Improving the prediction of asset returns with machine learning by using a custom loss function

Jean Dessain

IESEG, School of Management, Department of Finance, 3 rue de la Digue, 59000 Lille, France

Reacfin SA, 25 Place de l'Université , 1348 Louvain-La-Neuve, Belgique

Email: j.dessain@ieseg.fr

Abstract

The loss function in supervised deep learning is a key element for training AI algorithms. For models aiming at predicting asset returns, not all prediction errors are equal in terms of impact on the efficiency of the algorithm. Indeed, some errors result in poor investment decisions while other errors have no financial consequences. This is critical for the choice of the performance metric to assess the ability of machine learning algorithms to predict returns. This should also impact the choice for the most efficient loss function.

Mean Squared Error (MSE) is the most popular loss function for regression algorithms, but it is not the most efficient one for algorithms predicting asset returns. In this article: (a) we develop several custom loss functions considering the asymmetry in the objective of the algorithm. (b) We compare these custom loss functions to improve the algorithms predicting asset returns. (c) We identify a very efficient custom loss function that significantly improves the prediction of asset returns, and which we confirm to be robust.

Keywords: Machine learning; Deep learning; Loss function; Time series forecasting; Stock return predictability; Investment efficiency

JEL: C45, C53, G11, G17, N2.

---

1. Introduction

The finance industry has systematically looked for ways to predict future asset returns, and more generally to predict financial time series data. Regression and classification algorithms, as well as reinforcement learning, have been developed to predict the effective return of assets, either for very short periods of time or for longer horizons. But the task is undoubtedly difficult as financial markets are volatile and noisy environments, with short-term and long-term fluctuations and huge shifts in volatilities.

1.1 Situation

Numerous academic papers present algorithms aimed at improving investment strategy and optimizing the risk-adjusted returns of portfolios. The number of academic studies published on this topic has grown at an exponential rate and a comprehensive review of the literature is becoming increasingly challenging (Bustos & Pomares-Quimbaya, 2020; Huang et al., 2020; Meng & Khushi, 2019; Ozbayoglu et al., 2020; Sezer et al., 2020; Thakkar & Chaudhari, 2021), if feasible at all.

Dessain (2021) offers arguably the most comprehensive overview to date, with 190 articles reviewed over the period 2010 – June 2021, but with a narrow focus on the sole performance metrics used to compare algorithms predicting asset returns. He demonstrates that the mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE) or similar error-based performance metrics are not appropriate for assessing the results of algorithms predicting asset returns. The underlying reasoning is that error-based metrics treat all errors equally and do not differentiate an error that triggers a bad investment decision (an investment resulting in a negative return or a missed opportunity with no investment when the asset has led to a positive return) from a prediction error which does not have any adverse consequence, leading to a positive return or to a non-investment that avoided a negative return. The deductive reasoning is confirmed with an extensive empirical analysis.

The same MSE is one of the most common loss functions in regression algorithms. A loss function measures the prediction error of the model that the optimization process tries to minimize in updating the weights during the backward propagation. If MSE is inappropriate to assess the efficiency of an algorithm, our assumption is that MSE might not be the optimal loss-function for training a neural network performing a regression to predict asset returns. Indeed, as for the performance metric, all errors computed with MSE (and RMSE, MAE or MAPE) will be treated equally during the training phase, even though their consequences might be drastically different in terms of investment efficiency.

1.2 Contribution

The rest of the paper is structured as follows: In section two, we analyse the literature review in terms of custom loss functions for deep learning, with models predicting asset returns as well as models from other fields of computer science. Section three presents the framework and various custom loss functions as alternatives to the MSE. Section four presents the methodology for testing the various loss functions, and section five presents the results of the analyses. Section six concludes and draws some perspectives.

2. Common and custom loss functions in the literature

We analyse the loss functions in the literature from three sources: (i) the database of 190 articles predicting stock returns used by Dessain (2021), (ii) some articles of machine learning in finance not covered in (i) that analyse custom loss functions, and (iii) articles from other fields of AI that explicitly refer to the use of custom loss functions.

2.1 Financial articles predicting stock returns

The loss function is most of the time not considered as a relevant topic. Using Dessain's article database, we examined each article and searched for the loss function(s) applied. 39 articles[1] describe or mention the loss function(s) applied, and 151 don't address the topic. 20 out of the 39 articles perform regressions (sometimes together with classification tasks). Out of these 20 articles, 13 apply standard MSE or RMSE. 3 articles apply MAE or MAPE.

Four articles (Gu et al., 2020; Lim et al., 2019; Yun et al., 2020; Zhou et al., 2018) directly or indirectly investigate a possible customisation, but none of them introduce asymmetry in the loss function.

Gu et al. (2020) test various loss functions, starting with MSE, then using weighted MSE to underweight small stocks in favour of large stocks. They also test a Huber loss function as a mix of MSE (for the relatively small errors) and MAE (for relatively large errors). All errors are treated equally, without any consideration for the sign of the expected return versus the actual one.

Lim et al. (2019) enrich the MSE loss with a regularisation based on the Sharpe ratio. All errors are treated equally, without any consideration for the sign of the expected return versus the effective one. They obtain interesting results on 5 years out-of-sample data, as long as transactions costs are at 0, but they obtain negative Sharpe ratios for each neural network architecture (MLP, CNN or LSTM)[2] as soon as transaction costs reach or exceed 5 basis points.

Yun et al. (2020) combine the standard MSE to a directional cost in view of managing a portfolio of assets via a two-stage deep learning approach. For each asset within the portfolio, the joint cost function considers its absolute and relative performances, but it does not consider any asymmetry in the impact from the prediction error.

Zhou et al. (2018) combine a forecast error loss, MSE with a directional loss in a GAN[3] architecture that includes LSTM and CNN layers for high-frequency data. To the forecast error loss, they add a loss that tracks a directional change. The performance metrics applied by Zhou et al. (2018) are the root mean squared relative error (RMSRE) and the direction prediction accuracy (DPA), an error-based metric and an accuracy-based one, whose inefficiency have been demonstrated by Dessain. The true merit of the adjustment proposed can therefore not be confirmed.

2.2 Other financial articles

We also reviewed custom loss functions in articles dedicated to finance but not covered in 2.1. We found two papers that investigate custom loss functions.

[1] See Appendix 1. Loss functions applied by author

[2] MLP = Multi-Layer Perceptron, CNN = Convolutional Neural Network (in this case wavenet) and LSTM = Long-Short Term Memory neural network.

[3] GAN = Generative Adversarial Network, here with an LSTM generator and a CNN discriminator.

Ahmed et al. (2020) refer to the fact that algorithms and their constituents should be tailor made in respect of the field of application. They apply a custom loss function called Forex Loss Function (FLF) to predict forex movements with a LSTM model. Instead of computing the loss with the difference between the predicted exchange rate and the actual one, they adjust the MSE using the Open – High – Low – Close (OHLC) predicted prices and OHLC actual ones (inspired by the candlestick analysis). They then use MAE as performance metric to assess the higher efficiency of the custom FLF, leading to non-reliable results.

Tavakoli & Doosti (2021) propose a custom loss function that adjusts the MSE to account for (i) the expected future volatility of the asset and (ii) the expected gain defined a priori as target for the model. They compare the accuracy of the model with various standard losses (MSE, MAE, MAPE, Huber loss, Log-Cosh loss). They find a higher accuracy for the proposed loss function and assume that the superior accuracy should lead to a higher profit. In addition, the test set is three months of weekly returns or 13 data points. The approach is interesting but should be further confirmed, as 3-months test set of weekly data is obviously too short to draw a reliable conclusion, and the chosen performance metric cannot lead to any effective conclusion.

2.3 Other non-financial articles with custom loss functions

Eventually, we reviewed custom loss functions in other areas of research. From our analysis, medical and biomedical field and time series predictions are the main fields to propose several approaches. The loss function is not a popular theme.

Barton et al. (2021) use a weighted combination of global and local MSE to improve signal detection in biomedical science. Bhandari et al. (2020) combine a regression and a classification approach, with an MSE and a cross-entropy loss function to improve the detection of oral tumours.

To predict time-series, Cuturi & Blondel (2017) apply a soft dynamic time-wrapping (soft DTW) that allows to compare time series of variable size with a differentiable loss. Interestingly, Chen et al. (2021) explicitly search for a loss function tailor-made for wind speed prediction. A kernel-MSE is applied to cope with the nonlinearity of wind speed data forecasting. The kernel MSE is symmetric, positive, bounded, with a positive first order derivative and a negative second order derivative; therefore, it should be less sensitive to outliers than the standard MSE.

Eramo et al. (2021) propose an asymmetric loss function to predict traffic and cloud resource requirements for allocating cloud and bandwidth resources. The asymmetric loss function captures the dramatic increase in costs when the prediction error leads to the under-capacity of the cloud and to an excess traffic compared to the available bandwidth. The proposed model with asymmetric loss function has led to a significant cost advantage compared to standard model.

Finally, Ebert-Uphoff et al. (2021) propose a guide and a large collection of loss functions applied in environmental science, stating that “the loss functions required by environmental scientists are unlike any loss functions typically used in computer science, and the community has not yet developed comprehensive resources, such as a large collection of customized loss functions."

While none of these approaches are immediately relevant for our task of predicting returns, the process of tailoring the loss function for a specific purpose echoes our initial intuition.

3. Framework and custom loss functions

The prediction of asset return (or asset price) is driven by the asymmetry of goals. If the prediction error induces a wrong investment decision, its impact is significant and leads either to a loss-making investment or to a missed profit-making investment. If the error does not result in a wrong investment decision, its impact is negligible. Therefore, the optimization should not minimize the mean error, but the mean of the errors that cause a wrong investment decision. To achieve this objective, the investment strategy should be defined ex-ante, before the tailoring of the loss function.

3.1 Framework: data and investment strategy

To test our hypothesis of superior efficiency of a custom loss function, we define a framework. We will test various loss functions on a series of stocks whose daily returns are predicted by machine learning regression algorithms.

We use a long series of daily prices history to train our algorithm, and we will test[4] it with an out-of-sample data that will be long enough to ensure that most typical situations are properly represented: rally, crisis, recovery, periods of uncertainty with no clear direction. This is necessary to produce reliable results[5].

In both training and testing phases, the algorithm will receive a set of historical data up to the end-of-day $Close_t$ and will predict the next day return A transaction cost is charged for any purchase or sale of shares: if an open position exists and is rolled over, the transaction cost does not apply; if the position is opened or closed, the transaction cost is charged.

We test a very simple investment strategy: (i) if the predicted return of the next day is above a defined trigger, we invest or roll the open position, if any, for one day; (ii) otherwise, we take no open position or we close any previous open position.

[4] We do not use the traditional split between training set, validation set, and test set as we are just looking for algorithms to produce series of returns.

[5] This is a recurrent issue with the 190 surveyed articles in 2.1, where many papers apply a test set insufficiently long, sometimes even shorter than 2 years, thereby producing unreliable and not exploitable results.

3.2 Analyzed loss functions

To match the defined investment policy, the custom loss function should heavily penalize errors when the effective daily return y and the predicted return $\hat{y}$ are of opposite signs, and underweight loss when both actual and predicted returns are of the same sign.

We compare our analysis with the MSE loss function and design 3 main types of customisation[6]:

(i) an "adjusted loss 1" (AdjLoss1) where the squared error is multiplied by α if (y * $\hat{y}$) < 0 (when y and $\hat{y}$ are of opposite sign) and by 1 / α if (y * $\hat{y}$) > 0 (when y and $\hat{y}$ have the same sign), with α being a positive number greater than 1. (α =1 is equal to MSE). This AdjLoss1 is not fully differentiable, where either y or $\hat{y}$ is equal to 0.

(ii) an "adjusted loss 2" (AdjLoss2) that is fully differentiable. The squared error is multiplied by a sigmoid-like curve adjustment factor. This adjustment factor tends to a maximum for large negative values of (y * $\hat{y}$), is equal to the 1 for y or $\hat{y}$ equal to 0, and tends towards a minimum for large positive values of (y * $\hat{y}$).

$$AdjLoss2 \;=\; \frac{\beta * (y-\hat{y})^2}{\left(1+\beta-\frac{(\beta-0.5)}{(1+e^{(10000*y*\hat{y})})}\right)}$$ with β being a positive number greater than 1.

(iii) an "adjusted loss 3" (AdjLoss3) that, to some extent, mimics the ReLu activation function where the squared error is multiply by (1+γ) if (y * $\hat{y}$) < 0 (when y and $\hat{y}$ are of opposite sign) and multiplied by γ if (y * $\hat{y}$) > 0 (when y and $\hat{y}$ have the same sign), with γ being a positive number smaller than 1. This AdjLoss3 is not fully differentiable, when either y or $\hat{y}$ is equal to 0.

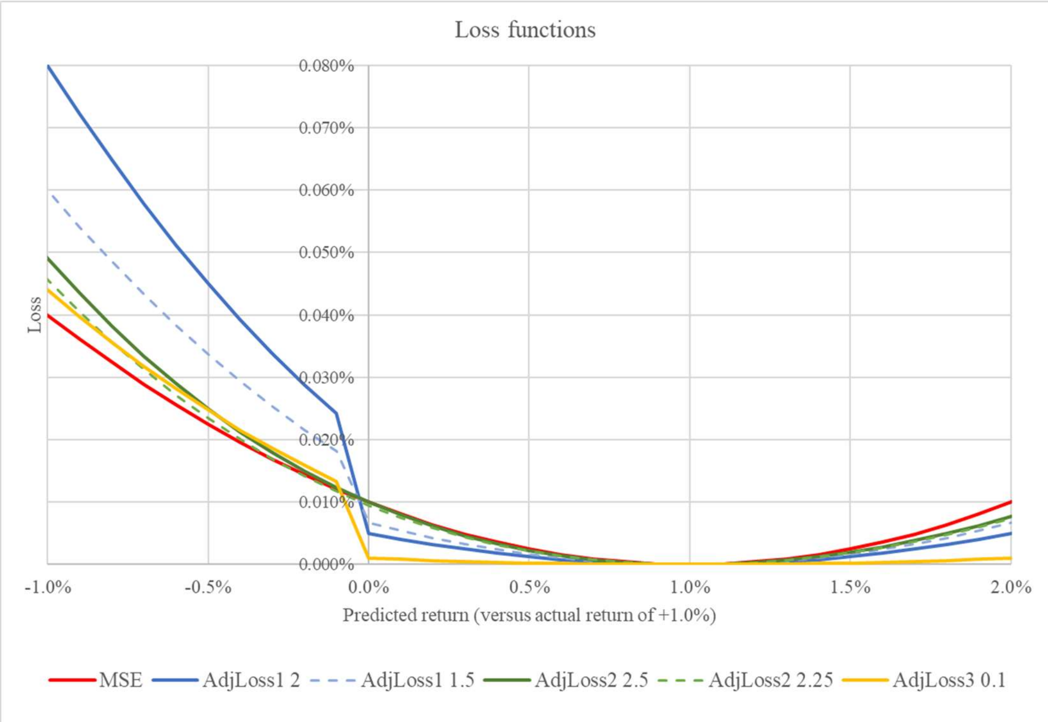
Visualisation of the loss functions

We compute the error values for 6 loss functions: MSE, AdjLoss1 with α = 2 and with α = 1.5 , AdjLoss2 with β = 2.5 and with β = 2.25 and AdjLoss3 with γ = 0.1.

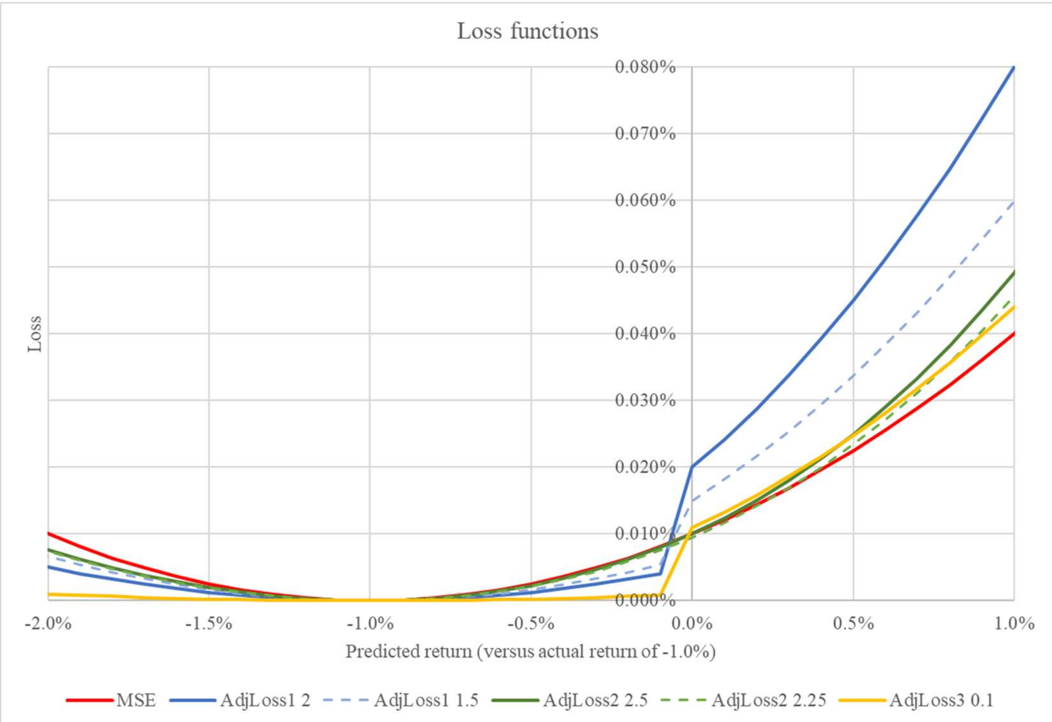
The curves of the various custom loss functions are presented in Graph 1. and Graph 2. MSE is the only symmetrical loss function. The non-complete differentiability of AdjLoss1 and AdjLoss3 is visible with Graphs 1 and 2.

Graph 1. Loss level per loss function with a true return of 1.00%

[6] The code of the custom loss functions in Python (Pytorch and TF2.0) is available on GitHub: https://github.com/JDE65/CustomLoss.



Graph 2. Loss level per loss function with a true return of -1.00%



4. Analysis

To compare the efficiency of the various loss functions, we organize our process as described hereafter, with the data management, the DL model and the performance measurement.

4.1 Data collection and data management

We collect publicly available data for large stocks traded on NYSE and Euronext from 01/01/1996 to 31/12/2020, and we reject any stock whose historical series is shorter than 15 years. We end up with a list of 105 stocks[7]: 43 traded in USD on NYSE and 62 traded in EUR on Euronext. As explained in 3.1, we organize for each stock a X matrix that is made of (i) OHLC+V daily data, (ii) a set of 14 technical indicators, and (iii) the closing prices of all the other stocks traded on the same exchange. In total, X contains 61 inputs for US stocks and 80 inputs for EU stocks. The array Y is made of the daily returns of the stock between the next opening and the opening of the day after: $y_t = \frac{\text{Open}_{t+2}}{\text{Open}_{t+1}} - 1$. The choice of considering the return from the next opening Open_{t+1} until the opening of the day after Open_{t+2} adequately integrates operational constraints such as the time required for collecting all the data and running the model, gathering all investment decisions and having controls performed before the execution of the generated buy or sell orders.

X and Y are split between a train set (X_train and y_train) and a "out-of-sample" test set (X_test and y_test. The size of the test set for each stock is 1260 trading days, or five years, between 01/01/2016 and 31/12/2020.

Graph 3. shows the cumulative result as percentage of the invested amount for buy & hold portfolios equally invested in the 43 US stocks and, respectively, for a portfolio equally invested in the 62 European stocks, between 01/01/2016 and 31/12/2020. The test set includes the various types of market sentiments we want to have to secure the reliability of our results, including a severe crisis and a rebound.

Graph 3. Cumulative results of buy & hold portfolios over 5 years

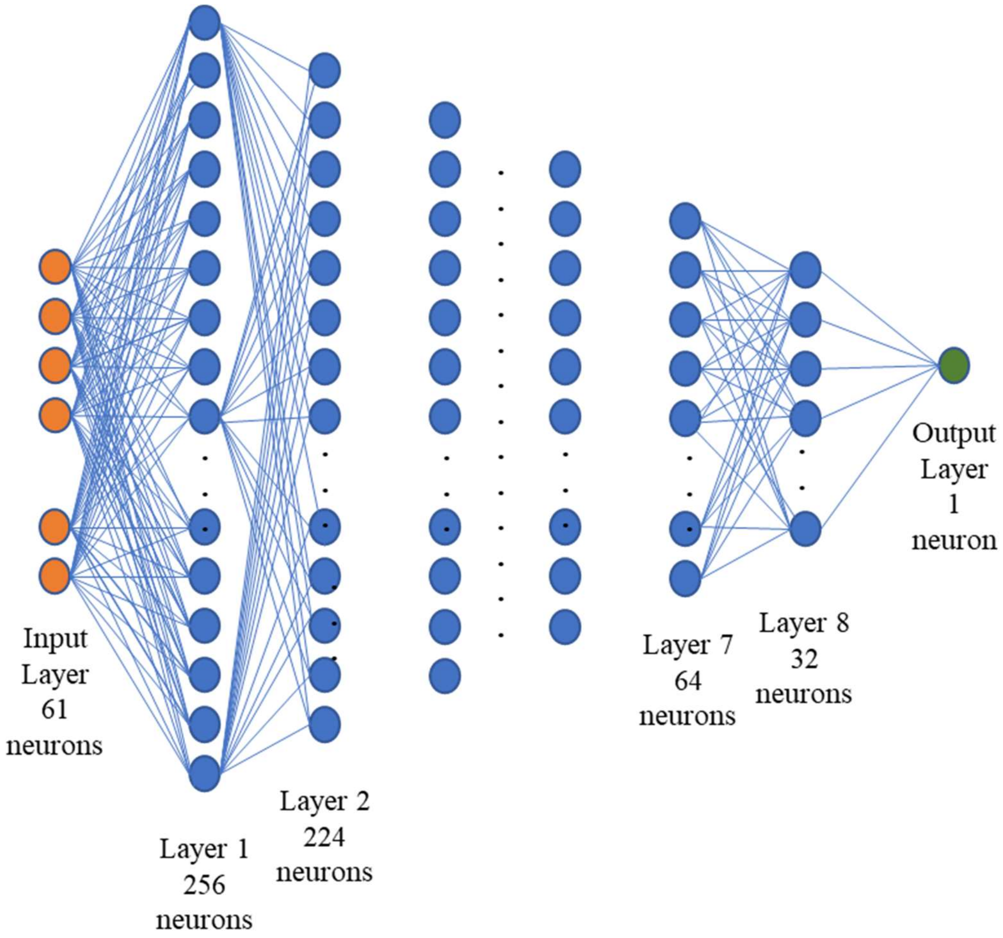
[7] See Appendix 2. List of stocks. The data set has been published on Mendeley Data doi:10.17632/nbwhzctrjp.2

Min-max normalisation is applied to X_train. The min and max values of X_train are applied to X_test for the min-max normalisation of the test set.

4.2 Model

In order to test the various loss functions, we opt for a straightforward MLP model, with all interconnected neurons, and one single neuron as output layer for the regression.

Graph 4. Simplified structure of the MLP for testing the loss functions.



The activation function is the standard Rectified Linear Unit (ReLu). With 61 inputs for US stocks and 8 hidden layers, we obtain a number of 188.893 learnable parameters. The number of parameters increases to 193.776 with the 80 inputs for EU stocks.

Table 1. Number of learnable parameters of the models

Layer	Input	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	Output	Total
Neurons	61	256	224	192	160	128	96	64	32	1	
Parameters		15677	57600	43232	30912	20640	12416	6240	2112	64	188,893
Layer	Input	*Layer 1*	*Layer 2*	*Layer 3*	*Layer 4*	*Layer 5*	*Layer 6*	*Layer 7*	*Layer 8*	Output	**Total**
Neurons	80	256	224	192	160	128	96	64	32	1	
Parameters		20560	57600	43232	30912	20640	12416	6240	2112	64	193,776

We run the model several times on several computers with various hyper-parameters in order to verify:

- The reproductibility[8] of the model: that secures exact same results when ran over the same computer with the same parameters;
- The replicability of the model: that leads to very similar results when ran over different computers and/or with marginally different hyper-parameters[9].

This way, we make sure that the results are reliable and consistent over time.

4.3 Investment strategy

To benchmark any strategy with the buy & hold strategy, a "reference number of shares" is initially determined as the number of shares that can be purchased with 100.000 USD (or 100.000 EUR for the European shares) on 01/01/2016. This reference number of shares is kept as open position over the 5 years until 31/12/2020 for the buy & hold strategy.

For each day, if the return predicted by the ML algorithm is above -0.20% (that is 2 times the transactions costs-0.20%, for out- and in- transactions compared to the buy & hold strategy), the algorithm invests in the asset: either it rolls over at no cost an existing open position or it purchases the shares with a transaction fee of 0.10% on the value of the deal. The number of shares purchased is the lower of (i) the reference number of shares and (ii) the maximum number of shares that could be purchased with the cash position.

If the predicted return is smaller than -0.20%, the algorithm either sells the entire existing open position with a transaction fee of 0.10% on the value of the sale or it maintains at no cost the open cash position. The 0.10% fee is above what is typically paid by institutional investors. It incorporates the direct costs paid to brokers and the indirect costs that might be linked to the lower liquidity at opening. Cash that is not invested in stocks is kept available without remuneration.

4.4 Performance criteria

The objective is to assess whether each custom loss function does significantly perform better than the standard MSE loss function, on average and consistently across the 105 stocks. For each of the 6 loss functions and for each of the 105 stocks[10], we compute the daily returns realized on the stock and we compare them with the returns achieved with the buy & hold strategy.

---

[8] In order to get reproducibility and replicability, we fix the seed for Python, for Numpy and for Pytorch.We also force Pytorch to work with deterministic CUDNN.

[9] We tested the model with various learning rates between 0.0001 and 0.001, with dropout values between 0.15 and 0.30, and with epochs between 100 and 300. We vary the number of hidden layers between 10 and 6.

[10] We therefore compute 630 series of 1260 daily returns generated by algorithms, and 105 daily returns for the buy & hold strategy.

We use the D ratio[11] as the main performance metric. Indeed, D ratio appears to be the best performing metric to assess machine learning algorithms predicting asset returns (Dessain, 2021). D ratio is a risk-adjusted return metric which is " $D = RtV_{algo} / RtV_{B\&H}$ ", where RtV_x is the return divided by VaR or "Return-to-VaR", respectively from the AI algorithm or from the buy & hold strategy. The D ratio indicates the extent by which the risk-adjusted return of the algorithm exceeds the risk-adjusted return of the buy & hold strategy: if the D ratio is above 1, the algorithm outperforms the buy & hold, if it is below 1 it underperforms the buy & hold. The measure of risk of the D ratio is the Value-at-Risk adjusted with the Cornish-Fisher expansion (CF-VaR)[12] in order to properly account for the effective skewness and kurtosis of the distribution of asset returns.

To complement the main metric, we also compute two sub-metrics: (i) the D-Return which measures the average return of the algorithm compared to the average return of the buy & hold strategy, and (ii) the D-VaR ratio that measures the relative risk (CF-VaR) of the returns from the algorithm compared to the riskiness of the buy & hold strategy.

To test the stability and reliability of the performance of the algorithm with each loss function, each ratio (D ratio, D-Return and D-VaR) is computed on the entire 5 years period, and for each of the two sub-periods of 2.5 years.

On top of the average D ratio, D-Return and D-VaR, we also perform a measure of the robustness of the loss function. For each loss function, we rank the 105 D ratios (respectively D-Return and D-VaR) and build a graph with the cumulative distribution of D ratios (respectively D-Return and D-VaR) for various values ranking from -5 to +5. From there, we compute the area-under-curve (AUC) for each cumulative distribution: the smallest AUC indicates the highest frequency of high D ratio values (respectively D-Return and D-VaR), and the highest AUC translates the highest number of lowest D ratios (respectively D-Return and D-VaR). This way, we can assess the robustness of the loss function. To make the comparison easier, we normalize the AUC to 1 for the buy & hold strategy. An AUC below 1 indicates that the model outperforms the buy & hold strategy; an AUC above 1 shows that the model underperforms the buy & hold strategy.

5. Detailed empirical results

The objective is to compare the efficiency of the various custom loss functions, compared to the standard MSE, and the impact of the loss functions on the performance of the algorithm. We first analyze wether the results of the algorithm with different custom loss functions are statistically different from the MSE. We then check whether the average performance of the loss functions are above the performance of the MSE loss. We eventually

[11] The code in Python for the D ratio is available on GitHub: https://github.com/JDE65/D-ratio.

[12] D ratio follows the same principle as the Sharpe ratio, but the risk measure (CF-VaR) is more robust than the standard deviation applied by Sharpe, as it does not assume a normal distribution of returns, an assumption that is most of the time not verified in practice.

verify that the superiority is consistent across most of the individual stock and we compare the distribution function of the performance metric of each loss function.

5.1. Dissimilarity of the D ratio from custom functions

To test the statistical difference between the series of D ratio obtained for each loss function, we use a Mann-Whitney U test (MWU test) which does not require any assumption about the implicit distribution function of the series. MWU test is applied for each series against the D ratios obtained with the MSE loss function.

We obtain p-values disclosed in Table 2. that are below a significance level of 0.01 for all loss functions, except for AdjLoss2 with β equal to 2.25, that is significantly different from MSE at a confidence level of 0.075.

Table 2. p-values of the Mann-Whitney U test

Loss function		**p value of MWU test**
AdjLoss1	2	8.54E-06
AdjLoss1	1.5	2.01E-06
AdjLoss2	2.5	4.19E-07
AdjLoss2	2.25	0.065475
AdjLoss3	0.1	1.90E-05

We conclude that the D ratios obtained with the various custom loss functions are statistically different from the D ratios obtained when the algorithm runs with the MSE loss function.

5.2. Average D ratio

The best algorithm will have the highest and most stable D ratio, that is the highest expected return adjusted for risk relative to the risk-adjusted return of the buy & hold strategy.

As shown in Table 3., with an average D ratio of 0.217, the MSE loss function achieves the lowest average D ratio of all tested loss functions. Every custom loss function achieves better results than MSE. Not only is it true for the entire period, it is also true with most custom loss functions for each sub-period. This confirms our intuition that asymmetric loss functions overperform the MSE.

The best loss function is the fully differentiable AdjLoss2 with β equal to 2.5 that achieves an average D ratio of 1.585 over the 105 stocks. With 1.081 for the first sub-period and 1.819 for the second sub-period, the AdjLoss2 with β of 2.5 overperforms the MSE for each sub-period. The D ratio above 1 also indicates an average risk-adjusted performance superior to the buy & hold strategy.

AdjLoss2 with β of 2.5 also reaches a D-Return of 1.322 over 5 years, significantly above the -0.024 of the MSE loss. The custom loss function significantly over-performs MSE in terms of expected return.

With D-VaR above 1 across the table, all loss functions achieve a risk reduction compared to the buy & hold strategy. The most efficient loss function for the risk reduction is AdjLoss2 with β of 2.25.

Table 3. Average D ratio, D-Return and D-VaR per loss function

Loss function	Param.	D ratio	*D ratio 1*	*D ratio 2*	D-Return	*D-return 1*	*D-return 2*	D-VaR	*D-VaR 1*	*D-VaR 2*
Buy & Hold		**1.000**	*1.000*	*1.000*	**1.000**	*1.000*	*1.000*	**1.000**	*1.000*	*1.000*
MSE		**0.217**	*-1.244*	*1.092*	**-0.024**	*-1.364*	*0.452*	**1.349**	*1.109*	*1.496*
AdjLoss1	2	**1.212**	*0.684*	*1.365*	**1.113**	*0.650*	*1.215*	**1.059**	*1.026*	*1.074*
AdjLoss1	1.5	**1.496**	*1.032*	*1.841*	**1.175**	*0.797*	*1.288*	**1.175**	*1.081*	*1.230*
AdjLoss2	**2.5**	**1.585**	*1.081*	*1.819*	**1.322**	*0.874*	*1.339*	**1.153**	*1.054*	*1.209*
AdjLoss2	2.25	**0.502**	*-1.357*	*0.358*	**0.203**	*-1.071*	*0.192*	**1.379**	*1.288*	*1.419*
AdjLoss3	0.1	**1.079**	*1.134*	*1.048*	**1.051**	*1.109*	*1.019*	**1.020**	*1.019*	*1.022*

5.3. Distribution of the D ratios and Area Under Curve (AUC)

The custom loss functions achieve an average performance significantly above the MSE loss. To reinforce these positive results, we test whether the average result is also consistent for most individual assets. Plotting the cumulative distribution of D ratios (Graph. 5) is a way to assess the consistency of the results across the 105 stocks. The graph demonstrates visually that MSE is the worst loss function with more D ratio below 1 and even below 0.

This plot of the cumulative distribution can be translated into a metric of Area Under Curve, presented in Table 4. The AUC of D ratio can also be computed for D-return and D-VaR.

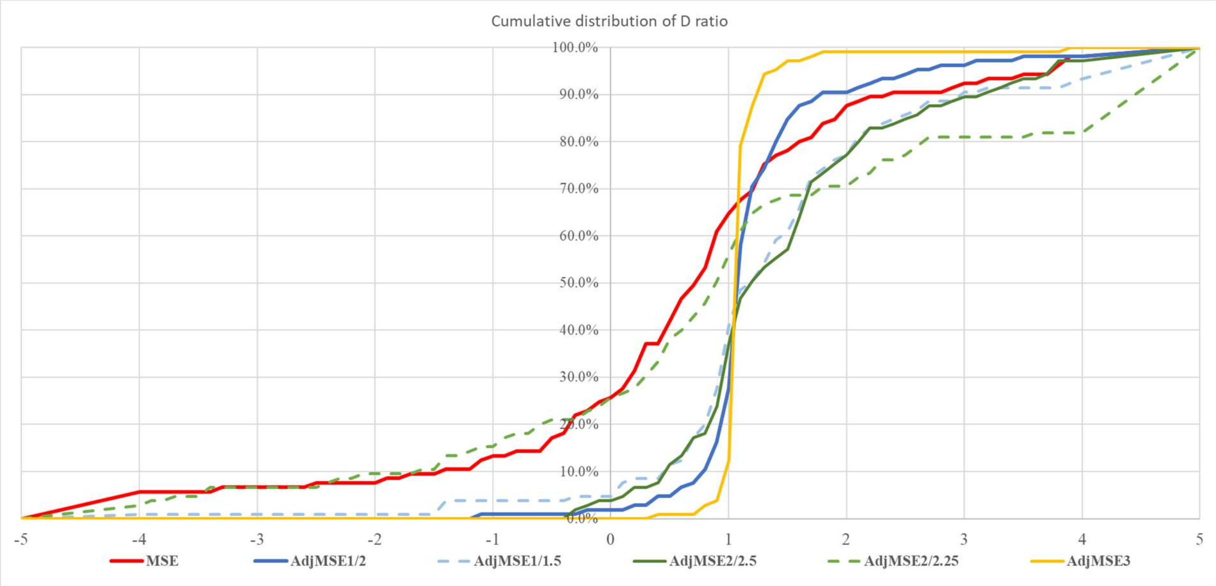
Table 4. Area Under Curve of D ratio per loss function

Loss function	Param.	D ratio	*D ratio 1*	*D ratio 2*	D-Return	*D-return 1*	*D-return 2*	D-VaR	*D-VaR 1*	*D-VaR 2*
Buy & Hold		1.000	*1.000*	*1.000*	1.000	*1.000*	*1.000*	1.000	*1.000*	*1.000*
MSE		1.123	*1.207*	*1.010*	1.181	*1.224*	*1.094*	0.913	*0.973*	*0.877*
AdjLoss1	2	0.947	*1.005*	*0.923*	0.969	*1.013*	*0.949*	0.983	*0.988*	*0.978*
AdjLoss1	1.5	0.906	*1.024*	*0.882*	0.962	*1.043*	*0.946*	0.956	*0.980*	*0.944*
AdjLoss2	**2.5**	**0.881**	*1.006*	*0.869*	**0.938**	*1.017*	*0.932*	0.961	*0.986*	*0.948*
AdjLoss2	2.25	1.024	*1.084*	*1.002*	1.088	*1.022*	*0.807*	**0.807**	*0.831*	*0.789*
AdjLoss3	0.1	0.975	*0.962*	*0.982*	0.981	*0.966*	*0.990*	0.987	*0.987*	*0.987*

The same custom loss function AdjLoss2 with β of 2.5 provides the lowest AUC for the D ratio, demonstrating that the custom loss function is also more effective than MSE for most of the individual stocks.

The AUC of D-Return of AdjLoss2 with β of 2.5 confirms that the custom loss function is more effective than MSE to produce effective returns for most stocks. As AUC is below 1, the loss function also overperforms the buy & hold strategy.

Graph 5. Cumulative distribution of D ratio per loss function



6. Conclusion and perspectives

In this final section, we conclude on the results from our analysis. We then draw some perspective for future research.

6.1. Conclusion

MSE is the most common loss function applied in ML and DL. While it is easy to apply, MSE delivers sub-optimal results once compared with asymmetric custom loss functions for algorithms predicting asset returns, as the consequence of the error in prediction does not deliver symmetrical consequences in terms of return.

Customization of the loss function to render the asymmetric consequence of the error in prediction is an easy way to significantly improve the results of simple algorithms aiming at predicting results. Not only do most custom loss functions perform much better than algorithms with MSE, but they also manage to achieve better results than a buy & hold strategy, a result that MSE never achieved with our MLP algorithm. Depending on the risk aversion of the investor and on the benchmark strategy for reference, various customizations can be contemplated. Fully differentiable loss function AdjLoss2 appears to be among the best performers, not only in terms of risk-adjusted return metric (D ratio) but also in terms of equilibrium between the effective return (D-return) and the effective risk-reduction (D-VaR). Eventually, this loss function provides a relative stability of the performance over time, when measured over several sub-periods.

6.2. Perspectives.

Loss function is a key component of any ML algorithm and a key input for computing the gradient descent. We demonstrate the advantage of tailoring the loss function with a simple deep learning model. Three possible next steps could be implemented that would generalize our results: (i) testing the algorithm with other types of assets (bonds, ETFs, commodities, crypto-currencies, …), (ii) testing the superiority of custom loss functions with more complex algorithms (LSTM, CNN and ResNet) for performing the same task of predicting asset returns. Eventually, (iii) generalizing the principle of custom loss function could be efficiently applied, mutatis mutanda, to some Reinforcement Learning (RL) algorithms (like an on-policy actor-critic PPO model).

Declaration of Competing Interest

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Warning

This research is for scientific purpose only and is not intended to support any model of investment or trading.

References

Abe, M., & Nakayama, H. (2018). Deep learning for forecasting stock returns in the cross-section. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10937 LNAI*, 273–284. https://doi.org/10.1007/978-3-319-93034-3_22

Abroyan, N. (2017). Neural Networks for Financial Market Risk Classification. *Frontiers in Signal Processing*, *1*(2), 62–66. https://doi.org/10.22606/fsp.2017.12002

Ahmed, S., Hassan, S.-U., Aljohani, N. R., & Nawaz, R. (2020). FLF-LSTM: A novel prediction system using Forex Loss Function. *Applied Soft Computing Journal*, *97*, 106780. https://doi.org/10.1016/j.asoc.2020.106780

Barton, S., Alakkari, S., O'Dwyer, K., Ward, T., & Hennelly, B. (2021). Convolution Network with Custom Loss Function for the Denoising of Low SNR Raman Spectra. *Sensors 2021, Vol. 21, Page 4623*, *21*(14), 4623. https://doi.org/10.3390/S21144623

Bhandari, B., Alsadoon, A., Prasad, P. W. C., Abdullah, S., & Haddad, S. (2020). Deep learning neural network for texture feature extraction in oral cancer: enhanced loss function. *Multimedia Tools and Applications 2020 79:37*, *79*(37), 27867–27890. https://doi.org/10.1007/S11042-020-09384-6

Börjesson, L., & Singull, M. (2020). Forecasting financial time series through causal and dilated convolutional neural networks. *Entropy*, *22*(10), 1–20. https://doi.org/10.3390/e22101094

Borovkova, S., & Tsiamas, I. (2019). An ensemble of LSTM neural networks for high-frequency stock market classification. *Journal of Forecasting*, *38*(6), 600–619. https://doi.org/10.1002/for.2585

Bustos, O., & Pomares-Quimbaya, A. (2020). Stock market movement forecast: A Systematic review. *Expert Systems with Applications*, *156*(October). https://doi.org/10.1016/j.eswa.2020.113464

Chen, J., Wu, W., & Tindall, M. L. (2016). Hedge Fund Return Prediction and Fund Selection: A Machine-Learning Approach. *Financial Industry Studies Department, Dallas Fed*, *November*. https://www.dallasfed.org/banking/fis/~/media/documents/banking/occasional/1604.pdf

Chen, L., Pelger, M., & Zhu, J. (2020). Deep Learning in Asset Pricing. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3350138

Chen, X., Yu, R., Ullah, S., Wu, D., Li, Z., Li, Q., Qi, H., Liu, J., Liu, M., & Zhang, Y. (2021). A novel loss function of deep learning in wind speed forecasting. *Energy*. https://doi.org/10.1016/j.energy.2021.121808

Cuturi, M., & Blondel, M. (2017). *Soft-DTW: a Differentiable Loss Function for Time-Series* (pp. 894–903). PMLR. https://proceedings.mlr.press/v70/cuturi17a.html

Dessain, J. (2021). Machine learning models predicting returns: why most popular performance metrics are misleading and proposal for an efficient metric. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3927058

Ding, X., Zhang, Y., Liu, T., Duan, J. (2015). Deep Learning for Event-Driven Stock Prediction Xiao. *Journal of Scientific and Industrial Research*, 2327–2333. https://www.aaai.org/ocs/index.php/IJCAI/IJCAI15/paper/view/11031/10986

Dingli, A., & Fournier, K. S. (2017). Financial time series forecasting - a deep learning approach. *International Journal of Machine Learning and Computing*, *7*(5), 118–122. https://doi.org/10.18178/ijmlc.2017.7.5.632

Ebert-Uphoff, I., Lagerquist, R., Hilburn, K., Lee, Y., Haynes, K., Stock, J. C., Kumler, C., & Stewart, J. Q. (2021). CIRA Guide to custom loss functions for neural networks in environmental sciences - Version 1. *ArXiv 2106.09757*. https://www.cawcr.gov.au/projects/verification/.

Elliot, A., & Hsu, C. H. (2017). Time Series Prediction : Predicting Stock Price. *ArXiv 1710.05751*, *2*. http://arxiv.org/abs/1710.05751

Eramo, V., Giacinto Lavacca, F., Catena, T., Jaime, P., & Salazar, P. (2021). *Application of a Long Short Term Memory neural predictor with asymmetric loss function for the resource allocation in NFV network architectures*. https://doi.org/10.1016/j.comnet.2021.108104

Feng, G., He, J., & Polson, N. G. (2018). Deep learning for predicting asset returns. *ArXiv 1804.09314*, 1–23.

Feng, G., Polson, N. G., & Xu, J. (2018). Deep Learning in Characteristics-Sorted Factor

Models. *ArXiv 1805.01104*, 1–41. http://arxiv.org/abs/1805.01104

Fischer, T., & Krauss, C. (2018). Deep learning with LSTM networks for Financial Market Predictions. *European Journal of Operational Research*, *270*(2), 1–34. https://www.econstor.eu/bitstream/10419/157808/1/886576210.pdf

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *Review of Financial Studies*, *33*(5), 2223–2273. https://doi.org/10.1093/rfs/hhaa009

Hansun, S., & Young, J. C. (2021). Predicting LQ45 financial sector indices using RNN-LSTM. *Journal of Big Data*, *8*(1). https://doi.org/10.1186/s40537-021-00495-x

Hao, Y., & Gao, Q. (2020). Predicting the trend of stock market index using the hybrid neural network based on multiple time scale feature learning. *Applied Sciences (Switzerland)*, *10*(11). https://doi.org/10.3390/app10113961

Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2018). Stock price prediction using support vector regression on daily and up to the minute prices. *Journal of Finance and Data Science*, *4*(3), 183–201. https://doi.org/10.1016/j.jfds.2018.04.003

Huang, J., Chai, J., & Cho, S. (2020). Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China*, *14*(1). https://doi.org/10.1186/s11782-020-00082-6

Kao, L. J., Chiu, C. C., Lu, C. J., & Yang, J. L. (2013). Integration of nonlinear independent component analysis and support vector regression for stock price forecasting. *Neurocomputing*, *99*, 534–542. https://doi.org/10.1016/j.neucom.2012.06.037

Lim, B., Zohren, S., & Roberts, S. (2019). Enhancing Time-Series Momentum Strategies Using Deep Neural Networks. *The Journal of Financial Data Science*, *1*(4), 19–38. https://doi.org/10.3905/jfds.2019.1.015

Liu, H. (2018). Leveraging Financial News for Stock Trend Prediction with Attention-Based Recurrent Neural Network. *ArXiv1811.06173*. https://arxiv.org/abs/1811.06173v1

Long, W., Lu, Z., & Cui, L. (2019). Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems*, *164*, 163–173. https://doi.org/10.1016/j.knosys.2018.10.034

Ma, Y., Han, R., & Wang, W. (2021). Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications*, *165*(September 2020). https://doi.org/10.1016/j.eswa.2020.113973

Mehtab, S., & Sen, J. (2020). A Time Series Analysis-Based Stock Price Prediction Using Machine Learning and Deep Learning Models. *ArXiv 2004.11697*, 1–46. https://doi.org/10.13140/RG.2.2.14022.22085/2

Meng, T. L., & Khushi, M. (2019). Reinforcement learning in financial markets. *Data*, *4*(3), 1–17. https://doi.org/10.3390/data4030110

Nikou, M., Mansourfar, G., & Bagherzadeh, J. (2019). Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithms. *Intelligent Systems in Accounting, Finance and Management*, *26*(4), 164–174. https://doi.org/10.1002/isaf.1459

Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications : A survey. *ArXiv*.

Parida, A. K., Bisoi, R., & Dash, P. K. (2016). Chebyshev polynomial functions based locally recurrent neuro-fuzzy information system for prediction of financial and energy market data. *The Journal of Finance and Data Science*, *2*(3), 202–223. https://doi.org/10.1016/J.JFDS.2016.10.001

Persio, L. Di, & Honchar, O. (2017). Recurrent neural networks approach to the financial forecast of Google assets. *International Journal of Mathematics and Computers in Simulation*, *11*, 7–13.

Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., & Cottrell, G. W. (2017). A dual-stage attention-based recurrent neural network for time series prediction. *IJCAI International Joint Conference on Artificial Intelligence*, *0*, 2627–2633. https://doi.org/10.24963/ijcai.2017/366

Rout, A. K., Dash, P. K., Dash, R., & Bisoi, R. (2017). Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach. *Journal of King Saud University - Computer and Information Sciences*, *29*(4), 536–552. https://doi.org/10.1016/J.JKSUCI.2015.06.002

Roy Choudhury, A., Abrishami, S., Turek, M., & Kumar, P. (2020). Enhancing profit from stock transactions using neural networks. *AI Communications*, *33*(2), 75–92. https://doi.org/10.3233/AIC-200629

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing Journal*, *90*(May). https://doi.org/10.1016/j.asoc.2020.106181

Shen, G., Tan, Q., Zhang, H., Zeng, P., & Xu, J. (2018). Deep Learning with Gated Recurrent Unit Networks for Financial Sequence Predictions. *Procedia Computer Science*, *131*, 895–903. https://doi.org/10.1016/J.PROCS.2018.04.298

Sim, H. S., Kim, H. I., & Ahn, J. J. (2019). Is Deep Learning for Image Recognition Applicable to Stock Market Prediction? *Complexity*, *2019*. https://doi.org/10.1155/2019/4324878

Song, D., Busogi, M., Chung Baek, A. M., & Kim, N. (2020). Forecasting stock market index based on pattern-driven long short-term memory. *Economic Computation and Economic Cybernetics Studies and Research*, *54*(3), 25–41. https://doi.org/10.24818/18423264/54.3.20.02

Tavakoli, M., & Doosti, H. (2021). Forecasting the Tehran Stock market by Machine Learning Methods using a New Loss Function. *Advances in Mathematical Finance and Applications*, *6*(2), 194–205. https://doi.org/10.22034/AMFA.2020.1896273.1399

Thakkar, A., & Chaudhari, K. (2020). CREST: Cross-Reference to Exchange-based Stock Trend Prediction using Long Short-Term Memory. *Procedia Computer Science*, *167*, 616–625. https://doi.org/10.1016/J.PROCS.2020.03.328

Thakkar, A., & Chaudhari, K. (2021). A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions. *Expert Systems with Applications*, *177*(March), 114800. https://doi.org/10.1016/j.eswa.2021.114800

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2020). Using Deep Learning for price prediction by exploiting stationary limit order book features. *Applied Soft Computing Journal*, *93*(October). https://doi.org/10.1016/j.asoc.2020.106401

Wen, M., Li, P., Zhang, L., & Chen, Y. (2019). Stock market trend prediction using high-order information of time series. *IEEE Access*, *7*, 28299–28308. https://doi.org/10.1109/ACCESS.2019.2901842

Wen, Q., Yang, Z., Song, Y., & Jia, P. (2010). Automatic stock decision support system based on box theory and SVM algorithm. *Expert Systems with Applications*, *37*(2), 1015–1022. https://doi.org/10.1016/j.eswa.2009.05.093

Yan, X., Weihan, W., & Chang, M. (2021). Research on financial assets transaction prediction model based on LSTM neural network. *Neural Computing and Applications*, *33*(1), 257–270. https://doi.org/10.1007/s00521-020-04992-7

Yun, H., Lee, M., Kang, Y. S., & Seok, J. (2020). Portfolio management via two-stage deep learning with a joint cost. *Expert Systems with Applications*, *143*, 113041. https://doi.org/10.1016/j.eswa.2019.113041

Zhou, X., Pan, Z., Hu, G., Tang, S., & Zhao, C. (2018). Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets. *Mathematical Problems in Engineering*, *2018*. https://doi.org/10.1155/2018/4907423

Appendices

Appendix 1. Loss functions applied by author

Author	**Loss functions**
Abe & Nakayama (2018)	Cross-entropy (classification)
Abroyan (2017)	Binary cross-entropy (classification)
Börjesson & Singull (2020)	MAPE (regression)
Borovkova & Tsiamas (2019)	Cross-entropy (classification)
Chen et al. (2016)	MSE (regression)
Chen et al. (2020)	Weighted mean pricing error (GAN)
Ding et al. (2015)	Margin loss (classification)
Dingli & Fournier (2017)	Margin loss (classification)
Elliot & Hsu (2017)	Unspecified
Feng, He, et al. (2018)	MSE (regression)
Feng, Polson, et al. (2018)	Economic-driven Fama-French
Fischer & Krauss (2018)	Cross-entropy (classification)
Gu et al. (2020)	MSE, MAE and Huber[13] (regression)
Hansun & Young (2021)	MSE (regression)
Hao & Gao (2020)	Binary cross-entropy (classification)
Henrique et al. (2018)	Vapnik (SVM)
Kao et al. (2013)	Cite Vapnik (SVM)
Lim et al. (2019)	MSE + sharpe (regression)
Liu (2018)	Cross-entropy (classification)
Long et al. (2019)	Cross-entropy (classification)
Ma et al. (2021)	MAE (regression)
Mehtab & Sen (2020)	MAE (regression)
Nikou et al. (2019)	MSE (regression), cross-entropy, Hinge (classification)
Parida et al. (2016)	MSE (regression)
Persio & Honchar (2017)	Log proba (classification)
Qin et al. (2017)	RMSE (regression)
Rout et al. (2017)	MSE (regression)
Roy Choudhury et al. (2020)	MSE (regression)

[13] Huber loss function as a mix of MSE and MAE

Shen et al. (2018)	Cross-entropy versus margin-based loss for SVM (classification)
Sim et al. (2019)	MSE (regression)
Song et al. (2020)	MSE (regression)
Thakkar & Chaudhari (2020)	MSE (regression)
Tsantekidis et al. (2020)	Categorical cross-entropy (classification)
Wen et al. (2019)	MSE (regression)
Wen et al. (2010)	Huber (for SVM)
Yan et al. (2021)	Categorical cross-entropy (classification)
Yun et al. (2020)	Joint cost function
Zhou et al. (2018)	Combined MSE + directional loss (regression)

Appendix 2. List of stocks

#	RIC	Asset	#	RIC	Asset
1	AAPL	Apple Inc	44	AGN.AS	Aegon NV
2	AIG	AIG	45	AKZA.AS	Akzo Nobel NV
3	AMGN	Amgen	46	ASML.AS	ASML Holding NV
4	AXP	American Express	47	DSM.AS	Koninklijke DSM NV
5	BA	Boeing	48	HEIO.AS	Heineken NV
6	BAC	Bank of America	49	INGA.AS	ING Groep NV
7	C	Citigroup	50	KPN.AS	Koninklijke KPN NV
8	CAT	Caterpillar	51	MT.AS	ArcelorMittal SA
9	CRM	Salesforces.com	52	PHIA.AS	Koninklijke Philips NV
10	CSCO	Cisco Systems	53	RAND.AS	Randstad NV
11	CVX	Chevron	54	RDSa.AS	Royal Dutch Shell PLC
12	DD	Dupont de Nemours	55	UNA.AS	Umicore SA
13	DIS	Walt Disney	56	URW.AS	Unibail-Rodamco-Westfield
14	FL	Foot Locker	57	WKL.AS	Wolters Kluwer NV
15	GE	GE	58	ABI.BR	Anheuser Busch Inbev NV
16	GS	Goldman Sachs	59	ACKB.BR	Ackermans & Van Haaren NV
17	GT	Goodyear Tire	60	AGS.BR	Ageas SA
18	HD	Home Depot	61	BAR.BR	Barco NV
19	HON	Honeywell International	62	COFB.BR	Cofinimmo SA
20	HPQ	Hewlett Packard	63	COLR.BR	Colruyt NV
21	IBM	IBM	64	GBLB.BR	Groep Brussel Lambert NV
22	INTC	Intel Corp	65	KBC.BR	KBC Groep NV
23	IP	International Paper	66	PROX.BR	Proximus NV
24	JNJ	Johnson & Johnson	67	SOF.BR	Sofina SA
25	JPM	JP Morgan	68	SOLB.BR	Solvay SA
26	KO	Coca Cola	69	TNET.BR	Telenet Group Holding NV
27	MCD	McDonald's	70	UCB.BR	Ucb SA
28	MDLZ	Mondelez Intl	71	UMI.BR	Umicore SA
29	MMM	3M	72	WDP.BR	Warehouses de Pauw
30	MO	Altria Group	73	AC.PA	Accor SA
31	MRK	Merck & Co	74	ACA.PA	Credit Agricole SA
32	MSFT	Microsoft	75	AI.PA	L'Air Liquide SA
33	NKE	Nike	76	AIR.PA	Airbus SE
34	PFE	Pfizer	77	ATO.PA	Atos SE
35	PG	Procter & Gamble	78	BN.PA	Danone SA
36	RTX	Raytheon Tech	79	BNP.PA	BNP Paribas SA
37	T	AT&T	80	CA.PA	Carrefour SA
38	TRV	Travelers Companies	81	CAP.PA	Capgemini SE
39	UNH	UnitedHealth Group	82	DG.PA	Vinci SA
40	VZ	Verizon Communications	83	DSY.PA	Dassault Systemes SE
41	WBA	Walgreens Boots Alliance	84	EN.PA	Bouygues SA
42	WMT	Walmart	85	ENGI.PA	Engie SA
43	XOM	Exxon Mobil	86	FP.PA	Total SA

#	RIC	Asset
87	HO.PA	Thales SA
88	KER.PA	Kering SA
89	MC.PA	LVMH
90	ML.PA	Michelin SA
91	OR.PA	L'Oreal SA
92	ORA.PA	Orange SA
93	PUB.PA	Publicis Groupe SA
94	RI.PA	Pernod Ricard SA
95	RMS.PA	Hermes International SCA
96	RNO.PA	Renault SA
97	SAF.PA	Safran SA
98	SAN.PA	Sanofi SA
99	SGO.PA	Compagnie de Saint Gobain SA
100	STM.PA	STMicroelectronics NV
101	SU.PA	Schneider Electric SE
102	TEP.PA	Teleperformance SE
103	UG.PA	Peugeot SA
104	VIE.PA	Veolia Environnement SA
105	VIV.PA	Vivendi SA