Deep Learning Methods in Finance
2023

Group Project - Enhancing Stock Price Prediction of the SMI Index Using Deep Learning Techniques

Ana Rita Azevedo – 20220524

João Gonçalo Morais – 20221237

Madalena Martins – 20221543

Sofia Gonçalves – 20220523

Vitor Santos – 20221238

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Abstract

Accurately predicting stock prices is crucial in financial markets, and deep learning has emerged as a powerful tool to improve prediction accuracy. Traditional statistical models often struggle to capture the intricate relationships inherent in stock market data. In contrast, deep learning algorithms, with their neural networks and multiple hidden layers, excel at extracting complex patterns and representations from raw data.

In the context of predicting stock prices, deep learning offers several advantages. Firstly, it effectively captures the dynamic nature of stock market data by incorporating temporal dependencies, enabling the exploitation of short-term trends and long-term patterns. Secondly, deep learning models are adept at handling large volumes of data, such as historical stock prices, trading volumes, and news sentiment. By integrating diverse data sources, these models can learn intricate relationships and uncover hidden factors influencing the stock price. This comprehensive approach provides a deeper understanding of market dynamics and enhances prediction accuracy.

Deep learning techniques hold promise in accurately predicting stock prices. Their ability to capture complex patterns, process extensive data, and adapt to changing market conditions can empower investors to make informed decisions and potentially gain a competitive edge in the financial landscape.

Key Words: Stock price prediction, Deep Learning, Financial Markets.

Introduction

We focused on predicting stock prices using the "Stock Exchange Data" dataset. From the dataset, which consists of 14 indexes, we specifically chose to predict the stock price of the SSMI Index, representing the Swiss market. The dataset comprises eight columns, namely Index, Date, Open, High, Low, Close, Adjusted Close, and Volume. Each column provides essential information about the stock market, such as the opening and closing prices, highest and lowest prices reached during the trading day, adjusted close price, and trading volume.

For the SSMI Index, our dataset consists of 7,671 rows, capturing a significant amount of historical market data, between 1990 and 2021. To ensure reliable predictions, we embarked on a comprehensive data preparation process. We initially split the data into three subsets: a training set, a test set, and a validation set. This division allowed us to train our models on past data, test their performance on unseen data, and validate their effectiveness.

To construct robust prediction models, we employed recurrent neural networks (RNNs) and long short-term memory (LSTM) models. Leveraging the temporal dependencies inherent in stock market data, these models can capture sequential patterns effectively. We explored several hypotheses and evaluated the performance of multiple models, ultimately, we identified and kept the two models with the best predictive capabilities—one based on RNNs and another based on LSTMs.

In addition, we investigated the influence of dropout layers on the performance of our selected models. By evaluating the impact of dropout layers, our objective was to optimize the performance of these models and make a comparison with the non-dropout models.

By conducting thorough data preparation, exploring various deep learning architectures, and evaluating the significance of dropout layers, our project aims to provide insights into accurate stock price prediction for the SMI Index. Through our findings, we aim to contribute to the field of deep learning in finance and enable investors to make more informed decisions in the Swiss market.

Data Preprocessing

In the pre-processing stage of our project, we found that the stock price dataset required minimal adjustments due to its nature. As stock prices are the focus, we did not encounter missing values, data errors, or outliers. As such, we began by transforming the data type of the "Date" column from object to datetime. Subsequently, we set the "Date" column as the index for easier time-based analysis. Additionally, we added three columns to the dataset, namely "year," "month," and "day," which provided further granularity for time-based analysis.

Considering the one-to-one architectures, we needed to select a specific variable for modeling. To do so, we created a new data frame containing only the "Open" column, which represents the opening prices of the stock, and converted this column into a NumPy array to facilitate further processing. Next, we split the data into three subsets: a training set consisting of 7,311 days, a test set spanning 270 days, and a validation set covering 90 days. This division allowed us to train our models on historical data, evaluate their performance on unseen data, and validate their effectiveness.

Figure 1 – Historical "Open" stock prices for the SMI Index. Light blue represents the training data, green for the validation and yellow for the test.

To ensure consistency and comparability across different variables, we standardized the data using the MinMaxScaler. Standardization is crucial as it brings the data into a consistent scale, preventing any single variable from dominating the modeling process.

Lastly, we prepared the data for training the models using the Time Series Generator from TensorFlow. With the time steps parameter set to 30, we created a generator that looked back 30 steps in the data, capturing past information for each observation. For the batch size, we chose 32 samples, which determined the number of samples fed into the model during each training iteration.

After completing these pre-processing steps, our data was appropriately prepared for training and evaluating the models. We considered that the standardized and time-series structured dataset provided a solid foundation for our deep learning models, enabling them to capture temporal patterns and make accurate predictions based on historical stock price information.

Approach

Deep learning models require extensive amounts of high-quality data to effectively learn and generalize. They can also be computationally intensive, requiring powerful hardware and sufficient memory. Other key factors such as choosing the right architecture and model variants or dealing with the right combination of hyperparameters are crucial to achieve optimal performance.

In this project, we analyze time-series data revealing a sequence over time of stock prices of the SSMI Index and attempt to predict the future behavior of this index. Stock prices exhibit temporal patterns and dependencies, where the current price is influenced by past prices and market conditions. RNNs and LSTMs excel at capturing such temporal dependencies due to their recurrent connections, which allow them to maintain a memory of past inputs.

In the training and modeling phase, we build two distinct architectures – a sequential model 1-to-1 containing at least one SimpleRNN layer and a sequential model 1-to-1 containing at least one LSTM layer.

After building the network's architectures, the approach taken consists of compiling the models, training them, and checking the results against unseen data.

For each test we employ a utility function that will allow us to collect and compare results. It casts several evaluation measures that should be summed up and analyzed in comparison to other models' results.

To correct the shapes to plot the predictions we apply the ravel function. We then plot our predictions to compare with the true labels.

We repeat this process several times, by changing the loss function and the number of epochs, testing 9 variations of both RNN models and LSTM models. We test 3 variations for each parameter (the loss function and the epochs), yielding 9 possible combinations.

After obtaining the best models from RNN and LSTM, we test the effect of dropout layers, plotting again the predictions to compare with the true labels. We also plot the comparison between RNN and LSTM models with and without dropout.

We have found these two types of models to be slower to train compared to other neural network architectures, favoring accuracy over efficiency.

Evaluation measures

In the iterative process mentioned above, we employed a comprehensive range of evaluation measures to assess the performance of the RNN and LSTM models including test loss, test MAE, validation loss, validation MAE, train loss, and train MAE. These metrics provide valuable insights into the models' performance and their ability to capture the underlying patterns in the time-series stock price data.

For each test we utilized the below hyperparameters, however on the LSTM model since we have more complex architectures (increased risk of overfitting) we used early stopping to find the optimal number of epochs:

	RNN				LSTM			
	Loss Function	Optimizer	Metrics	Epochs	Loss Function	Optimizer	Metrics	Epochs
Test 1	MSE	Adam	MAE	50	MSE	Adam	MAE	19/50
Test 2	MSE	Adam	MAE	20	MSE	Adam	MAE	20/20
Test 3	MSE	Adam	MAE	70	MSE	Adam	MAE	22/70
Test 4	MAE	Adam	MAE	50	MAE	Adam	MAE	16/50
Test 5	MAE	Adam	MAE	20	MAE	Adam	MAE	15/20
Test 6	MAE	Adam	MAE	70	MAE	Adam	MAE	22/70
Test 7	Mean_Squared_Error	Adam	MAE	50	Mean_Squared_Error	Adam	MAE	31/50
Test 8	Mean_Squared_Error	Adam	MAE	20	Mean_Squared_Error	Adam	MAE	17/20
Test 9	Mean_Squared_Error	Adam	MAE	70	Mean_Squared_Error	Adam	MAE	25/70

Table 1 – Hyperparameters used for each model.

By utilizing these standardized evaluation measures, we were able to assess the models' performance consistently and compare their results directly. This approach allowed us to make informed decisions regarding the effectiveness of each model architecture in predicting future stock market behavior.

To assess the performance of the proposed system, the test loss and test MAE are calculated for each model and approach. Lower values of test loss and test MAE indicate better performance, as they suggest that the model's predictions are closer to the true values. By comparing these metrics across different models and approaches, it is possible to identify the ones that demonstrate superior predictive capabilities and are more effective in capturing the underlying patterns in the time-series stock price data.

Based on the assumption that lower values of test loss and test MAE indicate better performance, **RNN_MODEL2** has been identified as the best performing model among the RNN models as we can check on the below table:

	model_name	test_loss	test_mae	val_loss	val_mae	train_loss	train_mae
0	RNN_MODEL1	0.0011	0.0262	0.0115	0.0792	0.0008	0.0192
0	RNN_MODEL2	0.0009	0.0226	0.0122	0.0794	0.0008	0.0190
0	RNN_MODEL3	0.0023	0.0419	0.0104	0.0810	0.0009	0.0221
0	RNN_MODEL4	0.0351	0.0351	0.0809	0.0809	0.0204	0.0204
0	RNN_MODEL5	0.0209	0.0209	0.0800	0.0800	0.0191	0.0191
0	RNN_MODEL6	0.0239	0.0239	0.0793	0.0793	0.0192	0.0192
0	RNN_MODEL7	0.4301	0.6438	0.3717	0.5875	0.1761	0.3746
0	RNN_MODEL8	0.0011	0.0252	0.0118	0.0797	0.0008	0.0195
0	RNN_MODEL9	0.0026	0.0455	0.0108	0.0831	0.0011	0.0254

Table 2 – Evaluation measures for each RNN model.

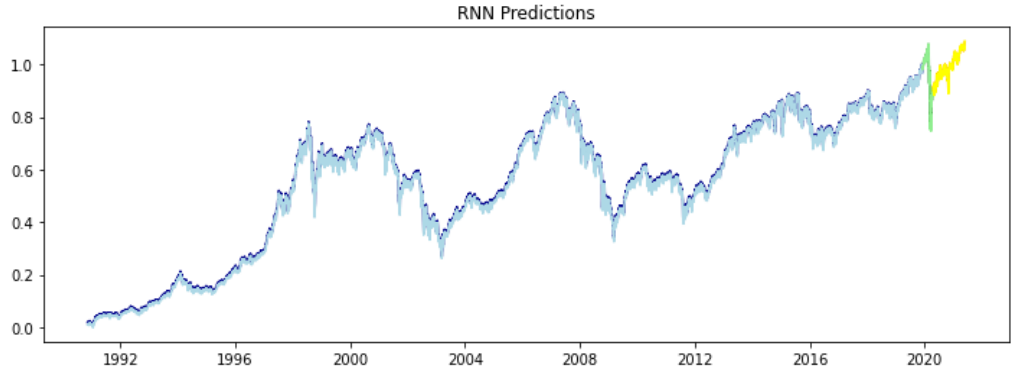

Figure 2- Best performing model for RNN (Model 2).

Regarding the LTSM models, and using the same assumption we identified **LSTM_MODEL2** as the best LSTM performing model:

	model_name	test_loss	test_mae	val_loss	val_mae	train_loss	train_mae
0	LSTM_MODEL1	0.0021	0.0394	0.0108	0.0817	0.0010	0.0239
0	LSTM_MODEL2	0.0018	0.0354	0.0110	0.0808	0.0009	0.0228
0	LSTM_MODEL3	0.0029	0.0486	0.0106	0.0835	0.0012	0.0272
0	LSTM_MODEL4	0.0235	0.0235	0.0791	0.0791	0.0203	0.0203
0	LSTM_MODEL5	0.0239	0.0239	0.0795	0.0795	0.0207	0.0207
0	LSTM_MODEL6	0.0231	0.0231	0.0790	0.0790	0.0202	0.0202
0	LSTM_MODEL7	0.0045	0.0619	0.0106	0.0862	0.0015	0.0300
0	LSTM_MODEL8	0.0026	0.0424	0.0107	0.0817	0.0011	0.0245
0	LSTM_MODEL9	0.0031	0.0499	0.0105	0.0834	0.0012	0.0276

Table 3 - Evaluation measures for each LSTM Model.

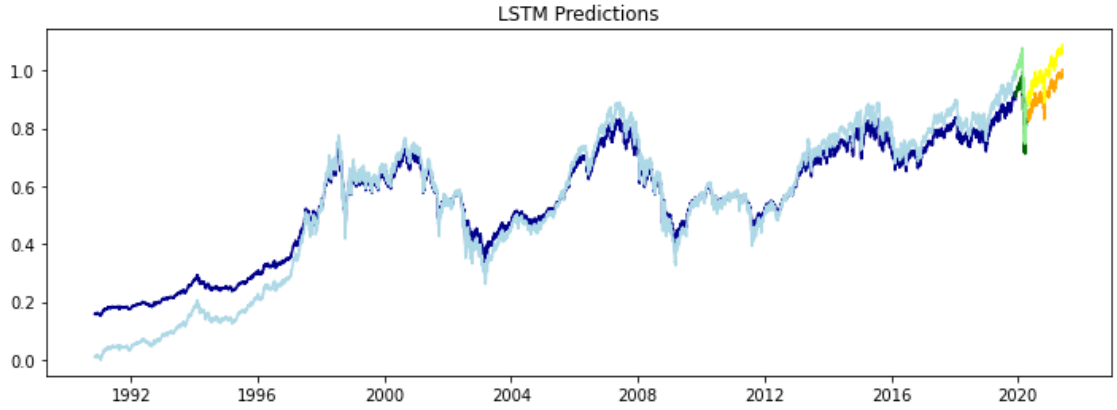

Figure 3 - Best performing model LSTM (Model 2).

These models showed superior predictive capabilities and the ability to capture underlying patterns in the time-series stock price data. The selection of RNN_MODEL8 for the RNN architecture and LSTM_MODEL8 for the LSTM architecture represents the best choices for predicting future stock market behavior based on the evaluation metrics utilized.

In summary, our approach involved analyzing the effects of different epochs on the training and validation functions, utilizing batch training for efficient processing, comparing model complexities to evaluate generalization ability, incorporating dropout mechanisms to prevent overfitting, and implementing early stopping on LSTM model to find an optimal number of epochs. These techniques collectively helped us understand and optimize the performance and properties of our system.

Error analysis & Discussion

In our quest to improve the performance and prevent overfitting in our best models, we conducted experiments using dropout mechanisms. Dropout is a regularization technique commonly used in neural networks, which randomly deactivates a fraction of neurons during training. By doing so, dropout forces the remaining neurons to learn more robust and independent features. This regularization technique helps prevent overfitting by reducing the model's reliance on specific neurons and promoting more generalized learning.

To compare the effectiveness of dropout, we evaluated our best models both without dropout and with dropout.

Best models without dropout mechanism:

	model_name	test_loss	test_mae	val_loss	val_mae	train_loss	train_mae
0	RNN_MODEL2	0.0009	0.0226	0.0122	0.0794	0.0008	0.0190
0	LSTM_MODEL2	0.0018	0.0354	0.0110	0.0808	0.0009	0.0228

Table 4 – Evaluation measures for the two best performing models without dropout.

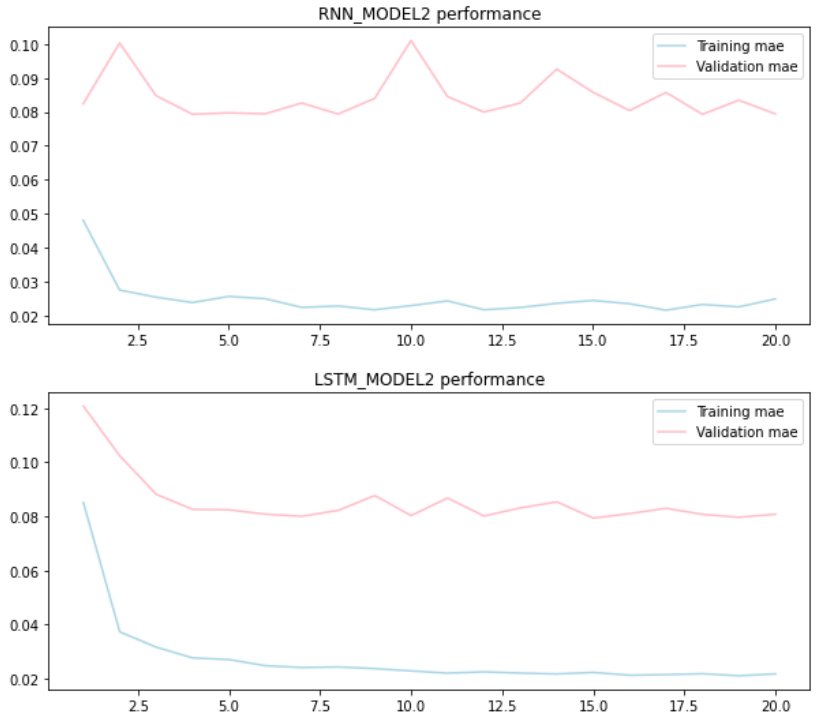

Figure 4 – Comparison of the two best models without dropout.

Best models with dropout mechanism:

	model_name	test_loss	test_mae	val_loss	val_mae	train_loss	train_mae
0	RNN_MODEL_DROPOUT	0.0011	0.0253	0.0126	0.0816	0.0009	0.0204
	model_name	**test_loss**	**test_mae**	**val_loss**	**val_mae**	**train_loss**	**train_mae**
0	LSTM_MODEL_DROPOUT	0.0007	0.0202	0.0136	0.0801	0.001	0.0227

Table 5 - Evaluation measures for the two best performing models with dropout.

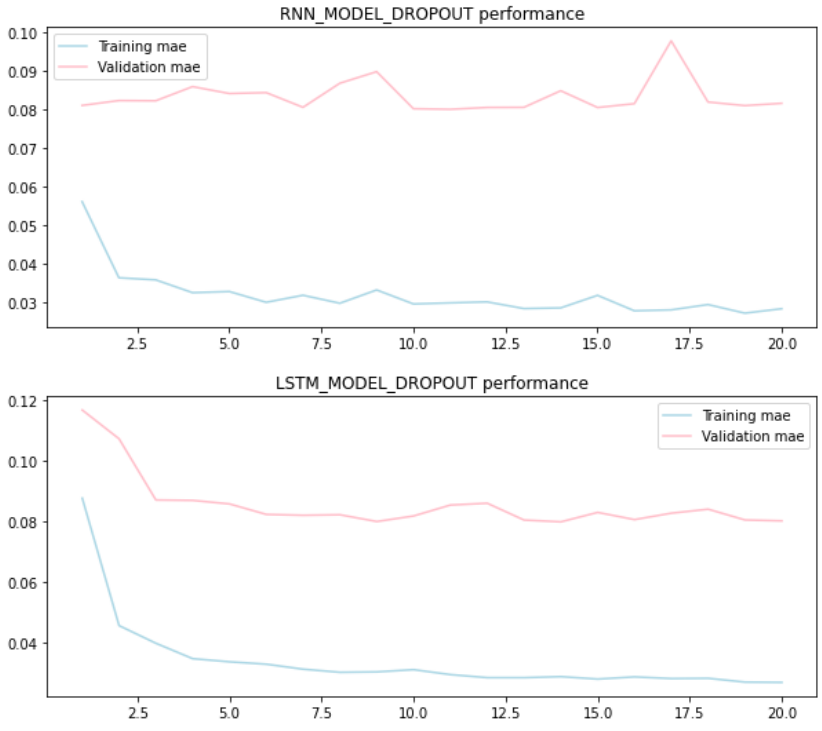

Figure 5 - Comparison of the two best models with dropout.

By incorporating the dropout mechanism into our models, we noticed a significant impact on the stability of the validation metrics compared to the models without dropout. When dropout was applied, we observed that the validation loss and validation MAE exhibited less erratic and more predictable variations during the training process. This stability in the validation metrics is a clear indication that the models with dropout were better able to generalize the underlying patterns in the dataset and effectively mitigate the risk of overfitting.

In contrast, the models without dropout showed more erratic fluctuations in the validation metrics, which suggests a higher susceptibility to overfitting. Without the regularization effect provided by dropout, these models may have been more prone to memorizing the training data, leading to poor generalization and unreliable performance on unseen data.

The stability observed in the validation metrics with dropout reflects the models' ability to capture essential patterns and features from the data while avoiding excessive reliance on specific neurons or features. This improved generalization capability resulted in more reliable and consistent performance across different evaluation metrics.

In summary, the inclusion of dropout in our models led to a more stable validation process, where the validation loss and validation MAE showed consistent trends. This stability indicates that the models with dropout effectively learned to generalize from the training data and exhibited improved performance on unseen data, providing more reliable and consistent predictions.

Conclusion

Our group focused on designing and evaluating models for time-series prediction using RNN and LSTM architectures. Through various experiments and analyses, we explored the impact of different factors on model performance, such as loss functions, optimizers, model complexities, and the inclusion of dropout mechanisms. The evaluation measures, including test loss, test MAE, validation loss, validation MAE, train loss, and train MAE, were used to assess the success of the models.

Our several tests point out the importance of selecting appropriate techniques and parameters to achieve optimal performance. We observed that the inclusion of dropout mechanisms improved generalization and prevented overfitting, as indicated by the stability and convergence of validation metrics. Also, the choice of loss functions and optimizers played a crucial role in model accuracy and generalization ability.

References

https://towardsdatascience.com/predicting-stock-prices-using-a-keras-lstm-model-4225457f0233

https://www.kaggle.com/code/ozkanozturk/stock-price-prediction-by-simple-rnn-and-lstm.

Stock Exchange Data | Kaggle

Notebooks & Slides provided during the classes.