Training Strategy As the first preprocessing step, we applied $\log(1+x)$ to the explanatory variables 'rv_lag_5', 'rv_lag_22', 'rv_plus_lag', 'medrv_lag' and the target variable 'rv_lead_1' due to their skewed distributions. This transformation makes the feature distributions closer to normal. For both explanatory and target variables, $\log(1+x)$ reduces the impact of outliers, improves gradient flow, and stabilizes training. It also aligns better with the assumptions of MSE, leading to faster convergence and more efficient model training.

We included a single interaction feature in our model, '$rv\_lag\_5 * rv\_plus\_lag$'. After testing various interaction terms, this was the only one that showed a significant impact on the MSE results. We removed the date and ticker variables from the model as explanatory variables.

Outlier capping was used to address extreme values by calculating the 99th percentile for each column based on the training set and capping any value exceeding it to the 99th percentile. This approach minimizes the impact of outliers, enhancing data quality, model performance, and fairness while maintaining dataset integrity.

We then standardized the features using the mean and standard deviation calculated soley from the training set. This stabilization ensures consistent input scales for LSTMs, preventing gradient issues, improving convergence, and enhancing the model's ability to learn patterns.

As a validation technique, we used an 80/20 split of the training data, with 80% allocated to the training set and 20% to the validation set. This split is based on date to preserve the temporal order of the data, as stock data follows a time series. By doing so, we ensure that the model is trained on historical data and validated on unseen, future data, simulating real-world forecasting while preventing data leakage.

Finally, we reversed the log transformation of the predicted and actual target values to compute the MSE in the original scale, allowing for a more meaningful comparison of different techniques.

Chosen Model For this task, we select Long Short-Term Memory (LSTM) model because of its suitability for time-series forecasting, especially with financial applications. LSTMs try to capture sequential dependencies, making them ideal for modeling the structure of stock data.

Table 3 (which are preliminary results) shows that the basic Neural Network model achieves the lowest mean squared error (MSE) compared to other models we tested, obtaining better results than the AR(1), ridge regression, and XGBoost. After improvements the final model resulted in a MSE of 15.3 for the training set, and a MSE of 12.1 for the validation set. Literature, like Shah et al., 2022, conclude that LSTMs outperform classical models like ARIMA in predicting stock prices, especially when capturing non-linear patterns and searching for long term dependencies. Similar results are reported by Fischer and Krauss, 2018, who demonstrated that LSTMs capture both temporal and non-linear dependencies in stock market data, significantly outperforming traditional models. Note that hybrid models like LSTM-CNN may result in better performances. However, we have a relatively small dataset and focus on prediction accuracy instead of intraday trends, so the LSTM structure should be sufficient. The LSTM model we used uses a single hidden layer along with one dense output layer, which optimizes efficiency and computational cost, while still being sufficient looking at the relatively small dataset. We used the ReLU activation function for the hidden layer to introduce non-linearity and to handle vanishing gradients efficiently. In the output layer a linear activation function is used to make sure there are precise numerical predictions for regression tasks.

In terms of interpretability, LSTMs are not as transparent as linear models but they do a good job in capturing temporal relationships which are present in time-series data. Simple models, like AR(1) and ridge regression, have difficulties capturing the dataset's complexity, and XGBoost required significant tuning for comparable results. Concluding that the LSTM model is the most suitable choice for this forecast assignment.

Hyperparameters The hyperparameters of the LSTM Neural Network model were optimizing utilizing the Keras Tuner's Bayesian Optimizaiton method. The hyperparameters being optimized are: the number of LSTM units, dropout rate, recurrent dropout rate, dense layer size, and learning rate. Bayesian Optimizaiton the relies on a Guassian process as its underlying probablisitc model to minimize the MSE of the validation set. The ranges for these hyperparameters, shown in Table 3, represent the search space explored during the optimization process, while Table 2 summarizes the Bayesian optimization setup. The suitiblity of the optimizer stems from its efficient tuning of multiple hyperparameters, fewer iterations compared to methods like grid search, and the fact that it does not require gradients which suits discrete parameters like dropout rate.

In regard to the hyperparameter search space, the code employs dropout rate and recurrent dropout rate as regularization techniques to prevent overfitting. It uses early stopping to reduce computation costs while running a sufficiently large number of epochs (50) to ensure convergence. A batch size of 32 is chosen to balance a smooth convergence path with the ability to escape local minima. An additional set of hyperparameters are derived from the Adam optimizer which uses adaptive learning rates to improve convergence, providing an additional hyperparameter for the learning rate of the optimizer which controls the step size weights for each iteration. Due to the sequential nature of the data and the structure of the LSTM Neural Network, Adam can aid in avoiding the vanishing gradient problem by adjusting updates based on past behavior.

Conclusion This assignment aimed to forecast stock prices, with the LSTM Neural Network ending up as the most effective model. The LSTM's good performance on time-series data has shown its capacity in capturing the temporal dependencies and nonlinear patterns in stock data. After hyperparameter optimization using Bayesian optimization, the final LSTM model achieved a MSE of 15.3 on the training set and 12.1 on the validation set, outperforming models like AR(1), ridge regression, and XGBoost. Effective preprocessing steps, like log transformations, outlier capping, and feature standardization, improved the stability and performance of the model. The LSTM model is not as interpretable as other models, but the accuracy has made it the most suitable method for forecasting the stock prices. Our results have enforced the potential of deep learning in financial forecasting.

Appendix

In this appendix, we tabulate the hyperparameters that were optimized during the tuning of the LSTM Neural Network. The ranges specified below represent the search space explored during the Bayesian optimization process. Note that Bayesian optimization does not rely on an exhaustive grid but rather adaptively selects values within these ranges based on model performance.

Table 1: Optimized Hyperparameters and Their Best Values for the LSTM Neural Network

Hyperparameter	**Search Range**	**Best Value**
Number of LSTM Units	{16, 32, 48, 64, 80, 96, 112, 128}	128
Dropout Rate	{0.1, 0.2, 0.3, 0.4, 0.5}	0.1
Recurrent Dropout Rate	{0.1, 0.2, 0.3, 0.4, 0.5}	0.5
Number of Dense Units	{16, 32, 48, 64, 80, 96, 112, 128}	128
Learning Rate (Adam Optimizer)	10^{-5} to 10^{-3} (logarithmic scale)	0.001

Table 2: Parameters used for the Bayesian optimization setup regarding search process

Parameter	**Value**
Number of Trials	100
Executions per Trial	3
Objective	Validation Mean Squared Error
Optimization Method	Bayesian Optimization

Table 3: Mean Squared Error (MSE) for each algorithm

Method	**MSE**
Ridge Regression (taking minimum λ)	57.85054
Ridge Regression (one-standard-error rule for λ)	76.11461
AR(1) Model	59.10731
XGBoosting	80.87218
Neural Network	55.38363
Neural Network (scaled data)	54.75982

References

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669.

Shah, J., Vaidya, D., & Shah, M. (2022). A comprehensive review on multiple hybrid deep learning approaches for stock prediction. *Intelligent Systems with Applications*, *16*, 200111. https://doi.org/https://doi.org/10.1016/j.iswa.2022.200111