# Deep Learning for Financial Time Series Prediction

Mirko Corletto

**Master's thesis**

# Deep Learning for Financial Time Series Prediction

Mirko Corletto

17. December 2020

Chair of Data Processing
Technische Universität München

Mirko Corletto. *Deep Learning for Financial Time Series Prediction.* Master's thesis, Technische Universität München, Munich, Germany, 2020.

Supervised by Prof. Dr.-Ing. Klaus Diepold and Matthias Kissel; submitted on 17. December 2020 to the Department of Electrical and Computer Engineering of the Technische Universität München.

Chair of Data Processing, Technische Universität München, 80290 München, Germany, http://www.ldv.ei.tum.de/.

# Abstract

Actively traded assets are among the most widely predicted time series in the world. When it comes to time series predictions, sequence to sequence deep learning models are particularly effective as they take into account the temporal properties of the input features. Yet, due to the competitive nature of the market, there is relatively little publicly available research on deep learning-based financial time series predictions. Many of the published papers promise excellent results. But most of them lack essential information, which makes the reproduction of their results impossible and thus highly questionable. With a transparent approach, in this thesis, I try to find out whether accurate predictions of indices, stocks, or cryptocurrencies are possible. In addition, I investigate if the predictions can be improved by using macroeconomic data as well as the volume of specific search terms on Google, which should reflect the general market sentiment. Neither the queried fundamental data nor the selected sentimental data could improve the predictions. While for most assets accurate predictions over the entire test period were not possible, a thoroughly good predictive accuracy could be observed on the most volatile days. This makes the prediction of high volatility assets like Bitcoin particularly interesting and provides a potential trading opportunity.

# Contents

# 1. Introduction

Forecasting price movements of financial time series has been of great interest since the early days of modern economy. Financial time series are highly non-stationary, noisy, and strongly affected by external influences like market psychology, macroeconomics, natural disasters, and politics, which makes them hard to predict. The efficient market hypothesis even goes so far as to claim that the market is unpredictable or, according to the semistrong form, at least only predictable through insider knowledge [60]. Yet the Medallion fund, a quantitative trading fund that was started in 1988 by Jim Simons and James Ax, achieved an impressive annual compound return of 63.3% in the period from 1988 to 2018 [20]. However, as the mutual fund study [36] by Gruber shows, it is by no means easy to outperform the market. According to the study, on average, actively managed funds were unable to outperform index funds such as the S&P 500 between 1985 and 1994.

In recent years machine learning techniques and mostly deep learning have disrupted many fields of research, such as computer vision, natural language processing, marketing, and robotics. Although machine learning is widely used in finance, be it for insurance, fraud detection, risk management, anti-money laundering, or trading, there is relatively little publicly available research on the subject. For financial time series predictions this is no surprise, since institutional investors and hedge funds are profit-oriented, disclosing their strategies would mean giving up their advantage over competitors and private investors.

Financial time series are sequences of multiple features. Sequence to sequence models are therefore particularly suited for their prediction. Over the last decade, recurrent neural networks (RNNs) and especially long short-term memory (LSTM) have shown impressive results in sequential problems like machine translation [86]. More recently, the focus has shifted to non auto-regressive models such as convolutional sequence to sequence learning [32] and transformers [89]. Although the encoder-decoder architecture of the transformer model is not particularly suited for financial time series predictions, some ideas such as the self-observation mechanism could potentially be used to improve the predictions of other sequence to sequence deep learning models.

Besides an adequate model that can handle sequential data, a carefully thought-out feature selection is necessary. Many different factors like technical indicators, patterns, fundamental data, as well as sentimental data could have an impact on the stock market. It would therefore make sense to use those features as inputs for the prediction task. In the case of few training data, throwing everything into one basket and letting the model decide which features to use would not work, however, as this would lead to overfitting on the training data due to the curse of dimensionality. Furthermore, the use of older observations could reduce the overfitting, but would lead to new complications. As the stock market continues to evolve, strategies tend to become outdated and stop working after a while. Therefore it is crucial to find a good compromise between the number of observations needed for sufficient generalization and how far in time these observations should date back. This makes the investigation of different preprocessing methods and the use of appropriate regularization techniques even more important for the prediction of financial time series.

## 1.1. Problem Formulation

By the end of this work, this thesis aims to answer the following two research questions:

Research Question 1

> ***"Can machine learning techniques and mostly deep learning, using historical price data and technical indicators, be effectively used to predict future prices of financial time series?"***

Research Question 2

> ***"Can the volume of specific search terms on Google, as well as macroeconomic data, improve the accuracy of machine learning-based financial time series predictions?"***

## 1.2. Thesis Contribution

This thesis analyzes the predictability of financial time series with three different sequence to sequence deep learning models, namely long short-term memory (LSTM), self-attention-based LSTM (ALSTM), and temporal convolutional network (TCN). To answer the two research questions, six different datasets are constructed which contain at least the base features of the underlying asset. The datasets are then expanded with either technical indicators, patterns, fundamental data, or sentimental data. The used features are partly based on the experience of experts and partly selected by means of causality tests. Besides feature scaling, the impact of time series denoising, dimensionality reduction, and transfer learning is investigated. The evaluation is done using hyperparameter tuning in a walk-forward nested cross-validation approach. The mean absolute scaled error (MASE) is used as a scaling and volatility invariant measurement metric to allow comparisons in different periods and between different assets. Furthermore, a long/short signal-based trading strategy is implemented and compared to the simple buy and hold strategy. In order to obtain meaningful results, different assets are evaluated.

## 1.3. Outline

In Chapter 2 I introduce the fundamentals of financial time series, machine learning (ML), and sequential deep learning, which are required for a basic understanding of the subject. In Chapter 3 I discuss the related work of financial time series prediction with special focus on deep learning-based approaches. Thereby I examine the limitations of the individual publications and try to address them in my work. In the first part of Chapter 4, I introduce six different datasets, whereas in the second part I describe the detailed deep learning frameworks. The most insightful results are presented and discussed in Chapter 5. A complete list of all results is provided in Appendix C. Finally, Chapter 6 concludes my findings.

# 2. Fundamentals

In the first section of this chapter, I introduce the basic principles of financial time series and the traditional analytical methods for their prediction. In Section 2.2 I give a short introduction to machine learning (ML) with special focus on deep learning for time series prediction. In the last section of this chapter, I describe three preprocessing techniques, namely data scaling, denoising of a time series, and dimensionality reduction. I only cover topics that are relevant to this work and recommend the book Deep Learning by Goodfellow et al. [33] for a deeper ML understanding.

## 2.1. Financial Time Series

Time series can be considered as sequences of random variables $\{X_t\}_{t\in T}$, which are often referred to as stochastic process in the field of probability theory. For financial time series the index set T is of discrete-time, resulting in $\underline{x}^{(t)} = \{x_1^{(t)}, x_2^{(t)}, \ldots x_n^{(t)}\}$ being a set of $n$ input features at the specific sampling points $t$ [83]. Time series that incorporate multiple input features are called multivariate time series. In his book [21], De Prado describes various methods of sampling from unstructured financial data and discusses their advantages and disadvantages. The most common approach is to sample at fixed time intervals, e.g. once per minute, hour, day, week, or month. Although this technique is the easiest method to implement, it has the drawback of over-sampling during periods of low trading activity and under-sampling during periods of high trading activity. This drawback can be avoided by sampling each time a predefined trading volume is exchanged.

Now that the basic principles of time series have been introduced, it is important to mention which input features make up a financial time series. The following five features form the basis, on which all further technical input features that will be introduced in the coming sections are built:

- Open (price of the first trade that happened in the sampling period)
- Close (price of the last trade that happened in the sampling period)
- High (highest price reached during the sampling period)
- Low (lowest price reached during the sampling period)
- Volume (number of shares traded during the sampling period)

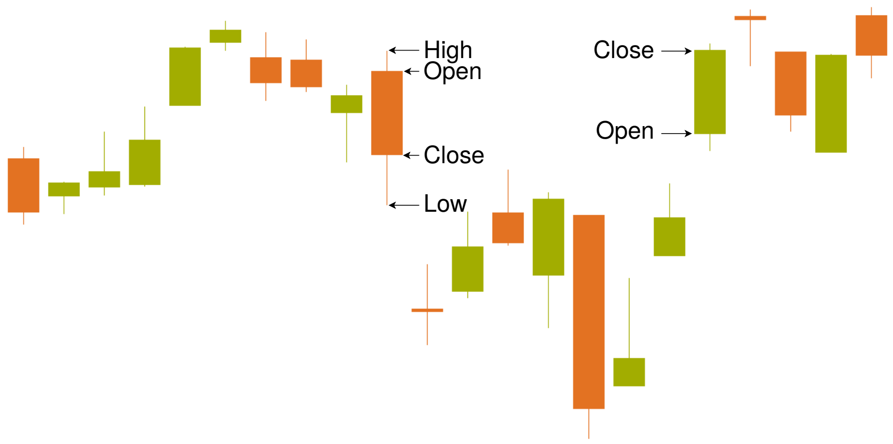


**Figure 2.1:** A candlestick chart where each candle corresponds to one sample.

In Figure 2.1 we can see a typical candlestick chart of a financial time series. Each candlestick corresponds to one sample showing the opening, closing, high and low price. For orange candlesticks, the closing price is lower than the opening price, indicating that the price has fallen during the sampling period. Conversely, for green candlesticks, the closing price is higher than the opening price, indicating that the price has risen during the sampling period.

In this thesis, I will use daily samples due to the limited availability of freely accessible financial data. This means that the opening and closing prices refer to the first and last trade that took place during the opening hours of the exchange.

### 2.1.1. Traditional Prediction Methods

The traditional stock market prediction can be subdivided into three categories, namely technical analysis, fundamental analysis, and sentimental analysis. Traders usually rely on these three analytical methods when making a trading decision.

#### Technical analysis

Technical analysis is the analysis of historical trading data, based solely on data from a multivariate financial time series. In Figure 2.2 we can see a typical candlestick chart where the stock price follows a specific trendline. In this case, the blue trendline acts as resistance, which means that every time the price approaches this trendline, there is a high chance that the price will fall. Once the price manages to close above the falling resistance trendline, the chances of a breakout rise. Conversely, trendlines can also act as support and their breakouts lead to a price drop [2]. Other frequently used tools

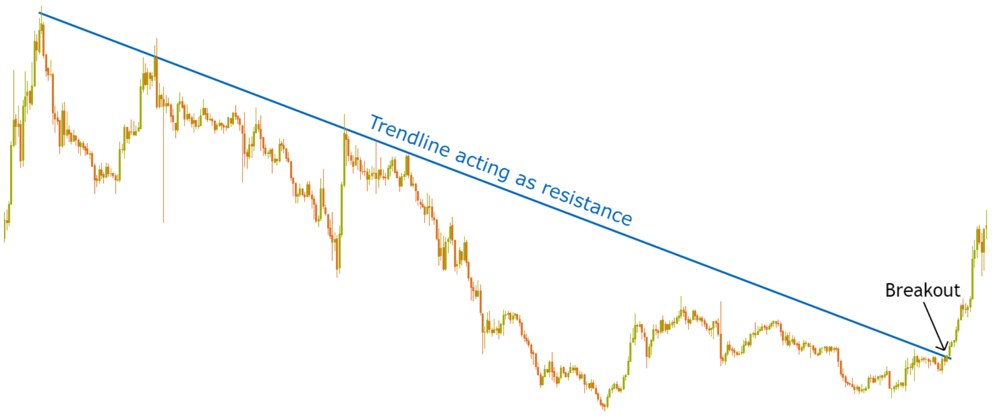


**Figure 2.2:** A candlestick chart with blue resistance trendline and subsequent breakout of the price trend.

in technical analysis are technical indicators. Technical indicators are mathematical and statistical tools used to gain insight into the trend, volatility, volume, and dynamics of financial time series [19]. In Subsection 4.1.1 several technical indicators used in this thesis will be introduced. Another common approach in technical analysis is the use of price patterns. Based on local minima and maxima, a specific pattern is identified, indicating the possibility of a trend continuation or reversal [81]. The price patterns used in this thesis will be discussed in Subsection 4.1.2.

### Fundamental analysis

Fundamental analysis is the analysis of financial ratios, reports, macroeconomics, and political events. It is used to determine the true value of a stock [51]. Financial reports contain data such as revenue, dividend payout, price to earnings ratio, profit, debt, invested capital, expenses, etc. Macroeconomics provide insights into the overall market and include data such as gross domestic product (GDP), inflation rate, interest rate, housing, average earnings, and unemployment rate.

### Sentimental analysis

Sentiment analysis is the analysis of market sentiment. It is used to capture public opinions and the feelings of the masses [56]. The market sentiment can be bearish, neutral, or bullish. A bearish sentiment means that the majority expects the price to move down, while a bullish sentiment means that the majority expects the price to move up.

### 2.1.2. Efficient Market Hypothesis

The efficient market theory states that stock prices reflect all relevant information and therefore in the long run it is impossible to outperform the market. The efficient market hypothesis is categorized in three forms, the weak, the semi-strong, and the strong form. The weak form implies that past market data is fully reflected in the current stock prices, therefore technical analysis cannot be used to predict future price movements. The semistrong form suggests that all publicly available information is already factored into current stock prices, therefore neither technical nor fundamental analysis can be used to predict future price movements. The strong form assumes that all information is incorporated into the current stock prices, whether public or insider information, which is why future price movements cannot be predicted at all [60]. Many economists and traders argue that the stock market is not perfectly efficient and thus future price movements can be predicted up to a certain degree [22, 27]. Although there are many active fund managers who have significantly outperformed the market in recent decades, supporters of the hypothesis claim that this is mainly due to higher risk exposure. This claim is supported by several studies showing that, on average, the returns of all actively traded mutual funds failed to outperform the market [11, 30, 36, 61]. One point on which proponents and opponents of the hypothesis tend to agree is that the market may at some point become irrational [61]. The time needed to restore its true value could then possibly be used to outperform the market.

Under the assumption that markets are not perfectly efficient, in this thesis I try to find out whether it is possible to predict financial time series using ML-based approaches, and if so, what kind of input features contribute to my findings.

### 2.1.3. Granger Causality

1969 C.W.J. Granger proposed a definition of causality involving two stationary time series $\{X_t\}_{t\in T}$ and $\{Y_t\}_{t\in T}$ that can be generalized for weakly stationary time series [35]. Let $X_t^{Lx} = (X_{t-Lx}, X_{t-Lx+1}, \dots, X_{t-1})$ represent the $Lx$-lagged vector of $X_t$ and $Y_t^{Ly} = \left(Y_{t-Ly}, Y_{t-Ly+1}, \dots, Y_{t-1}\right)$ represent the Ly-lagged vector of $Y_t$ with $Lx, Ly \in \mathbb{N}$ [41]. In addition let $P\left(X_t|\boldsymbol{U}_{t-1}\right)$ be the one-step-ahead least squares predictor of $X_t$ given the set $\boldsymbol{U}_{t-1}$ of past information that, among all available data, includes the two lagged vectors $X_t^{Lx}$ and $Y_t^{Ly}$. With the variance $\sigma^2(A|\boldsymbol{B})$ of the predictive error $\varepsilon_t(A|\boldsymbol{B})$, $Y_t$ is said to Granger cause $X_t$ with lag $Ly$ if and only if:

$$\sigma^2\left(X_t|\boldsymbol{U}_{t-1}\right) < \sigma^2\left(X_t|\boldsymbol{U}_{t-1} \setminus Y_t^{Ly}\right), \tag{2.1}$$

meaning that the past $Ly$ values of $Y_t$ help to predict $X_t$. To test for causality, the following vector autoregressive model is estimated by ordinary least squares:

$$\mathrm{X_t} = \alpha_0 + \sum_{\mathrm{k=1}}^{\mathrm{L}_x} \alpha_{11}^{\mathrm{k}} \mathrm{X_{t-k}} + \sum_{\mathrm{k=1}}^{\mathrm{L}_y} \alpha_{12}^{\mathrm{k}} \mathrm{Y_{t-k}} + \mathrm{u_{1,t}}. \tag{2.2}$$

Afterwards, a simple hpyothesis test (F-test or $\chi^2$-test) is applied with the null hypothesis beeing $H_0 : \alpha_{12}^1 = \alpha_{12}^2 = ... = \alpha_{12}^{Ly} = 0$. If the null hypothesis is rejected, $Y_t$ Granger causes $X_t$ with lag $Ly$ [50].

To remove any seasonality from the data, the Granger causality tests applied in this thesis are always performed on first-order derivatives.

### 2.1.4. Financial Ratios

Risk-adjusted return evaluation ratios are commonly used to compare the performance between different funds.

#### Sharpe Ratio

The Sharpe ratio describes the excess return of an investment by increasing the risk and can be computed as follows [82]:

$$S = \frac{\overline{r}_i - r_f}{\sigma_{r_i}}, \tag{2.3}$$

where $\overline{r}_i$ is the average return of investment and $r_f$ the risk-free rate, which can be set to $0$ due to the current interest policy.

#### Sortino Ratio

The Sortino ratio is similar to the Sharpe ratio, but only factors in the downside standard deviation, leaving the upside volatility out of the risk calculation [76]:

$$SR = \frac{\overline{r}_i - r_f}{\sqrt{\frac{1}{m}\sum_{j=1}^{m}\left[\min\left(0, r_i^{(j)} - r_f\right)\right]^2}}. \tag{2.4}$$

For highly volatile investments, the Sortino ratio is often preferred to the Sharpe ratio, since high upward volatility is actually a positive characteristic.

#### Mar Ratio

The Mar ratio is calculated by dividing the annualized return of an investment by the maximal drawdown of the investment. The maximal drawdown is the maximum proportional loss from a high of the investment to a low of the investment [59].

**Capital Asset Pricing Model**

The capital asset pricing model (CAPM) is a linear regression model that can be used to measure the return of an investment compared to the risk exposure [72]. Without interest, the CAPM can be described as follows:

$$\underline{r}_i = \beta \cdot \underline{r}_b + \alpha, \tag{2.5}$$

where $\underline{r}_i$ are the returns of the investment and $\underline{r}_b$ the returns of the benchmark. The constant $\alpha$ can be interpreted as the excess return of investment that cannot be explained by the benchmark. A good investment should have a positive $\alpha$. The factor $\beta$ can be interpreted as the risk exposure. A small $\beta$ indicates a low risk exposure compared to the benchmark, whereas $|\beta| > 1$ indicates a higher risk exposure compared to the benchmark.

## 2.2. Machine Learning-Based Prediction

Life is a continuous learning process. We educate ourselves through experience. When we as humans solve a task, we do not follow a mathematical formula, but rather we try to solve the problem intuitively. Tasks that we are confronted with on a daily basis will be very easy for us - think of walking, for example. Then there are tasks that can only be done by people who are specialized in these particular tasks, e.g. flying an airplane. Assuming you are not a pilot with many hours of experience, could you land a passenger plane safely? ML develops on the basis of these human-inspired ideas and combines them with ideas from linear algebra, probability theory, information theory, and numerical optimization [33]. Based on experience or example data, ML attempts to solve a specific task without explicitly programming the exact sequence of instructions. ML should not be confused with Artificial Intelligence (AI). ML is a sub-field of AI that focuses on solving individual tasks with the help of observations. The features of those observations should be carefully selected and tailored to the task to be solved. Just like for humans, a ML algorithm will perform poorly on tasks never experienced before.

ML is not about identifying the perfect algorithm. Rather, it is about identifying patterns, regularities, and irregularities in the input data and finding an approximation of the desired output using optimization techniques [3]. ML algorithms can be divided into three categories, namely supervised learning, unsupervised learning, and reinforcement learning. The basic concepts of unsupervised learning will be introduced in Subsection 2.3, while for reinforcement learning I refer to [87].

### 2.2.1. Supervised Learning

In supervised learning, the data is labeled, which means that given the input data we know the desired output. The ML algorithm then tries to approximate a function

$Y = f(X|\theta)$ by mapping the training data $X$ to its corresponding labels $Y$ and by optimizing the trainable parameters $\theta$ of a predefined model. After optimizing all the trainable parameters, the model is used to perform predictions on new, unseen data.

#### Classification

By assigning each label $Y$ a certain class code, we obtain a classification problem. Suppose we want to classify images that contain either a dog or a cat. Labeling all dog images with a 0 and all cat images with a 1 describes a typical two-class problem. After training a proper model, the ML-algorithm should be able to distinguish between new cat and dog images with reasonable accuracy.

#### Regression

In regression the data is labeled with real numbers instead of class codes. The task is then to find the relation that maps a given input data $X = \left\{\underline{x}^{(1)}, \underline{x}^{(2)}, \ldots, \underline{x}^{(N)}\right\}$ to its labeled outputs $Y = \left\{y^{(1)}, y^{(2)}, \ldots, y^{(N)}\right\}$. In case of a financial time-series prediction, the input data $X$ consists of a sequence with several features such as open, close, high, low, volume etc. If our goal is a one-step ahead prediction, we label the outputs $Y$ with the desired input feature (e.g. the closing price) one time step in the future.

### 2.2.2. Deep Learning

Deep Learning has its roots in the 1940s. Back then known as cybernetics, the first mathematical model of a human neuron was introduced in 1943 by McCulloch and Pitts [62]. Later, in 1958, Rosenblatt introduced the perceptron [77], the first model of a neuron that could learn the weights in a supervised fashion.

After decades without groundbreaking progress in the field, in the 1980s a second more successful wave of development called connectionism was underway. In 1979 Fukushima developed the neocognitron, a neural network architecture that uses 2d convolutions to recognize patterns in images [31]. In 1986 Rumelhart et al. published their paper [80] in which they successfully used backpropagation to train the weights of a neural network.

The financial time series prediction applied in this thesis is a special case of sequence to sequence learning, called many-to-one learning, where an input sequence is used to predict a single output. The greatest breakthrough in sequence to sequence learning was probably achieved by Hochreiter and Schmidhuber in 1997 with the introduction of the long short-term memory (LSTM) network [42].

Inspired by the neocognitron architecture, in 1998 LeCun et al. used backpropagation in a convolutional neural network (CNN) to recognize handwritten characters. Therefore they released the MNIST database, a database of handwritten digits which still serves as a benchmark today [54].

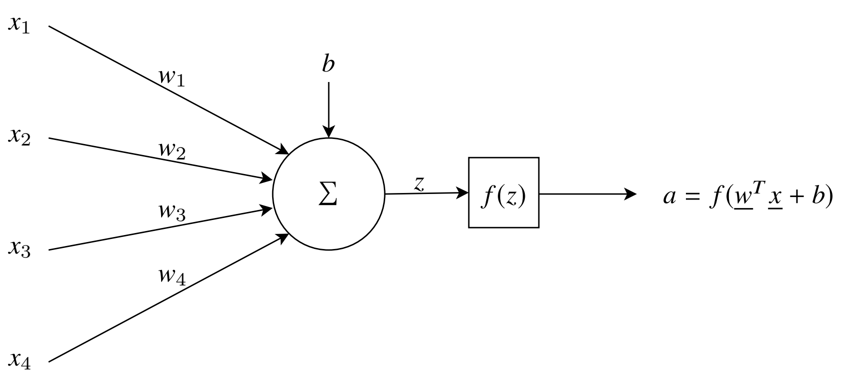


**Figure 2.3:** A single neuron, also called perceptron.

With the arrival of the new millennium, the second wave of development disappointed many investors. The expectations were just too high and could not be fulfilled [33].

During the last decade, deep learning has successfully disrupted many fields of research, such as natural language processing, computer vision, and robotics, and has become state of the art in many tasks. This was only partially caused by new discoveries such as greedy layer-wise training of deep networks [13], generative adversarial networks (GANs) [34] and transformers [47]. Back in the late 1990s, the main reason for failure was largely due to the lack of computational resources. The neural networks were simply too small and backpropagation was too complex for an enlargement of the nets. Today, some of the best performing deep learning architectures are still using the same methods discovered many years ago, with backpropagation, LSTM, and CNN beeing the most popular ones.

In Figure 2.3 we can see a single neuron with four input features $\underline{x} = \{x_1, \ldots, x_4\}$, their respective weights $\underline{w}$, the bias term $b$ and the activation function $f(z)$. A neural network consists of multiple such neurons. Figure 2.4 shows a simple feed-forward neural network with one hidden layer. Neural networks with more than one hidden layer are called deep neural networks (DNNs).

### Activation Function

The activation function $f(z)$ from Figure 2.3 can be used to transform the linear function $\underline{w}^T \underline{x} + b$ into a non-linear function. By applying a non-linear activation function to each hidden neuron of a sufficiently large network, it is possible to model any nonlinearity [55]. Without a non-linear activation function, a single output neural network could not solve non-linear problems such as the XOR problem.

Table 2.1 lists all activation functions that were used in this thesis. ReLU, which stands for Rectified Linear Unit, has become the preferred choice over sigmoid and tanh activation functions in recent years due to its advantageous properties in the learn-

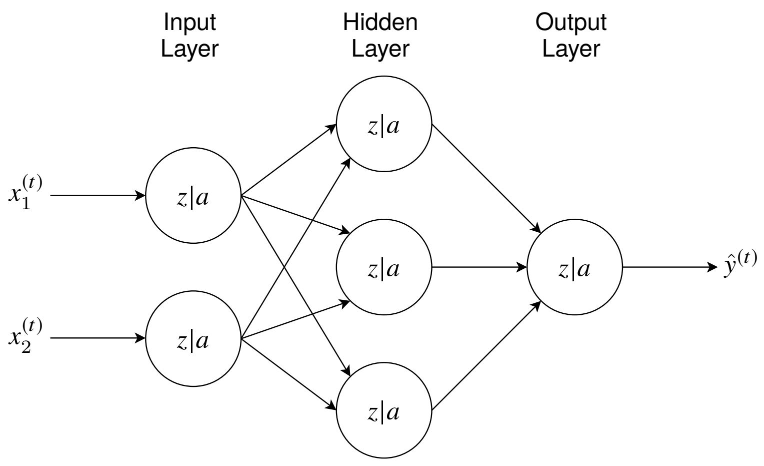


**Figure 2.4:** An exemplar feed-forward neural network with two input features, one hidden layer, and one output unit in the output layer.

ing process. Sigmoid and tanh are only sensitive to their input when $z$ is close to $0$. For large values of $z$, they saturate quite quickly, causing the gradient to go to 0. This makes the learning process with backpropagation very slow and difficult [33]. ReLU on the other hand does not saturate for positive values of $z$. The simple and consistent derivation of ReLU is also computationally more efficient, which usually leads to faster and better convergence in the training [52, 85]. One drawback is the dying ReLU phenomena, where neurons become inactive and the output of these neurons becomes 0 regardless of the input value [58]. Leaky ReLU and other similar rectified activation functions tackle this problem by assuring a non-zero gradient for every possible value of $z$ [92].

Neural networks with one unit in the output layer (see Figure 2.4) can be used for binary classification or regression. With a properly chosen cost function and the identity activation function for the output unit, we obtain a regression task.

## Cost Function

The cost function $J$ of a neural network is used to measure the deviation from the actual output value of the model to the target value. In regression we usually chose the mean squared error (MSE). This choise is supported by a simple maximum likelihood estimation with the modeling error assumed to be gaussian distributed [33]. The MSE can be calculated as follows:

$$\mathrm{J}_{\mathrm{MSE}} = \frac{1}{m} \sum_{t=1}^{m} \left\| \hat{y}^{(t)} - y^{(t)} \right\|^2, \tag{2.6}$$

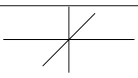
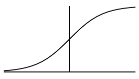
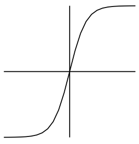
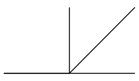
| Name | Plot | Function | Range | Derivative |
|---|---|---|---|---|
| Identity |  | $f(z) = z$ | $(-\infty, \infty)$ | $f'(z) = 1$ |
| Sigmoid |  | $\sigma(z) = \frac{1}{1+e^{-z}}$ | $(0, 1)$ | $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ |
| tanh |  | $f(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$ | $(-1, 1)$ | $f'(z) = 1 - f(z)^2$ |
| ReLU |  | $f(z) = \max(0, z)$ | $[0, \infty)$ | $f'(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$ |
| Softmax | | $f(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ | $(-1, 1)$ | $\frac{\partial f(z_i)}{\partial z_j} = \begin{cases} f(z_i)(1 - f(z_i)) & \text{if } i = j \\ -f(z_i) f(z_j) & \text{if } i \neq j \end{cases}$ |

**Table 2.1.:** An overview of the activation functions used in this thesis.

where $m$ is the number of observations, $\hat{y}^{(t)}$ the predicted value and $y^{(t)}$ the target value at time $t$.

### Gradient Descent

Gradient descent is an iterative optimization technique commonly used in deep learning to minimize the cost function $J$. Gradient descent relies on forward propagation and backpropagation algorithms. In forward propagation, the input data is propagated through the entire network to the output. Backpropagation, on the other hand, is used to efficiently calculate the gradients of the cost function with respect to the weights. For a detailed derivation of the forward propagation and backpropagation algorithms, I refer to [40], since their introduction would require a lengthy explanation of notation and therefore affect the reading flow.

The idea of gradient descent is to introduce a learning rate $\eta$, a small but positive scalar that determines the step size at each iteration. By gradually moving the weights in the opposite direction of the gradient with an adequate learning rate, gradient descent converges to a local minimum. Let $\boldsymbol{\theta}_i^{[l]} = [\boldsymbol{W}_i^{[l]}, \underline{b}_i^{[l]}]$ be the set of trainable parameters at layer $l$ and iteration $i$, then a gradient descent step can be performed as follows:

$$\boldsymbol{\theta}_{i+1}^{[l]} = \boldsymbol{\theta}_i^{[l]} - \eta \cdot \nabla_{\boldsymbol{\theta}_i^{[l]}} J\left(\boldsymbol{\theta}_i^{[l]}, \underline{x}^{(t:t+n)}, y^{(t:t+n)}\right), \tag{2.7}$$

while the set $\underline{x}^{(t:t+n)}$ depends on the gradient descent variant. With batch gradient descent, the entire training set is used for each optimization step, whereas with stochastic gradient descent (SGD) only one sample is used at a time. Mini-batch gradient descent combines the two variants, benefitting from the stable convergence of batch gradient descent and improved efficiency of SGD due to fewer gradient calculations [79].

### Adam Optimizer

The convergence of gradient descent strongly depends on the choice of the learning rate. A too small learning rate leads to very slow convergence and assuming the cost function is non-convex, it can also lead to convergence in suboptimal local minima. On the other hand, a too large learning rate can lead to a fluctuation around the minimum or even to divergence.

Adaptive moment estimation (Adam) is an optimization technique that similar to mini-batch gradient descent is used to minimize the cost function $J$ with the help of a learning rate. Adam, however, does not use a fixed learning rate. Instead, it adjusts the learning rate based on estimations of the first moment $\boldsymbol{m}_i$ (the mean) and second moment $\boldsymbol{v}_i$ (the uncentered variance) of the gradients [79]:

$$\begin{aligned} \boldsymbol{m}_i &= \beta_1 \boldsymbol{m}_{i-1} + (1-\beta_1)\,\nabla_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}_{i-1}, \underline{x}^{(t:t+n)}, y^{(t:t+n)}\right), \\ \boldsymbol{v}_i &= \beta_2 \boldsymbol{v}_{i-1} + (1-\beta_2)\,\nabla_{\boldsymbol{\theta}} \odot \nabla_{\boldsymbol{\theta}} J\left(\boldsymbol{\theta}_{i-1}, \underline{x}^{(t:t+n)}, y^{(t:t+n)}\right), \end{aligned} \tag{2.8}$$

with $\boldsymbol{m}_0 = \mathbf{0}$, $\boldsymbol{v}_0 = \mathbf{0}$ and decay rates $\beta_1$ and $\beta_2$ usually set to $0.9$ and $0.999$ respectively. Given the notations for element-wise division $\oslash$, for element-wise square root $\boldsymbol{A}^{\circ\frac{1}{2}}$ and for element-wise addition $\oplus$, an optimization step to update the parameters can be computed as follows:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta \hat{\boldsymbol{m}}_i \oslash (\hat{\boldsymbol{v}}_i^{\circ\frac{1}{2}} \oplus \epsilon), \tag{2.9}$$

where $\hat{\boldsymbol{m}}_i$ and $\hat{\boldsymbol{v}}_i$ are the bias-corrected moments:

$$\begin{aligned} \hat{\boldsymbol{m}}_i &= \frac{\boldsymbol{m}_i}{1-\beta_1^i}, \\ \hat{\boldsymbol{v}}_i &= \frac{\boldsymbol{v}_i}{1-\beta_2^i}, \end{aligned} \tag{2.10}$$

$\eta$ is the initial learning rate and $\epsilon$ a very small number for numerical stability [49].

### Regularization

With the methods that have been presented so far, we should be able to train a model that successfully maps some known inputs to the desired output. The goal of deep

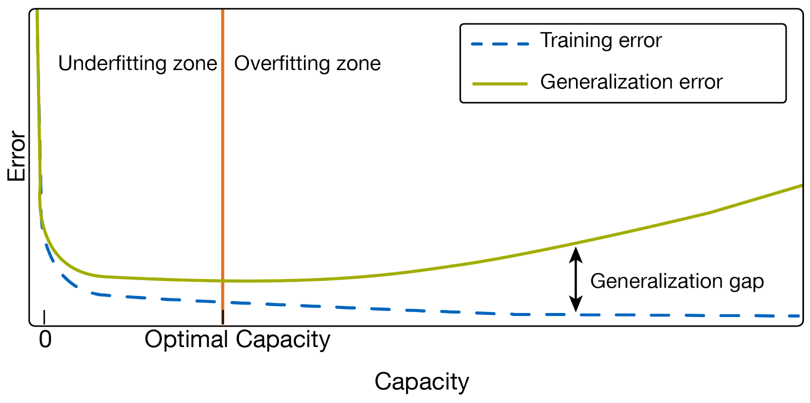


**Figure 2.5:** Typical U-shaped generalization error curve. The training error decreases as the model capacity increases. In the underfitting zone, the generalization error decreases until it reaches its minimum at optimal capacity. In the overfitting zone, the generalization error rises with increased capacity [33].

learning, however, is not just to achieve good performance on training data, but also on new unseen data.

In ML it is common practice to split the data into training, validation, and test sets. The training set is used to update all the trainable model parameters $\boldsymbol{\theta}$. The validation set is used to fine-tune and update the hyperparameters of the model (e.g. $\eta$) based on estimations of the generalization error (the expected error on a new input). After evaluating the estimated generalization error of the test set, no further changes to the model should be made. In Figure 2.5 we can see the typical U-shaped generalization error curve as a function of model capacity, which illustrates the famous bias-variance tradeoff in ML.

Regularization strategies aim to reduce the test error by preventing the model to overfit on the training data. The following are some commonly used regularization techniques:

- $L^2$ regularization adds a penalty term $\lambda||\theta||_2^2$ to the cost function, where $\lambda$ is a fixed regularization parameter. The $L^2$-norm regularization leads to a weight decay and prevents single weights from exploding.

- Activity regularization can be applied on a per-layer basis, as it adds the $L^1$-norm or $L^2$-norm as penalty term to the layer output. $L^1$ activity regularization encourages sparse layer outputs while $L^2$ activity regularization tries to reduce the output values.

- When using dropout, each neuron has a predefined probability that the output of the neuron is set to 0. For each input observation, $\underline{x}^{(t)}$ the neurons to be dropped are sampled independently at random. Thus, dropout is a computational very efficient strategy to prevent overfitting.

- Batch normalization is used to normalize the inputs of a layer at each mini-batch. As a result, the distribution of the inputs will not change dramatically from batch to batch. This has the positive effect of stabilizing and speeding up the training process [46].
- Early stopping is another type of regularization that requires an evaluation of the validation set after each epoch. As soon as the validation error has not improved for a predetermined amount of training steps, the training is stopped to prevent overfitting.

Finding the optimal capacity of a model is by no means easy. Therefore, the use of regularization techniques is a crucial step to prevent overfitting. Recent results have shown that once the interpolation point of a model is reached (i.e. the training error converges to 0), an increase in capacity in combination with adequate regularization leads to a decrease of the generalization error. A further increase in capacity can then lead to a lower generalization error than at optimal capacity [12].

### 2.2.3. Recurrent Neural Networks

In Section 2.1 multivariate financial time series were introduced as sequences with multiple input features. As input observation for a time series prediction task, it would therefore make sense to use input sequences as well. Feed-forward neural networks are not particularly suited for this type of task because they cannot properly capture the temporal domain of the inputs. Generally speaking, it would be possible to use sequences in a feed-forward neural network by stacking a predefined amount of $\tau$ vectors to one big input vector $\underline{x}^{(t:t+\tau)}$. Since this would require the training of separate weights for the same input features at different time steps, the neural network would have a hard time generalizing on the input data. Recurrent neural networks (RNNs) were designed to deal with sequential data. Unlike feed-forward neural networks, RNNs share their weights across time. In Figure 2.6 we can see a RNN that uses a $\tau + 1$ long sequence as input observation to predict a single output. The hidden states are defined as:

$$\underline{h}^{(t)} = f\left(\underline{b} + \boldsymbol{W}\underline{h}^{(t-1)} + \boldsymbol{U}\underline{x}^{(t)}\right), \tag{2.11}$$

where $\boldsymbol{W}$ are the hidden-to-hidden weights, $\boldsymbol{U}$ the input-to-hidden weights and $\boldsymbol{V}$ the hidden-to-output weights. Because of the recurrent nature of the hidden state computations, RNNs suffer from the vanishing and exploding gradient problem [33].

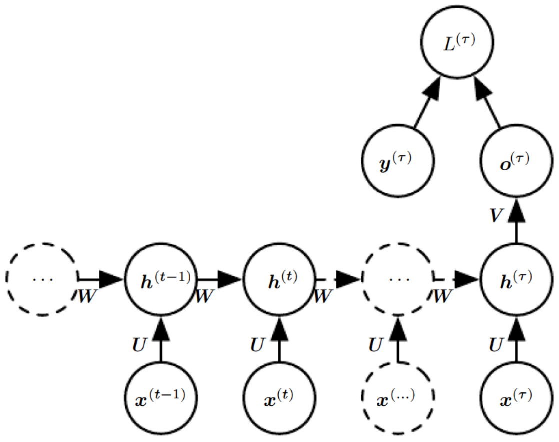


**Figure 2.6:** Recurrent neural network (RNN) with a single output prediction. For multivariate time series the inputs $x^{(t)}$ are vectors representing the features at time $t$ [33].

### Long Short-Term Memory

Long short-term memory (LSTM) is a special type of RNN that tackles the vanishing gradient problem by controlling the information flow through three additional gates. While the outer recurrent structure remains the same as in Figure 2.6, the hidden states are replaced by so-called LSTM cells (see Figure 2.7).

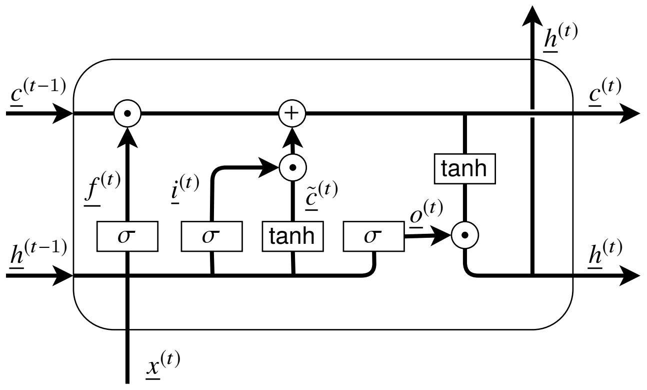


**Figure 2.7:** The structure of a LSTM cell with forget gate $\underline{f}^{(t)}$, input gate $\underline{i}^{(t)}$ and output gate $\underline{o}^{(t)}$.

The three sigmoid activation functions are used to ensure a gate value between 0 and 1, whereas the two tanh activation functions introduce the non-linearity. The memory cell $\underline{c}^{(t)}$ is computed with the forget gate $\underline{f}^{(t)}$, which controls how much information from the previous memory cell should be retained, and with the input gate $\underline{i}^{(t)}$, which controls the contribution of new information [68].

**Attention Mechanism**

The significance of an input feature can vary depending on its temporal occurrence in the sequence. The important part, however, may not be in the same position for every input of a training batch. In natural language processing, for example, a keyword could appear at the beginning or end of a sentence and still have the same meaning. Self-attention mechanism performs a softmax operation on the hidden states of the RNN. This can be understood as a weighting of the hidden states at each time step. In this way, the model learns to concentrate on the important parts of the sequences, which in turn means that the complexity of the network can be kept smaller [7].

### 2.2.4. Temporal Convolutional Network

In recent years convolutional neural networks (CNNs) have shown exceptional results in image processing tasks, as they are particularly suitable for input data with translation-invariant properties [48]. With few changes in the basic structure, CNNs can also be used for time series predictions with sequential data. Temporal convolutional network (TCN) is a special type of CNN that was designed for sequence to sequence predictions. Similar to DeepMinds WaveNet architecture [69], TCN uses dilated causal 1D convolutions for its prediction task as shown in Figure 2.8. The number of units in each layer corresponds to the length of the input sequence times the total number of filters. It is basically a 1D fully convolutional network with zero padding, where only past outputs of previous layers are used for the convolutions. A dilated convolution is defined as:

$$z^{(t)} = \sum_{i=0}^{k-1} W_i \cdot \mathbf{x}^{(t-d\cdot i)}, \tag{2.12}$$

where $d$ is the dilation factor, $k$ the kernel size, and $W_i$ the kernel weight which is shared within the same layer [8].

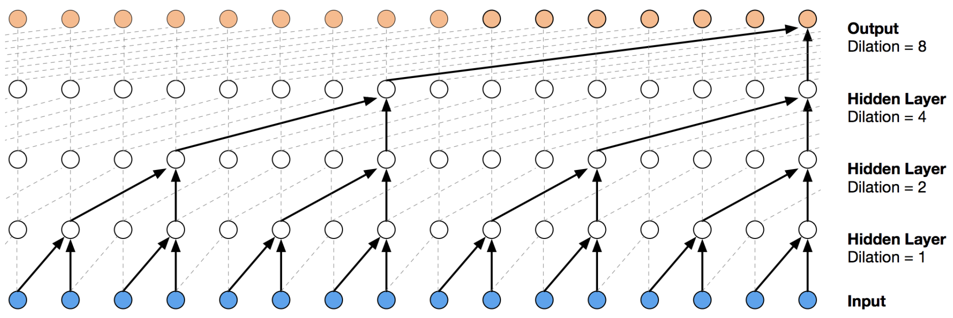


**Figure 2.8:** Dilated causal 1D convolution structure applied in TCNs. The dilation factor $d$ is increased exponentially with the depth of the network. All layers use a fixed kernel size $k = 2$ [69].

### Residual Learning

The best results in image recognition have been achieved by very deep CNNs and especially by residual neural networks. Residual neural networks use skip connections between layers to allow a direct gradient flow through backpropagation. This addresses the vanishing gradient problem, allowing networks to grow deeper and deeper [39].

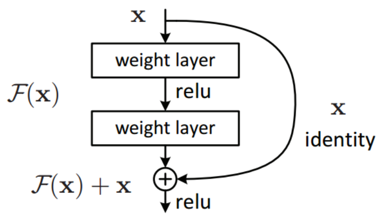


**Figure 2.9:** A residual block with identity mapping as skip connection [39].

For TCN predictions several residual blocks can be stacked together. Each residual block has the same structure as shown in Figure 2.8. Thus, for a one-step-ahead prediction with an input sequence length $\tau = 16$, two residual blocks with dilation factors $[1, 2, 4]$ could be stacked together instead of using a single block with dilation factors $[1, 2, 4, 8]$. To deal with different input-output shapes when using skip connections between residual blocks, TCN uses 1x1 convolutions instead of a direct identity mapping as shown in Figure 2.9.

## 2.3. Data Preprocessing

In Section 2.1 different sampling techniques for financial time series were outlined. The sampling of unstructured data is an essential preprocessing task that must be performed before the data can be fed into a predictor. While in theory the sampled data could be directly used for predictions, in practice further preprocessing steps are necessary to obtain reasonable results.

### 2.3.1. Feature Scaling

Usually, the input features of observations are of different scale. For a stable learning process, it is important that all features lie in the same order of magnitude. Especially in combination with regularization strategies, it is essential to rescale the data so that all features can be considered equally. The scaling is performed on each feature independently:

$$\underline{x}_i = \frac{\underline{x}_i - \min(\underline{x}_i^{(1:T_{\text{train}})})}{\max(\underline{x}_i^{(1:T_{\text{train}})}) - \min(\underline{x}_i^{(1:T_{\text{train}})})} * (LB - UB) + LB, \tag{2.13}$$

where $\underline{x}_i$ is the $i^{th}$ input feature vector and $(LB, UB)$ is the feature range which is usually set to $(0, 1)$. To prevent data leakage into the validation and test sets, the scaling factors are performed on the training set and then applied to the validation and test sets. As an alternative to the "MinMaxScaler" from equation 2.13, it is common practice to standardize the input features [71].

### 2.3.2. Denoising a Time Series

The most intuitive way to denoise a signal is probably to use the Fourier transform with subsequent filtering in the frequency domain. The Fourier transform decomposes a signal into a sum of sinusoids:

$$x(t) \xrightarrow{\mathcal{F}} X(\omega) := \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt. \tag{2.14}$$

The transformed signal $X(\omega)$ can then be denoised with an adequate filter and afterwards transformed back into the time domain. This procedure is suited for signals whose frequency does not change over time, e.g. for images or audio signals. For signals without a constant frequency over time (e.g. non-stationary time series), the so-called short-time Fourier transform (STFT) was introduced:

$$X(\tau, \omega) := \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j\omega t}dt, \tag{2.15}$$

where $w(t-\tau)$ is the window function with time index $\tau$. Basically, the signal is divided into segments of equal length, and the Fourier transform is applied to each segment separately. Although this method solves a limitation of the Fourier transform by introducing a time resolution, it has the disadvantage of having a fixed time and frequency resolution. Limited by the Heisenberg–Gabor uncertainty, the Fourier transform can either have a good time resolution or a good frequency resolution, but not both [37].

The wavelet transform partially overcomes the resolution limitations by introducing a multi-resolution structure:

$$X(a, b) := \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \Psi\left(\frac{t-b}{a}\right) x(t)dt, \tag{2.16}$$

where $\Psi$ represents the wavelet function, $a$ the scale parameter, and $b$ the position parameter. A small scaling factor $a$ compresses the wavelet, resulting in a good time resolution, but at the same time in a poor frequency resolution at high frequencies. Conversely, a large scaling factor $a$ stretches the wavelet, resulting in a good frequency resolution, but at the same time in a poor time resolution at low frequencies [4].

Based on the discrete wavelet transform, a continuous noisy signal $x(t)$ can be denoised as follows:

$$W_{j,k} = \int_{-\infty}^{+\infty} x_d(t)\psi_{j,k}(t)dt \quad (j = 1, 2, ..., J), \tag{2.17}$$

where $x_d(t)$ is the derivate of $x(t)$ and $W_{j,k}$ are the wavelet coefficients with $j, k \in \mathbb{N}$ beeing the scaling factor and position respectively. The basis function $\psi_{j,k}(t) = 2^{j/2}\psi\left(2^j t - k\right)$ is calculated with the mother wavelet $\psi(t)$. There exist different mother wavelets that applied to the same signal may produce different results [64]. One of these mother wavelets is the Haar wavelet:

$$\psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ -1 & \frac{1}{2} \leq t < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.18}$$

Since small scaling factors $j$ represent high frequencies, the signal can be denoised by setting the first $j_0$ wavelet coefficients $W_{j,k}$ to zero. The threshold $j_0$ can be determined with the universal thresholding method:

$$j_0 = \sigma\sqrt{2\log n}, \tag{2.19}$$

where $\sigma$ is the estimated standard deviation from the median absolute deviation of the wavelet coefficients at the finest frequency resolution level $J$ [25].

Afterwards, the denoised signal is reconstructed by inverse transforming the wavelet coefficients $W_{j,k}$:

$$\hat{x}_d(t) = \frac{1}{c_\psi}\sum_{j=0}^{J}\sum_{k=0}^{\infty} W_{j,k}\psi_{j,k}(t), \tag{2.20}$$

with the wavelet admissible constant:

$$c_\psi = \int_{-\infty}^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega, \tag{2.21}$$

where $\hat{\psi}$ is the Fourier transform of the mother wavelet.

Finally, the reconstructed signal $\hat{x}_d(t)$ is integrated, and we obtain the denoised signal $\hat{x}(t)$ [78].

Similar to the fast Fourier transform, there exists a fast wavelet transform for discrete signals [18].

### 2.3.3. Dimensionality Reduction Using Autoencoders

Autoencoders (AEs) are neural networks, where the desired output Y is an identical copy of the input X. Since there is no labeling involved, AEs are part of unsupervised learning. Unlike supervised learning, where we have labels that link the input data $X$ to a desired output $Y$, in unsupervised learning we try to find regularities and patterns in the input data. Among other things, unsupervised learning techniques are commonly used for clustering [93], dimensionality reduction [88], parameter estimation [63] and generative modeling [24].

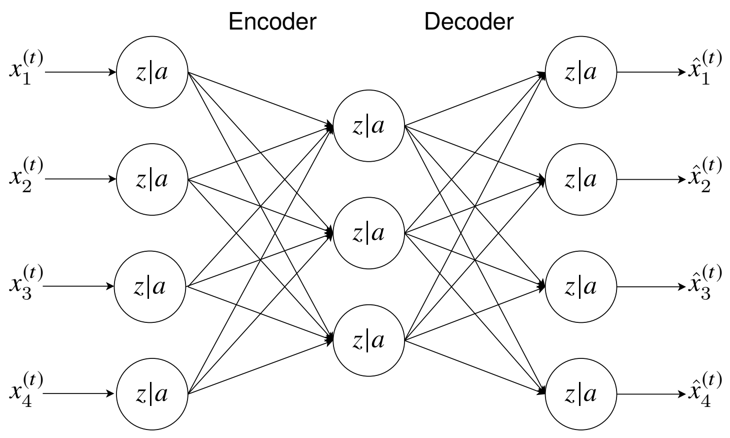


**Figure 2.10:** Exemplar undercomplete autoencoder with one hidden layer.

While AEs are capable of all the above mentioned tasks, here we are interested in the undercompleted AE for dimensionality reduction (see Figure 2.10). In the encoder part, the input features are encoded into a latent lower-dimensional representation. On the other side, the decoder tries to reproduce the original input data from the latent representation as accurately as possible. The latent representation can then be used to generate new lower-dimensional observations for the prediction task.

Deep autoencoders (DAEs) have more than one hidden layer. In the encoder part, the number of neurons in each layer is gradually reduced until the desired dimension is reached. Conversely, the decoder reconstructs the input in a mirrored fashion.

With linear activation functions and the MSE as cost function, the undercomplete AE will approximately span the same orthogonal subspace as the principal component analysis (PCA). By using non-linear activation functions, undercomplete AEs should be able to learn a more powerful non-linear representation of the input data [33].

# 3. Related Work

To be profitable, trading with the analytical methods described in Section 2.1.1 requires lots of experience and the right mindset. Most traders lose money due to emotional trading decisions [26]. Quantitative trading consists of trading strategies that rely on mathematical models and statistical analysis. The trading decisions are no longer made by humans, which excludes any involvement of emotions.

In this chapter, I review several papers that have investigated financial time series predictions. Note that due to the different evaluation approaches, no true state of the art could be explored.

## 3.1. Traditional Financial Time Series Prediction

Autoregressive moving average (ARMA) models make predictions based on combinations of an autoregressive model (AR) and a moving average model (MA). The MA model requires the time series to be stationary. For non-stationary time series like for stock predictions, Box and Jenkins introduced the autoregressive integrated moving average (ARIMA) process [15]. Although ARIMA models are used for stock predictions [5], their use is limited by their inability to model nonlinearities. Therefore, several hybrid models have been proposed that combine the linear ARIMA model with nonlinear ML algorithms. Zhang's hybrid model [95] combines ARIMA with a feed-forward neural network. Pai and Lin [70] used a hybrid ARIMA and support vector machine (SVM) model to predict several stocks. Other ML models that were used to predict the stock market are ensemble methods (Random Forest, AdaBoost, XGBoost) [9, 67] and SVR [94]. Although these ML methods have the advantage of requiring fewer observations for sufficient generalization, they are not able to capture the temporal characteristics of sequential data as RNNs do.

## 3.2. Deep Learning-Based Financial Time Series Prediction

The extraordinary results from deep learning models in the areas of computer vision and natural language processing have shifted researchers' interest from traditional stock predictions to deep learning-based predictions. Hsieh et al. [44] applied a wavelet transformation to denoise financial time series containing fundamental data and technical indicators. The predictions were performed with a RNN and artificial bee colony algorithm for hyperparameter tuning. Surprisingly, the authors only considered input features from the previous day instead of sequential observations. Because of the advantages of LSTM models compared to classical RNN, recent studies on deep learning-based stock predictions have focused on LSTM models (e.g. [17] and [28]). Both publications showed promising results, but unfortunately, their results can hardly be reproduced due to an incomplete description of their methodology. Fischer and Krauss [29], on the other hand, provided a very detailed description of their LSTM stock predictions. They used the daily return of 240 consecutive trading days from several stocks to predict whether a stock would be undervalued or overvalued. Even though they only used daily returns as input features, they managed to outperform the market. Nguyen et al. [66] used transfer learning with an LSTM model to address the problem of having too few observations for their predictions. They concluded that using stocks from a similar field would increase the accuracy of their predictions, but did not specify how the stocks were selected. Bao et al. [10] used wavelet denoising and stacked autoencoders to preprocess the financial time series before feeding the data into an LSTM model. The preprocessing steps drastically increased their predictive accuracy. The precise preprocessing and parameter optimization procedure was not specified, however, which makes it impossible to reproduce their results. Hollis et al. [43] and Qiu et al. [74] enhanced their LSTM stock predictions with an attention mechanism. Both papers concluded that an attention mechanism could improve LSTM-based stock predictions. Wan et al. [90] compared the performance of LSTM and TCN-based time series predictions, highlighting the training efficiency of the latter. Wiese et al. [91] introduced a generative adversarial network (GAN) that uses TCNs. The GAN could potentially be used to address the overfitting problem that often occurs with too few observations.

## 3.3. Predictions Based on Sentimental Analysis

The sentimental data used for stock predictions are usually obtained from the analysis of newspapers, social media, or search engines. Ding et al. [23] used structured events with Bloomberg and Reuters articles to classify whether the price would move up or down when a relevant article was published. They achieved good accuracy on their test data. If such an implementation would work with real-time data remains questionable, since stock prices usually react quite quickly to relevant news. In 2018 the hedge fund Two Sigma started a Kaggle competition[1] to predict stock movements based on technical data and Reuters news. Unfortunately, the competition is closed and the datasets are no longer accessible. Bollen et al. [14] used two Google mood tracking tools (OpinionFinder and GPOMS) on daily Twitter feeds and combined them with historical price data to predict the Dow Jones (DJI) closing price. Their results were evaluated on a single test set of just 15 trading days (December 1 to December

[1]https://kaggle.com/c/two-sigma-financial-news

19, 2008). Due to the limited availability of the Twitter API, it is not easy to verify whether their approach would be successful for longer test periods. Nguyen et al. [65] used the SentiWordNet opinion mining tool on the Yahoo finance message board to improve the average prediction accuracy on 18 different stocks. They used SVM as their base model, which in other publications like [66] turned out to be inferior to LSTM-based stock predictions. Price et al. [73] analyzed the predictive power of 98 different Google Trends search terms on the DJI closing price. They used a rather simple model that gave buy and sell signals based on the relative change in the search volume of a keyword. With several Google Trends search terms, they managed to outperform the market in the period from January 2004 to February 2011. Abraham et al. [1] analyzed the sentiment of tweets that included the hashtags #bitcoin and #ethereum to predict the daily closing price of the two cryptocurrencies. They concluded that using the volume of the two search terms instead of a bearish/neutral/bullish sentimental analysis gave better results for their cryptocurrency predictions. Their predictions were carried out using a simple linear regression model.

# 4. Methods

In this chapter, I describe the methods and the way in which experiments were conducted. To ensure the credibility of the results, the procedure is described in as much detail as necessary to facilitate a possible reproduction of the results.

## 4.1. Feature Selection

To check whether technical indicators, patterns, sentimental data, or fundamental data could improve my financial time series predictions, I created six different datasets for each asset. All six datasets include at least the daily sampled features open, close, high, low, and volume (OCHLV) of the underlying asset, which were fetched through the yfinance[1] API. The following subsections describe the remaining five datasets that were evaluated in addition to the OCHLV dataset.

### 4.1.1. Technical Indicators

The technical indicators used for the predictions were calculated using ta[2], the technical analysis library in python. I created two different technical indicator datasets, one dataset based on the experience of several traders (TA-experience) and a second dataset that uses all indicators of the ta library that passed the causality test (TA-causality). The technical indicators used in the TA-experience dataset are:

- $EMA_t = \begin{cases} x^{(1)}_{\text{close}} & \text{if } t = 1 \\ \frac{2}{T+1} \cdot x^{(t)}_{\text{close}} + (1 - \frac{2}{T+1}) \cdot EMA_{t-1} & \text{if } t > 1 \end{cases}$, for $T = 12, 26$
- $MACD = EMA_{12} - EMA_{26}$
- $RSI_T = 100 - \left[\frac{100}{1 + \frac{\text{avg. gain last T days}}{\text{avg. loss last T days}}}\right]$, for $T = 3, 7, 14$
- Bollinger bands $= \frac{\sum_{t=1}^{20} x^{(t)}_{\text{close}}}{20} \pm 2\sigma$
- $OBV_t = OBV_{t-1} + \begin{cases} \text{volume} & \text{if} x^{(t)}_{\text{close}} > x^{(t-1)}_{\text{close}} \\ 0 & \text{if} x^{(t)}_{\text{close}} = x^{(t-1)}_{\text{close}} \\ \text{- volume} & \text{if} x^{(t)}_{\text{close}} < x^{(t-1)}_{\text{close}} \end{cases}$

[1]https://github.com/ranaroussi/yfinance

[2]https://github.com/bukosabino/ta

- $ROC_{12} = \frac{x_{\text{close}}^{(t)} - x_{\text{close}}^{(t-12)}}{x_{\text{close}}^{(t-12)}}$

The technical indicators used for the TA-causality dataset are listed in Appendix A.

### 4.1.2. Patterns

For the TA-patterns dataset, eight different technical analysis patterns were identified, which are described in Lo et. al's paper [57]:

- $\text{HS} \equiv \begin{cases} E_1 \text{ is a maximum} \\ E_3 > E_1, E_3 > E_5 \\ E_1 \text{ and } E_5 \text{ are within } 1.5 \text{ percent of their average} \\ E_2 \text{ and } E_4 \text{ are within } 1.5 \text{ percent of their average} \end{cases}$

- $\text{IHS} \equiv \begin{cases} E_1 \text{ is a minimum} \\ E_3 < E_1, E_3 < E_5 \\ E_1 \text{ and } E_5 \text{ are within } 1.5 \text{ percent of their average} \\ E_2 \text{ and } E_4 \text{ are within } 1.5 \text{ percent of their average} \end{cases}$

- $\text{BTOP} \equiv \begin{cases} E_1 \text{ is a maximum} \\ E_1 < E_3 < E_5 \\ E_2 > E_4 \end{cases}$

- $\text{BBOT} \equiv \begin{cases} E_1 \text{ is a minimum} \\ E_1 > E_3 > E_5 \\ E_2 < E_4 \end{cases}$

- $\text{TTOP} \equiv \begin{cases} E_1 \text{ is a maximum} \\ E_1 > E_3 > E_5 \\ E_2 < E_4 \end{cases}$

- $\text{TBOT} \equiv \begin{cases} E_1 \text{ is a minimum} \\ E_1 < E_3 < E_5 \\ E_2 > E_4 \end{cases}$

- $\text{RTOP} \equiv \begin{cases} E_1 \text{ is a maximum} \\ \text{tops are within } 0.75 \text{ percent of their average} \\ \text{bottoms are within } 0.75 \text{ percent of their average} \\ \text{lowest top } > \text{ highest bottom} \end{cases}$

- $\text{RBOT} \equiv \begin{cases} E_1 \text{ is a minimum} \\ \text{tops are within } 0.75 \text{ percent of their average} \\ \text{bottoms are within } 0.75 \text{ percent of their average} \\ \text{lowest top } > \text{ highest bottom} \end{cases}$

$E_1$ to $E_5$ are five consecutive extrema which can be found using the trendln[3] library. If extrema $E_1$ to $E_5$ were not detected within 35 consecutive trading days, the patterns were not considered. If a pattern was detected within 35 consecutive trading days, the corresponding feature was set to 1 with a one-day delay. The delay is necessary because, in real-time, extrema can only be detected one day after their occurrence. Additionally, the following indicators from the ta library were used:

- Parabolic stop and reverse up/down indicator (PSAR)
- Bollinger higher/lower band indicator
- Keltner higher/lower indicator

### 4.1.3. Google Trends

Google Trends is an analytical tool that tracks the search volume of all keywords in the Google search engine. For the SA-trends dataset, a set of keywords that I believe could reflect the overall market sentiment was fetched with the pytrends[4] library. Google Trends restricts the query period for daily data from a single API call to 9 months. In addition, the fetched data is always scaled between 0 and 100. In order to obtain the search volume for a period of 10 years, I thus had to iteratively query multiple periods. These periods must partially overlap because the individual queries need to be rescaled [75].

The set of keywords that passed the Granger causality tests is listed in Appendix A.

### 4.1.4. Fundamental Data

The FA-fundamentals dataset (see Appendix A) uses the Federal Reserve Economic Data database FRED[5] that was retrieved from Quandl[6]. In addition, fundamental data from Wharton's Compustat North America - Daily database[7] was queried. When using fundamental data, we are interested in long-term dependencies and not short-term signals. Therefore, a resampling from the monthly and quarterly updated fundamental data to daily samples was performed.

## 4.2. Measurement Metrics

MSE is a good loss function for regression tasks, but it is quite meaningless as a measurement metric for evaluation purposes. A more intuitive metric is the mean absolute

[3]https://github.com/GregoryMorse/trendln
[4]https://github.com/dreyco676/pytrends
[5]https://fred.stlouisfed.org/
[6]https://github.com/quandl/quandl-python
[7]https://wrds-web.wharton.upenn.edu/wrds/

error (MAE):

$$\mathrm{MAE} = \frac{1}{m}\sum_{t=1}^{m}\left|\hat{y}^{(t)} - y^{(t)}\right|, \tag{4.1}$$

as it describes the average prediction error. Another very intuitive metric is the mean absolute percentage error (MAPE):

$$\mathrm{MAPE} = \frac{1}{m}\sum_{t=1}^{m}\left|\frac{\hat{y}^{(t)} - y^{(t)}}{y^{(t)}}\right|. \tag{4.2}$$

The MAPE is not defined for actual values $y^{(t)}$ equal to zero and very sensitive to small values. But since the values get scaled back before the metrics are evaluated, this is not a problem. Although MAPE represents a percentage error, it is not particularly suited for comparisons of stock predictions. When comparing a highly volatile stock with a very stable currency, for example, a MAPE of 1% could represent a thoroughly positive result for the stock prediction. On the other hand, it would be a very bad prediction for a currency that on average fluctuates only 0.1% per day.

A measurement metric that is scaling and volatility invariant is the mean absolute scaled error (MASE) for non-seasonal time series:

$$\mathrm{MASE} = \frac{1}{m}\sum_{t=1}^{m}\frac{\left|\hat{y}^{(t)} - y^{(t)}\right|}{\frac{1}{m-1}\sum_{j=2}^{m} \mid y^{(j)} - y^{(j-1)} \mid}. \tag{4.3}$$

MASE can be interpreted as the ratio of the MAE from the actual prediction to the MAE of a naive prediction [45].

## 4.3. Sliding Window

As input for my predictors, I used daily sampled sequential observations with sequence length $\tau$. The procedure to obtain the observations is described in Figure 4.1. For one-step ahead closing price predictions, the target value $y^{(t)}$ of the sequential observation $o(t)$ is the closing price $x_{\text{close}}^{(t+\tau)}$.

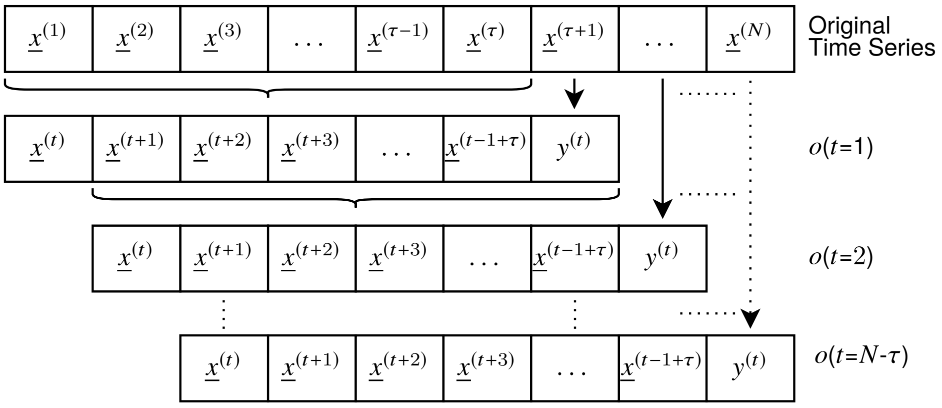


**Figure 4.1:** The sliding window procedure to extract sequential observations $o(t)$ of length $\tau$ from the original time series $\underline{x}^{(1:N)}$.

## 4.4. Cross-Validation

For small datasets, a hard split into train validation and test set is not feasible. As the number of input features increases, the predictors require more and more observations to generalize on the available data. On the other hand, we would like to have a large test set so that the generalization error can be estimated as accurately as possible. Cross-validation (CV) addresses this problem, which occurs with small datasets at the expense of computing time. $k$-fold CV splits the data into $k$ subsets of the same length. The training is performed $k$ times, leaving out one subset at a time for evaluation. The generalization error is then estimated by averaging the test errors that were computed on the left out folds [33].

### 4.4.1. Nested Cross-Validation for Hyperparameter Optimization

The intuitive way to optimize hyperparameters with CV is to fix a set of hyperparameters, perform CV, and then repeat the process with a new set of hyperparameters. In the end, the set of hyperparameters that yields the lowest averaged error on all test folds is selected. This procedure however has two limitations. Firstly, a wide set of hyperparameters will inevitably lead to overfitting on the test data, since the set of hyperparameters that minimize the error on the test folds is chosen. Secondly, even if the optimal set of hyperparameters is found, this could be caused by the data leak that occurs with this method [16]. To prevent an over-optimistic estimation of the generalization error when performing hyperparameter optimization, one should use nested CV.

In $k$-fold nested CV an additional inner loop is used to tune the hyperparameters. As with $k$-fold CV, the outer loop consists of $k$ training iterations in which one fold is always left out for evaluation. In the inner loop, the $k-1$ training folds are again split into $k_i$ subsets of the same length. The training is performed $k_i$ times, leaving out one fold for validation. The best set of hyperparameters is found by iterating through all sets of hyperparameters and selecting the set that minimizes the average error on the validation folds. Afterward, this set of hyperparameters is used to evaluate the test error on the left-out fold of the outer loop. After the $k$-th iteration in the outer loop, the generalization error is estimated by averaging all test errors. This method provides a better estimate of the generalization error but has the disadvantage that no global set of hyperparameters is found [53]. For online predictions, one would perform an additional inner loop step with all available data to find the global set of hyperparameters. The global set of hyperparameters is therefore not used to estimate the generalization error, but solely to make new predictions.

### 4.4.2. Walk-Forward Nested Cross-Validation

When performing CV, it is important that no information is leaked from the training into the test sets. Since time series are serially dependent, the test set of a time series must always follow the training set to avoid data leaks from the future to the past. Walk-forward nested CV enforces this by successively assigning one fold to the test set and only previous folds to the training set. The outer loop is executed until the entire dataset has been used for CV.

Figure 4.2 shows the walk-forward nested CV procedure that was used for my experiments. Since I used data from the last 10 years (from 11. May 2010 to 11. May 2020), the financial time series were split into 40 folds (Q1...Q40), each representing one quarter. With four iterations in the outer loop, I thus got one year (Q37...Q40) of test data for evaluation purposes. Instead of evaluating the generalization error by averaging the test errors among the four test folds, I decided to evaluate all test folds independently. This has two reasons. Firstly, the volatility of some stocks varies significantly from quarter to quarter. This may mean that a MAPE of 1%, for example, might be a very good result in one quarter and a very bad result in the next quarter. Secondly, the evaluation of a model with exactly the same hyperparameters can sometimes lead to significantly different predictions due to the random weight initializations. Financial time series are particularly affected by this phenomenon due to their partly unpredictable characteristics. Therefore, I decided to perform each outer loop iteration 10 times in order to calculate the mean and standard deviation of the measurement metrics. This resulted in a generalization error estimate for each of the four test quarters individually.

In each inner loop iteration, the respective training set was split into 10 folds. In the first inner loop iteration, 8 folds were used for training and the 9th fold for validation. Consequently, in the second inner loop iteration, 9 folds were used for training and the 10th fold for validation. The set of hyperparameters that minimized the average MSE on the two validation sets was then applied to the respective iteration of the outer loop.

To prevent data leakage from the training into the validation/test sets, the sliding window procedure (see Section 4.3) was always applied after the CV splitting into subfolds.

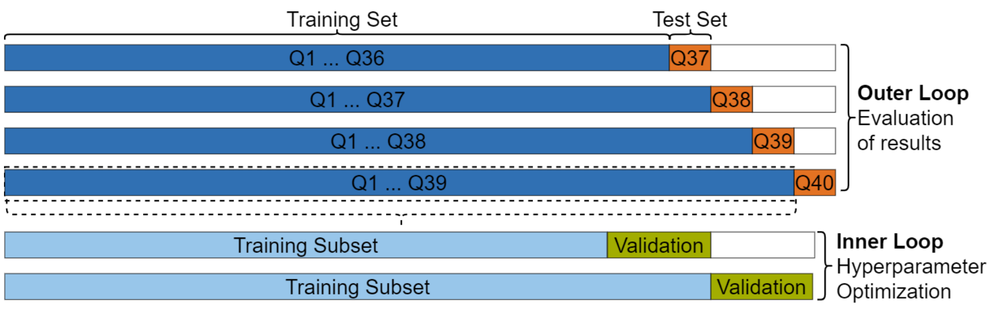


**Figure 4.2:** The walk-forward nested CV approach used for the experiments. Q1…Q40 are the 40 folds obtained by splitting 10 years of data into quarters.

## 4.5. Hyperparameter Tuning Using Bayesian Optimization

Tuning a set of hyperparameters requires considerable computational effort. Especially for models with multiple hyperparameters, tuning with grid search or random search may become impracticable as the number of possible combinations grows exponentially.

Bayesian optimization uses past experiences to determine the hyperparameter combination that should be evaluated next. An objective function (also called surrogate model) is successively approximated after a new combination of hyperparameters has been evaluated. This is commonly done by a gaussian process regression [71]. To decide which hyperparameters should be evaluated next, an acquisition function is optimized. The acquisition function uses the posterior distribution of the gaussian process regression and makes a tradeoff between exploration and exploitation [84].

For the experiments, I used the Bayesian optimization implementation of the python library scikit-learn[8]. The number of calls (i.e. the total number of different hyperparameter combinations that are evaluated) was set to 50.

## 4.6. Predictive Models and Their Hyperparameters

In this section, I give an overview of the three sequential deep learning models (LSTM, attention-based LSTM, and TCN) that were used for the predictions. Table 4.1 lists the hyperparameters and the corresponding boundaries of the parameter optimization. For all three models, I used the Keras[9] interface with Tensorflow backend. Additionally,

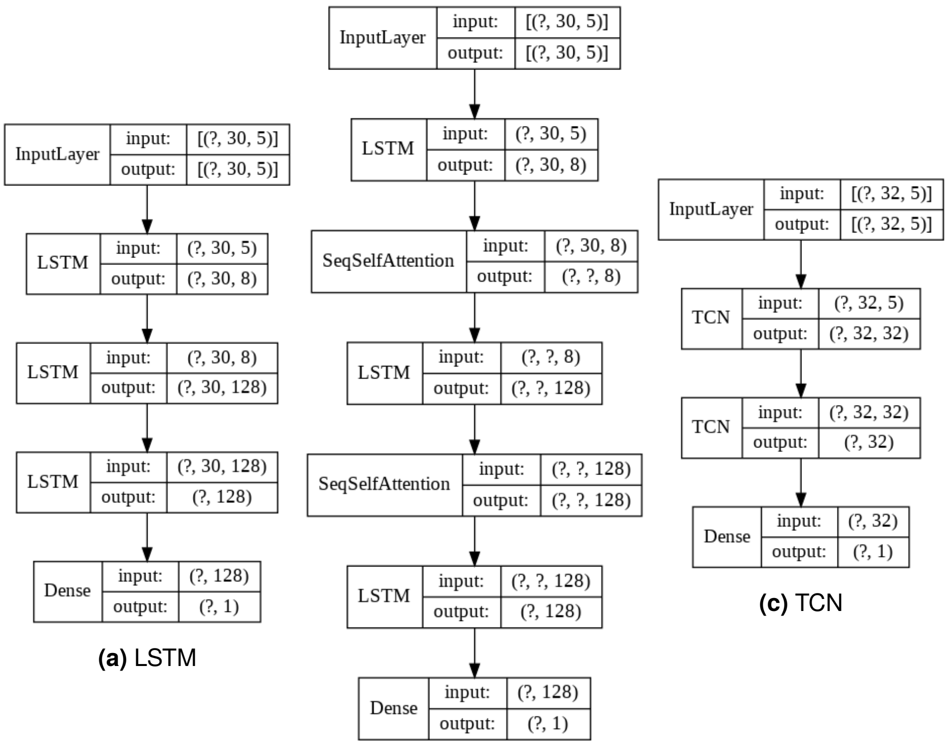
[8]https://scikit-optimize.github.io/stable/modules/generated/skopt.gp_minimize.html
[9]https://keras.io/api/

the keras-tcn[10] library was used for the TCN predictions. The attention-based LSTM (ALSTM) model was evaluated as a stand-alone model because of the interest in comparing the LSTM model with and without attention mechanisms. For the ALSTM model, I used the same hyperparameters as for the LSTM model but added a self-attention layer with sigmoid activation function between the LSTM layers. Figure 4.3 visualizes the three predictive models for an exemplary set of hyperparameters that lies within the optimization range.

| | LSTM / ALSTM | | TCN | |
|---|---|---|---|---|
| Hyperparameter | Range | Dimension | Range | Dimension |
| Observation length $\tau$ | [2,30] | Integer | [8,32] | Power of 2 |
| Mini-batch size | 256 | fixed | 256 | fixed |
| Learning rate | 1e-04 | fixed | [1e-5,1e-4] | log-uniform |
| Adam optimizer | Default | fixed | Default | fixed |
| Num. epochs | 3000 | fixed | 3000 | fixed |
| Num. hidden layers | [3,6] | Integer | | |
| Num. input units | [8,128] | Integer | | |
| Num. hidden units | [8,128] | Integer | | |
| Num. residual blocks | | | 2 | fixed |
| Num. filters | | | [8,128] | Power of 2 |
| Kernel size | | | 2 | fixed |
| Activity regularization | $L^2$(0.01) | fixed | | |
| Batch normalization | | | True/False | Boolean |
| Dropout | | | [0,0.5] | Real |

**Table 4.1.:** Overview of the hyperparameters that were used in combination with the three predictive models (LSTM, ALSTM, and TCN). The "fixed" in column "Dimension" means that the hyperparameter in column "Range" was applied without further parameter optimization. The number of hidden layers and the corresponding dilation factors of the TCN model depend on the observation length $\tau$ (see Figure 2.8). An observation length $\tau = 32$ corresponds to 8 hidden layers. With 2 residual blocks, this translates into two dilation factors of [1, 2, 4, 8] each.

[10]https://github.com/philipperemy/keras-tcn


InputLayer | input: [(?, 30, 5)] | output: [(?, 30, 5)]
LSTM | input: (?, 30, 5) | output: (?, 30, 8)
LSTM | input: (?, 30, 8) | output: (?, 30, 128)
LSTM | input: (?, 30, 128) | output: (?, 128)
Dense | input: (?, 128) | output: (?, 1)
InputLayer | input: [(?, 30, 5)] | output: [(?, 30, 5)]
LSTM | input: (?, 30, 5) | output: (?, 30, 8)
SeqSelfAttention | input: (?, 30, 8) | output: (?, ?, 8)
LSTM | input: (?, ?, 8) | output: (?, ?, 128)
SeqSelfAttention | input: (?, ?, 128) | output: (?, ?, 128)
LSTM | input: (?, ?, 128) | output: (?, 128)
Dense | input: (?, 128) | output: (?, 1)
InputLayer | input: [(?, 32, 5)] | output: [(?, 32, 5)]
TCN | input: (?, 32, 5) | output: (?, 32, 32)
TCN | input: (?, 32, 32) | output: (?, 32)
Dense | input: (?, 32) | output: (?, 1)

**(a)** LSTM

**(b)** ALSTM

**(c)** TCN

**Figure 4.3:** Exemplar LSTM model (a), ALSTM model (b) and TCN model (c). The input layers are basically tensors of shape (batch size, observation length, features). Both LSTM (a) and ALSTM (b) have 3 hidden layers with 8 units in the first LSTM layer (input units) and 128 hidden units in the consecutive LSTM layers. The TCN model (c) has 2 residual blocks with 4 hidden layers each. For the first two LSTM layers and the first TCN residual block, the argument return_sequence was set to true (returns (batch size, timesteps, units/filters)), while for the last LSTM layer and the second TCN residual block it was set to false (returns (batch size, units/filters)). The output layers are simple dense layers with a single unit and linear activation function.

## 4.7. Transfer Learning

Transfer learning (TL) is typically associated with the re-training of a pre-trained DNN. In computer vision, this is commonly done by loading a well-performing model (e.g. ResNet, InceptionV3, Xception, etc.) that has been trained on a large image database (e.g ImageNet). By fixing all weights except the weights from the last few layers, the model can be re-trained to classify new objects that were not part of the original database.

Instead of re-training an already existing model that solves a similar task, I used several indices and stocks to pre-train my own models. In this way, I tried to address the problem of having too few observations, which inevitably leads to overfitting on the training data. The stocks and indices in question were selected based on volume and market capitalization. Only stocks issued more than 10 years ago were considered (see Appendix B). I used 70% of each time series as training data and 10% for validation, leaving the last 20% unused to avoid any data leaks from the future to the past. The pre-training was performed as follows: The selected time series were initially rescaled using only the training data of the respective time series. Then the sequential observations of each time series were extracted using the sliding window procedure described in Section 4.3. Finally, the pre-training dataset was obtained by aggregating the sequential observations of all the time series. The pre-training was performed over 300 epochs and was stopped early if the validation loss had not improved over 50 consecutive epochs. After completion of the pre-training, the weights of the first LSTM layer (or the first TCN residual block for the TCN model) were fixed and the model was re-trained for another 100 epochs. The re-training used only the time series that were to be predicted (i.e. the training set in Figure 4.2) and the same hyperparameters that were used for the pre-training (except for the number of epochs). Apart from an additional pre-training step in each CV iteration, the procedure described in 4.4.2 remained exactly the same.

Preprocessing methods usually refer to the preparation of data prior to the actual predictions. In the following chapters, I will treat TL as a preprocessing method for a simpler and more understandable illustration of the results.

# 5. Results and Discussion

While the setups of the experiments were discussed in the previous chapter, in this chapter I present the results. Only the most insightful results are presented and discussed here. A complete list of all results can be found in Appendix C.

## 5.1. Overview

Before presenting the results, here is a short summary of the experiments that were conducted:

*Financial Time Series*

- I used 10 years of data (from 11. May 2010 to 11. May 2020).
- Six different datasets (see 4.1) with daily features were created for each evaluated asset.
- $\chi^2$ Granger causality tests with significance level $\alpha = .05$ were performed on the training data for feature selection. The tests were performed using the statsmodels[1] package.

*Preprocessing Methods*

- All features were rescaled with lower bound $LB = 0$ and upper bound $UB = 1$, using the MinMaxScaler[2] from scikit-learn.
- The discrete wavelet transformation was performed with the Haar wavelet and smoothing factor set to 2, using the PyWavelets[3] library.
- Autoencoder parameters were originally part of the hyperparameter optimization. To reduce the computational effort, they were fixed to a simple undercomplete autoencoder with linear activation function and a latent representation with half the original feature size.

[1]https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.grangercausalitytests.html
[2]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler
[3]https://github.com/PyWavelets/pywt

- The preprocessing steps wavelet transformation (WT), autoencoder (AE), and transfer learning (TL) were only applied to the setups marked as such. Setups that only used feature scaling as a preprocessing method are marked with (NA).

*Predictions*

- I performed one-step-ahead predictions of the closing price.
- LSTM, ALSTM, and TCN models were evaluated.
- I used a walk-forward nested cross-validation approach (see 4.4.2).
- The hyperparameters were tuned using bayesian optimization (see 4.5).
- Table 4.1 gives an overview of all hyperparameters.
- The mean value and standard deviation of the measurement metrics were calculated by performing 10 independent training runs with random weight initialization.
- MASE was used as a scaling and volatility invariant measurement metric.

## 5.2. Predicting the Dow Jones Industrial Average

First of all, I wanted to see whether the three preprocessing methods could improve my Dow Jones Industrial Average (DJI) LSTM predictions. Table 5.1 shows the evaluated measurement metrics for the OCHLV dataset. Both WT and AE disappointed with overall worse results than NA. In quarter Q40, however, the AE setups managed to achieve the smallest errors. The only preprocessing method that could lead to an overall improvement was TL. This finding suggests that even for the OCHLV dataset with only five input features there were too few observations for a decent generalization. In summary, WT and AE preprocessing could not improve the predictions, while TL successfully contributed to reduce overfitting.

Comparing the measurement metrics in the four quarters shows how essential a volatility invariant metric is. In Q40, the MAE and MAPE were significantly higher than in the other three quarters, while the MASE was lower. Looking at the DJI chart in Figure 5.1, we can recognize significantly higher volatility in Q40. The fact that the best results in terms of MASE were achieved in Q40 could not only be seen for the setups in Table 5.1 but for almost all setups (see Appendix C.1). This suggests that the forecasting accuracy is better in highly volatile periods.

Since the WT and AE setups for the DJI LSTM predictions were not completely convincing, in the following the different predictive models and datasets will be discussed exclusively for the preprocessing methods NA and TL. Figure 5.2 compares the LSTM, ALSTM and TCN models for the OCHLV dataset. The ALSTM NA setup outperformed

| Model: LSTM, Dataset: OCHLV, Ticker: DJI, Period: 10y, Sampling: Daily | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Preprocessing | | | | Measurement Metrics | | | | | |
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 206.68 | 21.87 | 0.78 | 0.08 | 1.15 | 0.12 |
| | | | 38 | 203.4 | 22.04 | 0.77 | 0.08 | 1.36 | 0.15 |
| | | | 39 | 180.78 | 32.63 | 0.63 | 0.11 | 1.26 | 0.23 |
| | | | 40 | 681.92 | 20.96 | 2.96 | 0.09 | 0.95 | 0.03 |
| | | | $\mu$ | 318.19 | | 1.29 | | 1.18 | |
| ✓ | | | 37 | 247.52 | 16.36 | 0.94 | 0.06 | 1.38 | 0.09 |
| ✓ | | | 38 | 203.12 | 18.14 | 0.76 | 0.07 | 1.36 | 0.12 |
| ✓ | | | 39 | 182.77 | 16.92 | 0.64 | 0.06 | 1.27 | 0.12 |
| ✓ | | | 40 | 783.44 | 19.02 | 3.38 | 0.08 | 1.09 | 0.03 |
| ✓ | | | $\mu$ | 354.21 | | 1.43 | | 1.27 | |
| | ✓ | | 37 | 197.7 | 11.98 | 0.76 | 0.04 | 1.24 | 0.07 |
| | ✓ | | 38 | 237.43 | 30.68 | 0.89 | 0.11 | 1.59 | 0.21 |
| | ✓ | | 39 | 156.27 | 6.04 | 0.55 | 0.02 | 1.09 | 0.04 |
| | ✓ | | 40 | 640.52 | 6.05 | 2.78 | 0.02 | 0.89 | 0.01 |
| | ✓ | | $\mu$ | 307.98 | | 1.25 | | 1.2 | |
| | | ✓ | 37 | 171.19 | 10.0 | 0.66 | 0.04 | 1.07 | 0.06 |
| | | ✓ | 38 | 190.54 | 32.32 | 0.72 | 0.12 | 1.27 | 0.22 |
| | | ✓ | 39 | 181.67 | 25.72 | 0.63 | 0.09 | 1.05 | 0.15 |
| | | ✓ | 40 | 689.19 | 11.72 | 3.0 | 0.05 | 0.96 | 0.02 |
| | | ✓ | $\mu$ | 308.15 | | 1.25 | | 1.09 | |
| ✓ | ✓ | | 37 | 221.04 | 4.23 | 2.92 | 0.01 | 1.38 | 0.03 |
| ✓ | ✓ | | 38 | 251.27 | 29.17 | 2.11 | 0.06 | 1.68 | 0.2 |
| ✓ | ✓ | | 39 | 175.33 | 18.85 | 2.02 | 0.01 | 1.22 | 0.13 |
| ✓ | ✓ | | 40 | 646.23 | 10.04 | 12.15 | 0.03 | 0.9 | 0.01 |
| ✓ | ✓ | | $\mu$ | 323.47 | | 4.8 | | 1.3 | |
| ✓ | | ✓ | 37 | 201.46 | 7.48 | 2.96 | 0.01 | 1.26 | 0.05 |
| ✓ | | ✓ | 38 | 149.62 | 13.51 | 1.7 | 0.02 | 1.05 | 0.09 |
| ✓ | | ✓ | 39 | 180.49 | 24.74 | 2.07 | 0.02 | 1.27 | 0.17 |
| ✓ | | ✓ | 40 | 719.0 | 9.42 | 12.24 | 0.07 | 1.0 | 0.01 |
| ✓ | | ✓ | $\mu$ | 312.64 | | 4.74 | | 1.14 | |
| | ✓ | ✓ | 37 | 235.52 | 33.39 | 2.91 | 0.04 | 1.47 | 0.21 |
| | ✓ | ✓ | 38 | 208.68 | 43.7 | 2.07 | 0.04 | 1.4 | 0.29 |
| | ✓ | ✓ | 39 | 158.55 | 12.48 | 2.03 | 0.01 | 1.11 | 0.09 |
| | ✓ | ✓ | 40 | 661.31 | 5.29 | 12.1 | 0.02 | 0.92 | 0.01 |
| | ✓ | ✓ | $\mu$ | 316.02 | | 4.78 | | 1.23 | |
| ✓ | ✓ | ✓ | 37 | 224.89 | 4.99 | 3.02 | 0.01 | 1.41 | 0.03 |
| ✓ | ✓ | ✓ | 38 | 217.64 | 13.65 | 2.12 | 0.03 | 1.46 | 0.09 |
| ✓ | ✓ | ✓ | 39 | 244.41 | 59.47 | 2.06 | 0.03 | 1.7 | 0.41 |
| ✓ | ✓ | ✓ | 40 | 657.72 | 10.55 | 12.24 | 0.04 | 0.92 | 0.01 |
| ✓ | ✓ | ✓ | $\mu$ | 336.16 | | 4.86 | | 1.37 | |

**Table 5.1.:** Measurement metrics for eight different preprocessing combinations.

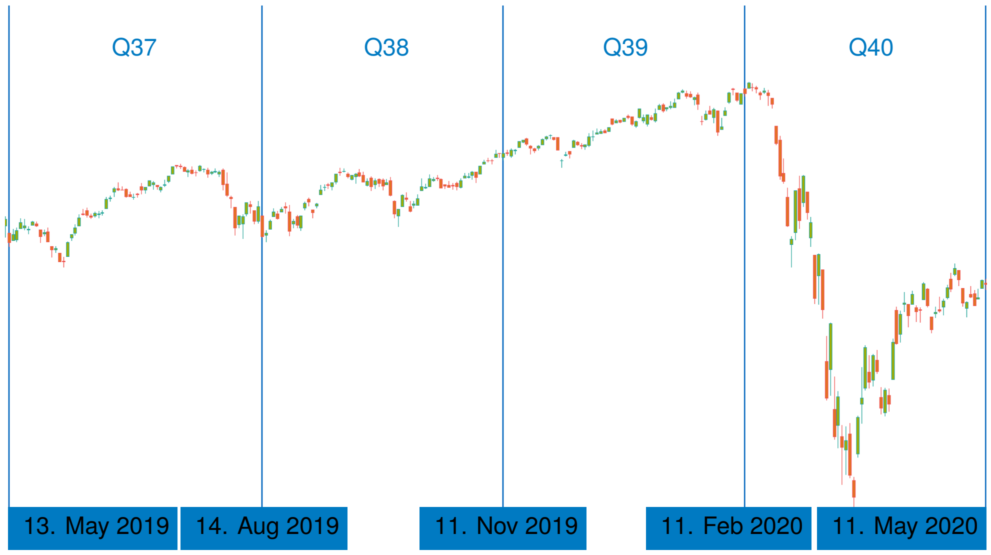


**Figure 5.1:** Dow Jones Industrial Average (DJI) chart in the quarters Q37 to Q40.

the LSTM and TCN NA setups in all quarters except Q39 where TCN performed best. A different situation was found with TL, where TCN outperformed both LSTM models in all quarters except Q40. Comparing the NA setups with the TL setups in Figure 5.2, we can see how the preprocessing method TL improved the predictive accuracy of all three models. But why had TL the strongest impact on the TCN model? Without TL, the hyperparameter optimization in the LSTM setups almost always chose the shortest observation length which was set to 2. Since the minimum observation length for the TCN model was set to 8, the TCN setups without TL were overfitting more on the training data than the LSTM setups without TL. To summarize, the ALSTM model performed best in the setup without TL, while with TL both the LSTM and TCN models performed better.

Finally, Figure 5.3 shows the errorbar plots that compare the different datasets for the LSTM and TCN models with TL. SA-trends and FA-fundamentals datasets performed very poorly in almost all quarters. The TA-causality TCN TL setup performed really well in the first three quarters, but extremely bad in Q40. This is the exact opposite of what we have seen in almost all other TA-based setups, where the best results were achieved in Q40. TA-experience performed well with the LSTM prediction but had a clear outlier in Q39 with the TCN prediction. The divergence occurred only once during the 10 evaluation iterations and would therefore not affect a long/short signal-based trading strategy in case of an ensemble decision. Overall, the OCHLV and TA-patterns datasets performed best for both LSTM and TCN predictions with TL. In summary, both fundamental data and sentimental data led to significantly worse results than simply using the OCHLV dataset. Technical indicators achieved similar results to the simple OCHLV dataset but could not improve overall predictions. For the

TCN model with TL, TA-patterns slightly improved the predictions.

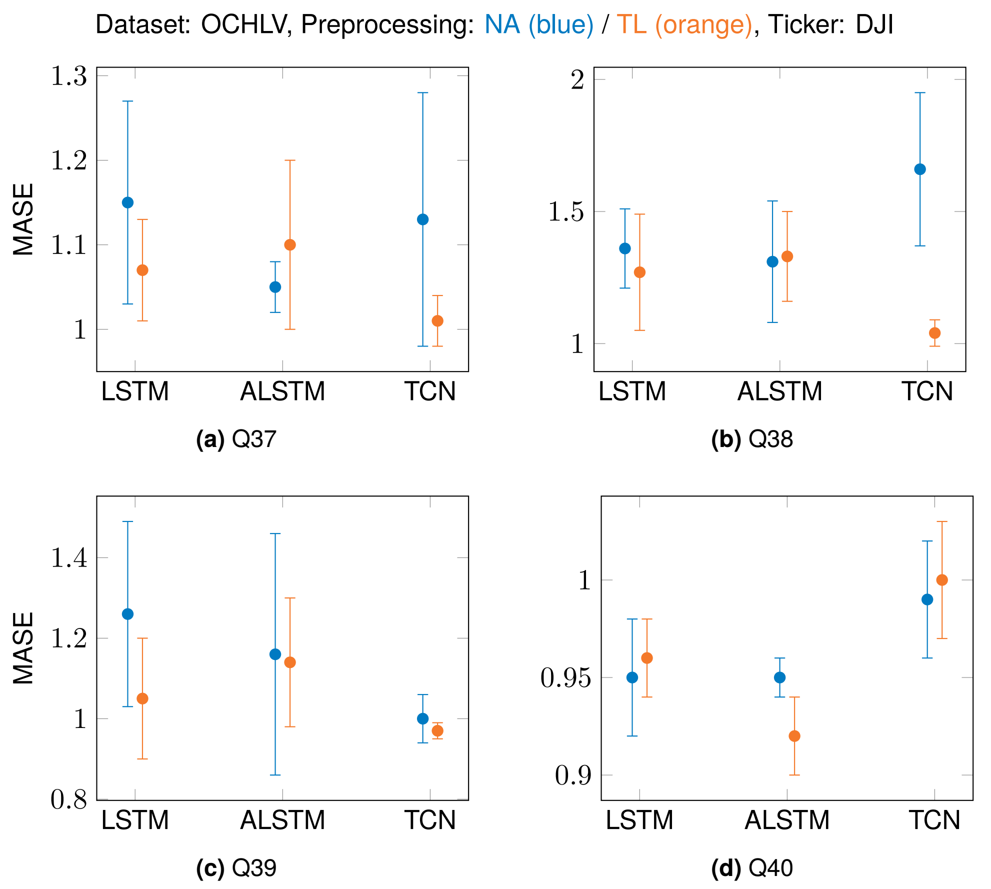
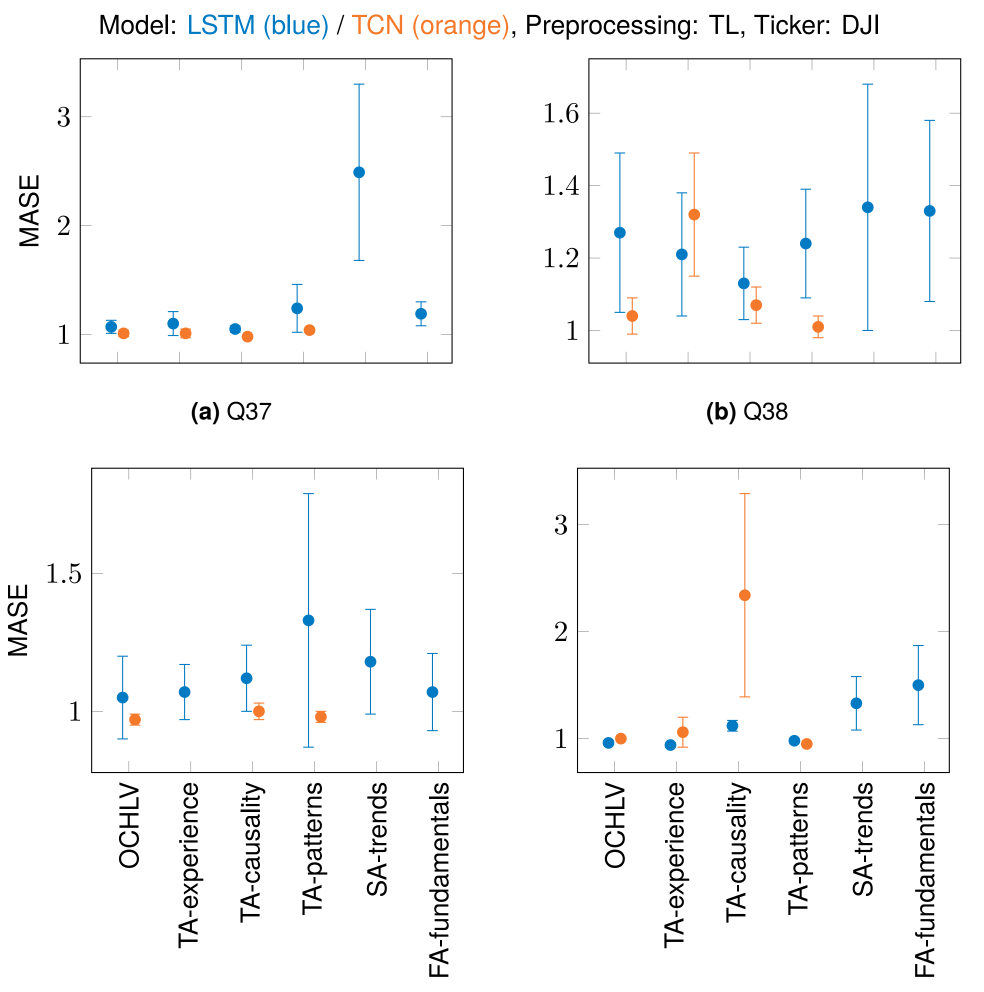


**Figure 5.2:** Errorbar plot with the MASE mean and standard deviation in the y-axis and the three predictive models LSTM, ALSTM, and TCN along the x-axis. Blue are the setups without further preprocessing (NA) and orange the setups with transfer learning (TL). In all four quarters, the ALSTM NA setup had a smaller mean error than the LSTM NA setup. The TCN NA setup performed worse than the ALSTM NA setup in all quarters except Q39. With TL, the ALSTM model performed worse than the LSTM model in the first three quarters, whereas it clearly outperformed the other setups in Q40. The TCN model with TL achieved a very stable MASE mean value around 1.0 in each quarter.



**(a)** Q37 **(b)** Q38

**(c)** Q39 **(d)** Q40

**Figure 5.3:** Errorbar plot for the LSTM model (blue) and TCN model (orange) with TL: with the MASE mean and standard deviation in the y-axis and the different datasets along the x-axis. Both SA-trends and FA-fundamentals datasets performed worse than the other datasets, which only contain technical analysis features. SA-trends and FA-fundamentals datasets performed very poorly for the TCN setup. They were omitted in the errorbar plot so as not to affect the visibility of the other datasets (see Appendix C.1 for the detailed results). In Q39 the TA-experience TCN setup had a clear outlier ($\mu$-MASE=10.87, $\sigma$-MASE=29.65). In Q37 the LSTM SA-trends setup performed very poorly and in Q40 the TA-causality TCN setup performed significantly worse than the other setups. Overall, the OCHLV and TA-patterns datasets performed best.

With these findings, I decided to evaluate several other assets based on the following setups:

- Model: LSTM, Dataset: OCHLV, Preprocessing: TL
- Model: ALSTM, Dataset: OCHLV, Preprocessing: NA
- Model: TCN, Dataset: TA-patterns, Preprocessing: TL

### 5.2.1. Ensemble Trading Strategy

To better interpret the results, I created a simple long/short signal-based trading strategy. Since 10 independent training iterations were used to evaluate the metrics, I decided to use the resulting predictions for an ensemble strategy. If 7 out of 10 one-step-ahead closing price predictions $\hat{y}^{(t+1)}$ were higher than the last known closing price $y^{(t)}$, the strategy went long. Conversely, the strategy went short if 7 out of 10 predictions were lower than the last known closing price. Figure 5.4 shows the long and short signals that were given for the TCN DJI predictions in quarters Q37 to Q40. Taking a closer look at the daily predictions, we can see how the predicted price $\hat{y}(t+1)$ closely follows the last actual known price $y(t)$. In fact, a MASE greater than $1.0$ suggests that a naive prediction $\hat{y}(t+1) = y(t)$ would result in an overall smaller predictive error in terms of MAE. So why even bother training a neural network that leads to an overall MASE of more than $1.0$? When making stock predictions for trading purposes, the most important thing is not about being right all the time or more than 50% of the time. It is about the risk/reward ratio, about getting it right on the trades that have the most impact. It is therefore crucial to achieve a small MASE in the highly volatile periods, and this is exactly what we could observe in the previous DJI predictions.

Table 5.2 compares the quarterly returns of the buy and hold strategy versus the long/short signal-based strategies for the DJI predictions. In quarters Q37 to Q39, when volatility was low, the three models were not really able to outperform the market. In Q40 on the other hand, all three models achieved impressive returns.

| | LSTM OCHLV TL | | | ALSTM OCHLV NA | | | TCN TA-Patterns TL | | |
|---|---|---|---|---|---|---|---|---|---|
| Q | $\tau$ | Buy & Hold | Pred. | $\tau$ | Buy & Hold | Pred. | $\tau$ | Buy & Hold | Pred. |
| 37 | 2 | -1.54% | -3.86% | 2 | -1.54% | -2.58% | 32 | 3.08% | 10.10% |
| 38 | 2 | 5.95% | -3.83% | 2 | 5.95% | -0.60% | 8 | 2.91% | -5.69% |
| 39 | 30 | 2.29% | -2.23% | 2 | 5.38% | 3.38% | 8 | 5.29% | 6.26% |
| 40 | 2 | -17.99% | 69.76% | 2 | -17.99% | 35.22% | 8 | -11.84% | 94.36% |

**Table 5.2.:** Quarterly returns of the DJI long/short signal-based strategies (without trading fees and slippage) versus simple buy and hold strategy. $\tau$ is the observation length that was introduced in Section 4.3. The reason for the partially different buy and hold returns between the three models is the varying observation length which is determined by the nested cross-validation.

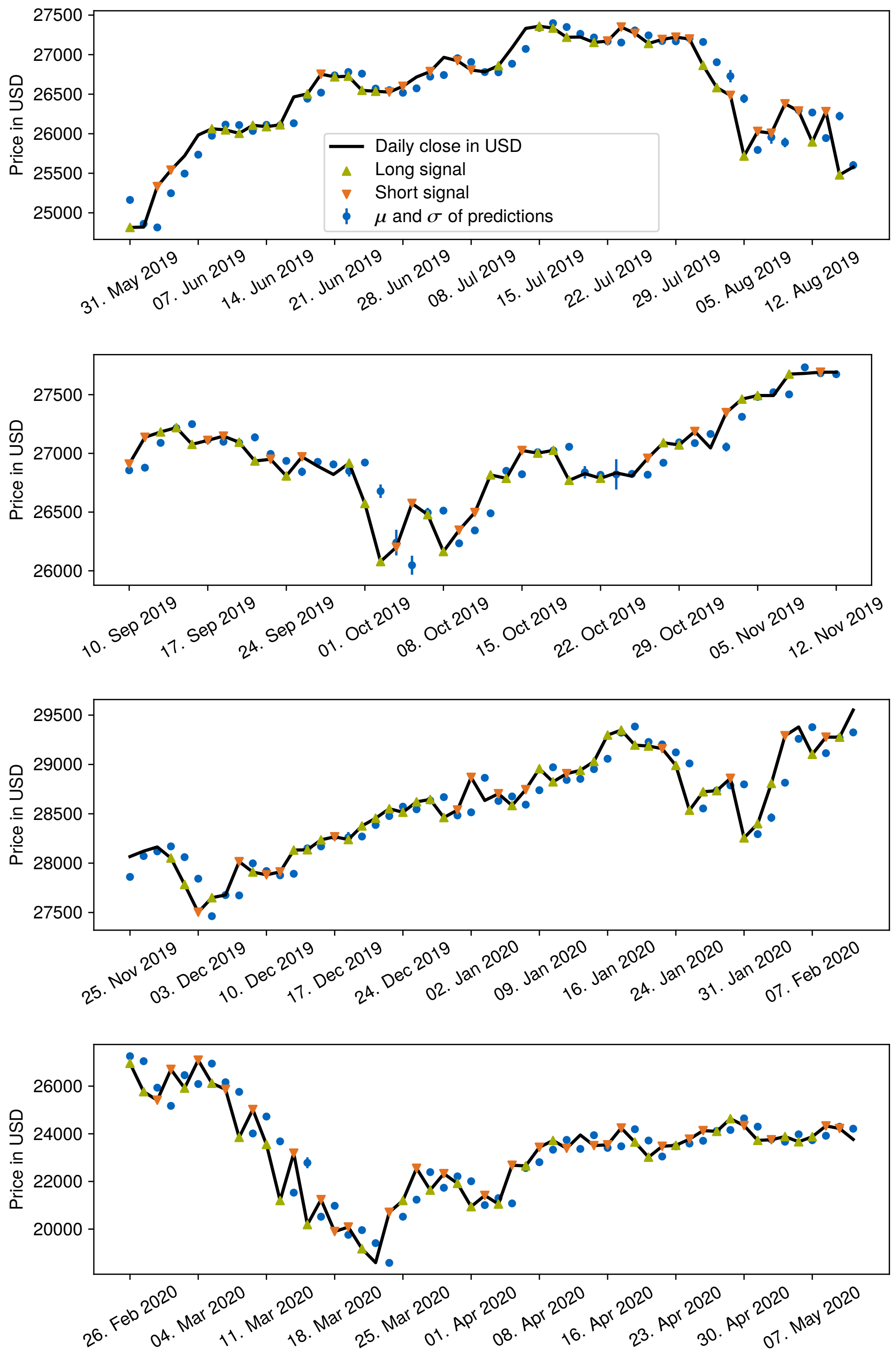


**Figure 5.4:** Long and short signals for the TCN TA-patterns TL setup predicting the DJI.

Finally, Figure 5.5 shows the cumulative return of the long/short signal-based strategies compared to the simple buy and hold strategy for the DJI. A trading fee of 3 USD was applied to each trade, slippage was not considered.

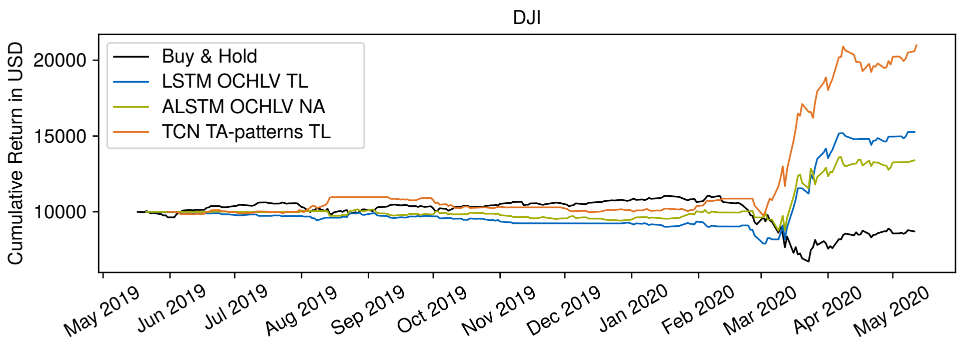


**Figure 5.5:** Cumulative return of the long/short signal-based strategies from Q37 to Q40. Note that at the beginning of each quarter no trades were made for $\tau$ days while buy and hold reflects the whole test period.

### 5.2.2. Risk Exposure

To check whether the models perform well in relation to the risk exposure, I evaluated the financial ratios that were introduced in Subsection 2.1.4.

Table 5.3 shows the $\alpha$ and $\beta$ values from the capital asset pricing model (CAPM) that were calculated using the ordinary least squares implementation from statsmodel[4]. The small $R^2$ values indicate a low correlation between the buy and hold returns and the returns of the long/short signal-based ensemble strategy. Power analysis, which determines the probability of correctly rejecting a false null hypothesis, shows that the CAPM is fundamentally unsuitable for modeling the returns of the presented trading strategies. With a given sample size of $m = 236$, a statistical power of $.95$, and a significance level of $\alpha = .05$, one would need a coefficient of determination of $R^2 = .053$ for a model of overall significance.

Figure 5.6 shows three annualized risk-adjusted return ratios in combination with the corresponding denominators in the x-axis and the total return in the y-axis. A Sharpe ratio above 1.0 indicates a positive excess return relative to risk. The Sharpe ratio can be translated into a t-statistic with the following formula [38]:

$$t_{r_i} = S_y \cdot \sqrt{\frac{m}{252}}, \tag{5.1}$$

[4]https://www.statsmodels.org/dev/examples/notebooks/generated/ols.html

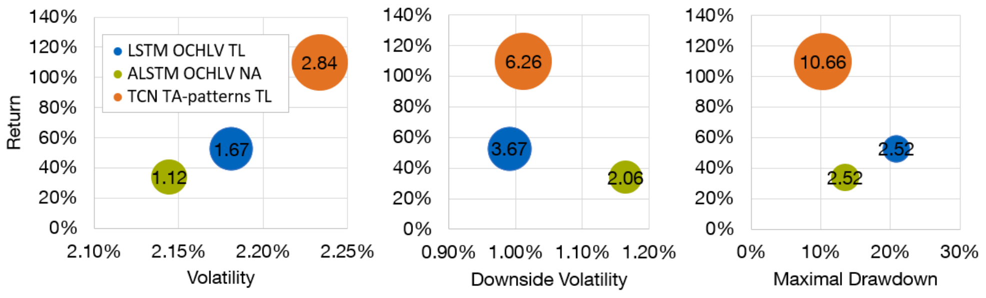
| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0024$ | $0.1860^{**}$ | 0.040 | 208 |
| | (0.001) | (0.063) | | |
| ALSTM OCHLV NA | $0.0015$ | $0.1240^{*}$ | 0.016 | 236 |
| | (0.001) | (0.063) | | |
| TCN TA-Patterns TL | $0.0040^{*}$ | $-0.0139$ | 0.000 | 204 |
| | (0.002) | (0.067) | | |

$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**Table 5.3.:** CAPM evaluation of the DJI using ordinary least squares. The standard deviations are shown in brackets.

where $S_y$ is the annualized Sharpe ratio and $m$ the sample size. If we consider the three different trading strategies as three tests against the null hypothesis of 0 profitability, we have to apply a correction in order to account for any random significance that could arise due to multiple testing. From the multiple testing corrections, Bonferroni is among the most conservative, so the false-negative rate could be very high [6]. With an annualized Sharpe ratio $S_y = 2.84$ and a sample size $m = 204$ we obtain a t-statistics $t_{r_i} = 2.56$. With a significance level of $\alpha = .05$ and the Bonferroni correction, this is sufficient to reject the null hypothesis.

The high Sortino ratios in comparison to the Sharpe ratios indicate a good predictive accuracy on the most volatile days. Since only one year was used for the evaluation, the Mar ratio is the ratio of the total return to the maximum drawdown in the test period.



**(a)** Sharpe ratio **(b)** Sortino ratio **(c)** Mar ratio

**Figure 5.6:** Risk-adjusted return ratios for the DJI trading strategies. The Sharpe and Sortino ratios have been annualized (multiplied by $\sqrt{252}$). The ratios are shown in the bubbles. The volatility in (a), downside volatility in (b), and maximal drawdown in (c) are related to the daily long/short strategy returns.

## 5.3. Predicting Several Indices and Stocks

Although I did not choose the three setups that performed best on the DJI predictions, selecting a small subset of setups and highlighting their results can definitely lead to an over-optimistic evaluation. Therefore, I considered it crucial to evaluate the three setups on different indices and on several stocks of companies operating in the largest financial sectors. The setup choice only involved the preprocessing method and dataset selection. Since the hyperparameter optimization for the different assets was carried out independently with the walk-forward nested cross-validation approach, the evaluation of the following assets is unbiased.

Table 5.4 shows the results of the evaluated assets. For a clear and compact representation, the results from the quarters Q37 to Q40 are shown in aggregated form. These results show once again how stock predictions with an overall MASE $>$ 1.0 can effectively be used for a successful trading strategy.

| | LSTM OCHLV TL | | | ALSTM OCHLV NA | | | TCN TA-Patterns TL | | |
|---|---|---|---|---|---|---|---|---|---|
| Ticker[5] | MASE | Buy & Hold | Pred. | MASE | Buy & Hold | Pred. | MASE | Buy & Hold | Pred. |
| ˆGSPC | 1.08 | -2.17% | **45.37%** | 1.10 | -2.17% | **40.10%** | 1.00 | 6.72% | **86.01%** |
| ˆGDAXI | 1.06 | -17.59% | **34.47%** | 1.04 | **-12.10%** | -22.37% | 1.05 | **-1.16%** | -2.19% |
| ˆHSI | 1.06 | **9.31%** | -5.37% | 1.03 | -10.66% | **4.84%** | 1.04 | **-6.26%** | -14.60% |
| AAPL | 1.20 | **59.74%** | -31.09% | 1.10 | **59.74%** | 11.67% | 1.10 | **82.35%** | 5.03% |
| BA | 1.01 | -64.90% | **7.70%** | 1.06 | -64.90% | **-17.98%** | 1.06 | -56.11% | **86.27%** |
| DIS | 1.05 | -25.44% | **28.39%** | 1.07 | -25.44% | **-5.32%** | 1.03 | -1.81% | **27.30%** |
| V | 1.01 | 3.96% | **104.64%** | 1.02 | 3.96% | **55.07%** | 0.99 | 26.50% | **177.22%** |
| KO | 1.10 | -12.87% | **-3.04%** | 1.07 | -8.23% | **17.50%** | 1.18 | -12.45% | **3.62%** |
| MCD | 1.01 | -11.81% | **80.34%** | 1.04 | -9.84% | **85.93%** | 1.00 | -2.54% | **60.67%** |
| RDS-B | 1.00 | -49.96% | **45.52%** | 1.02 | -49.96% | **173.28%** | 1.01 | -20.55% | **17.81%** |
| BMW | 1.05 | -23.86% | **46.79%** | 1.02 | -23.86% | **-0.32%** | 1.02 | -9.57% | **10.06%** |
| WMT | 1.04 | 12.17% | **51.76%** | 1.05 | 12.17% | **26.80%** | 1.04 | **11.30%** | -21.26% |
| JPM | 1.04 | -18.74% | **26.57%** | 1.06 | -18.74% | **8.87%** | 1.03 | 13.21% | **52.50%** |
| GOOG | 1.10 | **11.23%** | -2.50% | 1.09 | **15.38%** | 5.16% | 1.07 | **71.11%** | 4.33% |
| JNJ | 1.06 | 11.52% | **28.28%** | 1.11 | 11.52% | **41.59%** | 1.04 | 14.70% | **51.03%** |

**Table 5.4.:** Aggregated results of several indices and stocks. The returns include a trading fee of 3 USD per trade.

While for most stocks the long/short signal-based trading strategies outperformed simple buy and hold by a large margin, there are a few exceptions such as Apple (AAPL) and Alphabet (GOOG) where this was not the case. As we can see in Figure 5.7, the Apple stock had very little downside volatility in the quarters Q37 to Q39. During this period all three models performed poorly, highlighting a bad predictive accuracy in extremely bullish markets.

As for the poor performance on the DAX (ˆGDAXI) and Hang Seng Index (ˆHSI), this is probably related to them being indices outside the US markets. The full results of all evaluated indices and stocks are shown in Appendix C.2.

[5]https://finance.yahoo.com/

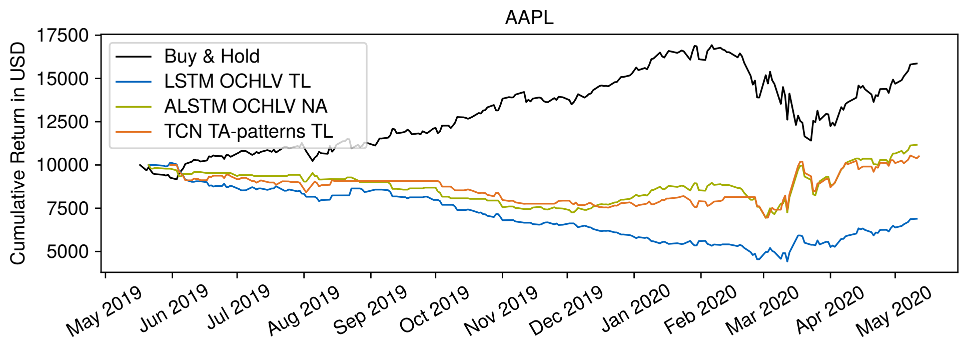


**Figure 5.7:** Cumulative return of the long/short signal-based strategies predicting the Apple stock versus simple buy and hold.

## 5.4. Predicting Bitcoin

Compared to the stock market, the cryptocurrency market is very immature and speculation driven. If the previously observed predictive accuracy in highly volatile periods holds true for all types of assets, the cryptocurrency market and its high volatility should offer ideal market conditions for the implemented models.

Since the historical Bitcoin trading data (BTC-USD) on Yahoo only dates back to late 2014, I used 5 years of historical data instead of 10 years. The data was split into 20 equal sized quarters, maintaining the same walk-forward nested CV approach as before with one year of test data (Q17 to Q20). A pre-training of the weights with several stocks and indices like was done before with TL makes no sense for Bitcoin, since the cryptocurrency market is fundamentally too different from the stock market. Therefore, I decided to evaluate the LSTM and TCN models without TL and only on the OCHLV dataset. Table 5.5 shows the corresponding quarterly results. As expected, the three models benefited strongly from the high volatility and outperformed a simple buy and hold by a large margin.

| | LSTM OCHLV NA | | | ALSTM OCHLV NA | | | TCN OCHLV NA | | |
|---|---|---|---|---|---|---|---|---|---|
| Q | $\tau$ | Buy & Hold | Pred. | $\tau$ | Buy & Hold | Pred. | $\tau$ | Buy & Hold | Pred. |
| 17 | 2 | -27.87% | 86.90% | 2 | -27.87% | 50.62% | 8 | -27.50% | -9.10% |
| 18 | 2 | -12.55% | 16.06% | 2 | -12.55% | 0.34% | 32 | -16.21% | -14.29% |
| 19 | 2 | -8.33% | 38.16% | 2 | -8.33% | 70.71% | 32 | -21.97% | 109.50% |
| 20 | 2 | 35.51% | 24.69% | 2 | 35.51% | 19.12% | 32 | 34.17% | 7.62% |

**Table 5.5.:** Quarterly returns of the long/short signal-based strategies predicting Bitcoin (without trading fees and slippage) versus simple buy and hold.

Unlike the evaluated indices and stocks, Bitcoin had relatively high volatility over the entire test period. As a result, the LSTM and ALSTM models achieved consistently good returns (see Figure 5.8). With fewer observations for training and the elimination of TL, the models had a hard time generalizing on the data. The TCN model was most affected by overfitting due to the minimum observation length in the hyperparameter optimization.

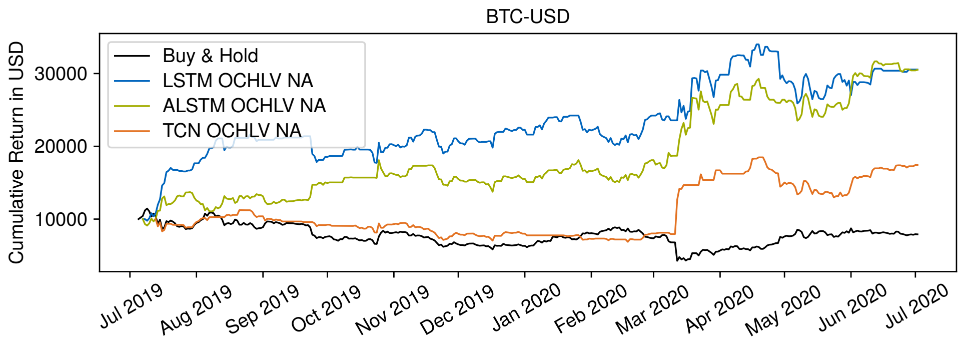


**Figure 5.8:** Cumulative return of the long/short signal-based strategies predicting Bitcoin in quarters Q17 to Q20.

Taking a look at the Sharpe ratios in 5.9a, we can see how part of the returns can be related to a high risk exposure due to Bitcoins volatility. Since the Sharpe ratio penalizes positive volatility, the Sortino ratio (see Figure 5.9b) is better suited for highly volatile assets such as Bitcoin. According to the Sortino ratio, the ALSTM strategy should be favored over the LSTM strategy because of the lower downside volatility while providing similar returns. With an almost identical annualized return of 205% and a similar maximum drawdown of approximately 24%, the LSTM and ALSTM models performed almost equally according to the Mar ratio (see 5.9c).

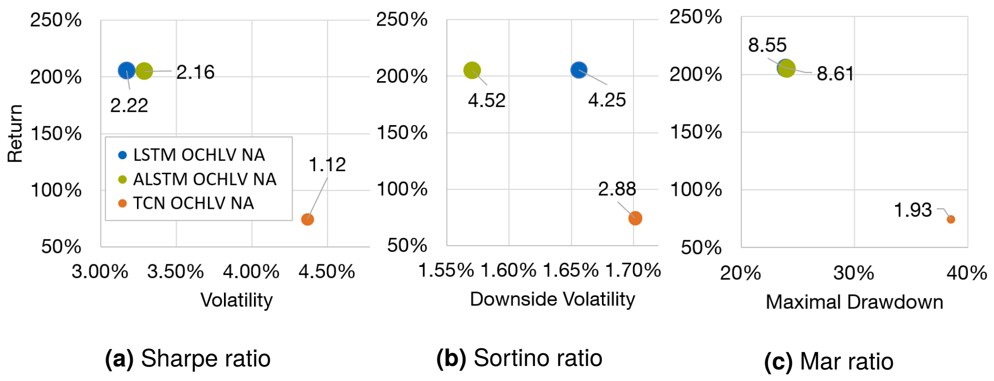


**Figure 5.9:** Risk-adjusted return ratios for the long/short trading strategies predicting Bitcoin. As BTC-USD is traded every day, the ratios were annualized by multiplying the daily Sharpe/Sortino ratios with $\sqrt{365}$.

Both the LSTM and ALSTM models had a sample size $m = 352$, which translates into the t-statistics $t_{r_{\text{LSTM}}} = 2.18$ and $t_{r_{\text{ALSTM}}} = 2.12$. With a significance level of $\alpha = .05$ the null hypothesis of 0 profitability would be rejected in both cases without the Bonferroni correction. But since we consider the trading strategies as three tests against the null hypothesis of 0 profitability, we fail to reject the null hypothesis and strive for more observations. At this point, we should keep in mind that the Bonferroni correction is quite conservative. Therefore the false-negative rate can be very high.

## 5.5. Limitations

The MASE was used as a scaling and volatility invariant measurement metric, but as we have seen, the predictive accuracy is strongly affected by the volatility of the underlying asset. Since $\tau$ observations are needed for the first prediction, the varying observation length led to slightly different test periods. This limits the comparability of the three models, even if they are used to predict the same asset. For larger test sets this difference would be negligible, but unfortunately, the publicly available historical data on stocks is very limited.

The DJI experiments have shown how TL improved the predictive accuracy by reducing the overfitting. According to Nguyen et al. [66], the predictive accuracy of a certain stock can be improved if only closely related stocks are used for pre-training. Therefore, more attention should be paid to the selection of pre-training assets.

Since overfitting could even be observed on the OCHLV dataset with only five input features (see Figure 5.2), giving up training data for more or larger test folds was not taken into consideration. For Bitcoin, the relatively small number of test samples led to a failed rejection of the corrected 0 profitability null hypothesis. Therefore, even if the LSTM and ALSTM Bitcoin trading strategies look promising, further testing is required for a high statistical significance.

The ensemble trading strategy provided long/short signals when 7 out of 10 one-step-ahead closing price predictions $\hat{y}(t+1)$ were higher/lower than the last known closing price $y(t)$. Besides choosing a more sophisticated ensemble strategy (e.g. using a simple neural network), additional predictions would be required for a robuster filtering of noisy predictions.

In quantitative trading, it is common practice to evaluate an investment with linear regression models that include various factors such as beta and momentum. However, the CAPM evaluation in 5.2.2 has shown that a simple linear regression is incapable in modeling the returns of the presented trading strategies. This makes it somewhat difficult to understand what the models learn, but at the same time, it shows how non-linear relations between input features and predicted output are identified.

# 6. Conclusion

The aim of this thesis was to investigate whether financial time series relating to publicly traded assets can be predicted with sequential deep learning models. Furthermore, it should be analyzed what kind of input features are best suited for the predictions.

To this end, six different datasets based on technical indicators, patterns, fundamental data, or sentimental data were constructed for each asset. In addition to the different datasets, the effects of the wavelet transformation for noise reduction, autoencoder for data compression, and transfer learning to prevent overfitting were explored. Three different deep learning models for sequential learning, namely long short-term memory (LSTM), self-attention-based LSTM (ALSTM), and temporal convolutional network (TCN) were used for the predictions. The models were evaluated using a walk-forward nested cross-validation approach with Bayesian hyperparameter optimization.

In a first experiment, I predicted the Dow Jones Industrial Average (DJI) index, considering several possible combinations of deep learning model, preprocessing method, and dataset. While wavelet transformation and autoencoder preprocessing methods failed to improve predictions, transfer learning successfully reduced overfitting in most setups. Regarding the different datasets, both fundamental data and sentimental data led to significantly worse results than simply using the open, close, high, low, and volume (OCHLV) as input features. Technical indicators achieved similar results to the simple OCHLV dataset but could not improve overall predictions. For the TCN predictions, adding patterns to the OCHLV dataset resulted in a small improvement.

With these results in mind, I conducted a second experiment in which I predicted several assets using one LSTM, one ALSTM, and one TCN setup. Thereby it became apparent that the chosen models are unable to accurately predict future price movements of indices, stocks, or cryptocurrencies. Since for most assets the predictions led to an overall mean absolute scaled error (MASE) greater than 1.0, even a naive prediction $\hat{y}(t+1) = y(t)$ would result in an overall lower mean absolute error (MAE). On the other hand, the models achieved very good results in highly volatile periods. This observation makes perfect sense, as the loss was calculated using the mean squared error (MSE), which gives greater significance to samples that lie further away from the mean value.

While accurate predictions were not possible, the models turned out to be particularly suited for trading purposes. Several indices and stocks, as well as Bitcoin, have been evaluated. Long/short signal-based trading strategies that used the predictions of the implemented models managed to outperform simple buy and hold on most assets. Since highly volatile trading days have the biggest impact on returns, the poor performance during periods of low volatility only slightly affected the cumulative returns. However, under extremely bullish market conditions with almost no downside volatility, buy and hold performed significantly better than the implemented strategies. Therefore, the strategies are unsuited for tech stocks. But on the other hand, due to the high volatility, the strategies performed particularly well on Bitcoin.

In a future work, it would be interesting to see whether classification with binary cross-entropy loss function could improve the predictive accuracy in periods of low volatility and how this would affect the accuracy on highly volatile days. Intraday predictions could be evaluated, as more observations should improve generalization and reduce overfitting. To assess the impact of slippage, the trading strategies should be tested on exchanges in real-time trading.

# List of Acronyms

| | |
|---|---|
| **AE** | Autoencoder |
| **AI** | Artificial Intelligence |
| **ARIMA** | Autoregressive Integrated Moving Average |
| **ARMA** | Autoregressive Moving Average |
| **CAPM** | Capital Asset Pricing Model |
| **CV** | Cross-Validation |
| **DAE** | Deep Autoencoder |
| **DJI** | Dow Jones Industrial Average |
| **CNN** | Convolutional Neural Network |
| **DNN** | Deep Neural Network |
| **FA** | Fundamental Analysis |
| **GAN** | Generative Adversarial Network |
| **GDP** | Gross Domestic Product |
| **LSTM** | Long Short-Term Memory |
| **MAE** | Mean Absolute Error |
| **MAPE** | Mean Absolute Percentage Error |
| **MASE** | Mean Absolute Scaled Error |
| **ML** | Machine Learning |
| **MSE** | Mean Squared Error |
| **NA** | No Additional Preprocessing |
| **OCHLV** | Open Close High Low Volume |

| | |
|---|---|
| **PCA** | Principal Component Analysis |
| **Q** | Quarter |
| **ReLU** | Rectified Linear Unit |
| **RNN** | Recurrent Neural Network |
| **SA** | Sentimental Analysis |
| **SGD** | Stochastic Gradient Descent |
| **SVM** | Support Vector Machine |
| **SVR** | Support Vector Regression |
| **TA** | Technical Analysis |
| **TCN** | Temporal Convolutional Network |
| **TL** | Transfer Learning |
| **WT** | Wavelet Transform |

# List of Figures

# List of Tables

# Bibliography

[1] J. Abraham, D. Higdon, J. Nelson, and J. Ibarra. “Cryptocurrency price prediction using tweet volumes and sentiment analysis”. In: *SMU Data Science Review* 1 (2018), p. 1.

[2] S. B. Achelis. *Technical Analysis from A to Z*. McGraw Hill New York, 2001.

[3] E. Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[4] R. M. Alrumaih and M. A. Al-Fawzan. “Time Series Forecasting Using Wavelet Denoising an Application to Saudi Stock Index”. In: *Journal of King Saud University-Engineering Sciences* 14 (2002), pp. 221–233.

[5] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo. “Stock price prediction using the ARIMA model”. In: *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*. IEEE. 2014, pp. 106–112.

[6] R. A. Armstrong. “When to use the B onferroni correction”. In: *Ophthalmic and Physiological Optics* 34 (2014), pp. 502–508.

[7] D. Bahdanau, K. Cho, and Y. Bengio. “Neural machine translation by jointly learning to align and translate”. In: *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

[8] S. Bai, J. Z. Kolter, and V. Koltun. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”. In: *arXiv preprint arXiv:1803.01271* (2018).

[9] M. Ballings, D. Van den Poel, N. Hespeels, and R. Gryp. “Evaluating multiple classifiers for stock price direction prediction”. In: *Expert Systems with Applications* 42 (2015), pp. 7046–7056.

[10] W. Bao, J. Yue, and Y. Rao. “A deep learning framework for financial time series using stacked autoencoders and long-short term memory”. In: *PloS one* 12 (2017), e0180944.

[11] L. Barras, O. Scaillet, and R. Wermers. “False discoveries in mutual fund performance: Measuring luck in estimated alphas”. In: *The journal of finance* 65 (2010), pp. 179–216.

[12] M. Belkin, D. Hsu, S. Ma, and S. Mandal. “Reconciling modern machine-learning practice and the classical bias–variance trade-off”. In: *Proceedings of the National Academy of Sciences* 116 (2019), pp. 15849–15854.

[13] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. “Greedy layer-wise training of deep networks”. In: *Advances in neural information processing systems*. 2007, pp. 153–160.

[14] J. Bollen, H. Mao, and X. Zeng. “Twitter mood predicts the stock market”. In: *Journal of computational science* 2 (2011), pp. 1–8.

[15] G. E. Box, G. M. Jenkins, and G. C. Reinsel. *Time series analysis: forecasting and control*. John Wiley & Sons, 2011.

[16] G. C. Cawley and N. L. Talbot. “On over-fitting in model selection and subsequent selection bias in performance evaluation”. In: *The Journal of Machine Learning Research* 11 (2010), pp. 2079–2107.

[17] K. Chen, Y. Zhou, and F. Dai. “A LSTM-based method for stock returns prediction: A case study of China stock market”. In: *2015 IEEE international conference on big data (big data)*. IEEE. 2015, pp. 2823–2824.

[18] M. A. Cody. “The fast wavelet transform: Beyond Fourier transforms”. In: *Dr. Dobb’s Journal* 17 (1992), pp. 16–28.

[19] R. W. Colby and T. A. Meyers. *The encyclopedia of technical market indicators*. Dow Jones-Irwin Homewood, IL, 1988.

[20] B. Cornell. “Medallion Fund: The Ultimate Counterexample?” In: *The Journal of Portfolio Management* 46 (2020), pp. 156–159.

[21] M. L. De Prado. *Advances in financial machine learning*. John Wiley & Sons, 2018.

[22] A. Degutis and L. Novickytė. “The efficient market hypothesis: A critical review of literature and methodology”. In: *Ekonomika* 93 (2014), pp. 7–23.

[23] X. Ding, Y. Zhang, T. Liu, and J. Duan. “Using structured events to predict stock price movement: An empirical investigation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1415–1425.

[24] C. DOERSCH. “Tutorial on Variational Autoencoders”. In: *stat* 1050 (2016), p. 13.

[25] D. L. Donoho and J. M. Johnstone. “Ideal spatial adaptation by wavelet shrinkage”. In: *biometrika* 81 (1994), pp. 425–455.

[26] M. J. Douglas. *Trading in the zone: master the market with confidence, discipline and a winning attitude*. Penguin, 2000.

[27] C. Eom, S. Choi, G. Oh, and W.-S. Jung. “Hurst exponent and prediction based on weak-form efficient market hypothesis of stock markets”. In: *Physica A: Statistical Mechanics and its Applications* 387 (2008), pp. 4630–4636.

[28] A. Essien and C. Giannetti. “A deep learning framework for univariate time series prediction using convolutional LSTM stacked autoencoders”. In: *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*. IEEE. 2019, pp. 1–6.

[29] T. Fischer and C. Krauss. “Deep learning with long short-term memory networks for financial market predictions”. In: *European Journal of Operational Research* 270 (2018), pp. 654–669.

[30] R. Fortin and S. Michelson. “Active international mutual fund management; can managers neat the index?” In: *Managerial Finance* (2005).

[31] K. Fukushima. “Neural network model for a mechanism of pattern recognition unaffected by shift in position-Neocognitron”. In: *IEICE Technical Report, A* 62 (1979), pp. 658–665.

[32] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. “Convolutional sequence to sequence learning”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 2017, pp. 1243–1252.

[33] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*. MIT press Cambridge, 2016.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[35] C. W. Granger. “Investigating causal relations by econometric models and cross-spectral methods”. In: *Econometrica: journal of the Econometric Society* (1969), pp. 424–438.

[36] M. J. Gruber. “Another puzzle: The growth in actively managed mutual funds”. In: *Investments And Portfolio Performance*. World Scientific, 2011, pp. 117–144.

[37] M. Hall. “Resolution and uncertainty in spectral decomposition”. In: *First Break* 24 (2006), pp. 43–47.

[38] C. R. Harvey and Y. Liu. “Evaluating trading strategies”. In: *The Journal of Portfolio Management* 40 (2014), pp. 108–118.

[39] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[40] R. Hecht-Nielsen. “Theory of the backpropagation neural network”. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[41] C. Hiemstra and J. D. Jones. “Testing for linear and nonlinear Granger causality in the stock price-volume relation”. In: *The Journal of Finance* 49 (1994), pp. 1639–1664.

[42] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9 (1997), pp. 1735–1780.

[43] T. Hollis, A. Viscardi, and S. E. Yi. “A comparison of LSTMs and attention mechanisms for forecasting financial time series”. In: *arXiv preprint arXiv:1812.07699* (2018).

[44] T.-J. Hsieh, H.-F. Hsiao, and W.-C. Yeh. “Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm”. In: *Applied soft computing* 11 (2011), pp. 2510–2525.

[45] R. J. Hyndman and A. B. Koehler. “Another look at measures of forecast accuracy”. In: *International journal of forecasting* 22 (2006), pp. 679–688.

[46] S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *International Conference on Machine Learning*. 2015, pp. 448–456.

[47] M. Jaderberg, K. Simonyan, A. Zisserman, et al. “Spatial transformer networks”. In: *Advances in neural information processing systems*. 2015, pp. 2017–2025.

[48] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi. “A survey of the recent architectures of deep convolutional neural networks”. In: *Artificial Intelligence Review* (2020), pp. 1–62.

[49] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[50] G. Kirchgässner and J. Wolters. *Introduction to modern time series analysis*. Springer Science & Business Media, 2007.

[51] M. Krantz. *Fundamental analysis for dummies*. John Wiley & Sons, 2016.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[53] D. Krstajic, L. J. Buturovic, D. E. Leahy, and S. Thomas. “Cross-validation pitfalls when selecting and assessing regression and classification models”. In: *Journal of cheminformatics* 6 (2014), pp. 1–15.

[54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86 (1998), pp. 2278–2324.

[55] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function”. In: *Neural networks* 6 (1993), pp. 861–867.

[56] B. Liu. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press, 2015.

[57] A. W. Lo, H. Mamaysky, and J. Wang. "Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation". In: *The journal of finance* 55 (2000), pp. 1705–1765.

[58] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis. "Dying ReLU and Initialization: Theory and Numerical Examples". In: *stat* 1050 (2019), p. 15.

[59] M. Magdon-Ismail and A. F. Atiya. "Maximum drawdown". In: *Risk Magazine* 17 (2004), pp. 99–102.

[60] B. G. Malkiel and E. F. Fama. "Efficient capital markets: A review of theory and empirical work". In: *The journal of Finance* 25 (1970), pp. 383–417.

[61] B. G. Malkiel. *A random walk down Wall Street: including a life-cycle guide to personal investing*. WW Norton & Company, 1999.

[62] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[63] T. K. Moon. "The expectation-maximization algorithm". In: *IEEE Signal processing magazine* 13 (1996), pp. 47–60.

[64] W. K. Ngui, M. S. Leong, L. M. Hee, and A. M. Abdelrhman. "Wavelet analysis: mother wavelet selection methods". In: *Applied mechanics and materials*. Trans Tech Publ. 2013, pp. 953–958.

[65] T. H. Nguyen, K. Shirai, and J. Velcin. "Sentiment analysis on social media for stock movement prediction". In: *Expert Systems with Applications* 42 (2015), pp. 9603–9611.

[66] T.-T. Nguyen and S. Yoon. "A Novel Approach to Short-Term Stock Price Movement Prediction using Transfer Learning". In: *Applied Sciences* 9 (2019), p. 4745.

[67] J. Nobre and R. F. Neves. "Combining principal component analysis, discrete wavelet transform and XGBoost to trade in the financial markets". In: *Expert Systems with Applications* 125 (2019), pp. 181–194.

[68] C. Olah and S. Carter. "Attention and Augmented Recurrent Neural Networks". In: *Distill* 1 (2016), e1.

[69] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[70] P.-F. Pai and C.-S. Lin. "A hybrid ARIMA and support vector machines model in stock price forecasting". In: *Omega* 33 (2005), pp. 497–505.

[71] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[72] A. F. Perold. “The capital asset pricing model”. In: *Journal of economic perspectives* 18 (2004), pp. 3–24.

[73] T. Preis, H. S. Moat, and H. E. Stanley. “Quantifying trading behavior in financial markets using Google Trends”. In: *Scientific reports* 3 (2013), p. 1684.

[74] J. Qiu, B. Wang, and C. Zhou. “Forecasting stock prices with long-short term memory neural network based on attention mechanism”. In: *PloS one* 15 (2020), e0227222.

[76] T. RollingeR and S. Hoffman. “Sortino ratio: A better measure of risk”. In: *Futures Magazine* 1 (2013), pp. 40–42.

[77] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65 (1958), p. 386.

[78] M. Roy, V. R. Kumar, B. D. Kulkarni, J. Sanderson, M. Rhodes, and M. vander Stappen. “Simple denoising algorithm using wavelet transform”. In: *AIChE Journal* 45 (1999), pp. 2461–2466.

[79] S. Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).

[80] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *nature* 323 (1986), pp. 533–536.

[81] G. Savin, P. Weller, and J. Zvingelis. “The predictive power of “head-and-shoulders” price patterns in the US stock market”. In: *Journal of Financial Econometrics* 5 (2007), pp. 243–265.

[82] W. F. Sharpe. “The sharpe ratio”. In: *Journal of portfolio management* 21 (1994), pp. 49–58.

[83] R. H. Shumway and D. S. Stoffer. *Time series analysis and its applications: with R examples*. Springer, 2017.

[84] J. Snoek, H. Larochelle, and R. P. Adams. “Practical bayesian optimization of machine learning algorithms”. In: *Advances in neural information processing systems*. 2012, pp. 2951–2959.

[85] D. Sussillo and L. Abbott. “Random walk initialization for training very deep feedforward networks”. In: *arXiv preprint arXiv:1412.6558* (2014).

[86] I. Sutskever, O. Vinyals, and Q. V. Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[87] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[88] L. Van Der Maaten, E. Postma, and J. Van den Herik. “Dimensionality reduction: a comparative”. In: *J Mach Learn Res* 10 (2009), p. 13.

[89] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[90] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang. “Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting”. In: *Electronics* 8 (2019), p. 876.

[91] M. Wiese, R. Knobloch, R. Korn, and P. Kretschmer. “Quant gans: Deep generation of financial time series”. In: *Quantitative Finance* (2020), pp. 1–22.

[92] B. Xu, N. Wang, T. Chen, and M. Li. “Empirical evaluation of rectified activations in convolutional network”. In: *arXiv preprint arXiv:1505.00853* (2015).

[93] R. Xu and D. Wunsch. *Clustering*. John Wiley & Sons, 2008.

[94] H. Yang, L. Chan, and I. King. “Support vector machine regression for volatile stock market prediction”. In: *International Conference on Intelligent Data Engineering and Automated Learning*. Springer. 2002, pp. 391–396.

[95] G. P. Zhang. “Time series forecasting using a hybrid ARIMA and neural network model”. In: *Neurocomputing* 50 (2003), pp. 159–175.

# A. Datasets

| Dataset | Considered Features | Final Features |
|---|---|---|
| OCHLV | **yfinance**: Open, Close, High, Low, Volume | **yfinance**: Open, Close, High, Low, Volume |
| TA-experience | **yfinance**: Open, Close, High, Low, Volume; **ta**: EMA12, EMA26, Bollinger l, Bollinger h, RSI3, RSI7, RSI14, MACD, OBV, ROC | **yfinance**: Open, Close, High, Low, Volume; **ta**: EMA12, EMA26, Bollinger l, Bollinger h, RSI3, RSI7, RSI14, MACD, OBV, ROC |
| TA-causality | **yfinance**: Open, Close, High, Low, Volume; **ta**: ta.volume: adi, obv, fi, em, vpt, nvi, cmf, sma_em, vwap; ta.volatility: atr, bbp, kcl, kcw, kcp, kcli, dcl, dch, bbm, bbh, bbl, bbw, bbhi, bbli, kcc, kch, kchi; ta.trend: macd, macd_signal, adx_pos, adx_neg, adx, vortex_ind_pos, vortex_ind_diff, cci, trix, dpo, ichimoku_b, macd_diff, kst, kst_sig, kst_diff, sma_fast, sma_slow, psar_up, psar_down, psar_up_indicator, psar_down_indicator, aroon_up, aroon_down, aroon_ind, ema_fast, ema_slow, vortex_ind_neg, mass_index, ichimoku_conv, ichimoku_base, ichimoku_a, visual_ichimoku_a, visual_ichimoku_b; ta.momomentum: mfi, rsi, tsi, uo, stoch, stoch_signal, wr, ao, roc, kama; others_dr, others_dlr | **yfinance**: Open, Close, High, Low, Volume; **ta**: ta.volume adi, obv, fi, em, vpt, nvi; ta.volatility: atr, bbp, kcl, kcw, kcp, kcli, dcl, dch; ta.trend: macd, macd_signal, adx_pos, adx_neg, vortex_ind_pos, vortex_ind_diff, cci, dpo, ichimoku_b; ta.momentum: mfi, rsi, tsi, uo, stoch, stoch_signal, wr, ao, roc; others_dr,others_dlr |

| Dataset | Considered Features | Final Features |
|---|---|---|
| TA-patterns | **yfinance**: Open, Close, High, Low, Volume; **Lo et. al** [57]: HS, IHS, BBOT, TTOP, TBOT, RTOP, RBOT; **ta**: ta.trend: psar_up_indicator, psar_down_indicator, bbhi, bbli, kchi, kcli | **yfinance**: Open, Close, High, Low, Volume; **Lo et. al** [57]: HS, IHS, BBOT, TTOP, TBOT, RTOP, RBOT; **ta**: ta.trend: psar_up_indicator, psar_down_indicator, bbhi, bbli, kchi, kcli |
| SA-trends | **yfinance**: Open, Close, High, Low, Volume; **keywords**: Stocks, Dow Jones, Bear market, Bull market, crisis, recession, buy stocks, sell stocks, rate cut, FED, unemployment rate, capital gains, stocks to buy, how to short, how to long, stock exchange, nasdaq | **yfinance**: Open, Close, High, Low, Volume; **keywords**: Stocks, Dow Jones, Bear market, Bull market, recession, buy stocks, rate cut, FED, capital gains, stocks to buy,stock exchange |
| FA-fundamentals | **yfinance**: Open, Close, High, Low, Volume; **FRED**: CPIAUCSL, DFF, DPRIME, UNRATE, PCE, PSAVERT, RRSFS, HOUST; **WRDS Compustat**: gicdesc, indret_ew, indret_vw, PEG_trailing_Mean, CAPEI_Mean, divyield_Mean, pcf_Mean, pe_inc_Mean, ptb_Mean, gpm_Mean, roa_Mean, roce_Mean, capital_ratio_Mean, totdebt_invcap_Mean, cash_debt_Mean, int_totdebt_Mean, sale_invcap_Mean, sale_nwc_Mean, rd_sale_Mean, adv_sale_Mean, staff_sale_Mean | **yfinance**: Open, Close, High, Low, Volume; **FRED**: DFF, DPRIME, UNRATE, PSAVERT, PCE, RRSFS, HOUST; **WRDS Compustat**: CAPEI_Mean, pe_inc_Mean |

# B. Transfer Learning Tickers

All stocks and indices were fetched through the yfinance API. Only stocks that were publicly listed before 11. May 2010 and are still actively trading were used for pre-training. The following is the list of tickers from Yahoo finance that was used for the transfer learning pre-training:

- N225
- HSI
- FTSE
- DJI
- GDAXI
- FCHI
- GSPC
- AMZN
- GOOG
- NFLX
- TSLA
- BRK-B
- FCAU
- ADDYY
- IFNNY
- AMD
- VOW
- SAP
- BMW
- RDS-B
- TM
- CVX
- NTDOY
- WMT
- RBGLY
- BAYRY
- PEP
- MCD
- SIE
- BAS
- SNE
- MA
- DIS
- BCS
- BA
- V
- DAI
- KO
- ALV
- JPM
- MSFT
- VOD
- AAPL
- NVDA
- LHA
- INTC
- ENEL

# C. Additional Results

## C.1. DJI Predictions

Model: LSTM, Dataset: TA-causality, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 243.42 | 48.18 | 0.92 | 0.18 | 1.32 | 0.26 |
| | | | 38 | 203.86 | 80.23 | 0.76 | 0.3 | 1.51 | 0.59 |
| | | | 39 | 247.14 | 39.88 | 0.86 | 0.14 | 1.69 | 0.27 |
| | | | 40 | 777.63 | 45.68 | 3.44 | 0.22 | 1.07 | 0.06 |
| | | | $\mu$ | 368.01 | | 1.5 | | 1.4 | |
| | | ✓ | 37 | 168.0 | 5.06 | 0.64 | 0.02 | 1.05 | 0.03 |
| | | ✓ | 38 | 164.23 | 14.56 | 0.62 | 0.05 | 1.13 | 0.1 |
| | | ✓ | 39 | 163.72 | 16.93 | 0.57 | 0.06 | 1.12 | 0.12 |
| | | ✓ | 40 | 814.76 | 35.18 | 3.66 | 0.17 | 1.12 | 0.05 |
| | | ✓ | $\mu$ | 327.68 | | 1.37 | | 1.1 | |
| | ✓ | | 37 | 179.34 | 2.09 | 0.69 | 0.01 | 1.12 | 0.01 |
| | ✓ | | 38 | 189.98 | 24.02 | 0.71 | 0.09 | 1.32 | 0.17 |
| | ✓ | | 39 | 201.5 | 23.79 | 0.7 | 0.08 | 1.38 | 0.16 |
| | ✓ | | 40 | 911.86 | 194.92 | 4.03 | 0.95 | 1.25 | 0.27 |
| | ✓ | | $\mu$ | 370.67 | | 1.53 | | 1.27 | |
| | ✓ | ✓ | 37 | 165.29 | 3.57 | 2.99 | 0.01 | 1.03 | 0.02 |
| | ✓ | ✓ | 38 | 201.06 | 46.2 | 2.02 | 0.05 | 1.38 | 0.32 |
| | ✓ | ✓ | 39 | 198.74 | 26.75 | 2.1 | 0.03 | 1.36 | 0.18 |
| | ✓ | ✓ | 40 | 1010.98 | 50.09 | 10.51 | 0.08 | 1.39 | 0.07 |
| | ✓ | ✓ | $\mu$ | 394.02 | | 4.41 | | 1.29 | |
| ✓ | | | 37 | 206.35 | 10.03 | 0.79 | 0.04 | 1.29 | 0.06 |
| ✓ | | | 38 | 193.32 | 12.52 | 0.73 | 0.05 | 1.34 | 0.09 |
| ✓ | | | 39 | 176.14 | 14.76 | 0.62 | 0.05 | 1.2 | 0.1 |
| ✓ | | | 40 | 828.0 | 62.37 | 3.7 | 0.31 | 1.14 | 0.09 |
| ✓ | | | $\mu$ | 350.95 | | 1.46 | | 1.24 | |
| ✓ | ✓ | | 37 | 207.12 | 19.36 | 0.79 | 0.07 | 1.29 | 0.12 |
| ✓ | ✓ | | 38 | 191.99 | 20.08 | 0.72 | 0.07 | 1.33 | 0.14 |
| ✓ | ✓ | | 39 | 178.29 | 10.84 | 0.62 | 0.04 | 1.22 | 0.07 |
| ✓ | ✓ | | 40 | 1290.48 | 249.32 | 5.92 | 1.18 | 1.75 | 0.34 |
| ✓ | ✓ | | $\mu$ | 466.97 | | 2.01 | | 1.4 | |

**Table C.1.:** Measurement metrics for the DJI LSTM TA-causality setups.

Model: LSTM, Dataset: TA-experience, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 182.33 | 12.86 | 0.7 | 0.05 | 1.14 | 0.08 |
| | | | 38 | 181.16 | 5.64 | 0.68 | 0.02 | 1.24 | 0.04 |
| | | | 39 | 183.89 | 19.04 | 0.64 | 0.07 | 1.26 | 0.13 |
| | | | 40 | 725.61 | 27.28 | 3.23 | 0.13 | 1.0 | 0.04 |
| | | | $\mu$ | 318.25 | | 1.31 | | 1.16 | |
| | | ✓ | 37 | 177.0 | 16.96 | 0.68 | 0.06 | 1.1 | 0.11 |
| | | ✓ | 38 | 173.87 | 24.26 | 0.65 | 0.09 | 1.21 | 0.17 |
| | | ✓ | 39 | 156.99 | 14.44 | 0.55 | 0.05 | 1.07 | 0.1 |
| | | ✓ | 40 | 678.03 | 14.83 | 2.98 | 0.07 | 0.94 | 0.02 |
| | | ✓ | $\mu$ | 296.47 | | 1.21 | | 1.08 | |
| | ✓ | | 37 | 178.63 | 22.35 | 0.68 | 0.08 | 1.11 | 0.14 |
| | ✓ | | 38 | 195.08 | 40.6 | 0.73 | 0.15 | 1.35 | 0.28 |
| | ✓ | | 39 | 198.12 | 27.57 | 0.69 | 0.1 | 1.35 | 0.19 |
| | ✓ | | 40 | 750.12 | 41.06 | 3.36 | 0.19 | 1.04 | 0.06 |
| | ✓ | | $\mu$ | 330.49 | | 1.37 | | 1.22 | |
| ✓ | | | 37 | 204.81 | 5.64 | 0.78 | 0.02 | 1.28 | 0.04 |
| ✓ | | | 38 | 192.87 | 22.73 | 0.72 | 0.08 | 1.34 | 0.16 |
| ✓ | | | 39 | 176.06 | 21.05 | 0.62 | 0.07 | 1.2 | 0.14 |
| ✓ | | | 40 | 742.49 | 33.56 | 3.28 | 0.16 | 1.03 | 0.05 |
| ✓ | | | $\mu$ | 329.06 | | 1.35 | | 1.21 | |
| ✓ | ✓ | | 37 | 216.57 | 6.03 | 0.83 | 0.02 | 1.35 | 0.04 |
| ✓ | ✓ | | 38 | 193.07 | 13.67 | 0.72 | 0.05 | 1.34 | 0.09 |
| ✓ | ✓ | | 39 | 193.87 | 35.54 | 0.68 | 0.12 | 1.33 | 0.24 |
| ✓ | ✓ | | 40 | 768.64 | 14.51 | 3.44 | 0.07 | 1.06 | 0.02 |
| ✓ | ✓ | | $\mu$ | 343.04 | | 1.42 | | 1.27 | |

**Table C.2.:** Measurement metrics for the LSTM TA-experience setups.

Model: LSTM, Dataset: TA-patterns, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 190.85 | 7.69 | 0.73 | 0.03 | 1.19 | 0.05 |
| | | | 38 | 216.83 | 33.01 | 0.81 | 0.12 | 1.48 | 0.23 |
| | | | 39 | 217.44 | 26.44 | 0.76 | 0.09 | 1.5 | 0.18 |
| | | | 40 | 642.89 | 7.92 | 2.79 | 0.04 | 0.89 | 0.01 |
| | | | $\mu$ | 317.0 | | 1.27 | | 1.27 | |
| | | ✓ | 37 | 198.15 | 34.66 | 0.76 | 0.13 | 1.24 | 0.22 |
| | | ✓ | 38 | 181.08 | 22.46 | 0.68 | 0.08 | 1.24 | 0.15 |
| | | ✓ | 39 | 191.92 | 66.97 | 0.67 | 0.23 | 1.33 | 0.46 |
| | | ✓ | 40 | 707.44 | 16.01 | 3.08 | 0.08 | 0.98 | 0.02 |
| | | ✓ | $\mu$ | 319.65 | | 1.3 | | 1.2 | |
| | ✓ | | 37 | 225.49 | 17.48 | 0.86 | 0.06 | 1.41 | 0.11 |
| | ✓ | | 38 | 274.41 | 43.12 | 1.03 | 0.16 | 1.87 | 0.29 |
| | ✓ | | 39 | 172.1 | 12.36 | 0.6 | 0.04 | 1.19 | 0.09 |
| | ✓ | | 40 | 656.13 | 9.95 | 2.86 | 0.04 | 0.91 | 0.01 |
| | ✓ | | $\mu$ | 332.03 | | 1.34 | | 1.35 | |
| | ✓ | ✓ | 37 | 179.73 | 13.78 | 3.01 | 0.02 | 1.12 | 0.09 |
| | ✓ | ✓ | 38 | 160.9 | 3.79 | 2.06 | 0.01 | 1.1 | 0.03 |
| | ✓ | ✓ | 39 | 226.49 | 44.65 | 1.32 | 0.09 | 1.25 | 0.25 |
| | ✓ | ✓ | 40 | 679.27 | 4.92 | 11.32 | 0.03 | 0.93 | 0.01 |
| | ✓ | ✓ | $\mu$ | 311.6 | | 4.43 | | 1.1 | |

**Table C.3.:** Measurement metrics for the DJI LSTM TA-patterns setups.

Model: LSTM, Dataset: SA-trends, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 345.38 | 76.76 | 1.31 | 0.29 | 2.16 | 0.48 |
| | | | 38 | 212.44 | 57.88 | 0.8 | 0.22 | 1.42 | 0.39 |
| | | | 39 | 185.46 | 25.03 | 0.65 | 0.09 | 1.29 | 0.17 |
| | | | 40 | 1452.92 | 277.95 | 6.5 | 1.29 | 2.02 | 0.39 |
| | | | $\mu$ | 549.05 | | 2.31 | | 1.72 | |
| | | ✓ | 37 | 398.66 | 129.26 | 1.51 | 0.49 | 2.49 | 0.81 |
| | | ✓ | 38 | 200.01 | 51.47 | 0.75 | 0.19 | 1.34 | 0.34 |
| | | ✓ | 39 | 169.69 | 27.66 | 0.59 | 0.1 | 1.18 | 0.19 |
| | | ✓ | 40 | 955.12 | 179.66 | 4.29 | 0.85 | 1.33 | 0.25 |
| | | ✓ | $\mu$ | 430.87 | | 1.79 | | 1.58 | |
| | ✓ | | 37 | 381.99 | 119.26 | 1.45 | 0.45 | 2.39 | 0.75 |
| | ✓ | | 38 | 205.75 | 29.34 | 0.77 | 0.11 | 1.38 | 0.2 |
| | ✓ | | 39 | 204.18 | 38.23 | 0.72 | 0.13 | 1.42 | 0.27 |
| | ✓ | | 40 | 1109.51 | 276.05 | 4.92 | 1.3 | 1.54 | 0.38 |
| | ✓ | | $\mu$ | 475.36 | | 1.96 | | 1.68 | |
| | ✓ | ✓ | 37 | 464.57 | 252.76 | 3.96 | 0.87 | 2.9 | 1.58 |
| | ✓ | ✓ | 38 | 234.15 | 41.13 | 2.2 | 0.12 | 1.57 | 0.28 |
| | ✓ | ✓ | 39 | 166.16 | 13.51 | 2.04 | 0.02 | 1.16 | 0.09 |
| | ✓ | ✓ | 40 | 1634.01 | 172.96 | 12.71 | 0.33 | 2.27 | 0.24 |
| | ✓ | ✓ | $\mu$ | 624.72 | | 5.23 | | 1.97 | |

**Table C.4.:** Measurement metrics for the DJI LSTM SA-trends setups.

Model: LSTM, Dataset: FA-fundamentals, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 179.94 | 10.54 | 2.9 | 0.06 | 1.12 | 0.07 |
| | | | 38 | 368.11 | 112.66 | 2.18 | 0.18 | 2.46 | 0.75 |
| | | | 39 | 241.84 | 52.1 | 2.03 | 0.05 | 1.68 | 0.36 |
| | | | 40 | 1059.31 | 213.73 | 12.67 | 0.62 | 1.47 | 0.3 |
| | | | $\mu$ | 462.3 | | 4.95 | | 1.69 | |
| | | ✓ | 37 | 190.68 | 18.16 | 0.73 | 0.07 | 1.19 | 0.11 |
| | | ✓ | 38 | 199.6 | 36.71 | 0.75 | 0.14 | 1.33 | 0.25 |
| | | ✓ | 39 | 154.41 | 19.97 | 0.54 | 0.07 | 1.07 | 0.14 |
| | | ✓ | 40 | 1075.79 | 267.67 | 4.63 | 1.12 | 1.5 | 0.37 |
| | | ✓ | $\mu$ | 405.12 | | 1.66 | | 1.27 | |
| | ✓ | | 37 | 338.43 | 47.86 | 1.29 | 0.18 | 2.12 | 0.3 |
| | ✓ | | 38 | 679.23 | 124.5 | 2.53 | 0.46 | 4.54 | 0.83 |
| | ✓ | | 39 | 326.07 | 83.05 | 1.14 | 0.29 | 2.27 | 0.58 |
| | ✓ | | 40 | 1578.75 | 597.52 | 6.78 | 2.52 | 2.2 | 0.83 |
| | ✓ | | $\mu$ | 730.62 | | 2.93 | | 2.78 | |

**Table C.5.:** Measurement metrics for the DJI LSTM FA-fundamentals setups.

Model: ALSTM, Dataset: OCHLV, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 167.7 | 4.53 | 0.64 | 0.02 | 1.05 | 0.03 |
| | | | 38 | 195.9 | 33.86 | 0.74 | 0.13 | 1.31 | 0.23 |
| | | | 39 | 166.71 | 43.31 | 0.58 | 0.15 | 1.16 | 0.3 |
| | | | 40 | 679.22 | 7.6 | 2.95 | 0.03 | 0.95 | 0.01 |
| | | | $\mu$ | 302.38 | | 1.23 | | 1.12 | |
| | | ✓ | 37 | 176.35 | 16.19 | 0.67 | 0.06 | 1.1 | 0.1 |
| | | ✓ | 38 | 198.94 | 26.07 | 0.75 | 0.1 | 1.33 | 0.17 |
| | | ✓ | 39 | 163.34 | 22.62 | 0.57 | 0.08 | 1.14 | 0.16 |
| | | ✓ | 40 | 661.73 | 13.44 | 2.89 | 0.06 | 0.92 | 0.02 |
| | | ✓ | $\mu$ | 300.09 | | 1.22 | | 1.12 | |
| | ✓ | | 37 | 194.65 | 7.43 | 0.74 | 0.03 | 1.22 | 0.05 |
| | ✓ | | 38 | 158.16 | 12.78 | 0.59 | 0.05 | 1.06 | 0.09 |
| | ✓ | | 39 | 169.84 | 17.63 | 0.6 | 0.06 | 1.18 | 0.12 |
| | ✓ | | 40 | 645.69 | 6.92 | 2.81 | 0.03 | 0.9 | 0.01 |
| | ✓ | | $\mu$ | 292.08 | | 1.19 | | 1.09 | |
| | ✓ | ✓ | 37 | 184.32 | 15.22 | 2.98 | 0.01 | 1.15 | 0.1 |
| | ✓ | ✓ | 38 | 173.41 | 13.38 | 2.02 | 0.03 | 1.16 | 0.09 |
| | ✓ | ✓ | 39 | 158.18 | 14.03 | 2.07 | 0.02 | 1.1 | 0.1 |
| | ✓ | ✓ | 40 | 659.84 | 9.87 | 12.09 | 0.04 | 0.92 | 0.01 |
| | ✓ | ✓ | $\mu$ | 293.94 | | 4.79 | | 1.08 | |
| ✓ | | | 37 | 225.43 | 25.87 | 0.86 | 0.1 | 1.41 | 0.16 |
| ✓ | | | 38 | 211.84 | 28.5 | 0.8 | 0.11 | 1.42 | 0.19 |
| ✓ | | | 39 | 187.36 | 27.01 | 0.66 | 0.09 | 1.32 | 0.19 |
| ✓ | | | 40 | 770.53 | 8.82 | 3.33 | 0.04 | 1.07 | 0.01 |
| ✓ | | | $\mu$ | 348.79 | | 1.41 | | 1.3 | |

**Table C.6.:** Measurement metrics for the DJI ALSTM OCHLV setups.

Model: ALSTM, Dataset: TA-patterns, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 176.46 | 5.14 | 0.68 | 0.02 | 1.1 | 0.03 |
| | | | 38 | 201.66 | 25.98 | 0.76 | 0.1 | 1.38 | 0.18 |
| | | | 39 | 186.86 | 24.76 | 0.65 | 0.09 | 1.29 | 0.17 |
| | | | 40 | 663.34 | 7.72 | 2.88 | 0.03 | 0.92 | 0.01 |
| | | | $\mu$ | 307.08 | | 1.24 | | 1.17 | |
| | | ✓ | 37 | 178.3 | 8.36 | 0.68 | 0.03 | 1.11 | 0.05 |
| | | ✓ | 38 | 173.99 | 13.37 | 0.65 | 0.05 | 1.19 | 0.09 |
| | | ✓ | 39 | 216.88 | 117.13 | 0.76 | 0.41 | 1.5 | 0.81 |
| | | ✓ | 40 | 679.08 | 16.53 | 2.96 | 0.07 | 0.94 | 0.02 |
| | | ✓ | $\mu$ | 312.06 | | 1.26 | | 1.19 | |
| | ✓ | | 37 | 237.67 | 31.7 | 0.9 | 0.12 | 1.49 | 0.2 |
| | ✓ | | 38 | 284.5 | 38.31 | 1.07 | 0.14 | 1.94 | 0.26 |
| | ✓ | | 39 | 176.76 | 16.44 | 0.62 | 0.06 | 1.22 | 0.11 |
| | ✓ | | 40 | 655.03 | 15.56 | 2.86 | 0.07 | 0.91 | 0.02 |
| | ✓ | | $\mu$ | 338.49 | | 1.36 | | 1.39 | |

**Table C.7.:** Measurement metrics for the DJI ALSTM TA-patterns setups.

Model: ALSTM, Dataset: TA-experience, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 170.46 | 33.89 | 0.65 | 0.13 | 1.06 | 0.21 |
| | | | 38 | 177.86 | 10.21 | 0.67 | 0.04 | 1.23 | 0.07 |
| | | | 39 | 203.02 | 42.42 | 0.71 | 0.15 | 1.39 | 0.29 |
| | | | 40 | 714.88 | 19.73 | 3.17 | 0.09 | 0.99 | 0.03 |
| | | | $\mu$ | 316.56 | | 1.3 | | 1.17 | |
| | | ✓ | 37 | 174.23 | 10.32 | 0.67 | 0.04 | 1.09 | 0.06 |
| | | ✓ | 38 | 170.87 | 26.51 | 0.64 | 0.1 | 1.17 | 0.18 |
| | | ✓ | 39 | 161.87 | 14.58 | 0.57 | 0.05 | 1.11 | 0.1 |
| | | ✓ | 40 | 679.67 | 36.09 | 2.97 | 0.17 | 0.94 | 0.05 |
| | | ✓ | $\mu$ | 296.66 | | 1.21 | | 1.08 | |
| | ✓ | | 37 | 169.62 | 4.28 | 0.65 | 0.02 | 1.06 | 0.03 |
| | ✓ | | 38 | 189.78 | 23.94 | 0.71 | 0.09 | 1.32 | 0.17 |
| | ✓ | | 39 | 180.65 | 14.08 | 0.63 | 0.05 | 1.24 | 0.1 |
| | ✓ | | 40 | 801.47 | 47.91 | 3.59 | 0.24 | 1.11 | 0.07 |
| | ✓ | | $\mu$ | 335.38 | | 1.4 | | 1.18 | |
| ✓ | | | 37 | 203.47 | 16.15 | 0.78 | 0.06 | 1.27 | 0.1 |
| ✓ | | | 38 | 182.78 | 6.63 | 0.69 | 0.02 | 1.27 | 0.05 |
| ✓ | | | 39 | 179.05 | 17.69 | 0.63 | 0.06 | 1.22 | 0.12 |
| ✓ | | | 40 | 728.2 | 12.0 | 3.2 | 0.06 | 1.01 | 0.02 |
| ✓ | | | $\mu$ | 323.37 | | 1.32 | | 1.19 | |
| ✓ | | ✓ | 37 | 199.84 | 4.82 | 0.76 | 0.02 | 1.25 | 0.03 |
| ✓ | | ✓ | 38 | 188.2 | 34.52 | 0.71 | 0.13 | 1.31 | 0.24 |
| ✓ | | ✓ | 39 | 176.81 | 23.7 | 0.62 | 0.08 | 1.19 | 0.16 |
| ✓ | | ✓ | 40 | 697.07 | 33.11 | 3.04 | 0.14 | 0.96 | 0.05 |
| ✓ | | ✓ | $\mu$ | 315.48 | | 1.28 | | 1.18 | |

**Table C.8.:** Measurement metrics for the DJI ALSTM TA-experience setups.

Model: ALSTM, Dataset: SA-trends, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 303.68 | 69.71 | 1.15 | 0.26 | 1.9 | 0.44 |
| | | | 38 | 191.2 | 23.59 | 0.72 | 0.09 | 1.28 | 0.16 |
| | | | 39 | 207.75 | 27.46 | 0.73 | 0.1 | 1.45 | 0.19 |
| | | | 40 | 2416.22 | 644.34 | 11.03 | 3.03 | 3.32 | 0.88 |
| | | | $\mu$ | 779.72 | | 3.41 | | 1.99 | |
| | | ✓ | 37 | 463.54 | 143.59 | 1.76 | 0.54 | 2.9 | 0.9 |
| | | ✓ | 38 | 227.92 | 30.57 | 0.86 | 0.11 | 1.52 | 0.2 |
| | | ✓ | 39 | 170.13 | 23.88 | 0.6 | 0.08 | 1.18 | 0.17 |
| | | ✓ | 40 | 1387.63 | 609.13 | 6.29 | 2.9 | 1.93 | 0.85 |
| | | ✓ | $\mu$ | 562.31 | | 2.37 | | 1.88 | |
| | ✓ | | 37 | 693.14 | 317.23 | 2.62 | 1.2 | 4.33 | 1.98 |
| | ✓ | | 38 | 211.97 | 29.28 | 0.8 | 0.11 | 1.42 | 0.2 |
| | ✓ | | 39 | 178.83 | 36.57 | 0.63 | 0.13 | 1.24 | 0.25 |
| | ✓ | | 40 | 1217.98 | 211.9 | 5.43 | 0.99 | 1.69 | 0.29 |
| | ✓ | | $\mu$ | 575.48 | | 2.37 | | 2.17 | |

**Table C.9.:** Measurement metrics for the DJI ALSTM SA-trends setups.

Model: ALSTM, Dataset: TA-causality, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 194.46 | 18.77 | 0.75 | 0.07 | 1.21 | 0.12 |
| | | | 38 | 181.96 | 35.05 | 0.68 | 0.13 | 1.26 | 0.24 |
| | | | 39 | 249.07 | 42.02 | 0.87 | 0.15 | 1.7 | 0.29 |
| | | | 40 | 784.33 | 44.65 | 3.46 | 0.21 | 1.08 | 0.06 |
| | | | $\mu$ | 352.46 | | 1.44 | | 1.31 | |
| | | ✓ | 37 | 167.07 | 4.71 | 0.64 | 0.02 | 1.04 | 0.03 |
| | | ✓ | 38 | 168.02 | 12.15 | 0.63 | 0.04 | 1.17 | 0.08 |
| | | ✓ | 39 | 199.65 | 43.35 | 0.7 | 0.15 | 1.35 | 0.29 |
| | | ✓ | 40 | 759.41 | 48.72 | 3.34 | 0.24 | 1.04 | 0.07 |
| | | ✓ | $\mu$ | 323.54 | | 1.33 | | 1.15 | |
| | ✓ | | 37 | 186.75 | 11.49 | 0.72 | 0.04 | 1.16 | 0.07 |
| | ✓ | | 38 | 180.4 | 40.45 | 0.68 | 0.15 | 1.24 | 0.28 |
| | ✓ | | 39 | 239.64 | 41.1 | 0.84 | 0.14 | 1.64 | 0.28 |
| | ✓ | | 40 | 829.21 | 78.33 | 3.7 | 0.38 | 1.14 | 0.11 |
| | ✓ | | $\mu$ | 359.0 | | 1.48 | | 1.3 | |
| ✓ | | | 37 | 200.19 | 9.83 | 0.77 | 0.04 | 1.25 | 0.06 |
| ✓ | | | 38 | 194.25 | 16.95 | 0.73 | 0.06 | 1.35 | 0.12 |
| ✓ | | | 39 | 172.5 | 5.83 | 0.6 | 0.02 | 1.18 | 0.04 |
| ✓ | | | 40 | 928.83 | 91.03 | 4.13 | 0.41 | 1.26 | 0.12 |
| ✓ | | | $\mu$ | 373.94 | | 1.56 | | 1.26 | |
| ✓ | ✓ | | 37 | 231.65 | 27.24 | 0.88 | 0.1 | 1.45 | 0.17 |
| ✓ | ✓ | | 38 | 223.06 | 25.77 | 0.84 | 0.1 | 1.55 | 0.18 |
| ✓ | ✓ | | 39 | 183.69 | 18.15 | 0.64 | 0.06 | 1.26 | 0.12 |
| ✓ | ✓ | | 40 | 846.28 | 64.62 | 3.75 | 0.31 | 1.16 | 0.09 |
| ✓ | ✓ | | $\mu$ | 371.17 | | 1.53 | | 1.35 | |

**Table C.10.:** Measurement metrics for the DJI ALSTM TA-causality setups.

Model: ALSTM, Dataset: FA-fundamentals, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 201.06 | 20.32 | 0.77 | 0.08 | 1.26 | 0.13 |
| | | | 38 | 264.81 | 56.92 | 0.99 | 0.21 | 1.77 | 0.38 |
| | | | 39 | 261.75 | 73.93 | 0.91 | 0.26 | 1.82 | 0.51 |
| | | | 40 | 843.75 | 111.57 | 3.68 | 0.47 | 1.17 | 0.16 |
| | | | $\mu$ | 392.84 | | 1.59 | | 1.51 | |
| | | ✓ | 37 | 178.52 | 15.65 | 0.68 | 0.06 | 1.12 | 0.1 |
| | | ✓ | 38 | 192.77 | 45.58 | 0.72 | 0.17 | 1.29 | 0.3 |
| | | ✓ | 39 | 182.05 | 29.71 | 0.64 | 0.1 | 1.27 | 0.21 |
| | | ✓ | 40 | 905.95 | 308.12 | 3.93 | 1.28 | 1.24 | 0.42 |
| | | ✓ | $\mu$ | 364.82 | | 1.49 | | 1.23 | |
| | ✓ | | 37 | 426.45 | 103.22 | 1.62 | 0.39 | 2.67 | 0.65 |
| | ✓ | | 38 | 360.55 | 56.77 | 1.35 | 0.21 | 2.41 | 0.38 |
| | ✓ | | 39 | 326.93 | 54.16 | 1.14 | 0.19 | 2.28 | 0.38 |
| | ✓ | | 40 | 1152.75 | 207.88 | 5.01 | 0.87 | 1.6 | 0.29 |
| | ✓ | | $\mu$ | 566.67 | | 2.28 | | 2.24 | |

**Table C.11.:** Measurement metrics for the DJI ALSTM FA-fundamentals setups.

| Model: TCN, Dataset: OCHLV, Ticker: DJI, Period: 10y, Sampling: Daily | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Preprocessing | | | | Measurement Metrics | | | | | |
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 207.66 | 26.81 | 0.78 | 0.1 | 1.13 | 0.15 |
| | | | 38 | 229.73 | 40.66 | 0.86 | 0.15 | 1.66 | 0.29 |
| | | | 39 | 149.93 | 8.58 | 0.52 | 0.03 | 1.0 | 0.06 |
| | | | 40 | 413.45 | 12.08 | 1.79 | 0.05 | 0.99 | 0.03 |
| | | | $\mu$ | 250.19 | | 0.99 | | 1.2 | |
| | | ✓ | 37 | 161.77 | 5.15 | 0.62 | 0.02 | 1.01 | 0.03 |
| | | ✓ | 38 | 143.79 | 6.8 | 0.54 | 0.03 | 1.04 | 0.05 |
| | | ✓ | 39 | 143.41 | 2.39 | 0.5 | 0.01 | 0.97 | 0.02 |
| | | ✓ | 40 | 750.57 | 10.79 | 3.3 | 0.05 | 1.0 | 0.01 |
| | | ✓ | $\mu$ | 299.88 | | 1.24 | | 1.0 | |
| | ✓ | | 37 | 177.2 | 4.15 | 0.68 | 0.02 | 1.11 | 0.03 |
| | ✓ | | 38 | 164.47 | 3.29 | 0.62 | 0.01 | 1.19 | 0.02 |
| | ✓ | | 39 | 155.36 | 16.08 | 0.54 | 0.06 | 1.05 | 0.11 |
| | ✓ | | 40 | 690.67 | 7.19 | 3.03 | 0.03 | 0.92 | 0.01 |
| | ✓ | | $\mu$ | 296.93 | | 1.22 | | 1.07 | |
| | ✓ | ✓ | 37 | 176.17 | 4.98 | 2.82 | 0.02 | 1.1 | 0.03 |
| | ✓ | ✓ | 38 | 208.85 | 60.78 | 1.92 | 0.18 | 1.51 | 0.44 |
| | ✓ | ✓ | 39 | 244.6 | 135.9 | 1.33 | 0.33 | 1.37 | 0.76 |
| | ✓ | ✓ | 40 | 686.74 | 2.23 | 9.87 | 0.01 | 0.91 | 0.0 |
| | ✓ | ✓ | $\mu$ | 329.09 | | 3.99 | | 1.22 | |
| ✓ | | | 37 | 189.34 | 9.02 | 0.72 | 0.03 | 1.25 | 0.06 |
| ✓ | | | 38 | 209.95 | 15.73 | 0.78 | 0.06 | 1.65 | 0.12 |
| ✓ | | | 39 | 167.64 | 6.13 | 0.59 | 0.02 | 1.13 | 0.04 |
| ✓ | | | 40 | 727.77 | 27.47 | 3.2 | 0.13 | 0.97 | 0.04 |
| ✓ | | | $\mu$ | 323.67 | | 1.32 | | 1.25 | |

**Table C.12.:** Measurement metrics for the DJI TCN OCHLV setups.

| Model: TCN, Dataset: SA-trends, Ticker: DJI, Period: 10y, Sampling: Daily | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Preprocessing | | | | Measurement Metrics | | | | | |
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 998.95 | 634.62 | 3.79 | 2.39 | 6.26 | 3.98 |
| | | | 38 | 230.2 | 32.17 | 0.87 | 0.12 | 1.66 | 0.23 |
| | | | 39 | 180.34 | 13.6 | 0.62 | 0.05 | 1.01 | 0.08 |
| | | | 40 | 848.05 | 371.25 | 3.68 | 1.61 | 2.04 | 0.89 |
| | | | $\mu$ | 564.39 | | 2.24 | | 2.74 | |
| | | ✓ | 37 | 1117.49 | 410.17 | 4.25 | 1.56 | 7.01 | 2.57 |
| | | ✓ | 38 | 169.75 | 35.01 | 0.64 | 0.13 | 1.22 | 0.25 |
| | | ✓ | 39 | 1708.02 | 3127.59 | 5.95 | 10.9 | 11.44 | 20.94 |
| | | ✓ | 40 | 2929.38 | 1373.17 | 13.48 | 6.29 | 3.9 | 1.83 |
| | | ✓ | $\mu$ | 1481.16 | | 6.08 | | 5.89 | |
| | ✓ | | 37 | 1025.78 | 336.93 | 3.88 | 1.27 | 6.43 | 2.11 |
| | ✓ | | 38 | 199.56 | 37.78 | 0.75 | 0.14 | 1.44 | 0.27 |
| | ✓ | | 39 | 157.68 | 12.52 | 0.55 | 0.04 | 1.06 | 0.08 |
| | ✓ | | 40 | 3644.59 | 1531.65 | 16.79 | 7.09 | 4.87 | 2.05 |
| | ✓ | | $\mu$ | 1256.9 | | 5.49 | | 3.45 | |
| | ✓ | ✓ | 37 | 2091.53 | 1513.37 | 8.87 | 5.44 | 13.77 | 9.96 |
| | ✓ | ✓ | 38 | 187.57 | 30.49 | 1.89 | 0.13 | 1.35 | 0.22 |
| | ✓ | ✓ | 39 | 645.08 | 597.1 | 2.85 | 1.69 | 4.32 | 4.0 |
| | ✓ | ✓ | 40 | 1120.41 | 373.02 | 10.31 | 0.65 | 1.49 | 0.5 |
| | ✓ | ✓ | $\mu$ | 1011.15 | | 5.98 | | 5.23 | |

**Table C.13.:** Measurement metrics for the DJI TCN SA-trends setups.

Model: TCN, Dataset: TA-experience, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 162.86 | 12.94 | 0.62 | 0.05 | 1.03 | 0.08 |
| | | | 38 | 181.21 | 48.12 | 0.68 | 0.18 | 1.4 | 0.37 |
| | | | 39 | 170.08 | 13.83 | 0.59 | 0.05 | 1.13 | 0.09 |
| | | | 40 | 1187.83 | 426.62 | 5.46 | 2.08 | 1.61 | 0.58 |
| | | | $\mu$ | 425.49 | | 1.84 | | 1.29 | |
| | | ✓ | 37 | 162.45 | 6.51 | 0.62 | 0.02 | 1.01 | 0.04 |
| | | ✓ | 38 | 171.28 | 22.25 | 0.64 | 0.08 | 1.32 | 0.17 |
| | | ✓ | 39 | 1939.1 | 5286.41 | 6.69 | 18.25 | 10.87 | 29.65 |
| | | ✓ | 40 | 780.85 | 105.47 | 3.49 | 0.5 | 1.06 | 0.14 |
| | | ✓ | $\mu$ | 763.42 | | 2.86 | | 3.57 | |
| | ✓ | | 37 | 164.38 | 12.74 | 0.63 | 0.05 | 1.02 | 0.08 |
| | ✓ | | 38 | 131.61 | 9.19 | 0.49 | 0.03 | 1.1 | 0.08 |
| | ✓ | | 39 | 202.91 | 16.68 | 0.7 | 0.06 | 1.14 | 0.09 |
| | ✓ | | 40 | 1315.48 | 368.25 | 6.07 | 1.72 | 1.79 | 0.5 |
| | ✓ | | $\mu$ | 453.59 | | 1.97 | | 1.26 | |
| | ✓ | ✓ | 37 | 161.62 | 4.92 | 2.63 | 0.03 | 1.01 | 0.03 |
| | ✓ | ✓ | 38 | 172.53 | 21.63 | 1.8 | 0.05 | 1.33 | 0.17 |
| | ✓ | ✓ | 39 | 207.48 | 17.47 | 1.27 | 0.02 | 1.16 | 0.1 |
| | ✓ | ✓ | 40 | 923.84 | 137.57 | 8.92 | 0.56 | 1.25 | 0.19 |
| | ✓ | ✓ | $\mu$ | 366.37 | | 3.65 | | 1.19 | |
| ✓ | | | 37 | 201.37 | 12.0 | 0.77 | 0.05 | 1.26 | 0.07 |
| ✓ | | | 38 | 159.69 | 8.15 | 0.6 | 0.03 | 1.3 | 0.07 |
| ✓ | | | 39 | 185.89 | 17.06 | 0.64 | 0.06 | 1.04 | 0.1 |
| ✓ | | | 40 | 1052.66 | 224.73 | 4.9 | 1.09 | 1.53 | 0.33 |
| ✓ | | | $\mu$ | 399.9 | | 1.73 | | 1.28 | |
| ✓ | | ✓ | 37 | 199.97 | 4.82 | 0.76 | 0.02 | 1.25 | 0.03 |
| ✓ | | ✓ | 38 | 145.63 | 6.56 | 0.54 | 0.02 | 1.19 | 0.05 |
| ✓ | | ✓ | 39 | 176.23 | 16.59 | 0.61 | 0.06 | 0.99 | 0.09 |
| ✓ | | ✓ | 40 | 865.15 | 109.42 | 3.88 | 0.53 | 1.17 | 0.15 |
| ✓ | | ✓ | $\mu$ | 346.74 | | 1.45 | | 1.15 | |
| ✓ | ✓ | | 37 | 212.33 | 12.16 | 0.81 | 0.05 | 1.32 | 0.08 |
| ✓ | ✓ | | 38 | 193.83 | 37.24 | 0.73 | 0.14 | 1.5 | 0.29 |
| ✓ | ✓ | | 39 | 157.54 | 10.47 | 0.55 | 0.04 | 1.04 | 0.07 |
| ✓ | ✓ | | 40 | 2264.67 | 1384.97 | 10.66 | 6.78 | 3.07 | 1.88 |
| ✓ | ✓ | | $\mu$ | 707.1 | | 3.19 | | 1.73 | |

**Table C.14.:** Measurement metrics for the DJI TCN TA-experience setups.

| Model: TCN, Dataset: TA-causality, Ticker: DJI, Period: 10y, Sampling: Daily | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Preprocessing | | | | Measurement Metrics | | | | | |
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 216.3 | 31.06 | 0.82 | 0.12 | 1.13 | 0.16 |
| | | | 38 | 153.88 | 18.22 | 0.58 | 0.07 | 1.19 | 0.14 |
| | | | 39 | 182.7 | 12.93 | 0.64 | 0.05 | 1.21 | 0.09 |
| | | | 40 | 1873.34 | 812.28 | 8.71 | 3.89 | 2.69 | 1.17 |
| | | | $\mu$ | 606.56 | | 2.69 | | 1.55 | |
| | | ✓ | 37 | 156.89 | 2.36 | 0.6 | 0.01 | 0.98 | 0.01 |
| | | ✓ | 38 | 138.23 | 6.78 | 0.52 | 0.02 | 1.07 | 0.05 |
| | | ✓ | 39 | 179.11 | 4.92 | 0.62 | 0.02 | 1.0 | 0.03 |
| | | ✓ | 40 | 1740.36 | 706.3 | 8.03 | 3.4 | 2.34 | 0.95 |
| | | ✓ | $\mu$ | 553.65 | | 2.44 | | 1.35 | |
| | ✓ | ✓ | 37 | 147.94 | 7.08 | 2.65 | 0.03 | 0.92 | 0.04 |
| | ✓ | ✓ | 38 | 122.96 | 7.1 | 1.88 | 0.03 | 1.03 | 0.06 |
| | ✓ | ✓ | 39 | 339.06 | 240.15 | 1.6 | 0.63 | 1.9 | 1.35 |
| | ✓ | ✓ | 40 | 1051.43 | 394.49 | 9.95 | 2.11 | 1.42 | 0.53 |
| | ✓ | ✓ | $\mu$ | 415.35 | | 4.02 | | 1.32 | |
| ✓ | | | 37 | 182.96 | 25.07 | 0.7 | 0.1 | 1.14 | 0.16 |
| ✓ | | | 38 | 183.06 | 29.77 | 0.68 | 0.11 | 1.42 | 0.23 |
| ✓ | | | 39 | 217.41 | 30.14 | 0.75 | 0.1 | 1.22 | 0.17 |
| ✓ | | | 40 | 2319.71 | 1113.68 | 10.72 | 5.31 | 3.12 | 1.5 |
| ✓ | | | $\mu$ | 725.79 | | 3.21 | | 1.73 | |
| ✓ | ✓ | | 37 | 257.31 | 149.04 | 0.97 | 0.56 | 1.63 | 0.94 |
| ✓ | ✓ | | 38 | 142.88 | 16.0 | 0.53 | 0.06 | 1.19 | 0.13 |
| ✓ | ✓ | | 39 | 188.67 | 18.52 | 0.65 | 0.06 | 1.06 | 0.1 |
| ✓ | ✓ | | 40 | 3154.78 | 1149.06 | 14.64 | 5.54 | 4.25 | 1.55 |
| ✓ | ✓ | | $\mu$ | 935.91 | | 4.2 | | 2.03 | |

**Table C.15.:** Measurement metrics for the DJI TCN TA-causality setups.

Model: TCN, Dataset: TA-patterns, Ticker: DJI, Period: 10y, Sampling: Daily

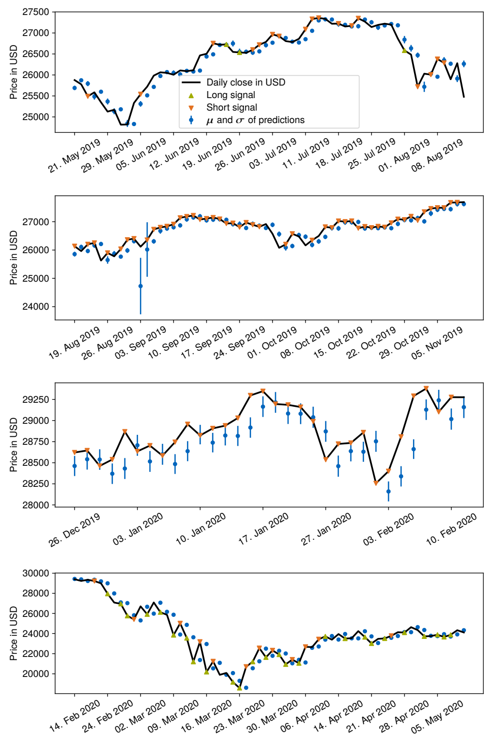
| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 201.82 | 13.39 | 0.76 | 0.05 | 1.09 | 0.07 |
| | | | 38 | 139.63 | 13.1 | 0.52 | 0.05 | 1.11 | 0.1 |
| | | | 39 | 228.65 | 28.97 | 0.8 | 0.1 | 1.53 | 0.19 |
| | | | 40 | 668.34 | 8.57 | 3.02 | 0.04 | 0.94 | 0.01 |
| | | | $\mu$ | 309.61 | | 1.28 | | 1.17 | |
| | | ✓ | 37 | 161.22 | 3.4 | 0.62 | 0.01 | 1.04 | 0.02 |
| | | ✓ | 38 | 126.49 | 3.49 | 0.47 | 0.01 | 1.01 | 0.03 |
| | | ✓ | 39 | 147.33 | 3.22 | 0.52 | 0.01 | 0.98 | 0.02 |
| | | ✓ | 40 | 715.67 | 10.41 | 3.15 | 0.05 | 0.95 | 0.01 |
| | | ✓ | $\mu$ | 287.68 | | 1.19 | | 1.0 | |
| | ✓ | | 37 | 232.83 | 13.67 | 0.88 | 0.05 | 1.51 | 0.09 |
| | ✓ | | 38 | 283.54 | 12.25 | 1.06 | 0.05 | 2.12 | 0.09 |
| | ✓ | | 39 | 190.5 | 22.5 | 0.67 | 0.08 | 1.27 | 0.15 |
| | ✓ | | 40 | 678.59 | 4.49 | 3.0 | 0.02 | 0.9 | 0.01 |
| | ✓ | | $\mu$ | 346.36 | | 1.4 | | 1.45 | |
| | ✓ | ✓ | 37 | 166.73 | 9.62 | 2.75 | 0.04 | 1.08 | 0.06 |
| | ✓ | ✓ | 38 | 227.9 | 56.47 | 1.88 | 0.15 | 1.7 | 0.42 |
| | ✓ | ✓ | 39 | 303.83 | 156.2 | 1.53 | 0.41 | 1.64 | 0.84 |
| | ✓ | ✓ | 40 | 681.89 | 10.65 | 9.46 | 0.04 | 0.9 | 0.01 |
| | ✓ | ✓ | $\mu$ | 345.09 | | 3.9 | | 1.33 | |
| ✓ | | | 37 | 218.38 | 8.15 | 2.82 | 0.03 | 1.41 | 0.05 |
| ✓ | | | 38 | 186.95 | 19.75 | 1.88 | 0.05 | 1.55 | 0.16 |
| ✓ | | | 39 | 166.22 | 4.53 | 2.02 | 0.01 | 1.11 | 0.03 |
| ✓ | | | 40 | 668.79 | 5.94 | 9.39 | 0.03 | 0.88 | 0.01 |
| ✓ | | | $\mu$ | 310.09 | | 4.03 | | 1.24 | |

**Table C.16.:** Measurement metrics for the DJI TCN TA-patterns setups.

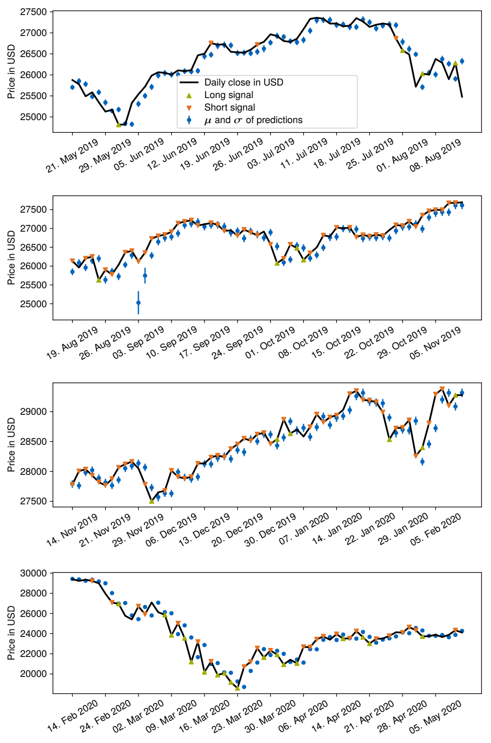
Model: TCN, Dataset: FA-fundamentals, Ticker: DJI, Period: 10y, Sampling: Daily

| Preprocessing | | | | Measurement Metrics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| WT | AE | TL | Q | $\mu$-MAE | $\sigma$-MAE | $\mu$-MAPE | $\sigma$-MAPE | $\mu$-MASE | $\sigma$-MASE |
| | | | 37 | 340.05 | 291.02 | 1.27 | 1.08 | 2.24 | 1.92 |
| | | | 38 | 482.92 | 368.05 | 1.8 | 1.36 | 3.48 | 2.66 |
| | | | 39 | 280.76 | 177.18 | 0.98 | 0.62 | 1.88 | 1.19 |
| | | | 40 | 1348.75 | 358.49 | 5.92 | 1.51 | 1.8 | 0.48 |
| | | | $\mu$ | 613.12 | | 2.49 | | 2.35 | |
| | | ✓ | 37 | 181.07 | 44.44 | 0.68 | 0.16 | 1.19 | 0.29 |
| | | ✓ | 38 | 204.32 | 103.28 | 0.76 | 0.38 | 1.47 | 0.75 |
| | | ✓ | 39 | 528.97 | 392.75 | 1.83 | 1.36 | 2.96 | 2.2 |
| | | ✓ | 40 | 1506.11 | 707.55 | 6.46 | 2.97 | 2.0 | 0.94 |
| | | ✓ | $\mu$ | 605.12 | | 2.43 | | 1.91 | |
| | ✓ | | 37 | 423.85 | 137.3 | 1.6 | 0.51 | 2.31 | 0.75 |
| | ✓ | | 38 | 446.14 | 161.92 | 1.67 | 0.61 | 3.22 | 1.17 |
| | ✓ | | 39 | 413.95 | 154.56 | 1.44 | 0.54 | 2.77 | 1.04 |
| | ✓ | | 40 | 2107.23 | 828.78 | 9.02 | 3.5 | 2.8 | 1.1 |
| | ✓ | | $\mu$ | 847.79 | | 3.43 | | 2.78 | |
| | ✓ | ✓ | 37 | 195.49 | 15.33 | 2.08 | 0.02 | 1.06 | 0.08 |
| | ✓ | ✓ | 38 | 343.86 | 544.62 | 2.36 | 1.85 | 2.48 | 3.93 |
| | ✓ | ✓ | 39 | 206.19 | 52.07 | 1.79 | 0.03 | 1.38 | 0.35 |
| | ✓ | ✓ | 40 | 3538.35 | 4518.19 | 17.21 | 19.0 | 8.49 | 10.84 |
| | ✓ | ✓ | $\mu$ | 1070.97 | | 5.86 | | 3.35 | |

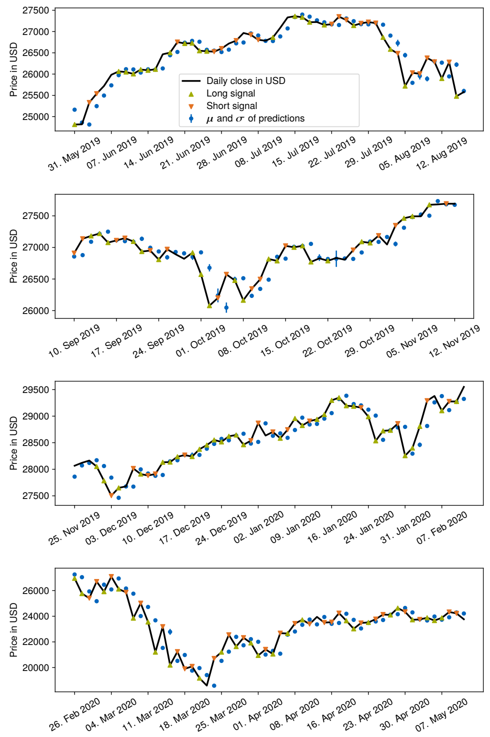
**Table C.17.:** Measurement metrics for the DJI TCN FA-fundamentals setups.



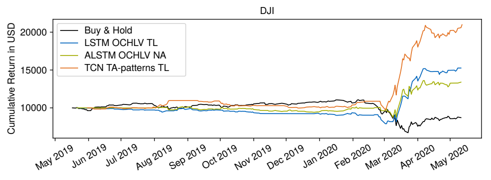
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

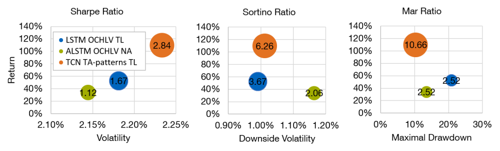


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | 0.0024 | 0.1860** | 0.040 | 208 |
| | (0.001) | (0.063) | | |
| ALSTM OCHLV NA | 0.0015 | 0.1240* | 0.016 | 236 |
| | (0.001) | (0.063) | | |
| TCN TA-Patterns TL | 0.0040* | −0.0139 | 0.000 | 204 |
| | (0.002) | (0.067) | | |

$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

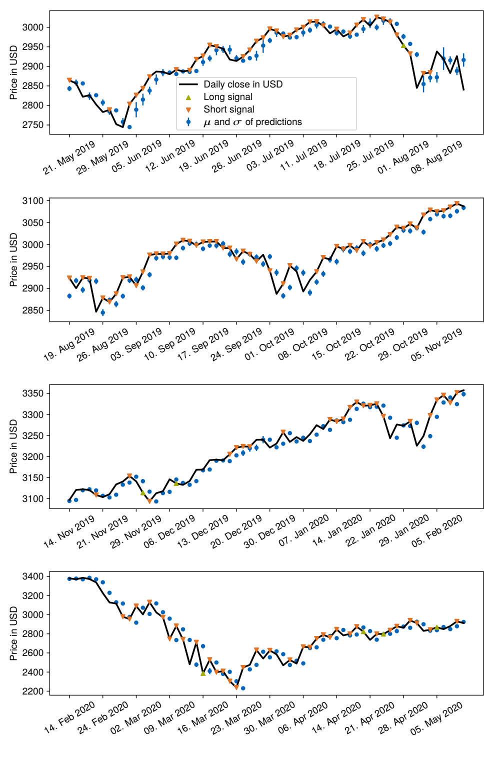
**(e)** CAPM



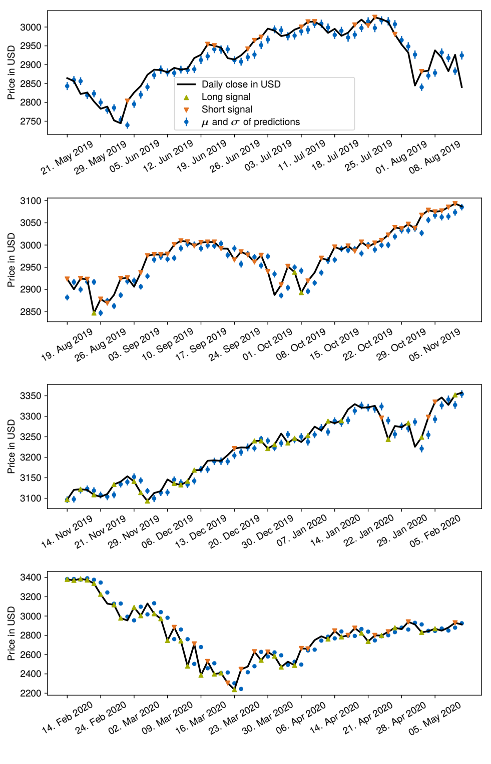
**(f)** Risk-adjusted return ratios

**Figure C.1:** Dow Jones (ˆDJI) predictions.

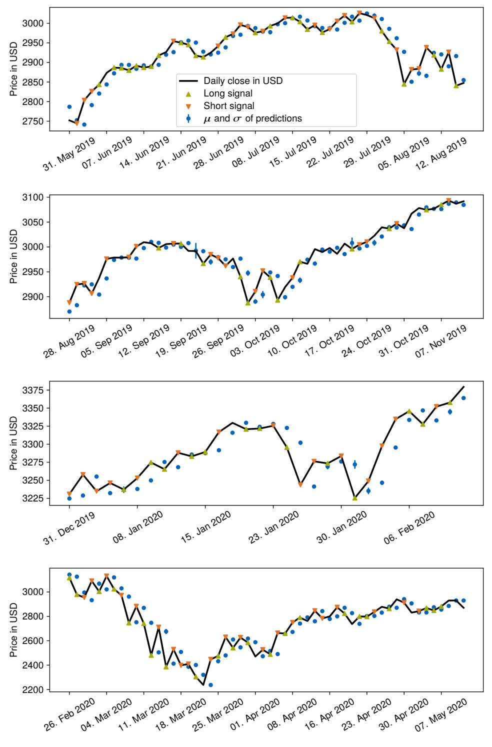
## C.2. Indices, Stocks and Bitcoin



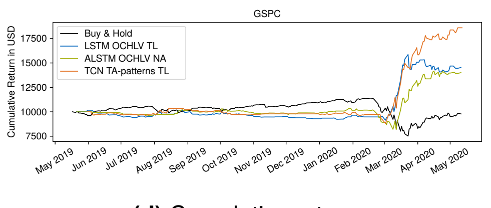
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

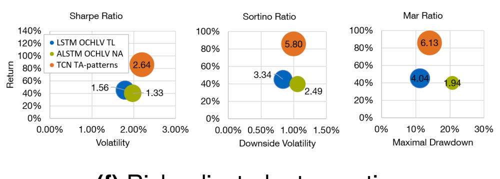


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0018^{*}$ | $-0.6125^{***}$ | 0.513 | 236 |
| | (0.001) | (0.039) | | |
| ALSTM OCHLV NA | 0.001 | $0.2893^{***}$ | 0.094 | 236 |
| | (0.001) | (0.059) | | |
| TCN TA-Patterns TL | $0.0036^{*}$ | 0.0337 | 0.001 | 188 |
| | (0.002) | (0.070) | | |

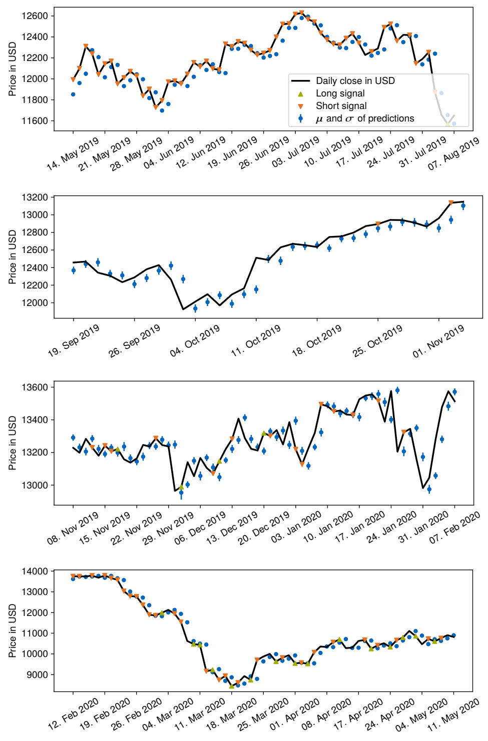
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM

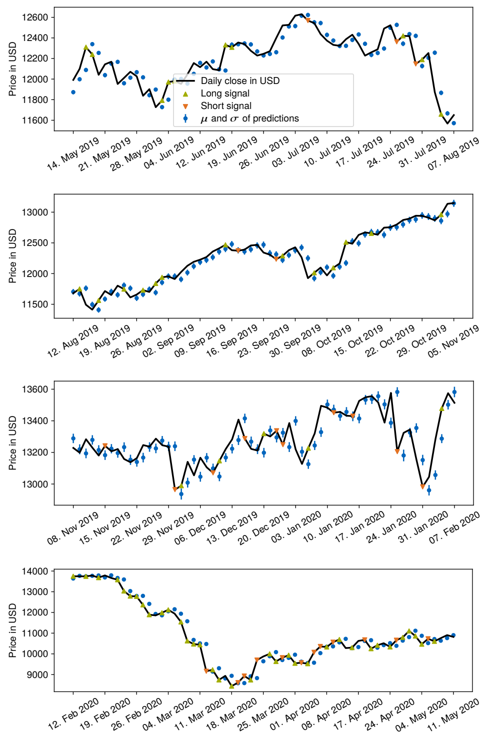


**(f)** Risk-adjusted return ratios

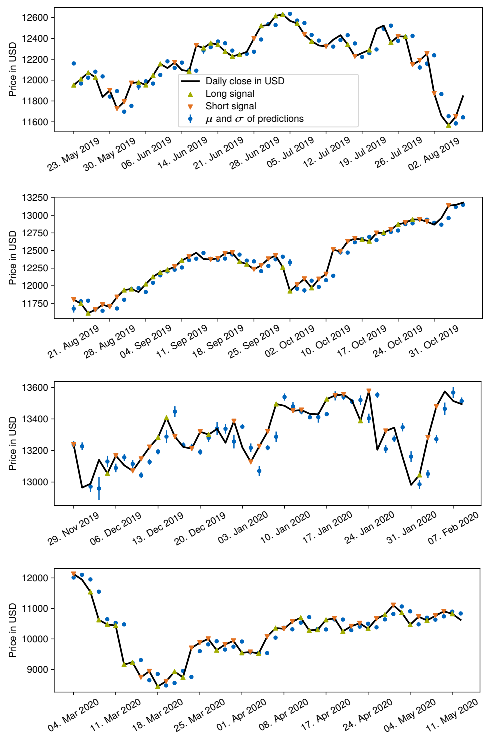
**Figure C.2:** S&P500 (ˆGSPC) predictions.



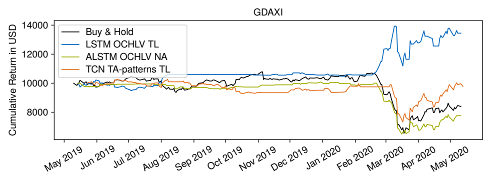
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

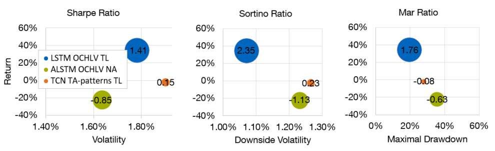


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0017$ | $0.1555^{*}$ | 0.029 | 212 |
| | $(0.001)$ | $(0.063)$ | | |
| ALSTM OCHLV NA | $-0.0006$ | $0.6511^{***}$ | 0.538 | 240 |
| | $(0.001)$ | $(0.039)$ | | |
| TCN TA-Patterns TL | $0.0001$ | $0.4322^{***}$ | 0.193 | 200 |
| | $(0.001)$ | $(0.063)$ | | |

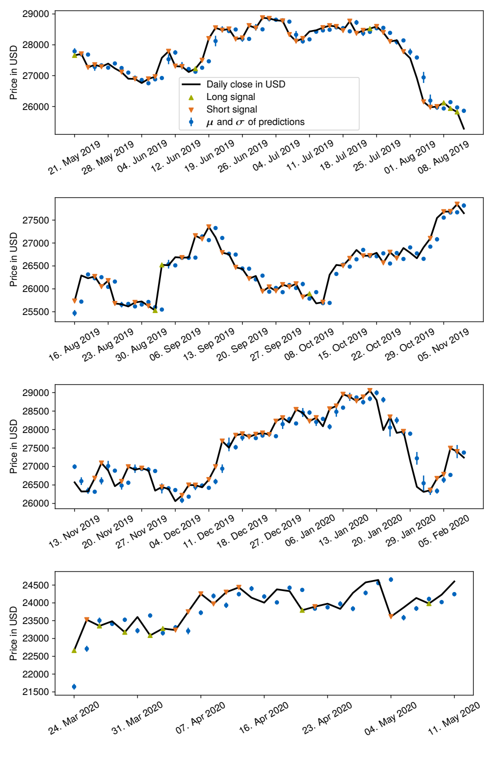
${}^{*}p < .05, \quad {}^{**}p < .01, \quad {}^{***}p < .001$

**(e)** CAPM

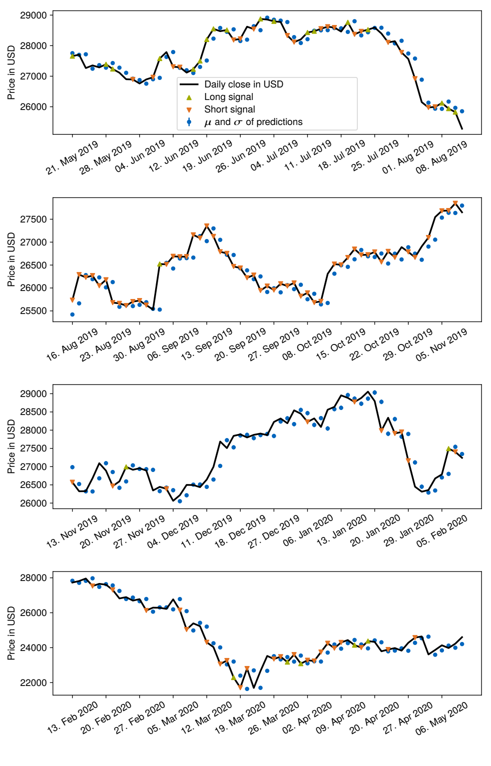


**(f)** Risk-adjusted return ratios

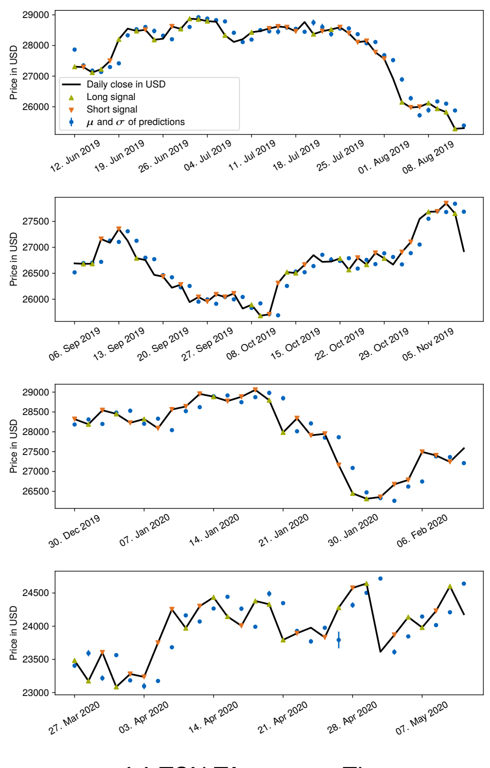
**Figure C.3:** DAX (ˆGDAXI) predictions.



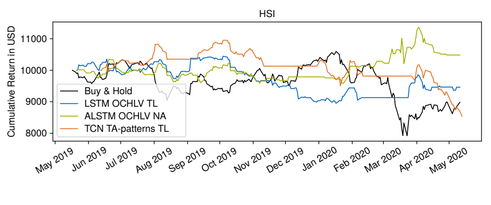
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

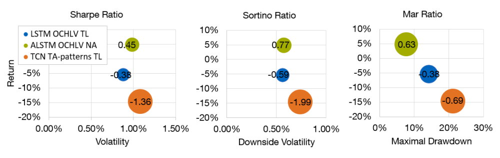


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $-0.0007$ | $-0.2778^{***}$ | 0.130 | 204 |
| | $(0.001)$ | $(0.051)$ | | |
| ALSTM OCHLV NA | $0.0001$ | $-0.3716^{***}$ | 0.268 | 232 |
| | $(0.001)$ | $(0.041)$ | | |
| TCN TA-Patterns TL | $-0.0010$ | $-0.1059$ | 0.013 | 144 |
| | $(0.001)$ | $(0.078)$ | | |

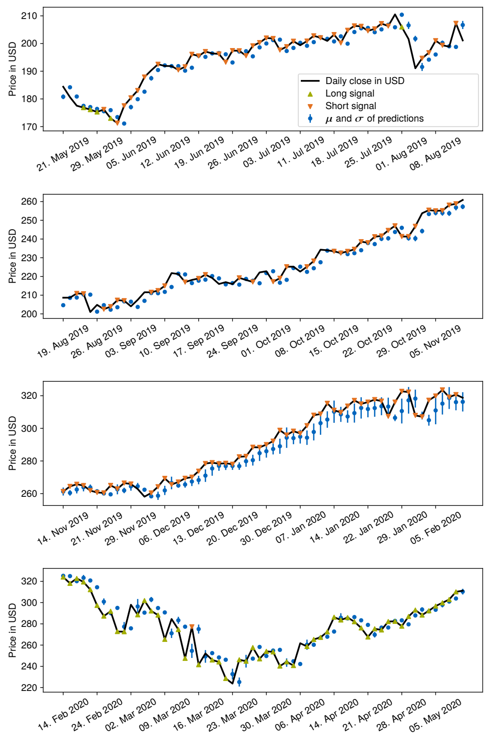
$^{*}p < .05,\quad ^{**}p < .01,\quad ^{***}p < .001$

**(e)** CAPM

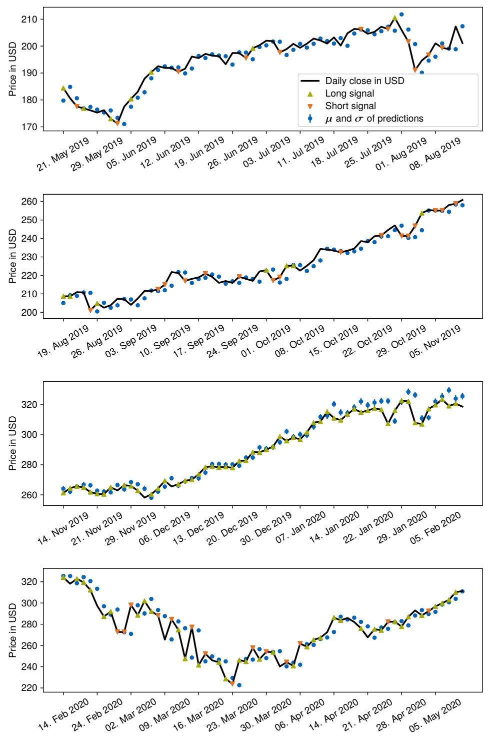


**(f)** Risk-adjusted return ratios

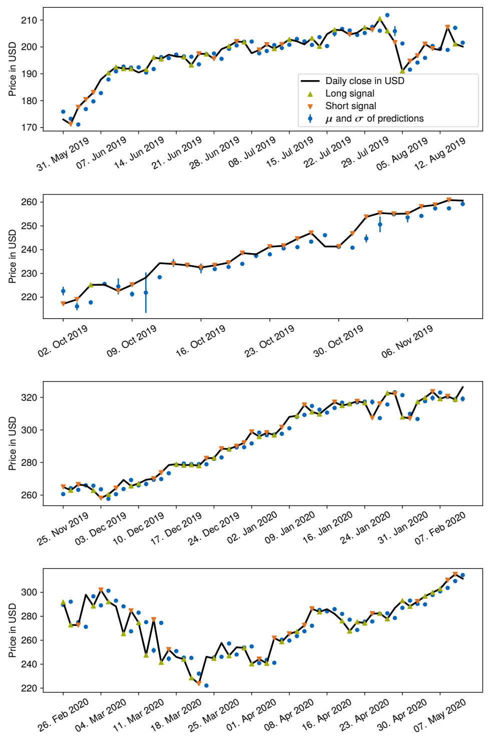
**Figure C.4:** Heng Seng Index (ˆHSI) predictions.



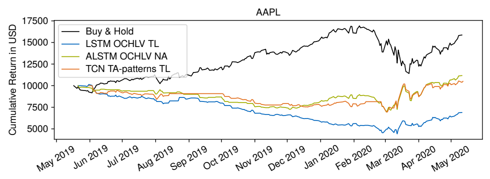
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

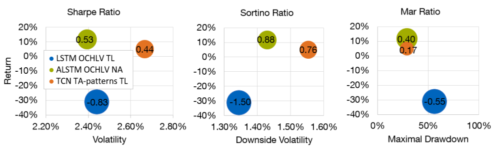


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $-0.0019$ | $0.2731^{***}$ | 0.084 | 236 |
| | (0.002) | (0.059) | | |
| ALSTM OCHLV NA | 0.0006 | 0.1058 | 0.013 | 236 |
| | (0.001) | (0.039) | | |
| TCN TA-Patterns TL | 0.0005 | 0.0671 | 0.005 | 188 |
| | (0.002) | (0.071) | | |

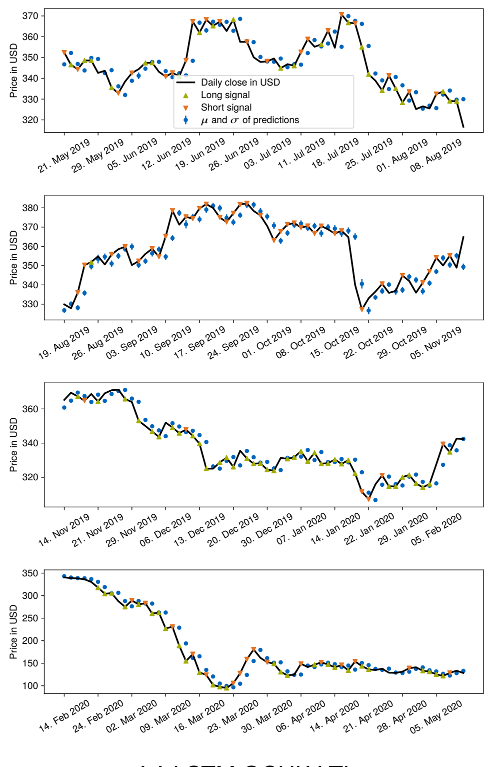
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM

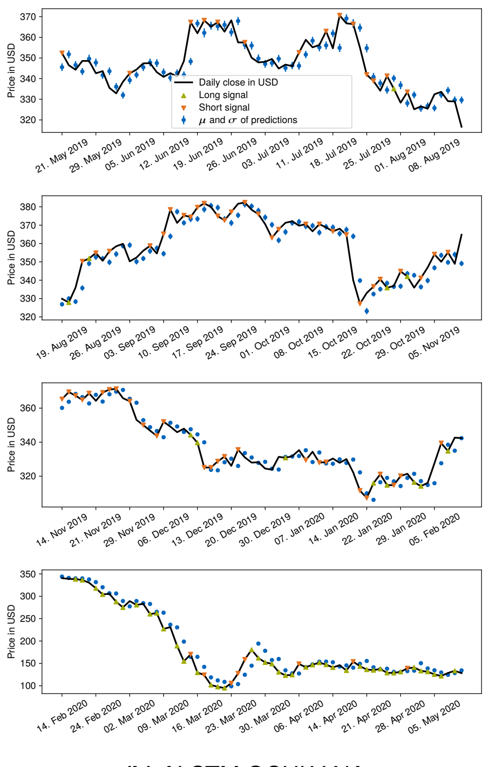


**(f)** Risk-adjusted return ratios

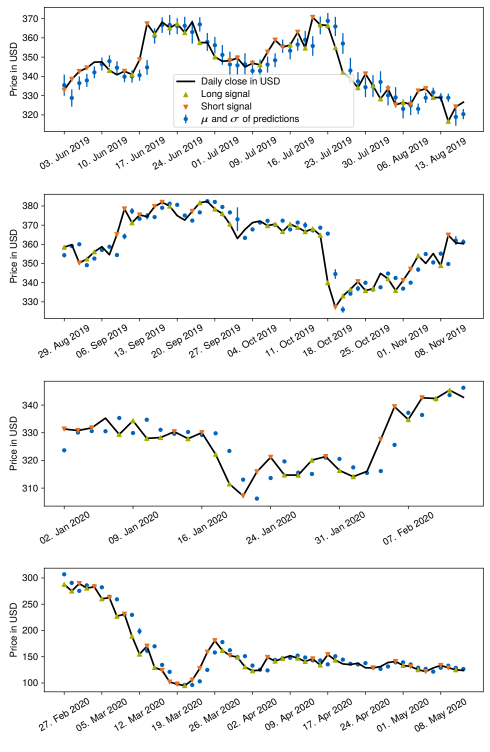
**Figure C.5:** Apple (AAPL) predictions.



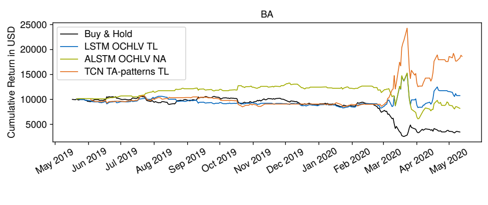
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

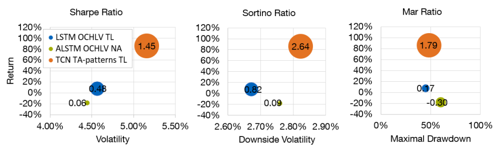


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0004$ | $-0.2864^{***}$ | 0.088 | 236 |
| | $(0.003)$ | $(0.060)$ | | |
| ALSTM OCHLV NA | $-0.0004$ | $-0.0586$ | 0.004 | 236 |
| | $(0.003)$ | $(0.061)$ | | |
| TCN TA-Patterns TL | $0.0036$ | $-0.3831^{***}$ | 0.150 | 188 |
| | $(0.001)$ | $(0.063)$ | | |

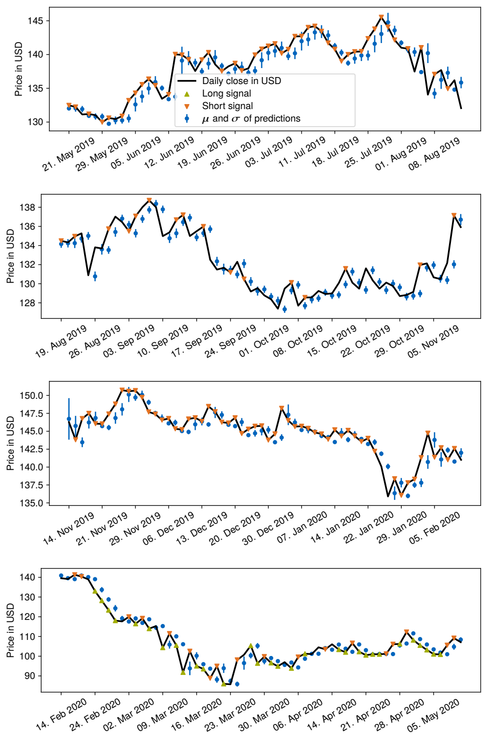
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM

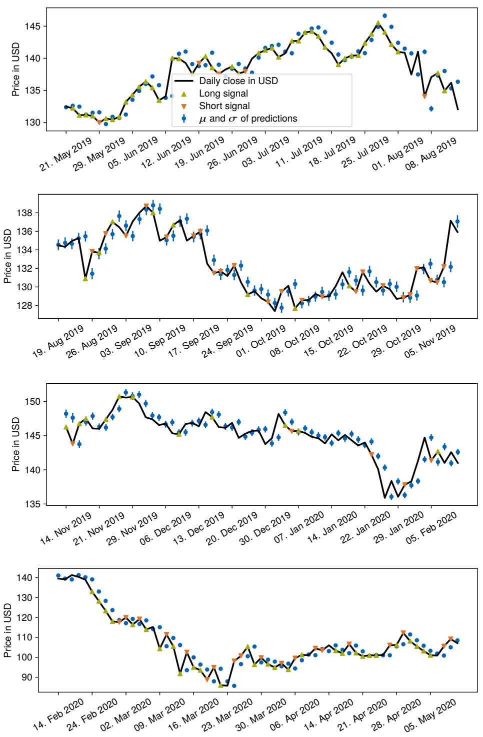


**(f)** Risk-adjusted return ratios

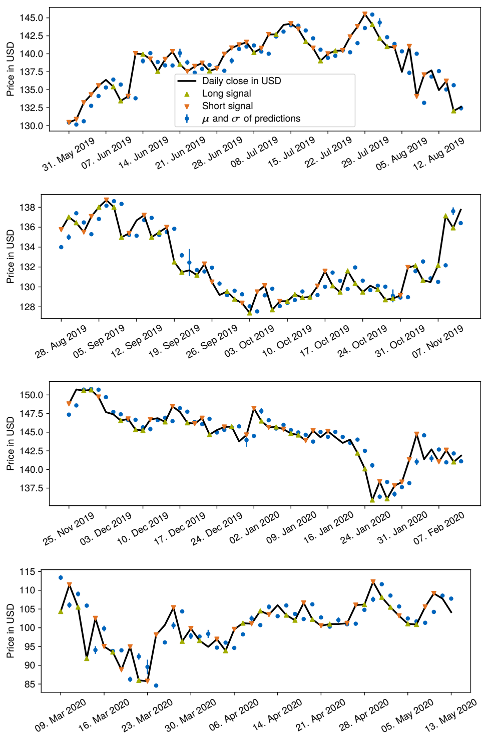
**Figure C.6:** Boeing (BA) predictions.



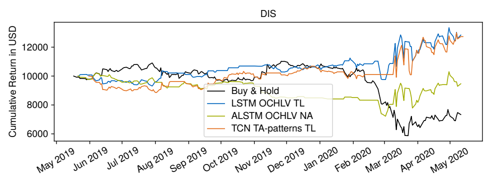
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

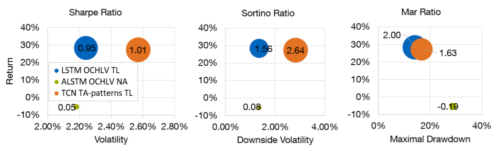


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0014$ | $0.0858$ | 0.010 | 236 |
| | (0.001) | (0.055) | | |
| ALSTM OCHLV NA | $0.0002$ | $0.1639^{**}$ | 0.039 | 236 |
| | (0.001) | (0.053) | | |
| TCN TA-Patterns TL | $0.0017$ | $-0.1268$ | 0.017 | 204 |
| | (0.002) | (0.068) | | |

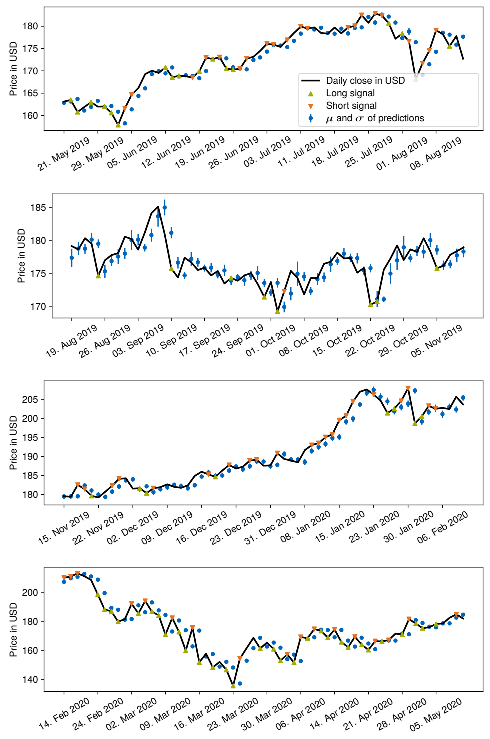
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM

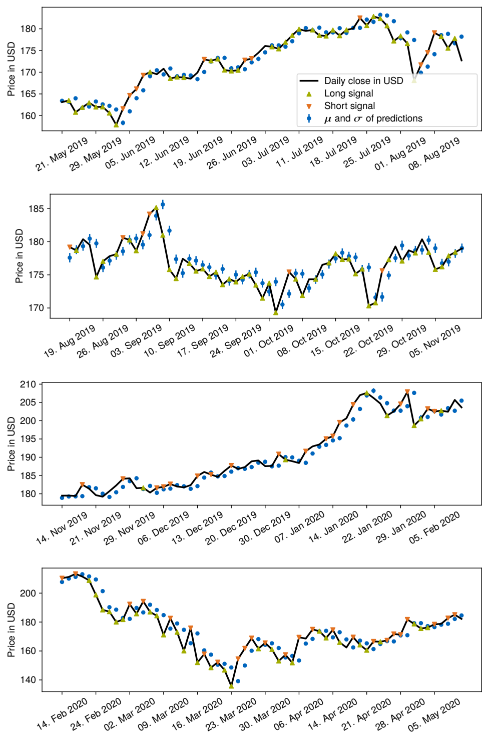


**(f)** Risk-adjusted return ratios

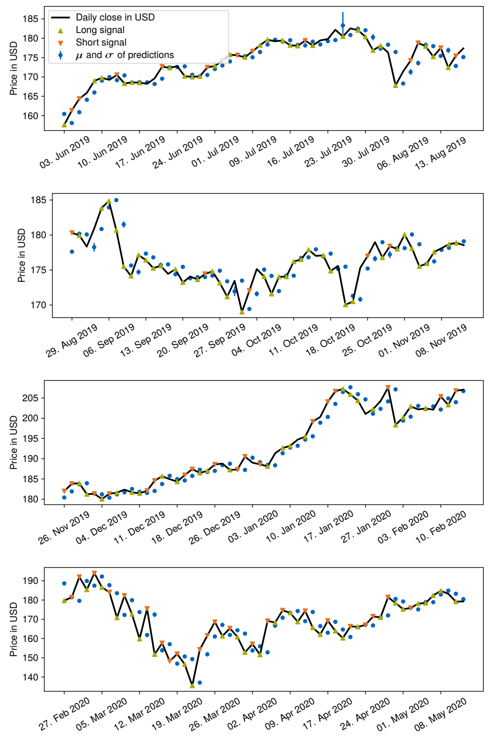
**Figure C.7:** Disney (DIS) predictions.



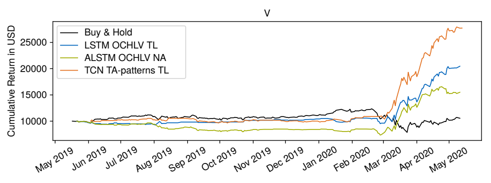
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

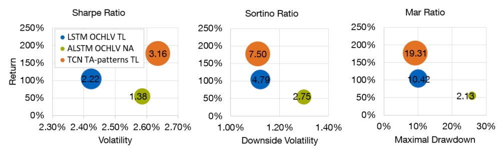


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0032^{*}$ | $0.2971^{***}$ | 0.103 | 235 |
| | (0.002) | (0.057) | | |
| ALSTM OCHLV NA | 0.0021 | $0.2887^{***}$ | 0.085 | 236 |
| | (0.002) | (0.062) | | |
| TCN TA-Patterns TL | $0.0050^{**}$ | $0.1751^{**}$ | 0.032 | 212 |
| | (0.002) | (0.067) | | |

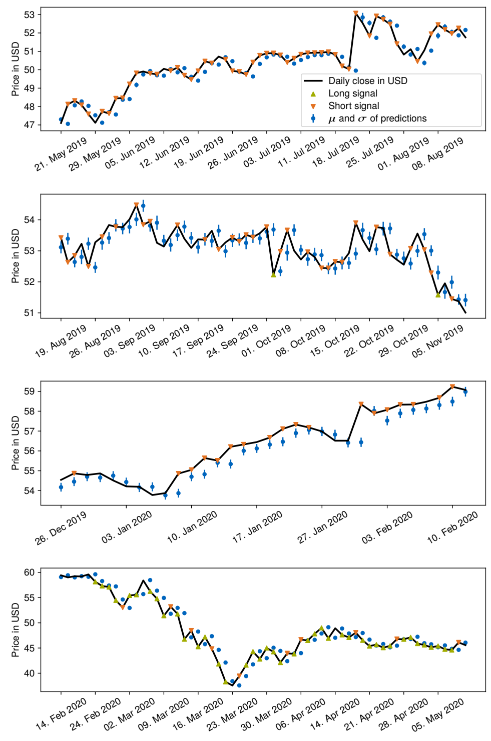
$^{*}p < .05,\quad ^{**}p < .01,\quad ^{***}p < .001$

**(e)** CAPM

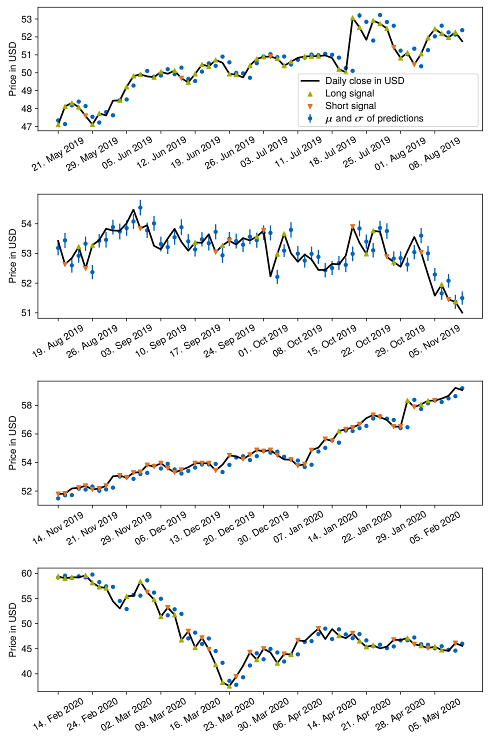


**(f)** Risk-adjusted return ratios

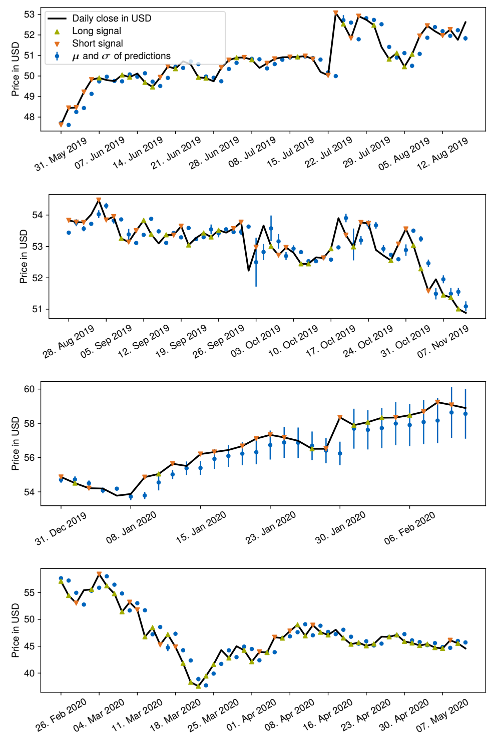
**Figure C.8:** Visa (V) predictions.



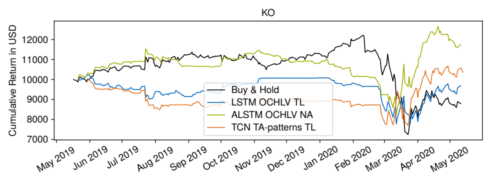
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

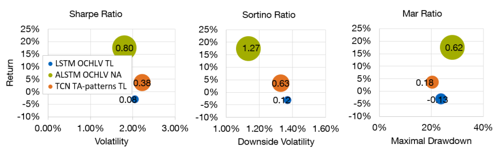


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | 0.0002 | 0.3453*** | 0.132 | 208 |
| | (0.001) | (0.062) | | |
| ALSTM OCHLV NA | 0.0009 | 0.2582*** | 0.085 | 236 |
| | (0.001) | (0.055) | | |
| TCN TA-Patterns TL | 0.0007 | 0.2997*** | 0.092 | 188 |
| | (0.002) | (0.069) | | |

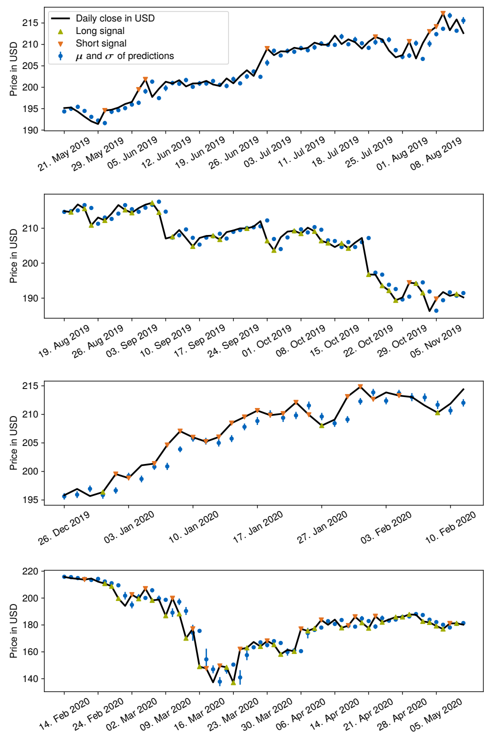
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM

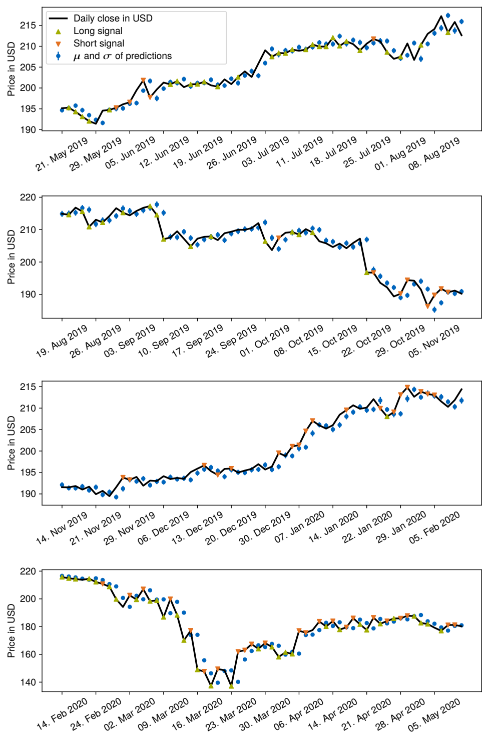


**(f)** Risk-adjusted return ratios

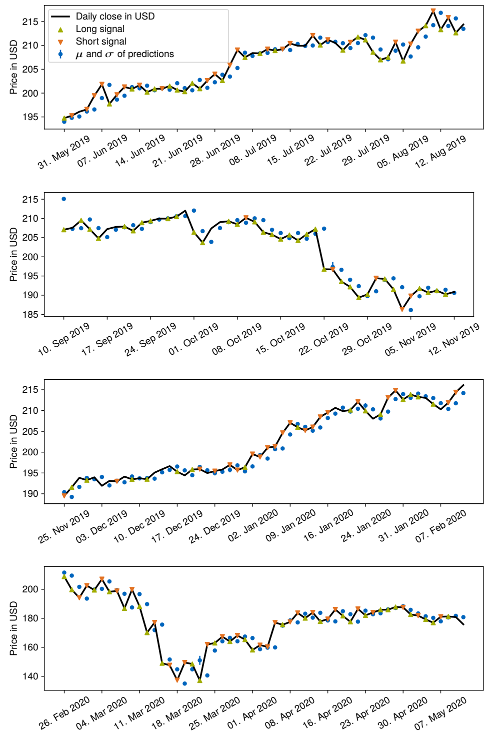
**Figure C.9:** Coca-Cola (KO) predictions.



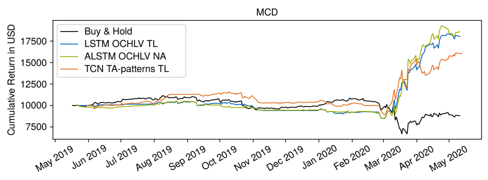
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

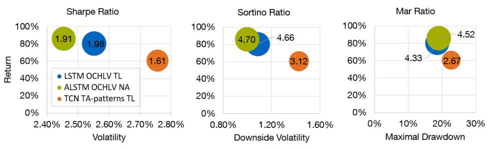


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | 0.0032 | 0.2451*** | 0.067 | 208 |
| | (0.002) | (0.064) | | |
| ALSTM OCHLV NA | 0.0030 | 0.2466*** | 0.065 | 236 |
| | (0.002) | (0.061) | | |
| TCN TA-Patterns TL | 0.0028 | 0.0019 | 0.000 | 204 |
| | (0.002) | (0.072) | | |

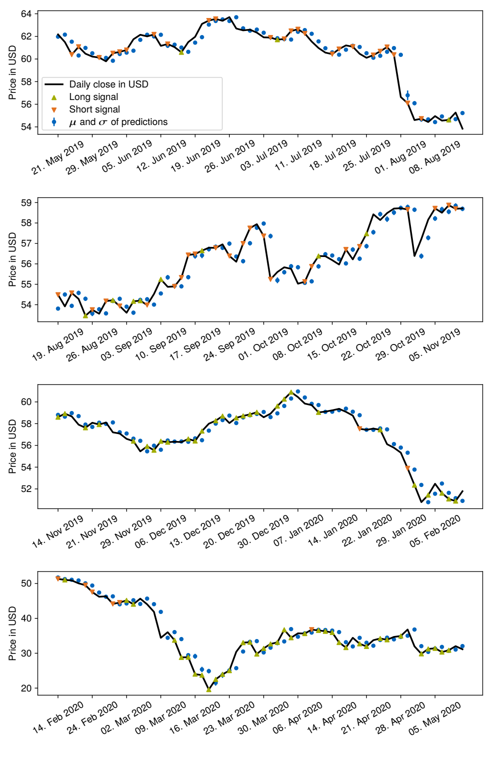
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM

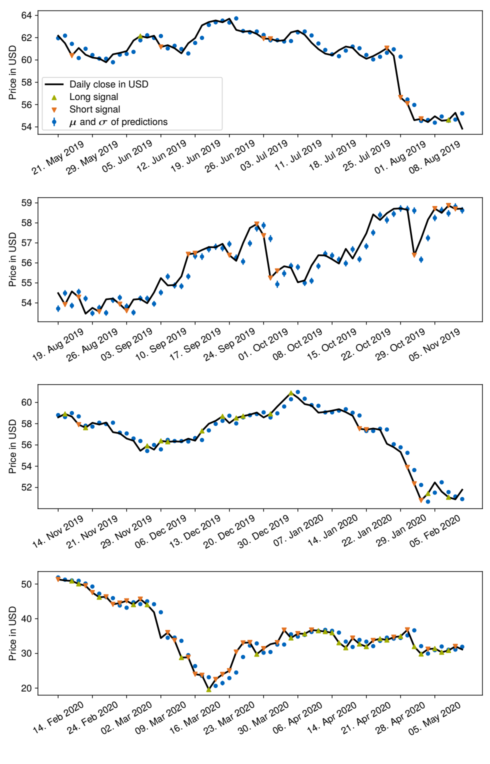


**(f)** Risk-adjusted return ratios

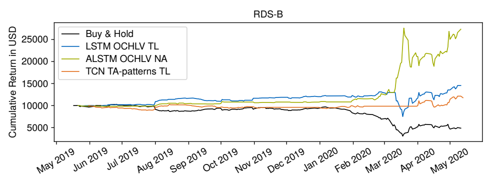
**Figure C.10:** McDonald's (MCD) predictions.



**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA

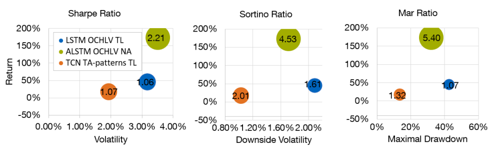


**(d)** Cumulative return

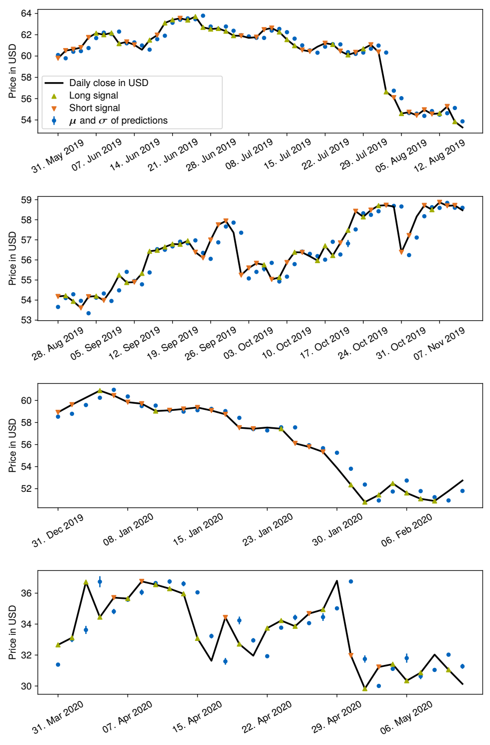
| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0036^{**}$ | $0.6541^{***}$ | 0.579 | 236 |
| | (0.001) | (0.036) | | |
| ALSTM OCHLV NA | 0.0039 | $-0.4568^{***}$ | 0.230 | 236 |
| | (0.002) | (0.055) | | |
| TCN TA-Patterns TL | 0.0016 | $0.2677^{***}$ | 0.109 | 164 |
| | (0.001) | (0.060) | | |

$^{*}p < .05,\quad ^{**}p < .01,\quad ^{***}p < .001$

**(e)** CAPM

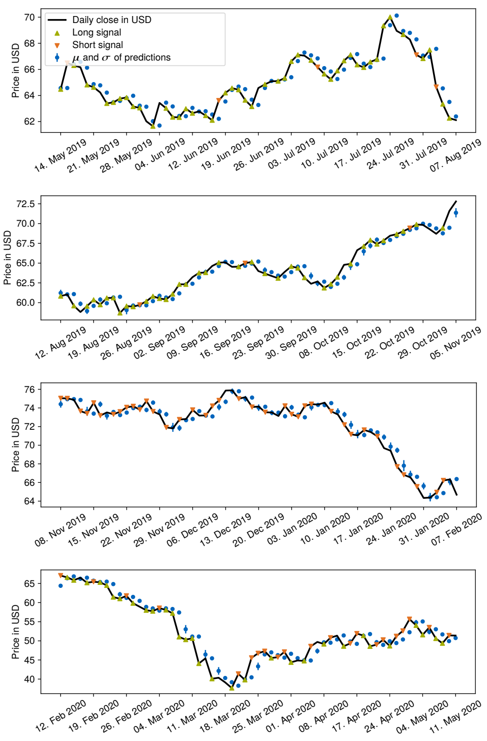


**(f)** Risk-adjusted return ratios

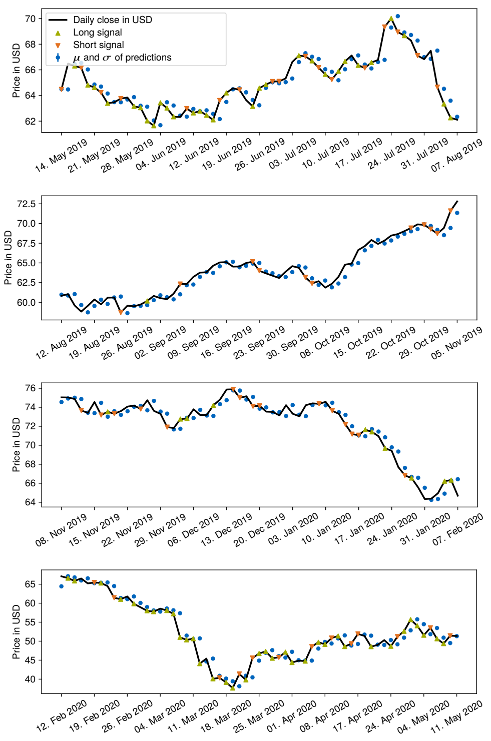


**(c)** TCN TA-patterns TL

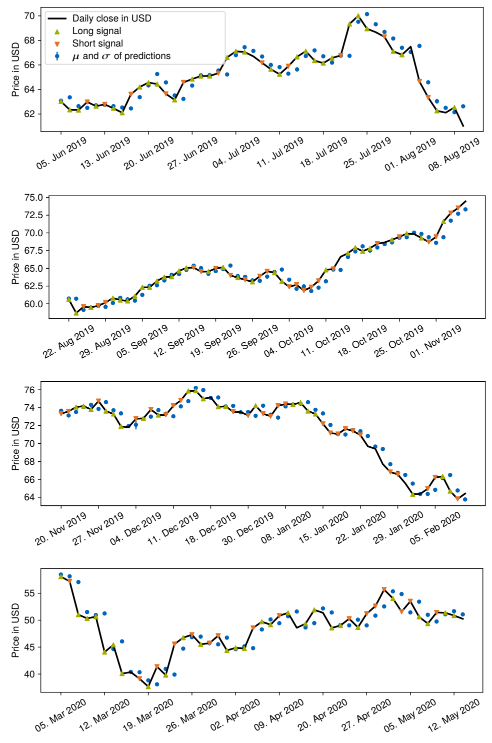
**Figure C.11:** Shell (RDS-B) predictions.



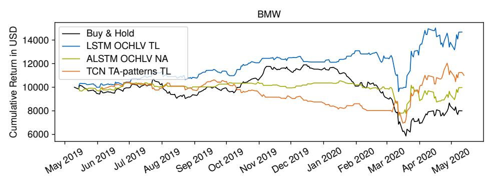
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

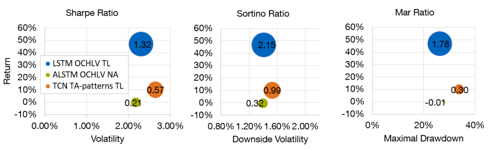


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0023^{*}$ | $0.4845^{***}$ | 0.284 | 240 |
| | (0.001) | (0.050) | | |
| ALSTM OCHLV NA | 0.0007 | $0.5336^{***}$ | 0.384 | 240 |
| | (0.001) | (0.044) | | |
| TCN TA-Patterns TL | 0.0010 | $0.3588^{***}$ | 0.133 | 200 |
| | (0.002) | (0.065) | | |

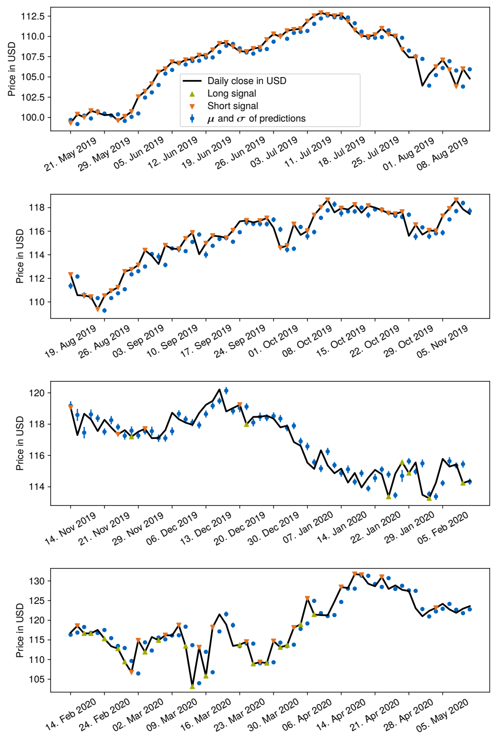
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM

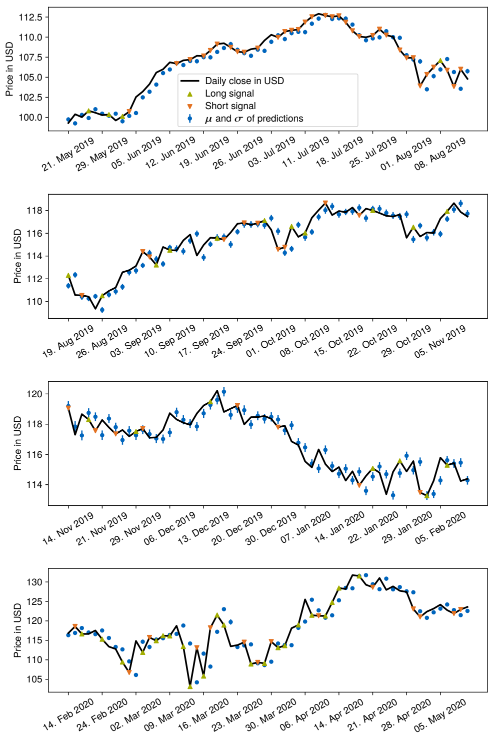


**(f)** Risk-adjusted return ratios

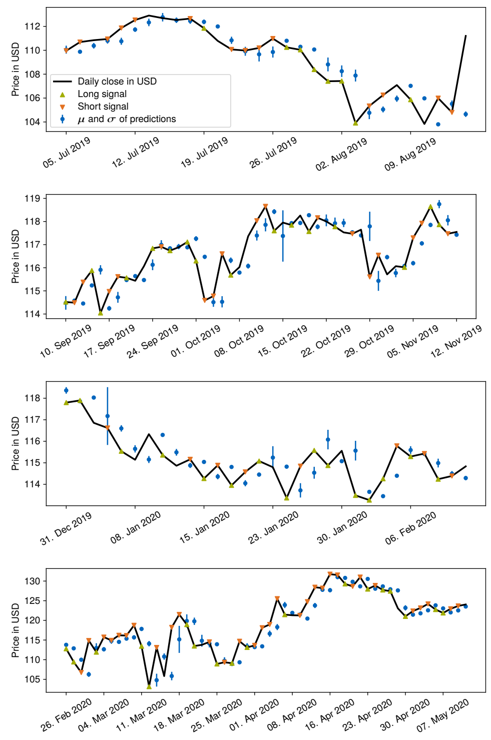
**Figure C.12:** BMW predictions.



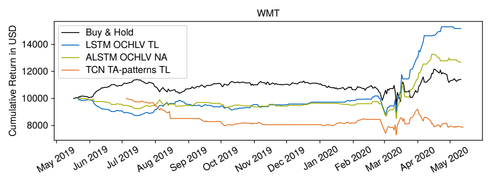
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

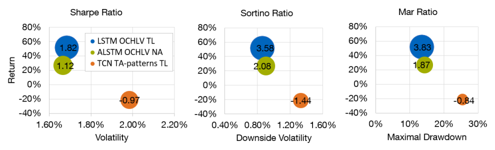


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0018$ | $0.2054^{**}$ | 0.050 | 236 |
| | $(0.001)$ | $(0.059)$ | | |
| ALSTM OCHLV NA | $0.0009$ | $0.3549^{***}$ | 0.152 | 236 |
| | $(0.001)$ | $(0.055)$ | | |
| TCN TA-Patterns TL | $-0.0012$ | $-0.0215$ | 0.001 | 156 |
| | $(0.002)$ | $(0.071)$ | | |

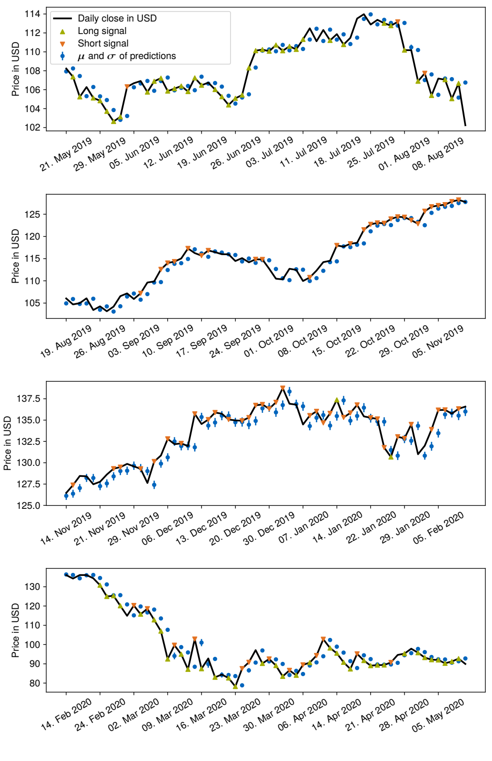
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM

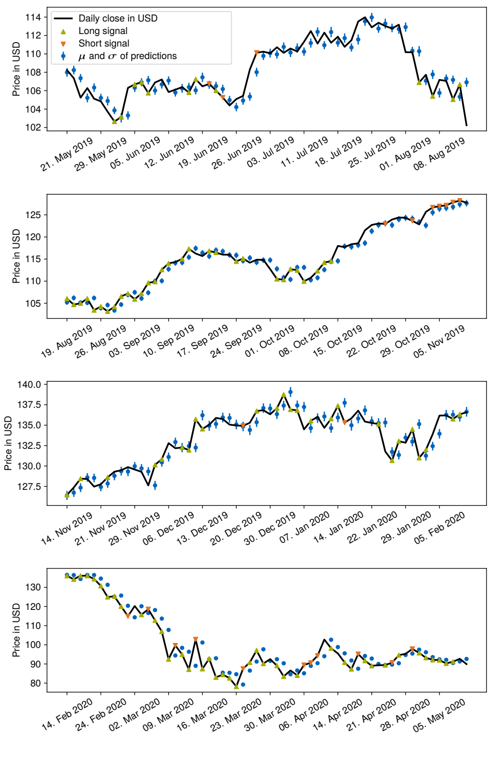


**(f)** Risk-adjusted return ratios

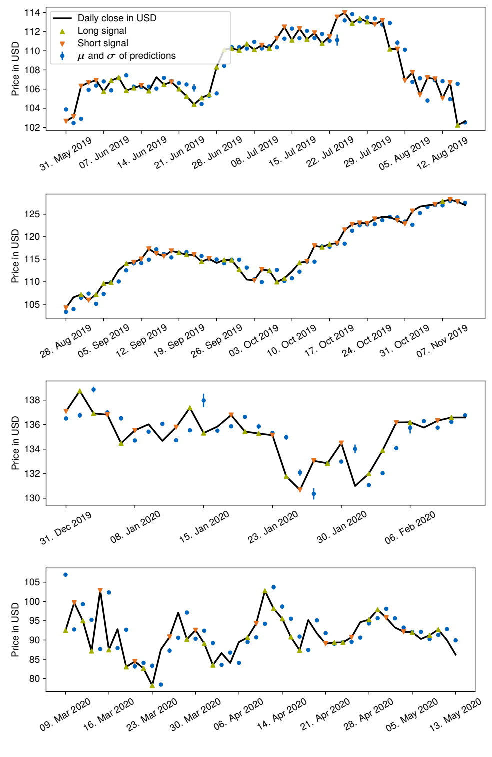
**Figure C.13:** Walmart (WMT) predictions.



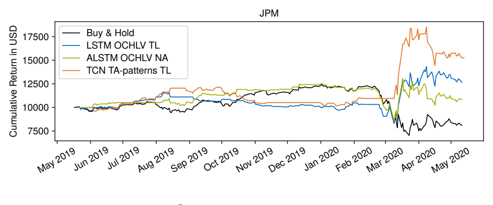
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

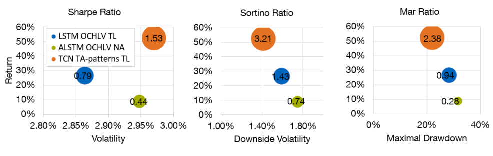


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0016$ | $0.3169^{***}$ | 0.113 | 236 |
| | $(0.002)$ | $(0.058)$ | | |
| ALSTM OCHLV NA | $0.0010$ | $0.4856^{***}$ | 0.251 | 236 |
| | $(0.002)$ | $(0.055)$ | | |
| TCN TA-Patterns TL | $0.0020$ | $0.3001^{***}$ | 0.092 | 212 |
| | $(0.001)$ | $(0.065)$ | | |

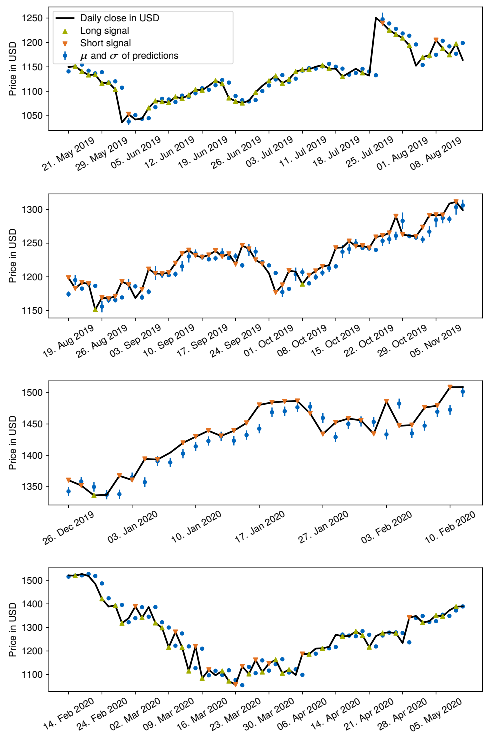
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM

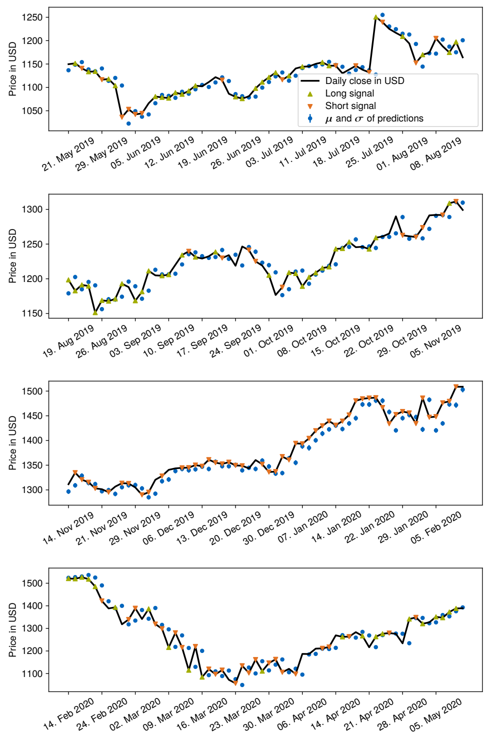


**(f)** Risk-adjusted return ratioss

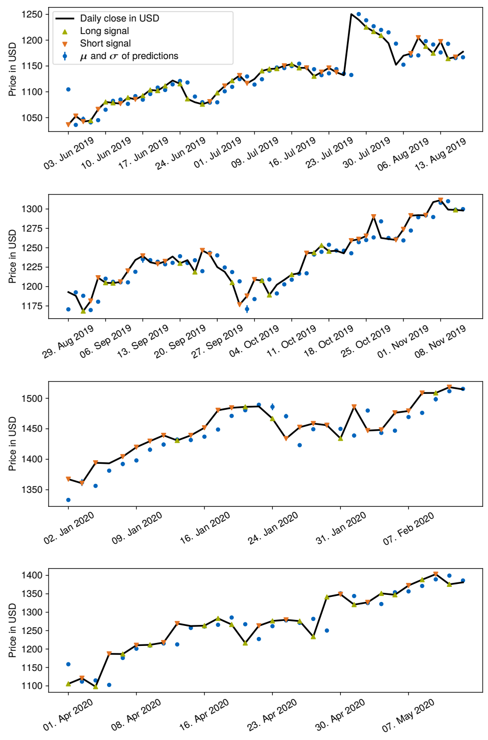
**Figure C.14:** JPMorgan (JPM) predictions.



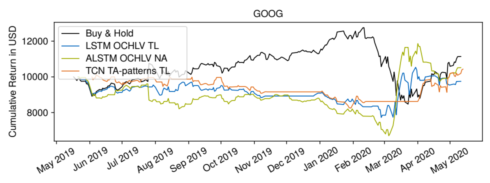
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

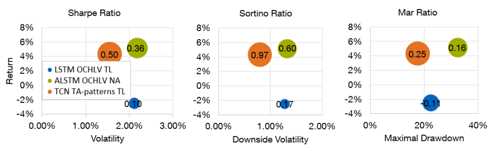


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | $0.0004$ | $0.1134$ | 0.018 | 208 |
| | (0.001) | (0.059) | | |
| ALSTM OCHLV NA | $0.0008$ | $-0.3214^{***}$ | 0.119 | 236 |
| | (0.001) | (0.057) | | |
| TCN TA-Patterns TL | $-0.0003$ | $0.2430^{***}$ | 0.078 | 164 |
| | (0.001) | (0.065) | | |

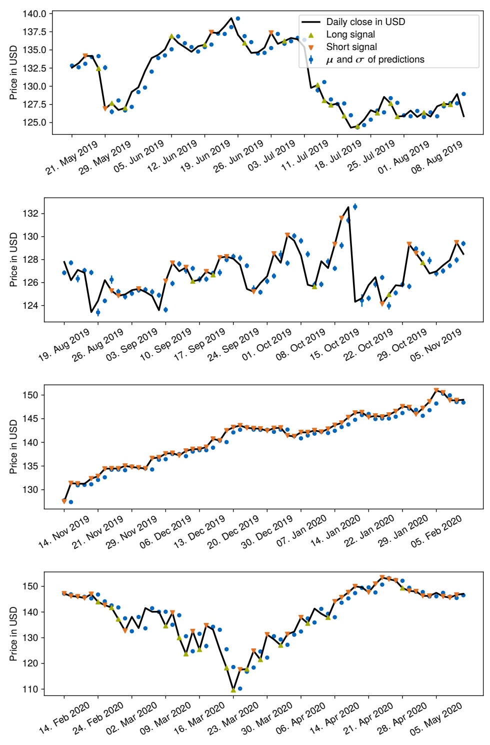
$^{*}p < .05,\quad ^{**}p < .01,\quad ^{***}p < .001$

**(e)** CAPM

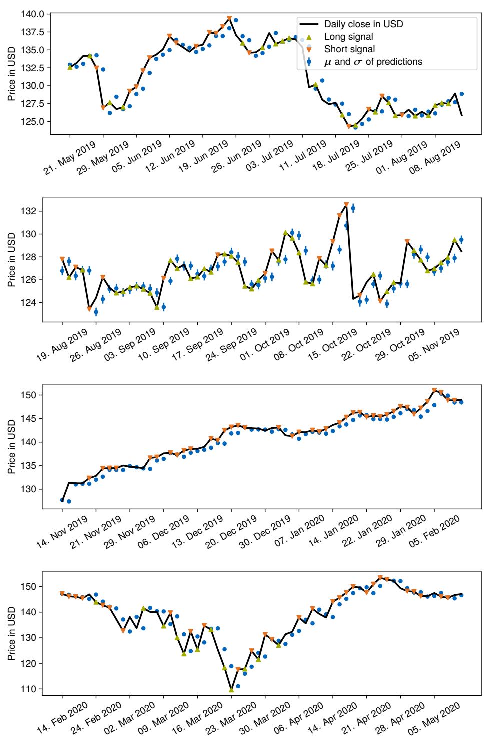


**(f)** Risk-adjusted return ratios

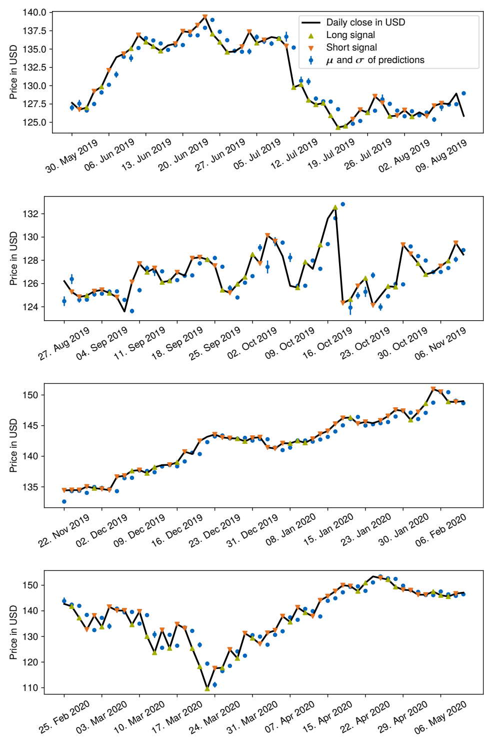
**Figure C.15:** Alphabet (GOOG) predictions.



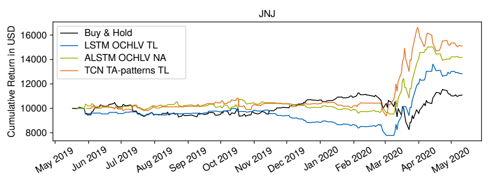
**(a)** LSTM OCHLV TL



**(b)** ALSTM OCHLV NA



**(c)** TCN TA-patterns TL

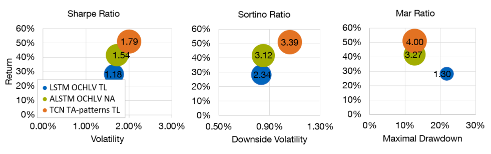


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV TL | 0.0010 | 0.3029*** | 0.132 | 236 |
| | (0.001) | (0.051) | | |
| ALSTM OCHLV NA | 0.0015* | 0.2298*** | 0.069 | 236 |
| | (0.001) | (0.055) | | |
| TCN TA-Patterns TL | 0.0020 | 0.3001*** | 0.092 | 212 |
| | (0.001) | (0.065) | | |

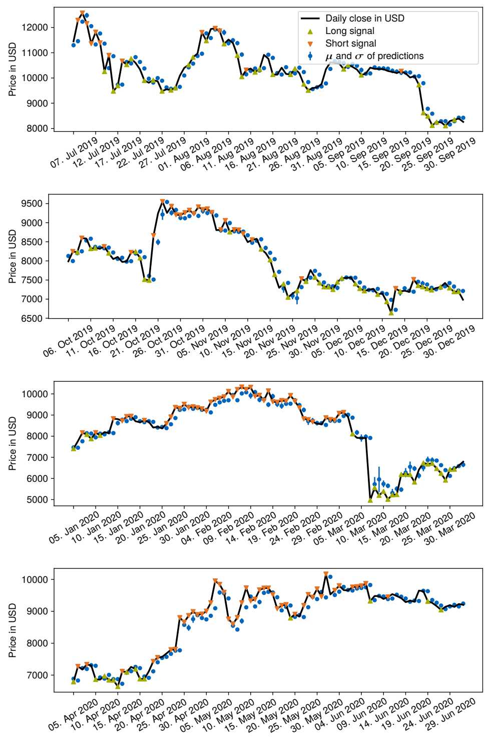
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM

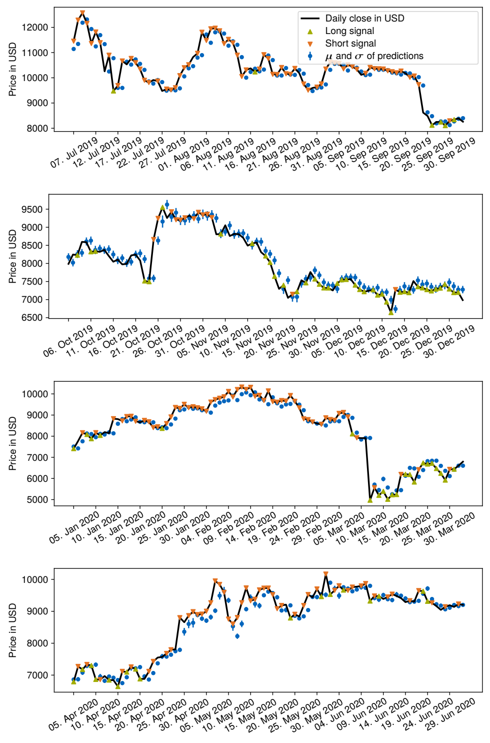


**(f)** Risk-adjusted return ratios

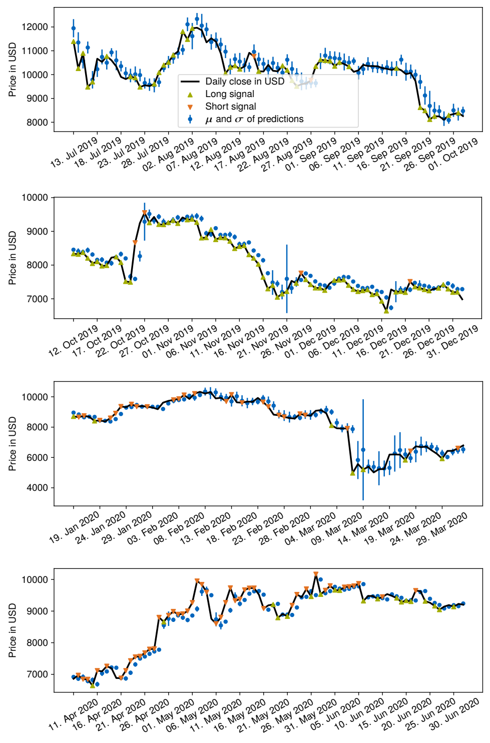
**Figure C.16:** Johnson & Johnson (JNJ) predictions.



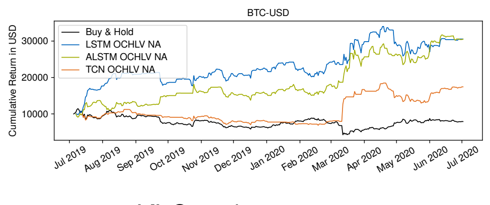
**(a)** LSTM OCHLV NA



**(b)** ALSTM OCHLV NA



**(c)** TCN OCHLV NA

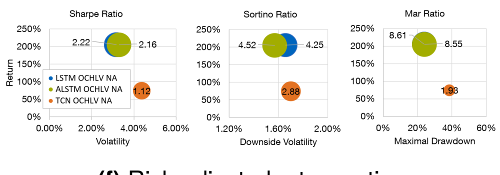


**(d)** Cumulative return

| | $\alpha$ | $\beta$ | $R^2$ | m |
|---|---|---|---|---|
| LSTM OCHLV NA | $0.0037^{*}$ | $0.1199^{*}$ | 0.023 | 352 |
| | (0.002) | (0.042) | | |
| ALSTM OCHLV NA | $0.0037^{*}$ | $-0.0693$ | 0.007 | 352 |
| | (0.002) | (0.043) | | |
| TCN OCHLV NA | 0.0024 | $-0.2406^{***}$ | 0.050 | 320 |
| | (0.002) | (0.059) | | |

$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$

**(e)** CAPM



**(f)** Risk-adjusted return ratios

**Figure C.17:** Bitcoin (BTC-USD) predictions.