Advances in Artificial Intelligence and Machine Learning; Research 3 (4) 1640-1653 Received 2-10-2023; Accepted 14-11-2023; Published 21-11-2023

Improving the Prediction of Asset Returns With Machine Learning by Using a Custom Loss Function

Jean Dessain j.dessain@ieseg.fr
Department of Finance
IESEG School of Management
3 rue de la Digue
59000 Lille, France

Corresponding Author: Jean Dessain

Abstract

Not all errors from models predicting asset returns are equal in terms of impact on the efficiency of the algorithm: a small error could trigger poor investment decisions while a significant error has no financial consequences. This economic asymmetry, critical for assessing the performance of algorithms, can usefully be replicated within the machine learning algorithms itself through the loss function to improve its prediction capability. . In this article: (a) we analyze symmetric and asymmetric loss functions for deep learning algorithms. We develop custom loss functions that mimic the asymmetry in economic consequences of prediction errors. (b) We compare the efficiency of these custom loss functions with MSE and the linear-exponential loss "LinEx". (c) We present an efficient custom loss function that significantly improves the prediction of asset returns with improved risk-return metrics (like Sharpe ratio twice better), and which we confirm to be robust.

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

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 non-stationary timeseries that suffer from short- and long-term fluctuations and huge shifts in volatilities. Therefore, it regularly happens that ML models suffer from instability and non-replicability.

Citation: Jean Dessain. Improving the Prediction of Asset Returns With Machine Learning by Using a Custom Loss Function. Advances in Artificial Intelligence and Machine Learning. 2023;3(4):93.

1.1 Situation

Deep learning algorithms have become common algorithms tested for predicting asset returns. Deep learning algorithms rely on a loss function that measures the distance between the actual return and the predicted return. This loss is used during the backward propagation for training the algorithm. The magnitude of the loss is not a good indicator of the economic efficiency of the algorithm: a minor prediction error can lead to an investment that will cause a negative result or a missed opportunity. A significant error can lead to a profitable investment or to a non-investment that avoids a loss. Let us consider a strategy of investing if the predicted return is positive and to have no open position if the predicted return is negative. If the algorithm predicts a positive return of 0.25% and that the actual return is 3.0%, the loss is 2.75% but the prediction triggers a gain. If the actual return is -1.0%, the prediction loss is smaller with 1.25%, but the prediction triggers a negative result. This simple example can be further extended. With this simple proposed investment strategy, the asymmetry of the result is linked to the sign of the actual return versus the predicted one. If they have the same sign, the algorithm triggers the right investment decision. If they are of opposite signs, the algorithm will cause a bad investment with a loss or a missed opportunity. We want to test if this asymmetry around the signs of the actual and predicted returns could be replicated within the loss function to improve the efficiency of the algorithm.

1.2 Contribution

This paper analyses how a custom loss function can be organised to render the asymmetry of the objective function and can be implemented in deep neural networks. The rest of the paper is structured as follows: In section two, we analyse the literature review in terms of asymmetric or custom loss functions in finance for deep learning. Section three presents the framework of building various custom loss functions as alternatives to the MSE. Section four presents the methodology for testing the various loss functions. Section five presents the empirical results and demonstrates that a custom loss function significantly outperforms MSE. Section six concludes and draws some perspectives.

2. LOSS FUNCTIONS IN THE LITERATURE

Numerous academic papers present algorithms aimed at improving investment strategies 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 [1–6], if feasible at all. [7] offers the most comprehensive overview to date, with 190 articles reviewed over the period 2010 – June 2021, but with a narrow focus on the performance metrics used to compare algorithms predicting asset returns.

We analyze the loss functions in the literature from four sources: (i) articles analyzing the loss functions in prediction problems without referring to machine learning algorithms, (ii) the database of 190 articles predicting stock returns used by [7], (iii) some articles of machine learning in finance not covered in (ii) and that specifically analyze custom loss functions, and (iv) articles from other fields of AI that explicitly refer to the use of custom loss functions. Besides (ii), we search with Google

Scholar for the most relevant articles with "custom loss function", "asymmetric loss functions", and performing the same search replacing "loss" with "cost".

2.1 Asymmetric Losses Besides ML Models

The concept of asymmetric loss is not new. [8] tests an asymmetric loss function called Linear Exponential, often referred to as "LinEx" loss:

$$\text{LinEx}(y - \hat{y}, \gamma) = \frac{2}{\gamma^2}\left(e^{\gamma(y-\hat{y})} - \gamma(y - \hat{y}) - 1\right) \quad (1)$$

where $\gamma \neq 0$ is a parameter and Equation 1 is the custom loss function.

This loss function is a key element in [9] analysis of the properties of optimal forecasts under asymmetric loss and non-linearity. They demonstrate the properties of the LinEx loss, including its differentiability and stationarity. Among other time-series, [9] test the LinEx loss function versus the MSE for various applications, including for predicting the annualized return of Exxon stock over a period from 1970 to 2003.

LinEx loss has been widely tested since [8]. [10] use LinEx for solving prediction problems of asset returns. [11] extend the results to forecasting a volatility process as the sum of conditional volatility and an adjustment factor. [12] attempts a comprehensive overview of loss functions used in the financial sector. Besides the symmetric MSE, MAE and all their variations, he noticed three key asymmetric losses: the LinEx, Linear-Linear ("LLL") and Quadratic-Quadratic ("QQL") losses. [12] presents the LinEx as "particularly well-suited in financial applications". The LinEx[1] loss function organizes the asymmetry around the sign of the loss $(y - \widehat{y})$. The asymmetry of the investment problem, as presented in a simple version, is linked to the difference in signs between the actual return y and its predicted value $\widehat{y}$, or to the sign of the product $(y * \widehat{y})$. The sign of the loss is irrelevant for the economic efficiency of the algorithm.

2.2 Financial Articles Related to Algorithms Predicting Stock Returns

The loss function is, most of the time, not considered as a relevant topic. Using the aforementioned database, we examined each of the 190 articles and searched for the loss function(s) applied. 39 articles[2] describe or mention the loss function(s) applied, and 151 do not address the topic at all. 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 and 4 articles investigate a possible customisation, but none of them introduce asymmetry in the loss function.

[13] test various loss functions, starting with MSE, then using weighted MSE to underweight small stocks in favor 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 sign of the actual one.

[1] This is also true for LLL and QQL

[2] See Appendix A. of [7] Loss functions applied by author.

[14] enrich the MSE loss with a regularization based on the Sharpe ratio. All errors are treated equally. They obtain interesting results on 5 years out-of-sample da-ta, when transactions costs are at 0; but they obtain negative Sharpe ratios for each neural network architecture (MLP, CNN or LSTM)[3] as soon as transaction costs reach or exceed 5 basis points.

[15] combine the standard MSE to a directional cost to manage 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.

[16] combine a forecast error loss, MSE with a directional loss in a GAN[4] 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 [16] are the root mean squared relative error (RMSRE) and the direction prediction accuracy (DPA), unreliable error-based and an accuracy-based metrics[5].

2.3 Other Financial Articles

We also reviewed custom loss functions in articles dedicated to finance but not to prediction of returns, hence not covered in 2.2. We found three papers that investigate custom loss functions.

[17] test various asymmetric loss functions with linear regressions, decision trees, SVM and MLP to forecast the value of resold leasing cars. They compare the Quadratic-Quadratic Loss "QQL" function with the MSE. The trigger for asymmetry of the QQL is determined by the sign of the error, that is relevant for their purpose.

[18] suggest that algorithms and their constituents should be tailor-made for their 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.

[19] 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 and find a higher accuracy for the proposed custom loss function. They assume but do not prove that the superior accuracy should lead to a higher profit. Furthermore, the test set (out-of-time) is too limited (13 datapoints) to draw any reliable conclusion.

[3] MLP = Multi-Layer Perceptron, CNN = Convolutional Neural Network and LSTM = Long-Short Term Memory neural network, a form of Recurrent Neural Network (RNN).
[4] GAN = Generative Adversarial Network, here with an LSTM generator and a CNN discriminator.
[5] as demonstrated by [7]

2.4 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, even if the loss function is not a popular theme. 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. There-fore, the optimization should not minimize the mean error, but only 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 it with an out-of-time 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, high and low volatility periods. This is a necessary condition to produce reliable results[6].

In both training and testing phases, the algorithm will receive a set of historical data up to the end-of-day Closet 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 either 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.

[6] This is a recurrent issue with the 190 surveyed articles in 2.1, where many papers apply a test set insufficiently long, sometimes even equal or 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 $\widehat{y}$ 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[7] loss function and design 3 main types of customisation:

(i) an "adjusted MSE 1" (AdjMSE1) where the squared error is multiplied by α if (y * $\widehat{y}$) < 0 (when y and $\widehat{y}$ are of opposite sign), and by (1 / α) if (y * $\widehat{y}$) ≥ 0, (when y and $\widehat{y}$ have the same sign), with α being a positive number greater than 1. (when α equal, AdjLoss1 is equal to MSE).

(ii) an "adjusted MSE 2" (AdjMSE2) where the squared error is multiplied by α if (y * $\widehat{y}$) < 0 (when y and $\widehat{y}$ are of opposite sign), and by 1 if (y * $\widehat{y}$) ≥ 0, with α being a positive number greater than 1. (when α equal 1, AdjMSE2 is equal to MSE).

(iii) an "adjusted MSE 3" (AdjMSE3) that, to some extent, mimics the ReLu activation function where the squared error is multiply by (1+γ) if (y * $\widehat{y}$) < 0 (when y and $\widehat{y}$ are of opposite sign) and multiplied by γ if (y * $\widehat{y}$) > 0, with γ being a positive number smaller than 1.

AdjMSE1, AdjMSE2 and AdjMSE3 are not fully differentiable, when either y or $\widehat{y}$ is equal to 0.

(iv) Adj4 is a sigmoid-like curve adjustment factor that we apply to MSE and LinEx. Adj4 tends to a maximum for large negative values of (y * $\widehat{y}$) and tends towards a minimum for large positive values of (y * $\widehat{y}$). This adjustment is fully differentiable.

$$\text{Adj4} = \frac{\alpha}{\left(1 + \alpha - \frac{(\alpha - 0.5)}{(1 + e^{(\beta * y * \widehat{y})}}\right)} \tag{2}$$

with δ being a positive number greater than 1. We derive: AdjMSE4 = Adj4 * MSE and AdjLinEx4 = Adj4 * Linex.

We compute the error values for 7 loss functions with the parameters presented in TABLE 1.

Table 1: Tested parameters for each loss function.

Loss	α	β	γ
MSE			
LinEx			4
AdjMSE1	2		
AdjMSE2	2.5		
AdjMSE3	0.25		
AdjMSE4	2.5	90.000	
AdjLinEx4	2.5	90.000	4

[7] We fist compared MSE with RMSE, MAE, MAPE loss functions. MSE was the best performing loss function that has been retained as basis for custom loss functions.

FIGURE 1 presents the amount of loss for the main custom functions analyzed, for an actual return of -0.50%. MSE is the only symmetric loss function. LinEx is asymmetric around a loss of 0%, all other functions increase the loss when signs of actual and predicted returns are opposite and reduce the loss when actual and predicted returns have the same sign. The non-differentiability of AdjMSE1, AdjMSE2 and AdjMSE3 when the predicted return is 0% is visible with FIGURE 1.

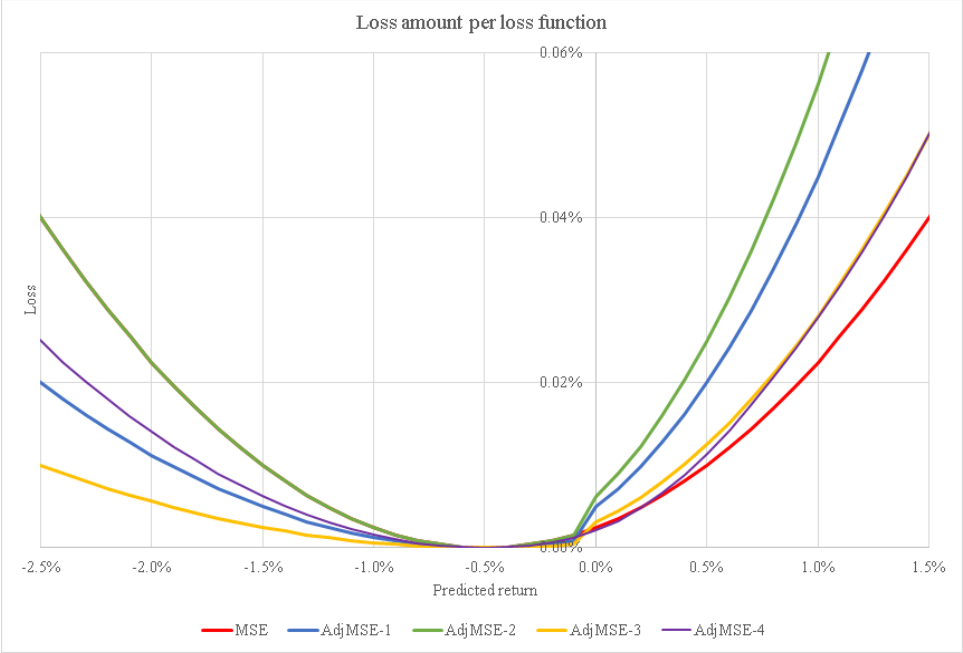

Figure 1. Loss level per loss function with a true return of -0.50%

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. The data set is made of 43 traded in USD on NYSE and 62 traded in EUR on Euronext. We organize for each stock a X matrix that is made of (i) Open – High – Low – Close - Volume daily data (OHLCV), (ii) a set of 14 technical indicators[8], and (iii) the closing prices of all the other stocks traded on the same exchange.

[8] 5 days moving average; 12-, 26- and 50-days exponential moving average, MACD Moving Average Convergence Divergence, Bollinger bands up & down 20 days, CCI Commodity Channel Index 14 days, ATR Average True Range 10 days, ADX Average Directional moving indeX 5 and 14 days, RSI Relative Strength Index 5 and 14 days and momentum 1 day.

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

FIGURE 2 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, high and low volatility periods, etc.

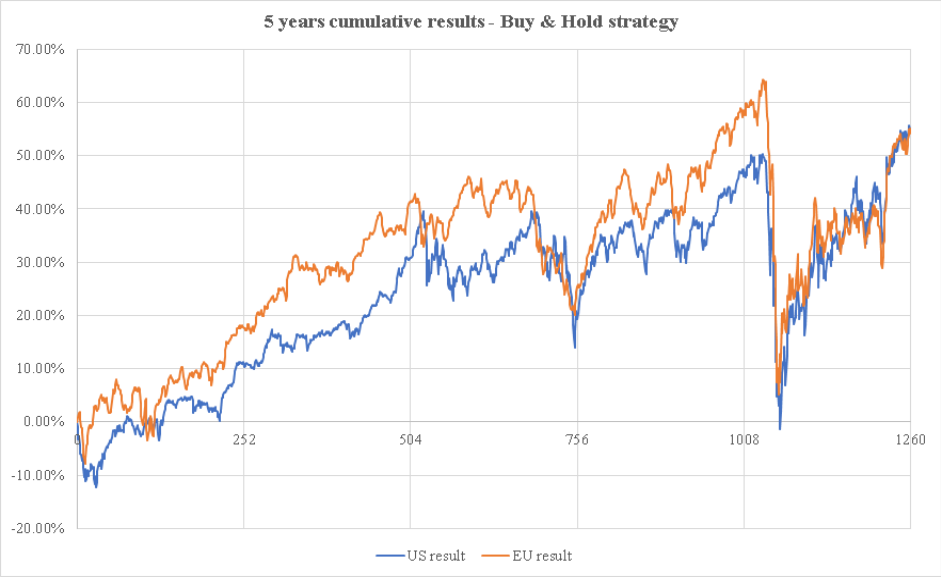

Figure 2. Cumulative results of buy & hold portfolios over 5 years

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

4.2 Model

To assess the various loss functions, we tested various straightforward MLP models for regression. We present herewith the results of one architecture with 8 hidden layers. The activation function is the standard Rectified Linear Unit (ReLu). With 61 inputs for US stocks and 8 hidden layers, we

obtain 188.893 learnable parameters. The number of parameters increases to 193.776 with the 80 inputs for EU stocks.

Other tested architectures delivered very similar results in term of relative efficiency of the various loss functions[9].

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

- The reproductibility[10] 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[11].

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, for each stock, we repeat this investment process: 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 of the OOT test set, if the return predicted by the ML algorithm is above -0.20% (that is 2 times the transactions costs[12] of 0.10%), 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 outstanding cash position for the stock resulting from previous investment decisions.

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 small 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 the stock is kept available without remuneration.

[9] Detailed results available upon request

[10] 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. Details about hardware and software are available upon request

[11] 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.

[12] The reason for this negative trigger is to avoid selling shares for avoiding a daily loss below the transactions costs

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[13], we compute the daily returns realized on the stocks and we compare them with the daily returns achieved with the buy & hold strategy. We obtain 105 series of 1260 daily returns per loss function. To test the effectiveness of each tested loss functions, we compare the risk-adjusted performances of the returns generated by the AI algorithm and with the risk-adjusted average return of the buy & hold strategy:

- We compare the absolute risk-adjusted performance with Sharpe and Sortino ratios of all the stocks;
- We compare the relative risk-adjusted performance of the algorithms with the D-ratio that measures the riskiness with the Value-at-Risk adjusted with the Cornish-Fisher expansion (CF-VaR).

$$D\text{-}ratio = \frac{\text{return}_{\text{algo}}/\text{CF-VaR}_{\text{algo}}}{\text{return}_{\text{b\&h}}/\text{CF-VaR}_{\text{b\&h}}} \tag{3}$$

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 strategy; if it is below 1, the algorithm under-performs the buy & hold. It also provides two sub-metrics: (a) the D-Return ratio which measures the average return generated by the algorithm divided by the average return of the buy & hold strategy, and (b) the D-VaR ratio that measures the relative risk (CF-VaR) of the returns from the buy & hold divided by the CF-VaR of the returns of the algorithm. With each sub-metric, a result above 1 indicates that the algorithm over-performs the buy & hold strategy, respectively with higher returns or with lower riskiness. To test the stability over time and the reliability of the performance of the algorithm with each loss function, each metric (D-ratio, D-Return and D-VaR) is computed on the entire 5 years period, and on two sub-periods of 2.5 years.

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 analyse whether 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 is above the performance of the MSE loss. We eventually verify that the superiority is consistent across most of the individual stocks and we compare the distribution function of the performance metric of each loss function.

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

5.1 Dissimilarity of the D Ratio From Custom Functions

To test the statistical difference between the series of Sharpe, Sortino and 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.

No loss function has distribution of Sharpe or Sortino ratios significantly different from MSE. D-ratio distribution for AdjMSE1, AdjMSE2 and AdjMSE3 are significantly different from MSE with a level well below 0.01; LinEx, AdjMSE4 and AdjLinEx4 don't.

5.2 Average Performance Metrics

The best algorithm has the highest and most stable Sharpe, Sortino and/or D-ratio, that is the highest expected return adjusted for risk, in absolute terms with Sharpe and Sortino, or relative to the risk-adjusted return of the buy & hold strategy for the D-ratio. TABLE 2 presents the average and median Sharpe and Sortino ratios per loss function. AdjMSE1, AdjMSE2 and AdjMSE3 significantly over-perform the MSE and LinEx and present interesting results, benefiting from the significantly increased loss when actual and predicted return change of sign. MSE achieves the lowest performance of all tested asymmetric loss functions.

Table 2: Average and median Sharpe and Sortino ratios per loss function

		Average ratio			**Median ratio**		
Loss function	**Param.**	**RoI**	Sharpe	*Sortino*	**RoI**	Sharpe	*Sortino*
Buy & Hold		**6.10%**	0.213	*0.343*	4.81%	0.18	*0.281*
MSE		1.88%	0.138	*0.238*	1.45%	0.094	*0.144*
LinEx		−0.72%	0.021	*0.064*	0.05%	0.003	*0.004*
AdjMSE1	2	7.13%	0.283	*0.451*	5.53%	0.234	*0.378*
AdjMSE2	2.5	6.85%	**0.31**	***0.496***	**7.06%**	**0.302**	***0.468***
AdjMSE3	0.25	6.88%	0.267	*0.427*	4.65%	0.204	*0.313*
AdjMSE4	2.5	2.72%	0.171	*0.286*	3.10%	0.154	*0.229*
AdjLinEx4	2.5	2.91%	0.187	*0.314*	0.00%	0.182	*0.248*

The results are confirmed with the D-Ratio. As shown in TABLE 3, MSE under-performs the buy & hold strategy with an average D-ratio of 0.354. Every custom loss function achieves better results than MSE, for the entire period, and for each sub-period. This con-firms our intuition that asymmetric loss functions over perform the MSE.

The best loss function is the AdjMSE2 with an average D-ratio of 1.722 over the 105 stocks (1.041 for the first sub-period and 1.859 for the second sub-period). AdjMSE2 out-performs the MSE and the buy & hold strategy for each sub-period.

AdjMSE2 also reaches a D-Return of 1.289 over 5 years, significantly above the -0.022 of the MSE loss. The custom loss function significantly over-performs MSE in terms of expected return and

Table 3: Average D ratio per loss function (the higher the better)

Loss function	**Param.**	**D ratio**	*D ratio 1*	*D ratio 2*	**D-Return**	*D-return 1*	*D-return 2*	**D-VaR**
Buy & Hold		**1.000**	*1.000*	*1.000*	**1.000**	*1.000*	*1.000*	**1.000**
MSE		0.354	*−0.746*	*1.395*	−0.022	*−0.990*	*0.546*	**1.518**
LinEx		1.120	*−0.854*	*0.124*	−0.480	*−1.180*	*0.045*	1.439
AdjMSE1	2	1.400	*0.722*	*1.424*	1.161	*0.708*	*1.202*	1.108
AdjMSE2	2.5	**1.722**	*1.041*	*1.859*	**1.289**	*0.888*	*1.371*	1.234
AdjMSE3	0.25	1.241	*1.122*	*1.214*	1.060	*1.114*	*1.048*	1.082
AdjMSE4	2.5	1.055	*1.249*	*1.305*	0.533	*0.505*	*0.619*	1.406
AdjLinEx4	2.5	0.760	*0.492*	*1.475*	0.098	*0.468*	*0.624*	**1.457**

beats the buy & hold strategy by 72.2% in risk-adjusted return, by 28.9% in absolute return and by 23.4% in risk measured with the CF-VaR.

With D-VaR above 1 across the table, all loss functions achieve a risk reduction compared to the buy & hold strategy.

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 effective realized 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 investors and on the benchmark strategy for reference, various customizations can be contemplated. Loss functions AdjMSE2 and AdjMSE3 appear to be among the best performers, not only in terms of risk-adjusted return metric (Sharpe more than doubled and D-ratio more than threefold) but also in terms of equilibrium between the effective return (D-return) and the effective risk-reduction (D-VaR). Eventually, the performances of these loss functions are relatively stable 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. Four 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. (iii) The loss functions could be tested with more complex investment strategies, possibly including long-short positions. Eventually, (iv) generalizing the principle of custom loss function could be efficiently applied, mutatis mutandis, to some Reinforcement Learning (RL) algorithms (like an on-policy actor-critic PPO model).

7. 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.

8. FUNDING

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

References

[1] Bustos O, Pomares-Quimbaya A. Stock Market Movement Forecast: A Systematic Review. Expert Syst Appl. 2020;156:113464.

[2] Huang J, Chai J, Cho S. Deep Learning in Finance and Banking: A Literature Review and Classification. Front Bus Res China. 2020;14:1-24.

[3] Meng TL, Khushi M. Reinforcement Learning in Financial Markets. Data. 2019;4:110.

[4] Ozbayoglu AM, Gudelek MU, Sezer OB. Deep Learning for Financial Applications: A Survey. Appl. Soft Comput. 2020;93:106384.

[5] Sezer OB, Gudelek MU, Ozbayoglu AM. Financial Time Series Forecasting With Deep Learning: A Systematic Literature Review: 2005-2019. Appl Soft Comput J. 2020;90:106181.

[6] Thakkar A, Chaudhari K. A Comprehensive Survey on Deep Neural Networks for Stock Market: The Need, Challenges, and Future Directions. Expert Syst Appl. 2021;177:114800.

[7] Dessain J. Machine Learning Models Predicting Returns: Why Most Popular Performance Metrics Are Misleading and Proposal for an Efficient Metric. Expert Syst Appl. 2022;199:116970.

[8] Zellner A. Bayesian Estimation and Prediction Using Asymmetric Loss Functions. J Am Stat Assoc. 1986;81:446-451.

[9] Patton AJ, Timmermann A. Properties of Optimal Forecasts Under Asymmetric Loss and Nonlinearity. J Econ. 2007;140:884-918.

[10] Christoffersen PF, Diebold FX. Further Results on Forecasting and Model Selection Under Asymmetric Loss. J Appl Econ. 1996;11:561-571.

[11] Huang J, Chai J, Cho S. Deep Learning in Finance and Banking: A Literature Review and Classification. Front Bus Res China. 2020;14:1-24.

[12] Singh BK. Loss Functions in Financial Sector: An Overview. Asian J Math Stat. 2015;8:35-45.

[13] Gu S, Kelly B, Xiu D. Empirical Asset Pricing via Machine Learning. Rev Financ Stud. 2020;33:2223-2273.

[14] Lim B, Zohren S, Roberts S. Enhancing Time-Series Momentum Strategies Using Deep Neural Networks. The J Financ Data Sci. 2019;1:19-38.

[15] Yun H, Lee M, Kang YS, Seok J. Portfolio Management via Two-Stage Deep Learning With a Joint Cost. Expert Syst Appl. 2020;143:113041.

[16] Zhou X, Pan Z, Hu G, Tang S, Zhao C. Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets. Math Probl Eng. 2018:1-11.

[17] Dress K, Lessmann S, Von Mettenheim HJ. Residual Value Forecasting Using Asymmetric Cost Functions. Int J Forecasting. 2018;34:551-565.

[18] Ahmed S, Hassan SU, Aljohani NR, Nawaz R. Flf-Lstm: A Novel Prediction System Using Forex Loss Function. Appl Soft Comput J. 2020;97:106780.

[19] Tavakoli M, Doosti H. Forecasting the Tehran Stock Market by Machine Learning Methods Using a New Loss Function. Adv Math Fin Appl. 2021;6:194-205.