PEDEBOY
M2 Risk and Asset Management
Université Paris-Saclay
francois.pedeboy@etud.univ-evry.fr

Abdelkader Bousabaa
Associate Professor of Finance
Université Paris-Saclay
abdelkader.bousabaa@univ-evry.fr

25/03/2024

1 Introduction

This project aims to forecast a time series of financial prices taken from the S&P500 index with its underlying equities and make predictions on the future prices using different machine learning techniques that have been developed in the machine learning module.

In order to perform the forecast and run the different machine learning techniques, we first start by handling the dataset, cleaning any missing observations or data that could bias the model output in order to offer better insights. After importing the dataset of S&P500 data covering 41265 observations with its underlying stocks, we made a preliminary analysis of the return profile and characteristics. From the preliminary analysis of the dataset, we filtered the column into seconds to better interpret the date column. From the figure 5a, we can see that the S&P500 was evolving in an uptrend. The data starts at roughly 2360 and finishes above this level reaching 2480 price level, representing a increase of 5.08% across the period analysed. From the drawdown plot 7 for the index, we can see that the index losses where fluctuating in a narrow corridor of zero and nearly 3% during that period. In terms of cumulative return 6 , the S&P500 index managed to roughly make around 5% over the analysed period. Looking at the histogram 5b, we can see that the returns are all close to zero. This means that during this trading day, the price of the S&P 500 did not fluctuate much and stayed around zero. It is normal for a very liquid market index like the S&P 500 to be centred around the mean. This is because price changes are usually small when looked at over short periods of time. We can therefore assume there was no big economic or news events that impact the index level. The distribution is leptokurtic and shows that extreme returns, both positive and negative, happen less often but are still there. These outliers are times when prices suddenly go up or down. This could be because of events happening in real time or big trades happening during the day. Also, the fact that the histogram is symmetric around the centre axis shows that the S&P 500's price went up and down almost equally many times during the period. It is worth mentioning that the histogram only shows how often returns happen, but not how big they are. The effect on an investor's portfolio could be very different based on the size of the returns, even if the amount of positive and negative returns is equal. After analysing the dataset and analysing the overall performance, we can dive into the different parts of the report and elaborate more on the methodology and results obtained. The structure of the project is based on the following:

- **Implementation of MLP and LSTM models on S&P500 data**: In this part of the project, we focus on applying MLP and LSTM machine learning models to forecast the S&P500 price based on a training period of 250 data points and forecast for the rest of the training period.
- **Implementation of MLP and CNN on the underlying equities**: In this part of the project, we perform the same procedure as in the first part but the approach entails forecasting the 500 different stocks in the dataset in order to produce a forecast.
- **Implementation of alternative machine learning models**: For the third part, we dive deeper into alternative machine learning techniques and implement them into S&P500 data in order to perform forecasting and compare the different machine learning methods in terms of their accuracy compared to the training dataset.

2 Implementation of Multilayer Perceptron (MLP) and a Long-Short Term Memory (LSTM) model on S&P500 data

In this section, we implement the MLP and LSTM on S&P500 prices and compare results.

The Multilayer Perceptron (MLP) is based on the following structure. The model use the sequential function to be able to manipulate this model in a more personalised way. Using the dense function, each layer of this perceptron is fully connected in the sense that the neurons at one layer are fully linked with the neurons of the previous layers. The rectified linear unit (ReLU) serves as an activation mechanism to be able to trigger the neurons of the model. This method introduces non linearity which comes very handy when performing analysis on complex data structures. The model drops over 25% of the neurons in the training set in order to preserve the results form overfitting, a common pitfall in machine learning. Finally, this perceptron aims at forecasting S&P500 prices and so we have a final dense function equal to one which indicates that the model trains only one continuous data for this purpose. The MLP model is set to run for 10 epochs, meaning the entire dataset will pass through the MPL model 10 times with a batch size at 16. This means the model's weights will be updated after every 16 samples are processed.

The Long-Short Term Memory (LSTM) model is based on the following structure. The model, just like the MLP, use the sequential function to be able to manipulate this model in a more personalised way. The model can be deconstructed into two distinct parts. The first LSTM layer has 128 neurons and is set to return sequences. This means that it will return the full sequence to the next layer rather than just the output of the last time step, an essential step when stacking LSTM layers. The second LSTM layer has 64 neurons and does not return sequences, which indicates this is the last recurrent layer and will only pass on the last step's output to the next layer. The model, just like MLP, drops over 25% of the neurons in the training set in order to preserve the results form overfitting. Finally, the LSTM adds two dense layer. The first has 25 neurons with ReLU activation, which introduces non-linearity to the model. The final dense layer has 1 neuron with no activation specified, which is appropriate for regression-type problems where we predict a single continuous value. The LSTM model is set to run for 10 epochs, meaning the entire dataset will pass through the LSTM model 10 times with a batch size at 16. This means the model's weights will be updated after every 16 samples are processed.

From the figure 1, we can see that both models managed to capture the overall shape of the S&P500 prices, but we cannot determine which model performed better based only on the figure on itself.

The table 1 presents the results of the model in terms of Mean Average Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). For the statistical analysis, these metrics can be analysed on the following basis. Regarding the MAE, it measures the average size of the errors in a set of forecasts, without considering the direction of errors. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight. For the MSE, it is the average of the squares of the errors. This will mechanically place a higher weight on larger errors. This is because the squaring of each difference magnifies the impact of larger differences. RMSE is the square root of MSE and

Figure 1: MLP and LSTM forecast of S&P500 price time series.

represents the sample standard deviation of the differences between predicted values and observed values. Across all the metrics, the lower the value, the better the result.

The results obtained for both MLP and LSTM are close enough and the differences are marginal. The LSTM performs slightly better across the three performance metrics 1.

Metric	**MLP Model**	**LSTM Model**
MAE	8.71	7.88
MSE	79.89	78.41
RMSE	8.94	8.85

Table 1: Forecasting performance comparison between MLP and LSTM models

3 Implementation of MLP and Convolutional Neural Networks (CNN) on equities

For this section, we implement a MLP and CNN model for the equities in the dataset and compare forecasting power.

As with the previous implementation of MLP on S&P500 prices, we implement a similar procedure applied to the stocks in the dataset. It initialise a sequential model based on a linear stack of

layers. The data flattened and is then converted into a 1D array. The input sample has 500 features. Same as before, the model drops 25% of the neurones to preserve the model from overfitting. The model is run ten times (10 epoch) and with a batch size at 16. This means the model's weights will be updated after every 16 samples are processed.

Convolutional Neural Network is a machine learning technique, precisely a type of deep learning model that can be used across different usages [1]. Regarding the characteristics of the model, it is based on two 1-dimensional convolutional layers (Conv1D) which are added with 64 filters each, a kernel size of 3, and a ReLU activation functions. The usage of Conv1D layers are typically used for sequences, such as time-series data in this instance. The model also has a 1-dimensional max pooling layer (Maxpooling1D) with a pool size of 2. Max pooling is a downsampling technique used to reduce the dimensionality of the input. It helps to prevent overfitting and reduce computational load. In addition to that, the model, just like the previous implementations, drops out 25% of the neurons (units) to prevent overfitting of the model. The model use a flattening layer to harmonize the input so that it can be fed into a dense layer. It is a necessary step because convolutional and max pooling layers output multi-dimensional arrays, but dense layers expect 1-dimensional arrays as input. The final dense layer outputs a single continuous value, which suits best for a regression task. The model is run ten times (10 epoch) and with a batch size at 16.

Metric	**MLP Results**	**CNN Results**
MAE	599.99	16.42
MSE	360050.77	320.54
RMSE	600.04	17.90

Table 2: Comparison of MLP and CNN models performance metrics

From table 2, we can clearly see that the MLP model was terrible at forecasting stock prices, with a very wide margin of difference when compared to CNN, which managed to capture the overall price forecast that match to the actual prices from the original dataset. The CNN model outperforms the MLP model by a very large difference for this particular task. The errors from the CNN model are considerably smaller, meaning its predictions are more accurate. Graphically, we can see from figure 2 that there is a very wide difference between MLP forecast and actual prices.

Implementation of an enhanced MLP on equities

In order to improve the results of the MLP, we added more epochs in order to see if it would actually enhance the results of the prediction. The idea for increasing the number of epochs would help the modle to train more times and therefore improve the output. Training the MLP for more epochs typically allows the model to learn the data better and would help to achieve better performance. By increasing the number of epochs from 10 to 30, the model has more opportunities to adjust its weights on the training data, potentially lowering the training and validation loss if the model had not yet converged after 10 epochs. However, the risk of increasing the number of epoch is that the model falls in an overfitting of the data and bias the results. If the increase in epoch has decreased RMSE, then we would argue that this approach helped to enhance the model forecasting

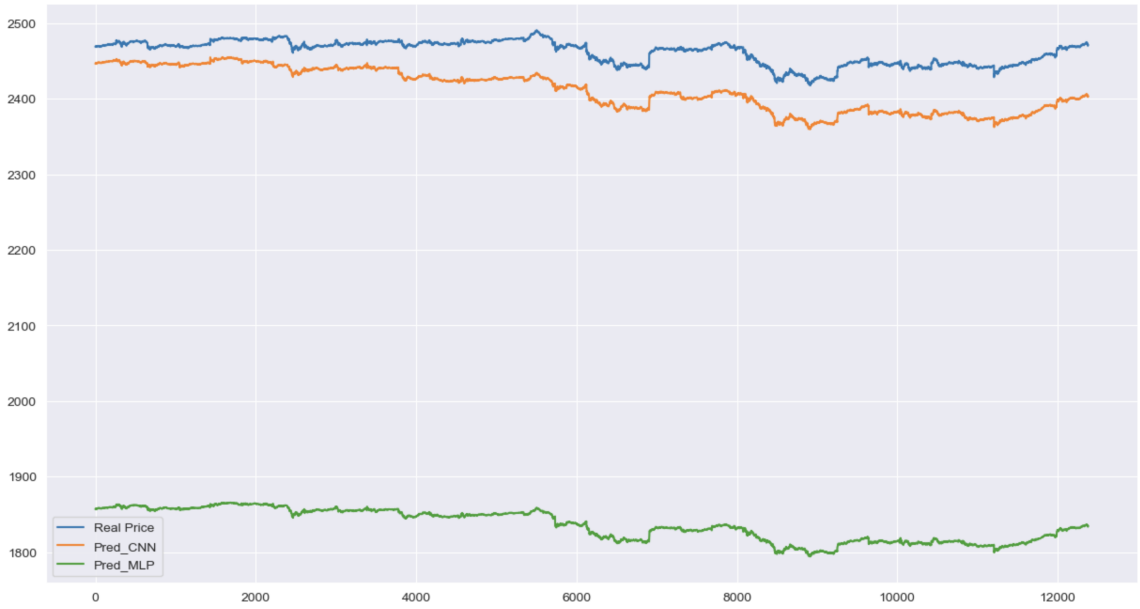


Figure 2: MLP and CNN forecast of S&P500 underlying equity prices.

abilities. If not, then the increase in epoch did not achieve its objective of enhancing the results. Looking at table 3, the modified MLP, which was trained for 30 epochs as opposed to 10 for the simple MLP, actually shows worse performance across all three metrics. The modified MLP has a slightly higher MAE, inidicating that the average prediction error increased with more epochs. The MSE also increased significantly for the modified MLP, which indicates that the additional epochs may have caused the model to overfit the training data, leading to weaker performance on the test data. Consistent with the MSE, the RMSE is higher for the modified MLP, which again suggests a decline in model performance. Visually, figure 3 do not show any sign of model improvement overall.

Metric	**Modified MLP results**
MAE	624.05
MSE	389492.99
RMSE	624.09

Table 3: Performance metrics for the modified MLP model (increase of epoch 10-30.)

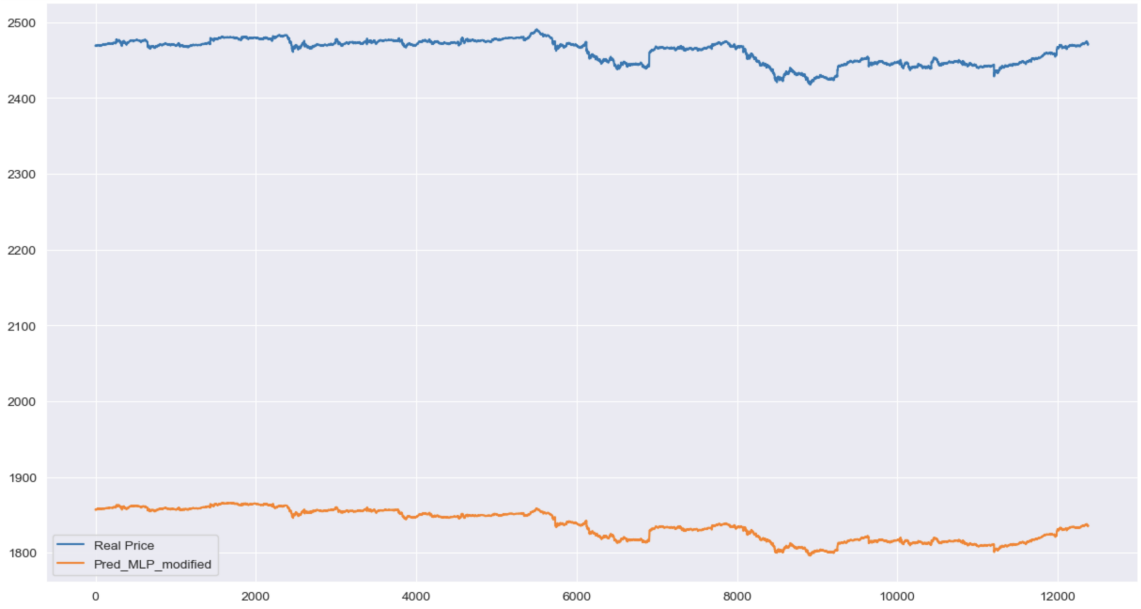


Figure 3: Enhanced MLP forecast on S&P500 underlying equity prices.

4 Implementation of alternative machine learning models: Suport Vector Machine (SVM), Gradient Boosting and Random Forest

We implement three different models to test forecasting of S&P500 price data, namely Suport Vector Machine (SVM), Gradient Boosting and Random Forest, and compare the five machine learning techniques implemented to forecast stock index price.

Suport Vector Machine (SVM) is a non-parametric machine learning technique that uses the concept of Support Vector Regression (SVR) to address regression problems [1]. The C parameter is set to 1.0, controlling the trade-off between the model's complexity (and accuracy on training data) and the degree to which it generalizes.

Random Forest (RF) is a machine learning technique that works by building many decision trees during training and then showing the average forecast of all the trees. When using individual decision trees, especially for regression tasks, the risk of overfitting is higher. The ensemble method lowers this risk. In our project, it is created with 50 trees (n_estimators) in the forest. The random_state ensures that the results are reproducible.

Gradient Boosting (GB) creates an additive model one step at a time, and it can be used to improve any differentiable loss function. It is done by fitting a regression tree to the negative gradient of the given loss function at each stage. This lowers the residuals of the earlier models. The

boosting method focuses on areas where the old models don't do well by adding weak learners one at a time. In our project, the model uses 50 trees with a learning rate of 0.1 and a maximum depth of 3 for each tree.

From the table 4, we can highlight the following observations.

The Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM) models made the most accurate predictions of S&P 500 prices. This shows that these models did a good job of finding the trends in the time-series data. LSTM did better than the MLP and than the whole sample of machine learning techniques with the lowest MAE, MSE, and RMSE numbers One explanation that could promote the performance of LSTM is its ability to capture long-term relationships and sequences in the data, which is common in financial time series like stock prices.

The Support Vector Machine (SVM) with an RBF kernel has worse error rates than the MLP and LSTM. Although SVMs are great for classification and can also be used for regression (SVR), the kernel (C and gamma) have a big effect on how well they work.

Lastly, the Gradient Boosting and Random Forest models have the most important errors and deviate the most from actual prices. Even though robust and powerful techniques like GB and RF can deal with non-linear relationships, they may not have worked as well here because they were overfitted or because the S&P 500 price movements are not predictable and don't follow clear trends.

Model	**MAE**	**MSE**	**RMSE**
MLP	8.7106	79.8918	8.9382
LSTM	7.8847	78.4090	8.8549
SVM	18.9365	437.2680	20.9109
Gradient Boosting	28.8977	1000.9598	31.6379
Random Forest	26.7723	1017.4265	31.8971

Table 4: Forecasting Performance Comparison Across Models

The observations made can be captured visually on figure 4.

Figure 4: SVM, Gradient Boosting, Random Forest, LSTM, MLP forecast on S&P500 price data.

5 References

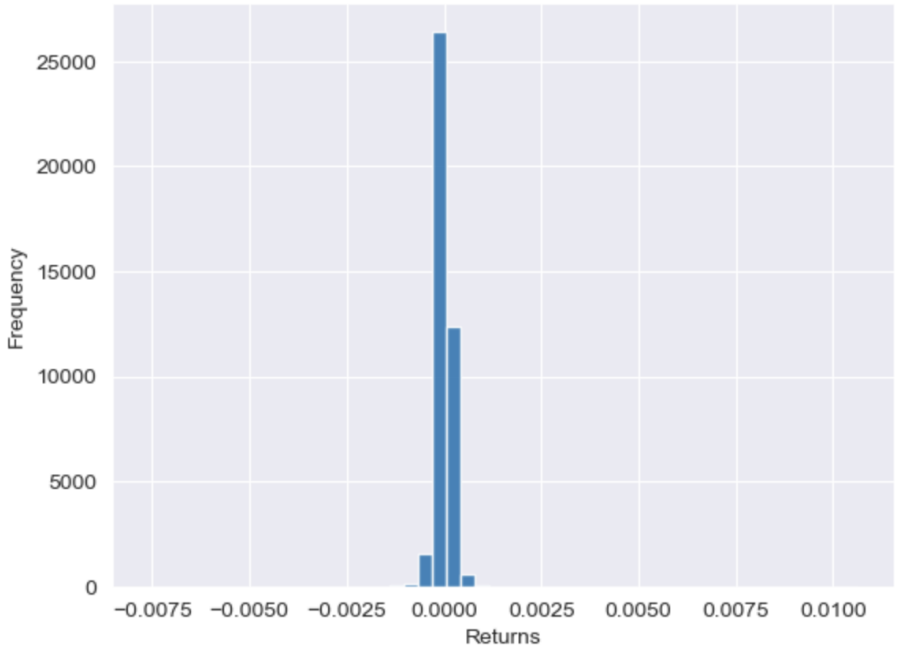
References

[1] Bousabaa, A. (2024). Machine Learning (2024) presentation slides.

‘

Appendices

(a) S&P500 equity index price series.



(b) S&P500 equity index histogram of returns.

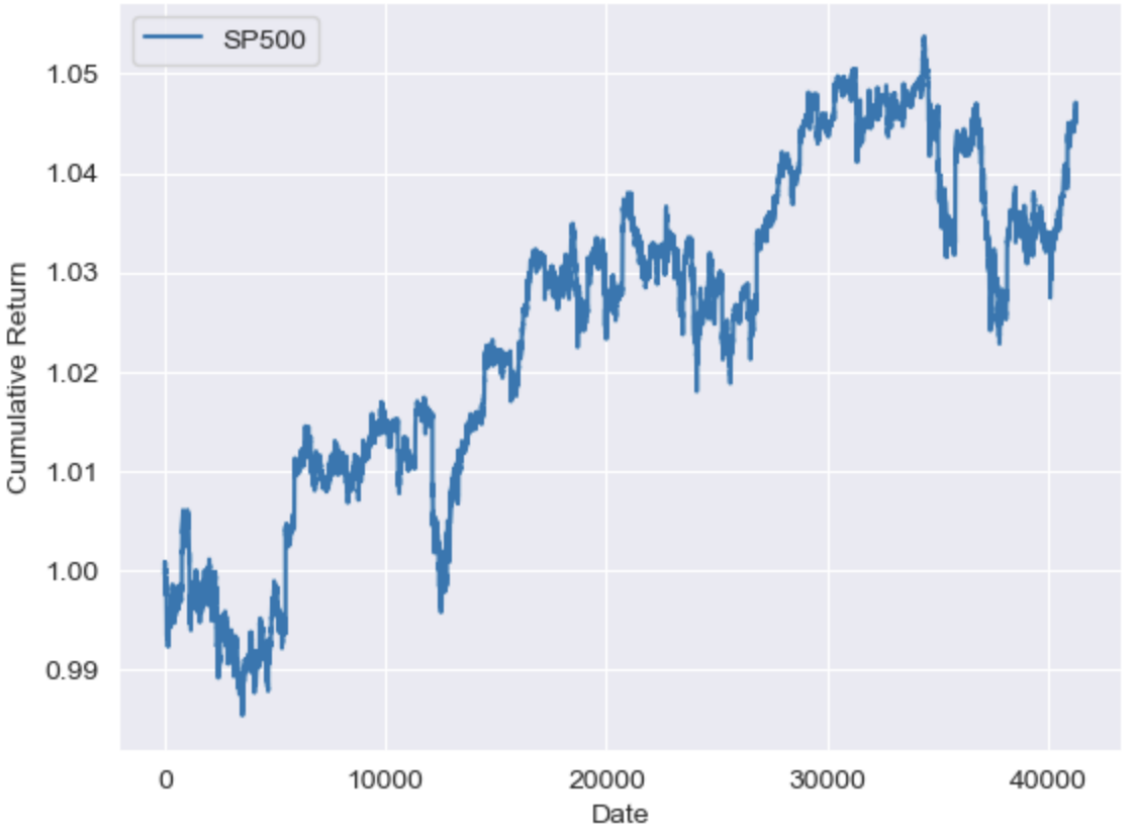


Figure 6: S&P500 equity index cumulative returns.

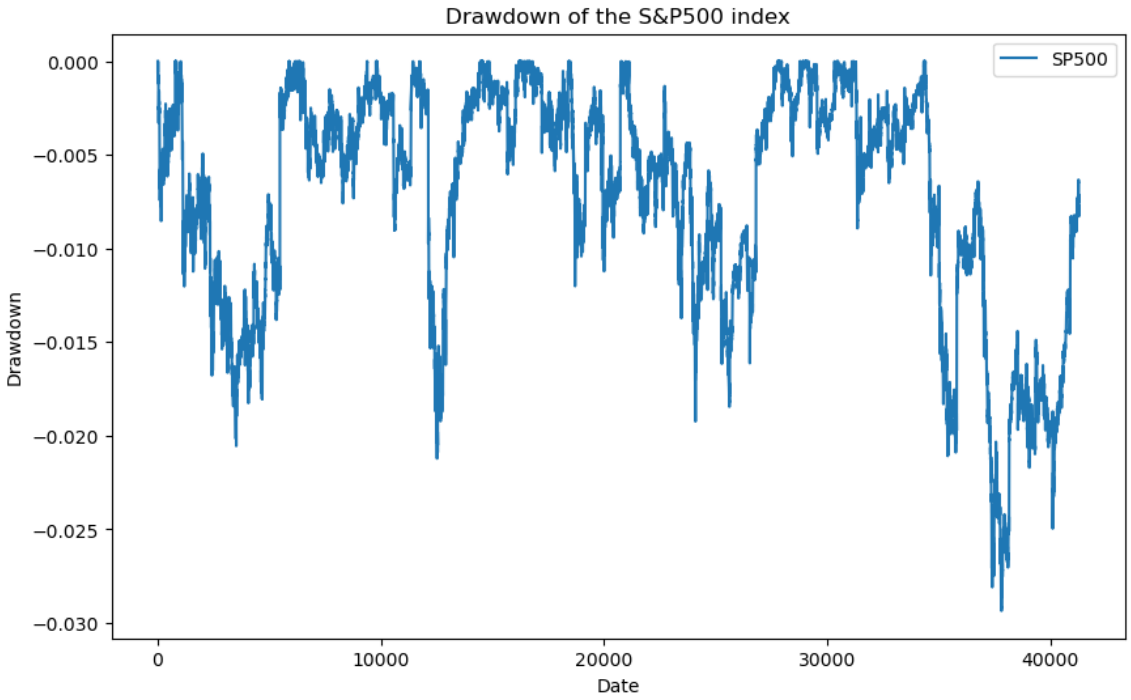


Figure 7: S&P500 equity index drawdown.