Research in International Business and Finance 62 (2022) 101747

Contents lists available at ScienceDirect

Research in International Business and Finance

journal homepage: www.elsevier.com/locate/ribaf

Synthetic data generation with deep generative models to enhance predictive tasks in trading strategies

Daniel Carvajal-Patiño, Raul Ramos-Pollán *

Universidad de Antioquia, Facultad de Ingeniería, Medellín, Colombia

ARTICLE INFO

JEL classification:
O160
G400
G170
G150
G110
F370

Keywords:
Trading strategies
Machine learning
Synthetic data
Deep generative models
Deep learning
Trading simulations

ABSTRACT

This work develops machine learning (ML) predictive models on price signals for financial instruments and their integration into trading strategies. In general, ML models have been shown powerful when trained with large amounts of data. In practice, the time-series nature of financial datasets limits the effective amount of data available to train, validate and retrain models since special care must be taken not to include future data in any way. In this setting, we develop deep generative models to produce synthetic time-series data, enhancing the amount of data available for training predictive models. Synthetic data obtained this way replicates the distribution properties of real historical data, leads to better performance, and enables thorough validation of predictive models for price signals. We leverage machine-generated predictive signals on synthetic data to build trading strategies. We show consistent improvement leading up to profits in our simulations for commodities and forex exchange markets.

1. Introduction

Grace (2020) mentions that one difficulty when working in financial fields is data quality, which can be caused by issues such as missing values, discrepancies, inconsistencies, or simply access to data due to restrictions or privacy. This also affects emerging fields such as artificial intelligence (AI) since large amounts of data are required for most kinds of analytics, including building predictive models. In AI, this can be caused by several factors, such as the cost of obtaining data, specially when we focus on rare or scarce events, or the intrinsic limits of the training and validation methods we use. In finance, examples of critical challenges for researchers include access to recent real transaction data for money laundering detection (Jullum et al., 2020) or enhancing data for performance comparison with Monte Carlo study methodologies (Susan et al., 2021).

Financial markets operate at high frequency, meaning that new market information is readily available in very short periods of time. However, only large companies operating in the market for years are in a position to construct large historical datasets. In addition, many methodological requirements limit the amount of effective data available to build predictive models, such as discarding old noisy data, carefully splitting data for validation of time-series information, etc. This is specially delicate when working with financial time series such as market price signals, and different studies address it in many ways. Liu et al. (2021) highlight this and suggest producing synthetic data using various techniques, including ML models. Duan et al. (2022) generate new features based on market signals and macroeconomic indicators to train models. Jørgensen and Igel (2021) resort to transfer-learning, which uses models trained to generalize over different companies from only a few available. On the other hand, it is also

* Correspondence to: Universidad de Antioquia, Calle 70 No. 52-21, Medellín, Colombia.
E-mail addresses: danielcarvajalpatino@gmail.com (D. Carvajal-Patiño), raul.ramos@udea.edu.co (R. Ramos-Pollán).

https://doi.org/10.1016/j.ribaf.2022.101747
Received 14 June 2021; Received in revised form 8 August 2022; Accepted 14 August 2022
Available online 23 August 2022

common to use information from other sources such as Twitter (Jiahang and Chi, 2022). Another methodology to address this is data augmentation (David and Xiao-Li, 2001), which consists of increasing the number of samples of data using techniques such as generating synthetic data or oversampling. Depending on the type of data, one approach may be more beneficial than another. For example, oversampling of time-series data could be a challenging task due to the consecutive correlation of the data.

In this paper, we use deep generative models for synthetic data augmentation, so that more price scenarios become available for training predictive models, enhancing their robustness when facing new data. However, the use of deep generative models represents a significant challenge, since training them requires delicate computational and numerical balances, and the careful monitoring of the representativeness of the generated data (Jonathan, 2019). In fact, we required several iterations modifying the architectures and configurations of generative models, applying different tricks and best practices reported in the literature (Pan et al., 2019).

Currently, there are several deep generative models, including GANs (Generative Adversarial Networks), VAEs (Variational AutoEncoders), RBMs (Restricted Boltzmann Machines), or GTNs (Generative Teaching Networks). In this work, we propose using GANs and VAEs to generate synthetic data and increase the number of scenarios to train each predictive model. We discarded GTNs and RBMs since the output of these models is not comparable with the data they are trained with, taking into account that one of the objectives of the present work is to generate a larger number of market scenarios, and preliminary experiments showed non satisfactory results. We used the statistical properties of real and synthetic data to measure their similarity to select generative models.

The purpose of this work is to test a momentum-based trading strategy (Adam, 2021) through trading simulations and market predictions. The trading strategy uses a trend signal produced by a predictive model fed with historical market and synthetic data produced by deep generative models. Based on how the predictive task is defined, it can be challenging to achieve a high level of accuracy due to the nature of the data, as mentioned in Novak and Velušček (2016). Financial series are sequences of highly volatile data and are influenced by themselves (technical analysis) or external events (fundamental analysis). Some works use notions of momentum to perform trading and support decisions on when to start trading operations, such as Panagiotis et al. (2020). In this paper, we use ML models that produce an estimation of price velocity as a predictive signal. This required extensive experimentation and computational power over many configurations of ML models and signal preprocessing options. Finally, simulations of different trading strategies in various markets were performed, measuring their ability to generate profits by performing buy and sell operations at different times according to the predictive signals. The trading strategy consists of buying when a high and positive velocity value is estimated and selling on high and negative estimates. While the position is open we monitor the price to close it according to a predetermined training stop, or different mitigation actions aimed at identifying wrong predictions or signal properties as soon as possible. The experimentation results show the advantage of using synthetic data in trading strategies running simulations in XAU/USD and EUR/USD markets. The performance of trading strategies was better for models trained with synthetic data.

This paper is structured as follows. In Section 2, we present a literature review on the impact of synthetic data in predictive models for trading strategies. In Section 3, the strategy and prediction task implemented in this work are detailed. The parametrization, implementation, and experimentation of the methodology proposed are specified in Section 4. In Section 5, the results of the trading strategies simulations in two different markets are shown. Finally, in Section 6, the conclusions and future direction of this work are discussed.

2. Related works

As mentioned in Dev et al. (2019) and Aziz et al. (2021), there has been a significant increase in studies that perform predictions in financial markets during the last few years, primarily due to the rise of techniques and models developed in the area of AI. Today one can find simple studies that use statistical models such as ARIMA (Rafiqul and Nguyet, 2020) or complex studies that make use of the transformation of market signals to images to train complex ML models such as convolutional neural networks (Ghosh et al., 2022). Furthermore, to showcase and determine the scope of their findings, studies test their models during trading simulations and achieve the highest possible profit, as in the case of Uzunlu and Hussain (2020).

As mentioned above, studies done in financial markets present a severe problem: the lack of data. Studies, such as Lahmiri (2020) or Braun et al. (2020), deal with this problem by performing wavelet transformations of the data or using techniques such as embedding on news or blogs, respectively. In another study, Luo et al. (2018) used the continuous wavelet transform to decompose the oil price signal and train a deep predictive model with these data. The experiments conducted in this study show how using the decomposed signal helps in the performance of predictive models. However, thanks to advances in deep learning, one of the fields in AI, models have been developed to generate synthetic data that recreate (Magnus et al., 2020) in a good way the data they are trained with. In this way, it is possible to have more data to train the required predictive models; these are called deep generative models.

In this sense, the methodology proposed in this article consists of a complete experimentation cycle that includes the generation of synthetic data through deep generative models, market estimations performed by ML mechanisms using a momentum-based approach, and testing of trading strategies through market simulations. The experimentation will be detailed in the following sections. We focus on measuring the impact of using synthetic data in financial markets, as opposed to works which focus on the generation of synthetic market data without paying much attention to its effectiveness in predictive or classification tasks. Furthermore, not all studies decide to address the lack of data by making data augmentation through deep generative models. Deep generative models try to learn the data distribution they are trained with to generate samples of the learned distribution. The hope is that if the model succeeds in determining the original distribution of the data, the generated samples can be used as real samples.

GANs (Generative Adversarial Networks) is one of these models, with an architecture of two competing neural networks, known as discriminator and generator, which have become very popular in the last couple of years. It was introduced in Goodfellow et al. (2014). During the training process, the network known as the discriminator is fed both real and synthetic data and tries to classify them apart. The generator updates its weights to make its data generation deceive the classification of the discriminator. Since their creation, GANs have been frequently used in the generation of synthetic images.

Regarding financial data generation, Fernando (2019) takes data from the S&P500 indicator and generates new signals through WGAN, Wasserstein's GAN, a particular type of this network. Then, he checks the quality of the generated signals by verifying Kurtosis, autocorrelation functions, and other properties of the signals. Finally, with the real and synthetic data, he trains a ResNet model to predict the trend of this indicator, obtaining a level of accuracy of about 70%. Tizzano (2018) uses recurrent GAN to generate new market scenarios to train a reinforcement learning model, a branch of ML that trains agents to perform actions in an environment to maximize a reward. In this case, the prize is the income obtained in the market (environment), with a series of buying or selling actions. Tizzano (2018) shows how the use of synthetic data helps the agent to obtain better results since the agent is more robust because it has more scenarios from where to learn.

VAEs (Variational Autoencoders) are deep generative models whose architecture is based on Autoencoders. While an Autoencoder looks for a way to represent the training data in a lower-dimensional space, with the help of two neural networks, encoder and decoder, a VAE model tries to determine the distribution from which the data come through the same architecture of two networks, encoder, and decoder. This model was proposed by Diederik and Max (2014).

Gu et al. (2019) used VAE models to generate features based on market signals and macroeconomic indicators. Next, these new features (VAE model output) are used in training an LSTM, a particular type of recurrent neural network, to generate market predictions. Finally, these predictions are used as input to trading strategies. The models implemented by Chen-Sheng Gu and his team show an accuracy of close to 83%. In this study, predictions of the S&P 500 signal are made.

RBMs (Restricted Boltzmann Machines) are models that, through a two-layer architecture, can learn the distribution of the data they are trained with. These models are based on Boltzmann machines which are computationally complex to train. However, restricting the number of connections of these models makes it possible to reduce this complexity. These models were proposed by Salakhutdinov et al. (2007).

Liang et al. (2017) use RBM models to generate new features based on a binarized market signal. These features are used for market prediction through different classifiers, such as RandomForest, SVM, and logistic regression. Comparisons between the models trained with original data and models trained with generated by the RBMs show that the performance of the models improves with the use of the new features. In this study, accuracy levels of 61% are achieved. In Li et al. (2017) these models are also used to make price predictions using the S&P 500 signal.

GTNs (Generative Teaching Networks) (Such et al., 2020) are deep generative models that can generate synthetic data that help to improve the training of predictive models with a two-network architecture (based on GANs). GTNs architecture has a generator with the task of producing synthetic data, and a network called learner is trained with generator output. During the training process, the learner will try to predict real data by training with the synthetic data produced by the generator. The failures of the learner will make the generator produce better data through the update of the weights of both networks. Due to this process, the data generated by the GTNs does not have to be similar to the original.

3. Methodology

This section will explain the notions behind the proposed trading strategy, predictive task, and generative task. Our experimentation cycle is shown in Fig. 1, where a periodic training of generative and predictive models is performed through a sliding window (Phase 1 and Phase 3). In turn, a trading strategy uses the issued predictions of the market signal (Phase 2). At each training instant, a sample price signal feeds the generative model. The predictive model trains with features from the market signal and a synthetic signal produced by the generative model. Trading strategies constantly monitor price behavior and predictions to decide when to sell or buy.

Each value and parameter used in the mechanisms presented in this section will be detailed in the experimentation section.

3.1. Trading strategy

Like many trading strategies, we devised a trading strategy based on the behavior of the price, measured as the amount of price movement, also known as momentum. Depending on the trend, buy or sell operations are carried out when the price carries a considerable momentum. Then, the trade is closed when the momentum observed at the opening changes or fades. For example, suppose in an instant it is determined that the price is rising with considerable momentum. In that case, a buy operation is performed. When such behavior is ending, that is to say, the price will stop rising or will start to fall, a sell operation is performed. Thus generating a profit since the sold operation was at a higher price than the bought operation. When the price does not have considerable momentum, trading operations are not carried out, with such market behavior would make no profit.

In this way, two situations must be considered: when a trade is opened (the price has momentum) and when it is closed (the price has lost momentum). If a high difference is achieved between the price at the entry and exit moments, ensuring the highest price in the sale operation will profit. Observe that an entry moment can be marked by a buy or a sell operation, depending on whether the price is rising or falling. Illustrative examples are given below.

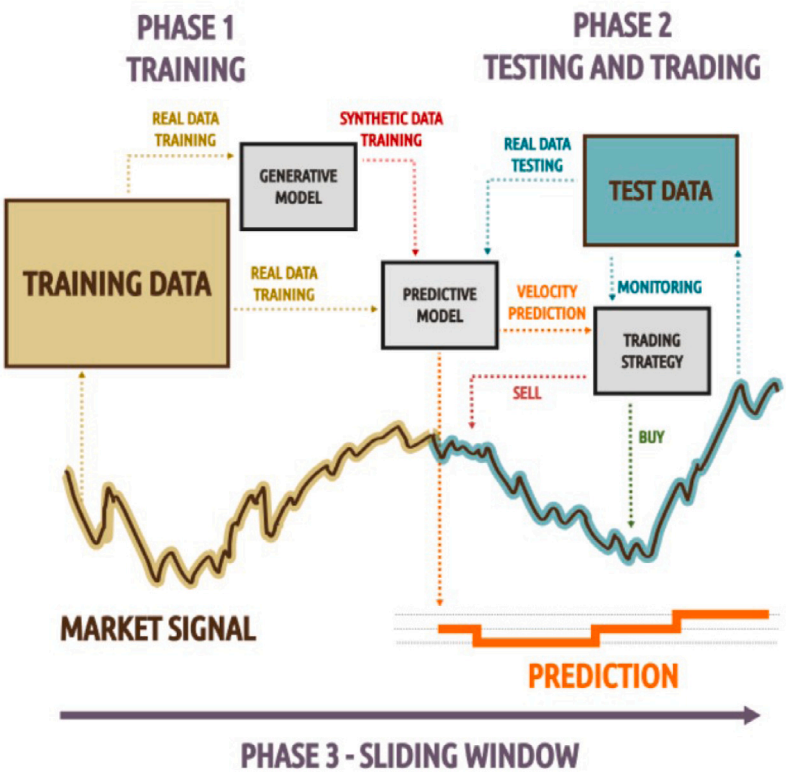


Fig. 1. Experimentation cycle. It includes synthetic data generation for predictive model training and testing in trading strategies.

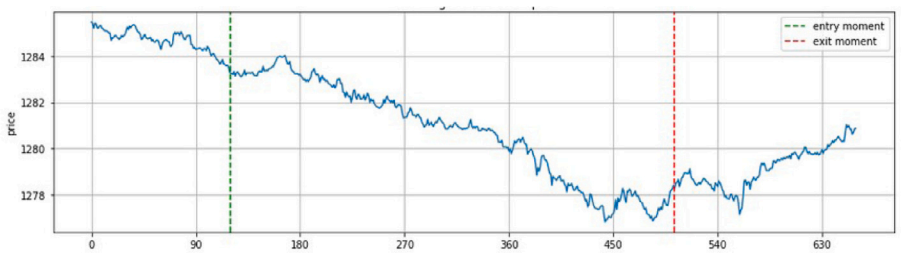


Fig. 2. Example of a trade with a downward price trend. Entry moment of the trade in momentum detection and exit moment in end-momentum detection.

Fig. 2 shows an example of a trade that is opened once a considerable downtrend price movement is detected (green line) and closed when it is determined that such behavior is lost (red line). Since it is decided to open a trade, it is assumed that the price will continue with such a trend. Thus, the operation that marks the moment of entry is a sale, thus obtaining a negative position (position to be purchased). However, when it is detected that the price behavior has changed, the position is closed with a buy trade. In this way, a profit is obtained, the selling price is higher than the buying price.

As in the previous example, Fig. 3 shows two ideal moments to perform trading operations, but this time, with an upward trend in the price. For this example, after detecting an amount of movement in the price, a purchase operation is performed. Then when it is determined that such behavior has ceased, the sale is executed, generating a profit in the same way. The above examples could ideally locate the entry and exit moments a few moments before to obtain greater profits. However, it should be considered that these trading processes are performed in real-time, and at every moment, you do not know what can happen with the price. Here it is important to be made clear that determining a good entry moment does not ensure a good exit moment since the price could have abrupt or unexpected behaviors. Determining a good entry moment is crucial since it determines an opportunity to generate profits.

This strategy was developed through a set of functions, which during the price data traversal, detects those appropriate moments to open and close a trade while keeping track of the transactions to measure the performance of each simulated strategy. It should be noted that the simulations were carried out by opening one trade at a time.

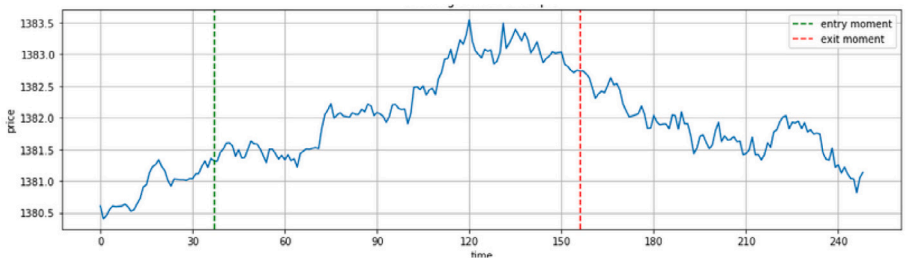


Fig. 3. Example of a trade with a rising price trend. Entry moment of the trade in momentum detection and exit moment in end-momentum detection.

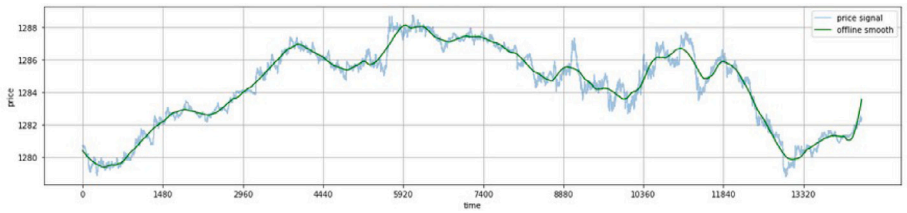


Fig. 4. Offline smooth. The smooth signal is produced by the Savitzky–Golay filter using the derivative 0 in a price signal.

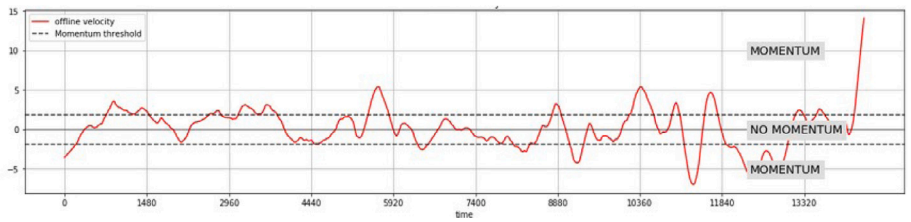


Fig. 5. Offline velocity. The velocity signal is produced by the Savitzky–Golay filter using the derivative 1 in a price signal.

3.2. Entry mechanisms

The first challenge consists of defining or quantifying this momentum. Such quantification must be robust to price volatility and follow price trends. For the detection of entry moments, a predictive task was designed and implemented to determine the price momentum at each instant through estimates of the price velocity. In other words, the notion of momentum was represented as the price velocity. For example, if a positive velocity is estimated at an instant, it is interpreted that the price is rising. Therefore, it is advisable to open a trade with a buy operation. On the other hand, if a negative velocity is estimated, it is interpreted that the price is falling. Therefore, it is advisable to open a trade with a sell operation. This predictive task was implemented using supervised learning algorithms and a cross-validation scheme for time series (Bergmeir et al., 2018).

In supervised learning, predictive tasks need to have a target to predict, in this case, which recreates the price velocity at each moment. For this, we use the Savitzky–Golay filter (Savitzky and Golay, 1964) to compute the velocity of the price signal. The Savitzky–Golay filter allows smoothing and calculating the derivatives of a signal, performing computation with data before and after each signal value. A velocity signal is obtained by calculating the price signal's first derivative using this filter since the price signal can be interpreted as a position through time. This signal will be called offline velocity since it is calculated directly over the entire price signal. Each value was estimated using previous and following data through the Savitzky–Golay filter, which can be interpreted as using data from the future and past to calculate a current value. This offline velocity is only used as a target for the predictive models.

Fig. 4 shows the result of using the Savitzky–Golay filter to smooth the price signal. Fig. 5 shows the price velocity calculated through the first derivative of the Savitzky–Golay. Fig. 4 and Fig. 5 use the XAU/USD signal sampled every 5 s.

The trading strategy is based on the existence or absence of momentum in the price. Therefore, the offline velocity signal must be transformed into a discrete signal using intervals, which mark momentum in an uptrend and a downtrend, as shown in Fig. 5. Thus, the offline velocity values were divided into three classes, the first class (−1) indicates that the price is falling, the second class (0) suggests that the price is stable. Finally, the last and third class (1) suggests that the price is rising.

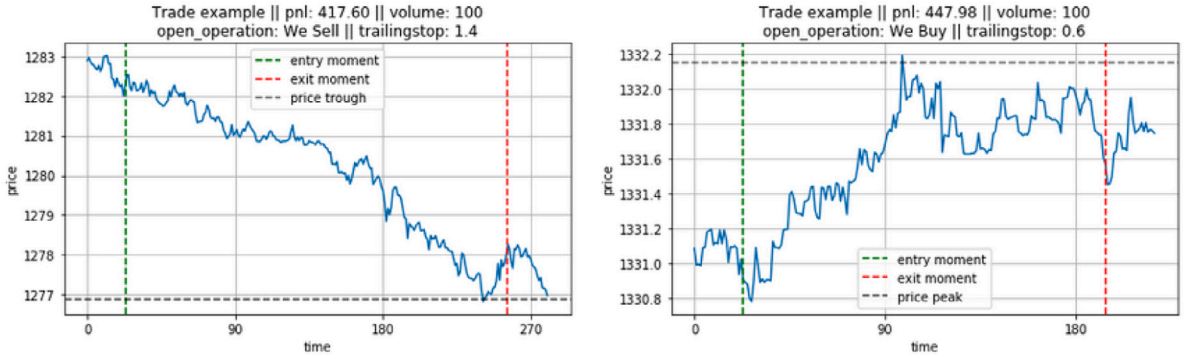


Fig. 6. Trailing stop. Examples of the trailing stop mechanism in trading operations. Each time a trade is open, a trailing stop is assigned to monitor the price and determine when to close the trade.

3.3. Exit mechanisms

The second challenge consisted in determining the appropriate moments to close the trading operations or trades. For this, we implemented price behavior monitoring mechanisms. First, we implemented a mechanism currently used in numerous trading strategies called trailing stop. However, we found that trailing stop alone might miss or falsely react to undesired price behaviors and, thus, we implemented additional monitoring mechanisms in the form for mitigations.

3.3.1. Trailing stop

Trailing stop is defined as the necessary change in the price from the last peak or valley observed to consider that the price has changed its trend. When this change is observed, the open position is closed. For example, if it is decided to obtain a position by purchasing since the price has momentum and an uptrend, as soon as the price goes down we measure the difference since the last high, and when this difference is greater than the trailing stop, the position is closed by making a sell operation. In a downtrend, the position is opened with a sell trade, and the trailing stop is measured based on the last low observed.

Fig. 6 illustrates an example of a trailing stop in an uptrend and a downtrend. The green line indicates the time at which a position is obtained by making a trade. The red line indicates the time at which the obtained position is closed with the opposite trade. The horizontal dotted black line shows the last peak or trough to which the price is compared at the time of exit moment.

Ideally, the price should have risen or fallen enough not to be affected by the difference between the closing price and the last observed peak or trough. Setting the trailing stop correctly is important, as too large a trailing stop can lead to the trade lasting too long without closing, while too small a trailing stop can lead to the trade being closed by small price volatilities.

The magnitude of the trailing stop is assigned based on the price volatility up to the moment at which a position is decided to be open. If the price has high volatility, a high tailing stop is assigned. If the price has low volatility, a low trailing stop is assigned.

3.4. Mitigations

Closing a trade based solely on the trailing stop is not a perfect methodology. Once a trailing stop is assigned to trade, the price may have behaviors that mock it. For example, if there is a rising trend at the moment of opening a trade, the price could remain in a kind of plateau with low volatility instead of continuing to rise or start to fall. This case would cause a trade to last without closing for several hours or days since it would not exceed the trailing stop at any time after that. Therefore, we developed other exit mechanisms.

3.4.1. No momentum mitigation

We look back at any time after the decision to open a trade and see if the price was trending or had very low volatility at the time of the decision. If the volatility is below a certain threshold, the trade is closed, because since there is no trend even some time after the decision, the trailing stop is not helpful.

3.4.2. Wrong prediction mitigation

Once enough time has passed since the momentum prediction that triggered an open position, we can compute the actual velocity of the price, and check whether such a decision was correct or not. If it was not, the trade is closed. It is necessary to mention that we must wait to compute the velocity of the open moment because we calculate it with the following and previous values of the market price using the Savitzky–Golay filter.

3.4.3. Hold trading mitigation

After having closed a trade, the prediction is rechecked to decide to open a new one. When using the previously mentioned mitigations, an inconvenience arises that the strategy will open more trades since the trades will last less time. We decide not to open new trades until the trailing stop of the previous trade is reached, even if the previous trade has been closed by mitigation. This allows us to control the number of trades opened during a simulation.

3.4.4. End day

During trading simulations, new trades are not open as we reach the daily closing time of the market to which the price signal belongs. Additionally a few minutes before the day ends, all trade that is open at that time will be closed.

3.4.5. Liquidity start

The idea of this mechanism is to avoid having open trades when there is no liquidity in the market. Liquidity is referred to the number of offer/bid providers available in the price books. That is when few providers are participating in it. That usually occurs at the opening moments of each market.

3.5. Cross-validation scheme

In the cross-validation scheme for time series data, predictions are periodically performed by simulating a real-time execution. During this simulation, we periodically train the model with the most recent information avoiding the influence of the older information. With this in mind, each simulation is governed by a set of parameters:

- *model*: Predictive algorithm to build the models to be used throughout the simulation.
- *sample_freq*: Sampling frequency of the signals used. Each signal can have data with an irregular frequency.
- *train_period*: The size of the historical price signal needed to train a model at any given moment.
- *test_period*: Period during which a model will make predictions until a new model is retrained.

The cross-validation scheme for time series, as shown in Fig. 7, consists of using a sliding window to train and test the predictive model. At each window position, part of the data is used to train the model (training dataset) and another part is used to test the model (test dataset). Each position of the sliding window is called a fold. The purpose of using the sliding window in testing predictive models is to simulate real time, because in a production environment it would be desirable to have a model that has been trained with the latest data and retrain it regularly. In the cross-validation scheme for time series, the sliding window should move according to the time series used to avoid using future information for prediction. This window slides every *test_period*, i.e., every *test_period*, a new predictive model is trained using a historical price signal of a *train_period* size and makes the corresponding predictions for that *test_period*. This way, we will have velocity predictions each instant with a trained model for each *test_period* with the latest *train_period* data from the market. It should be made clear that the data in the time series corresponding to the *train_period* (training dataset) and *test_period* (test dataset) do not overlap within the same fold.

It is worth mentioning that the *sample_freq*, *train_period*, and the set of features to be used determine the size of the dataset to be used to train each predictive model.

For training each model, we use a set of features of various price-derived signals. This includes smoothed price signals or price velocity signals. These signals are generated using the Savitzky–Golay filter, which simulates real time through sliding windows, avoiding the use of forward-looking data. It should be noted, however, that the signals calculated in real time differ from the actual price behavior since we cannot use future data.

3.6. Generating price data

We use a specific period of the real data as the source to train generative models. This ensures that the generative process has a chance to observe and characterize the real signal, with the hope to produce synthetic data with similar features.

There are many deep generative models (DGM), such as GAN, RBM, VAE, or GTN. Through neural network architectures, these models aim to generate synthetic data that are generally used in the training of predictive models in tasks in which there is not enough data. In the case of this work, the fact that we must use a windowing validation methodology for time series limits the use of the available data.

Computing velocity estimations requires the calculation of real-time velocities of the price signal. Therefore, the generated signal must also allow such measures since each DGM is trained to produce price signals. It is worth mentioning that synthetic and real data must be in the same format used in the predictive models. Also, each predictive model to train must have the possibility of accessing a broad and sufficient amount of synthetic data. Each time a DGM is trained, it must produce a considerable amount of data, and the necessary processing for velocity estimation must be performed. When a predictive model is used, it simply accesses the most recently generated synthetic data. For the production of synthetic data, a parameter called *generative_period* was defined, which indicates the training period of each generative model and, in turn, the valid period for using a synthetic dataset.

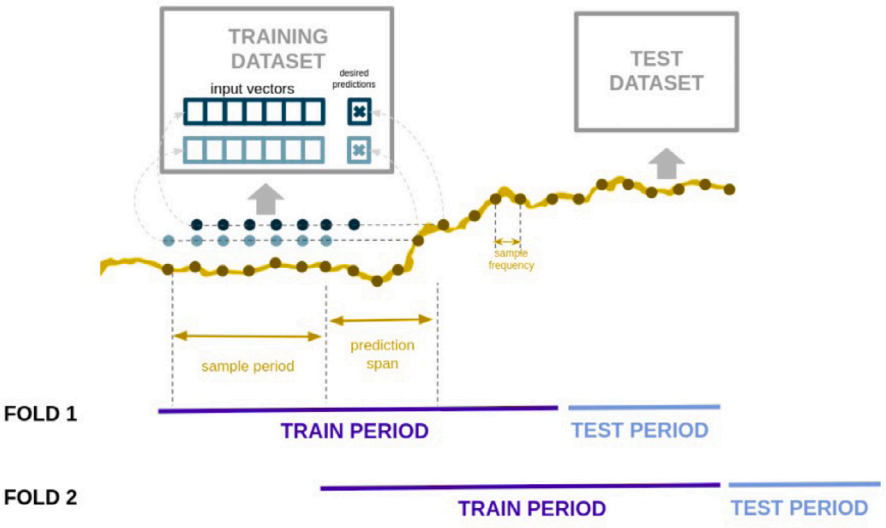


Fig. 7. Cross-validation scheme. Cross-validation scheme for time series used in predictive and generative tasks. Each test period, a new model is trained with a signal history with a *train_period* size, and predictions are made for the indicated *test_period*.

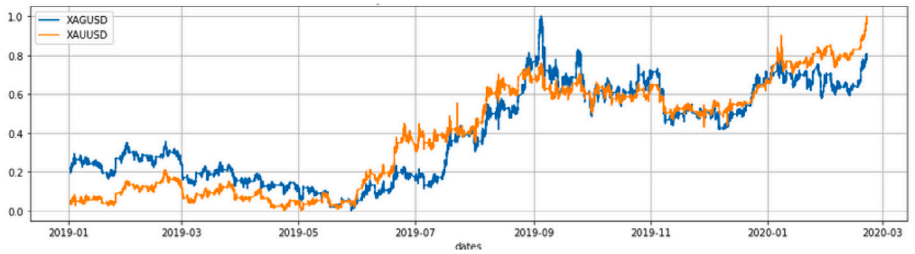


Fig. 8. Precious metal markets. Price signals for XAG and XAU. It can be seen how the price of XAU and XAG have similar behavior with very similar trends. Therefore, we decided to use only one of the two signals for experimentation.

4. Experimentation

To perform all the experimentation mentioned below and in the following sections, several functions and procedures were developed in the Python 3 programming language, using libraries such as Pandas, NumPy, Scikit-learn, TensorFlow, and others.

We tested the predictive task, synthetic data generation, and trading strategy simulations mentioned above with the price signals of the XAU/USD and EUR/USD markets. Data was obtained from `tickstory`,[1] a desktop software to download historical trading data. For the XAU/USD (gold-dollar) market signal, we used data between 2019-01-01 and 2020-02-22. For the EUR/USD (euro-dollar) market signal, we use data between 2020-05-01 and 2021-04-01. We decided to work with the signal sampled every 5 s since we designed the proposed strategy to work in an intraday trading environment. After taking the signal sampled every 5 s, we eliminate those outliers present in the price distribution of the whole signal. Thus, predictive, generative models and trading strategies make use of these signals sampled every 5 s. It is worth mentioning that the development of the article was carried out mainly with the XAU/USD market. Experimentation with the EUR/USD market was carried out later as a way of validating our approach. Fig. 8 shows the XAU/USD signal for the selected period, and Fig. 9 shows the EUR/USD signal.

XAU/USD and EUR/USD signals were selected by their availability in the data source, the high frequency of the data, and their change to the dollar, the latter to have a clearer view of the performance of the trading strategies. It is also considered that the selected markets are representative compared to other markets available in the mentioned data source, as shown in Figs. 8 and 9. Here other signals in the same markets show similar behaviors. Finally, it is worth noting that for the experimentation, we take signal data sampled every 5 s.

We use the XAU/USD signal to illustrate out implementation of the proposed methodology, and in the results section below you will find both the XAU/USD and EUR/USD comparative results.

[1] https://tickstory.com/

Fig. 9. Forex markets. Price signals for foreign exchange markets. It can be seen how the price of EUR, GBP, and NZD have similar behavior with very similar trends. Therefore, we decided to use only one of the two signals for experimentation.

Table 1
Class distribution for velocity estimation XAU/USD. The proposed velocity prediction is defined as a 3-class classification task. −1 refers to high and negative velocity values, 1 to high and positive velocity values, and 0 to low-velocity values. It is at high-velocity values that trading operations are proposed.

Class −1	Class 0	Class 1
18%	62%	20%

Table 2
Savitzky–Golay filter configurations. Set of Savitzky–Golay filter parameters used to construct the online velocities required by the predictive model. Two online velocity features are used for training and prediction. *window_length* and *polyorder* are parameters for using the Savitzky–Golay filter algorithm.

window_length	polyorder
721	3
1441	3

We performed a wide range of experiments considering several options and values for most the configuration parameters described below, with the intent to cover the most reasonable choices given the amount of computing resources at hand. We herewith provide details of the best choices leading to the results described in Section 5.

4.1. Velocity estimation

The Savitzky–Golay filter was used for offline velocity with a window size of 841 values for the entire 14-month price signal. The intervals to determine the classes to predict were (−inf, −1.55), [−1.55, 1.55], (1.55, inf). These intervals were selected, obtaining the percentile 62 of the absolute value of the offline velocity. Thus the target to be predicted has the distribution of classes shown in Table 1.

The real-time calculated velocities mentioned above and the features with which the trained models are derived correspond to signals calculated through sliding windows on the price signal. This sliding window method is used to simulate a real-time approach. In each window, a Savitzky–Golay filter is used, taking the last value of the generated signal. The first derivative delivered by the filter is interpreted as the velocity and the 0s derivative as the smoothed signal. Each of the velocities used in the experiment was calculated using the values shown in Table 2 for the Savitzky–Golay filter parameters. To capture short and long-term trends of the price signal were selected these two values.

To train models, a dataset was built using five signals: the price signal itself, two smoothed signals, and two real-time velocity signals. This dataset uses the range and standard deviation for 30 s, 1 min, and 1 h of the five signals. It also takes the differences of each smoothed signal with the first and second previous moment for periods of 1 min, i.e., the differenced signal. Each 1-minute interval, the values are transformed into −1, 0, or 1, based on the offline velocity intervals for one of the real-time velocities. Then, these values are added as other features.

The different parameters used in the experimentation were selected after trying out several different configurations and selecting the best performing one.

4.2. Mitigations

The trailing stop is computed at the beginning of each trade by using the percentile divided by 10 of the spread value at the open instant from the historical spread signal. We selected the parameters of each mitigation to give a chance for the trailing stop to be executed.

4.2.1. No momentum mitigation

This mitigation is used 15 min after opening the trade and closes those trades whose standard deviation of 10 min before entry is less than 0.25.

Table 3
Deep generative models dataset parametrization. Set of parameters to build the dataset with which the generative models are trained. *sample_freq* is the sampling frequency of the market signal, *sample_period* is the period of data used in each row of the dataset, *overlap_spam* is the data overlap between each row of the dataset, and *rescale_rows* is the scaling technique of the data in each row of the dataset.

Parameter	Value
sample freq	5 s
sample period	1 h
overlap spam	1 min
rescale rows	minmax

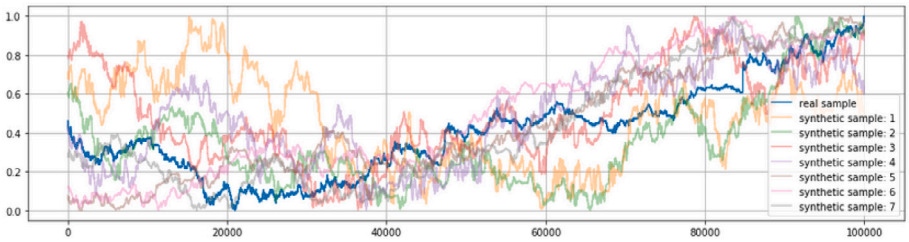


Fig. 10. Synthetic and real data samples. Set of synthetic samples from the XAU/USD market and a sample of real data from the same market.

4.2.2. Wrong prediction mitigation

The price signal is sampled every 5 s. The Savitzky–Golay filter is used with a value of 361, which means 15 min of previous and following data was necessary to estimate each offline velocity value. Thus, during the simulation, each time a trade is opened, it is required to wait 15 min from opening the trade to determine the offline velocity value for that moment.

4.3. Synthetic data

The use of deep generative models involves a lot of work. It is not enough to implement a deep generative model and make use of its data. You must ensure that the model has a good training process since using data produced by poorly trained models would only add noise to the predictions and trading strategies. In the same way, it is expected that the generated data present similarities with the real data.

Therefore, we carried out a series of experiments to verify the stability of the training process and the level of similarity produced between synthetic and real data for different generative models implemented. In the trading simulations, the data produced by the model that presented the best results in the experimentation mentioned above were used. The experiment consisted of making synthetic price data using the first 15 days of August 2019. Then, the dataset to train each DGM was built using the real price signal with the parameters shown in Table 3.

The *overlap_spam* parameter refers to the overlapping period between each row in the dataset. The *rescale_rows* parameter refers to the scaling of the data in the dataset. In this case, each row is scaled between values of 0 and 1 using the minimum and maximum of each row. Finally, the *sample_period* parameter refers to the sample size of the signal used in each dataset row.

Each DGM produces enough data to be compared with the 15 days mentioned above of real data. In other words, each model generated 15 days of synthetic data. It should say that the output of the generative model has the same format as the input with which it is trained, i.e., a matrix with the parameters in the table above. Therefore, once the data is generated, a process is done to obtain a single signal and compare it with the real price signal. That process consists of concatenating each row of the matrix using the differences between the values with the following value. It is necessary to take into account that there is overlapping between the rows.

Once the signal produced by each model is obtained, the similarity with the real data is determined. Such similarity is measured using the Wasserstein distance (Earth mover's distance) (Levina and Bickel, 2001), which is a distance function between probability distributions that considers the minimum cost of converting one distribution into another as if they were two piles of soil. What we compared is the difference in distribution of each signal time with a previous period, in other words, the distributions of the once-differenced signals. Fig. 10 shows examples of synthetic signals produced by various generative models.

Around 130 deep generative models were implemented for the experimentation described above. These models were based on GAN and VAE architectures, varying their complexity using different layers, such as dense, convolutional, dropout, among others, and modifying the number of layers and nodes per layer. The models that showed the best performance are in Table 4. The architecture for the best model is shown in Table 5.

Specifically, a value of 15 days was used as the *generative_period* for the synthetic data generation tasks. It means that the DGM was trained every 15 days of simulation with 15 days of price data. During 15 days of simulation, the predictive model will use synthetic data from the same deep generative model. The deep generative model selected for this task was the GAN-Dense. Each

Table 4
Generative models experiment. Performance of the four best generative models implemented and tested. The performance of each model was measured using the Wasserstein distance metric, which indicates the similarity between two distributions. The lower the value of the Wasserstein distance metric, the more similar the synthetic data produced by each generative model is compared to the real data. The architecture of each model is displayed in Table 5.

Architecture	Layers	Wasserstein
VAE	Dense	1,801
GAN	Conv1D	2.155
GAN	Dense	**1,355**
VAE	Conv1D	23,220

Table 5
Model architecture of the GAN Dense model which is the best generative models implemented and tested. We used the Adam (Kingma and Ba, 2014) optimizer with a learning rate of 10^{-3}. No early stopping was used.

Discriminator	Generator
Dense (256 units)	Dense (128 units)
LeakyReLU	Dropout (rate=0.5)
Dropout (rate=0.5)	ReLU
Dense (256 units)	Dense (512 units)
LeakyReLU	ReLU
Dropout (rate=0.5)	

experiment was conducted with epochs parameters of 200 and *batch_size* of 10. We use Eq. (1) (binary cross-entropy) as loss function for the GAN-Dense model.

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\ \hat{y}_i + (1 - y_i) \cdot \log\,(1 - \hat{y}_i) \tag{1}$$

In general, the use of synthetic data in predictive models consists in training a model with real data and an extra amount of synthetic data. That will produce different predictions depending on the percentage of real and synthetic data in each experiment. Then, these predictions are used in trading strategies, each generating profits or losses based on the decisions made by observing each prediction made.

5. Results and discussion

With everything mentioned so far, several factors could affect the behavior of trading strategies and predictions. One of them, and the motivation of this work, is the amount of data available in each training, either real or synthetic data. In any work, not having a large amount of real data and counting with synthetic data could be a great advantage. One of our goals is to understand how trading strategies and predictions are affected by the variability of synthetic and real data.

To this end, several trading strategy simulations were performed using predictive models trained with different amounts of synthetic and real data. This experimentation was accompanied by graphs that could give a global view of the impact of synthetic and real data on trading strategies. When training any predictive model, we use different proportions of real and synthetic data. The amount of real data was denoted as *train_period*, while synthetic data was denoted as *synthetic_size*. The *train_period* is given in time units since the real data is associated with a time instant. At the same time, *synthetic_size* represents the number of new rows added to the dataset from the signal produced by the deep generative model, on the other hand, we could express the *train_period* in the number of rows. We know each row in the dataset is sliding 5 s of the next one. It is worth mentioning that the predictions made by the predictive models, which will later be used in trading strategies, are made using only the real price data. In the first step, the predictions were made using the parametrization shown in Table 6.

The RandomForest model used had a *max_depth* of 4, 30 estimators and the balanced option on class weights to prevent the model from being affected because it is trained with an unbalanced dataset. The model parameters and *test_period* were selected through parameter tuning tests. The results of the mentioned experimentation are in Tables 7 and 8

As can be observed with the increase of real data, the predictive models show better results, which is expected. On the other hand, in the use of synthetic data, it can be observed how the general performance of the models worsens (accuracy in the test set). However, when observing the performance by classes, there is a better performance for classes 1 and −1, which indicates a strong trend in the price, generating a need to execute trading strategies that make use of these predictions. This is actually desirable, meaning that synthetic data helps us have better predictions when there is a chance to profit from market changes. Observe as well that overfitting smoothly decreases with the increase of real and synthetic data.

Before continuing with the simulation of trading strategies and going deeper into the exploration of the prediction experiments, we found that the probabilities delivered by the models for each prediction in each experiment also show different behaviors. Each trained model provides a list of probabilities when it makes a prediction corresponding to the confidence that the model deposits

Table 6
Velocity estimation experiments. Set of parameters used with predictive models trained with real and synthetic data for price velocity estimation. *model* refers to the model used. *sample_freq* to the sampling done on the price data. *train_period* to the period of data each model is trained with. *test_period* to the period for which a trained model will make predictions before being retrained. *synthetic_size* to the number of new rows of synthetic data added to the dataset to train each predictive model.

Parameter	Values
model	RandomForestClassifier
sample_freq	5 s
test_period	10 min
train_period	1 h, 3 h, 10 h
synthetic_size	0, 14000, 21000

Table 7
Velocity estimation results (Accuracy) in XAU/USD. Accuracy for velocity estimation by varying the *train_period* and *synthetic_size* parameters in the test set on real data. *pct_syntetic* refers to the percentage representation of the synthetic data in the dataset built. The accuracy is always on real data.

train_period	real_size	synthetic_size	pct_synthetic	accuracy_train	accuracy_test
1 h	720	0	0,0%	0,9999	0,495
1 h	720	14000	95,1%	0,9287	0,367
1 h	720	21000	96,6%	0,8768	0,348
3 h	2160	0	0,0%	0,9733	0,557
3 h	2160	14000	86,6%	0,8700	0,527
3 h	2160	21000	90,6%	0,8409	0,518
10 h	7200	0	0,0%	0,8314	**0,590**
10 h	7200	14000	66,0%	0,7542	**0,575**
10h	7200	21000	74,4%	0,7414	**0,577**

Table 8
Velocity estimation results (Accuracy per class) in XAU/USD. Accuracy per class for velocity estimation by varying the *train_period* and *synthetic_size* parameters in the test set on real data. With the increase of real and synthetic data, the accuracy per class increases while the overfitting decreases slightly. *pct_syntetic* refers to the percentage representation of the synthetic data in the dataset built. The accuracy is always on real data.

Train period	Real size	Synthetic size	pct_synthetic	Accuracy class −1	Accuracy class 0	Accuracy class 1
1h	720	0	0,0%	0,243	0,648	0,260
1h	720	14000	95,1%	0,208	0,453	0,252
1h	720	21000	96,6%	0,207	0,425	0,244
3h	2160	0	0,0%	0,303	0,714	0,313
3h	2160	14000	86,6%	0,369	0,613	0,409
3h	2160	21000	90,6%	0,373	0,594	0,419
10h	7200	0	0,0%	**0,468**	**0,663**	**0,478**
10h	7200	14000	66,0%	**0,519**	**0,597**	**0,561**
10h	7200	21000	74,4%	**0,523**	**0,597**	**0,565**

on each class prediction. The class with the highest probability is the one assigned as the definitive prediction for that instant. We define the *proba_diff* by taking the difference between the highest value and the summation of the other ones. This indicator helps us determine the overall certainty that the model gives to the predictions. Other studies such as Omid et al. (2019) show more complex mechanisms to determine the certainty of a predictive model. Our approach was simple and enough for us to observe behaviors from which to take advantage. Fig. 11 shows the distribution of *proba_diff* for each of the experiments. *proba_diff* is interpreted as the reliability given for the trading strategy to each prediction made by the predictive models.

There is an advantage in the models that do not use synthetic data since they show a higher certainty in their predictions. But, how is the distribution if we also discriminate by class? In Fig. 12 we observe the distributions of the experiments with a *train_period* of 10 h.

Although there is less certainty in the predictions in general, there is some behavior in synthetic data that could favor trading strategies. First, it should be made clear that making erroneous predictions of classes 1 or −1 in trading strategies generates a greater risk than erroneously predicting class 0. That is because opening a trade at the wrong time could generate losses, while not opening it when we should can be seen as a missed opportunity but not a loss. As shown in Fig. 12, if discrimination is made in the predictions based on the *proba_diff*, erroneous predictions can be avoided to a greater extent. Due to this, we decided to endow trading strategies with the capability to determine the validity of each prediction based on the probability given by the model. In other words, every time a trading strategy uses a prediction to decide whether or not to open a trade, it can discard that prediction if it considers that the probability provided by the model is too small. If so, the prediction is discarded, and the trade is not opened. Several strategies were run with different levels of validation for the predictions or, in other words, different probability thresholds. Trading simulations were performed for each prediction experiment shown above by varying the probability threshold between the values in Table 9.

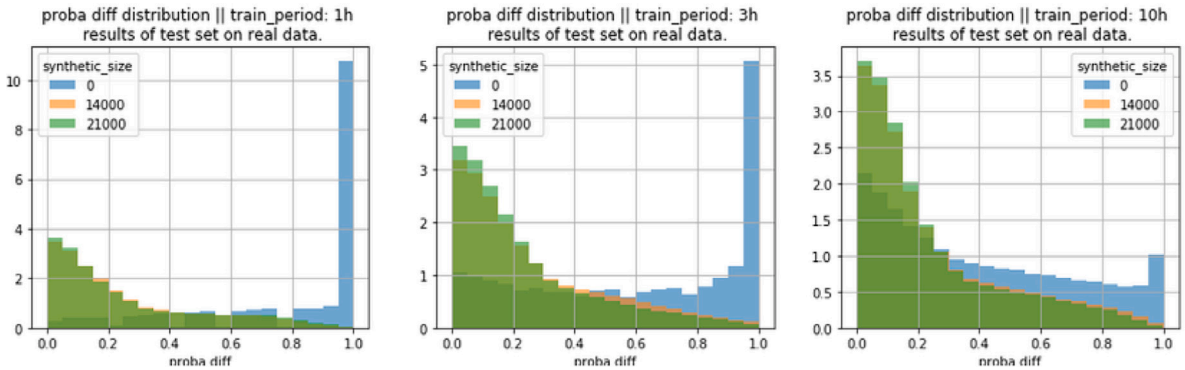


Fig. 11. *proba_diff* distribution in XAU/USD. *proba_diff* distribution for the different prediction experiments in the test set on real data. The three plots can be observed how the distributions of the predictions that do not make use of synthetic data have more population around 1. It means an improved certainty on those predictions. Target and prediction discrimination of 10 h experiments are detailed in Fig. 12.

Table 9
proba_diff thresholds. Thresholds based on the probabilities delivered by the model for each prediction, used in trading strategies to determine which estimates to discard.

proba diff
0
0,7
0,9

Table 10
Class distribution for velocity estimation EUR/USD. The proposed velocity prediction is defined as a 3-class classification task. −1 refers to high and negative velocity values, 1 to high and positive velocity values, and 0 to low-velocity values. It is at high-velocity values that trading operations are proposed.

Class −1	Class 0	Class 1
22%	55%	23%

A threshold of 0 means that all predictions are considered valid. While 0.9 means that only predictions with large confidence (large *proba_diff*) are considered valid. We obtained the results in Fig. 13 executing these strategies for the entire period of data and comparing profits through the Return on Investment (ROI) (Phillips and Phillips, 2009) metric.

As can be seen in Fig. 13, the different behaviors of the predictive models caused by synthetic data impact the trading strategies. Trading strategies using more synthetic data, produce better results. The ROI determines the efficiency of each strategy through the relative difference between the money obtained and delivered in each trade. Higher values mean better performance. This improvement in the performance of trading strategies is mainly observed using a threshold of 0.9 in the *proba_diff* of the predictions (Fig. 13(c)). Also, it can be observed how, with the increase of synthetic data in the predictive models, the strategies have better efficiency, thus demonstrating the impact of using synthetic data produced by deep generative models in predictive tasks in trading strategies.

5.1. Experimentation in the EUR/USD market

As mentioned above, the results presented so far were obtained using the XAU/USD market price signal as a basis. The results of applying the methodology proposed in this article using the EUR/USD market price signal are below. It is worth mentioning that the predictive models, deep generative models, and the parametrization used for this experimentation, are the same as described in XAU/USD experimentation. The only difference is the signal data used to feed each model since the real-time and offline velocities are built from the EUR/USD signal and not XAU/USD. This means that the interval selected for the offline velocity also changes. In this case the interval (−inf, −0.001), [−0.001, 0.001], (0.001, inf) was used. These intervals also were computed obtaining the percentile 62 of the absolute value of the offline velocity. Thus the target to be predicted has the distribution of classes shown in Table 10.

The predictions made in this market yield the results in Tables 11 and 12. As can be seen in Tables 11 and 12, the predictions show similar behavior to that observed in the XAU/USD market predictions, i.e., the use of synthetic data increases the performance per class and reduces the overfitting of the models to a small extent.

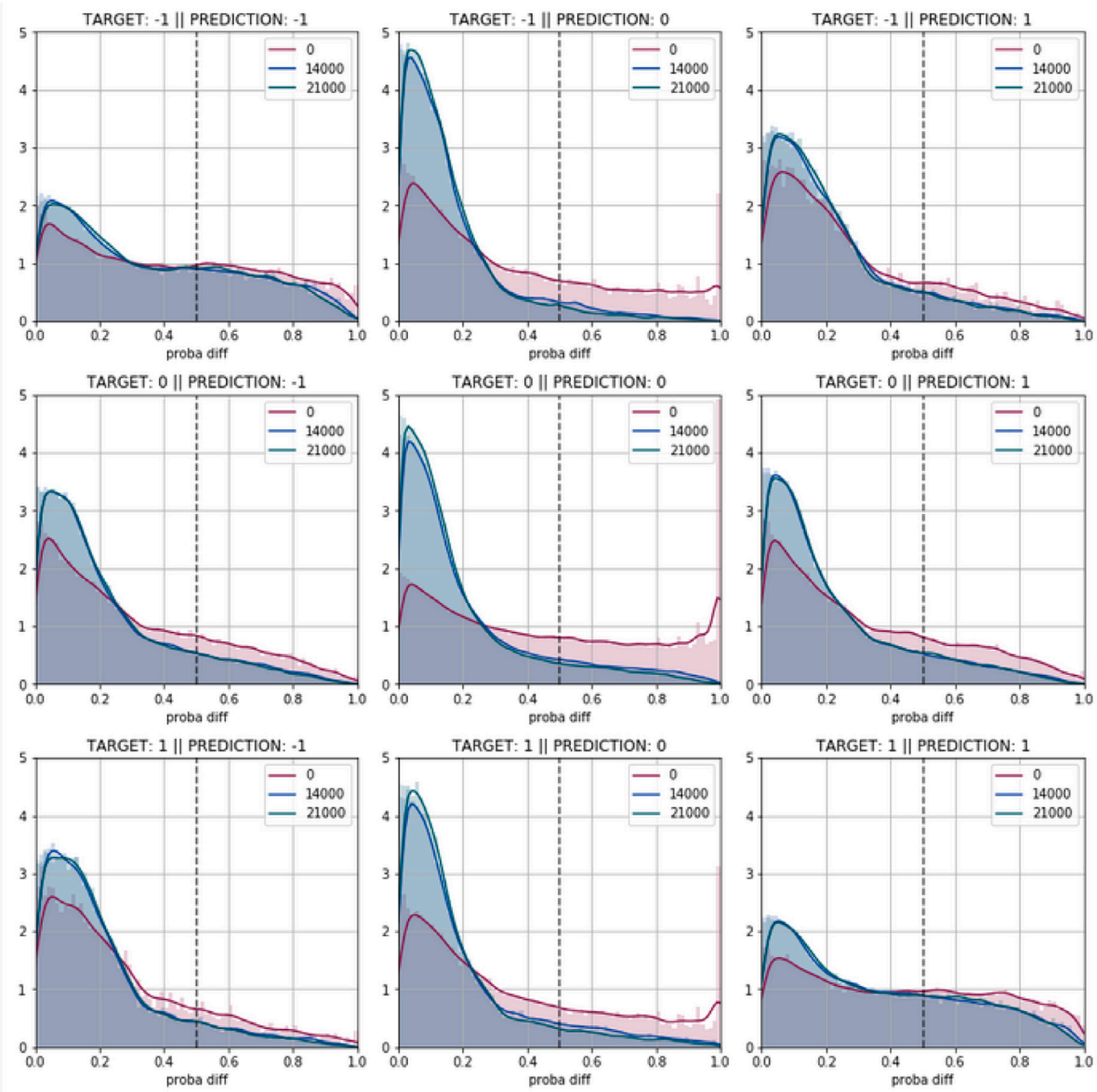
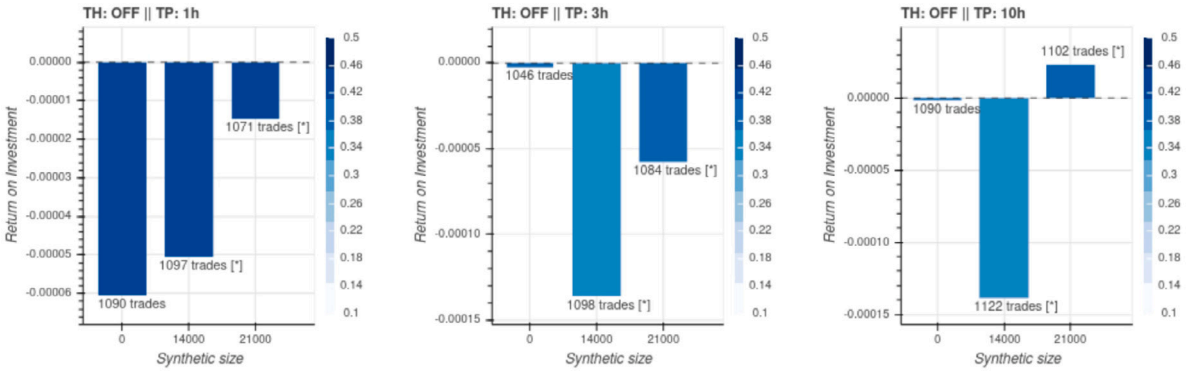


Fig. 12. Discriminated *proba_diff* distribution in XAU/USD. *proba_diff* distribution distinguished by prediction and target combinations for each experiment in the test set on real data. The predictions with the same value show less population in erroneous estimates while the *proba_diff* value increases.

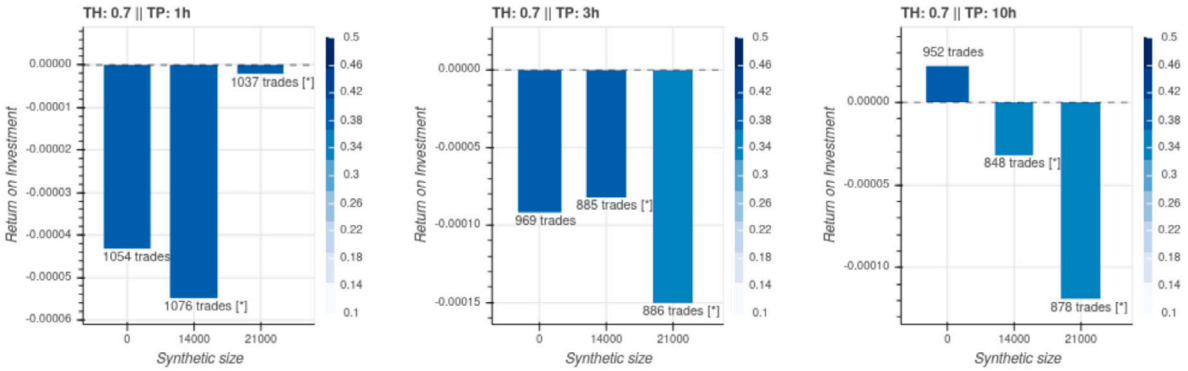
In the same way, similar behavior is observed in the probabilities delivered for each of the predictions (Figs. 14 and 15). Still, this time there is a tiny population of predictions with a *proba_diff* higher than 0.6, so the trading simulations for this market were performed using smaller thresholds than those used in the trading strategies of the XAU/USD market. Trading simulations on the EUR/USD market show the results in Fig. 16.

Finally, the trading strategies using the EUR/USD signal present a similar behavior to that obtained in the experimentation with the XAU/USD signal. The use of a larger amount of real and synthetic data and the application of a threshold for the predictions positively impact the performance of the strategies. It can be observed through the ROI metric comparing Figs. 13 and 16.

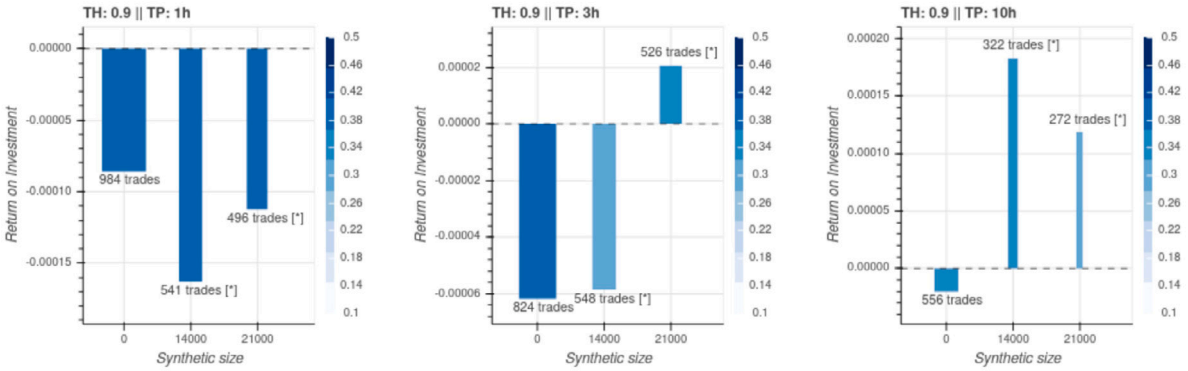
Thanks to the simulation and cross-validation mechanism we used, it is possible to take the whole experimentation cycle or some of its parts to a production environment. For example, in a market data stream environment, a system could be implemented that collects the market price at each instant and, in turn, builds each of the signals and datasets required for the proposed task since the only data needed for the whole experimentation are the market prices. Once datasets or signals of the necessary size to train predictive models have been built, one could start making predictions at every instant while collecting data to keep the datasets and signals updated with the most recent data. These velocity predictions could very well be used as a trading signal or indicator in trading strategies other than the one developed in the methodology. It should be made clear that each of the proposed mechanisms



(a) ROI for trading strategies that make use of a threshold of 0.



(b) ROI for trading strategies that make use of a threshold of 0.7.



(c) ROI for trading strategies that make use of a threshold of 0.9.

Fig. 13. Trading strategies result in XAU/USD. Results of trading strategy simulations measured as Return on Investment. The strategies make use of different amounts of synthetic data produced by deep generative models. Results are for data between 2019-01-01 and 2020-02-22. The color of the bars represents the percentage of positive trades and the width, the number of trades. TP means *train_period*, and TH means threshold. [*] signals strategies using synthetic data. Observe how in the majority of the cases, synthetic data improves ROI, either by losing less money or by producing benefits (subplot (c), center and right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

or methods, such as exit mechanisms in trading strategies, could also be implemented to different trading strategies used today, especially those executed by automatic high-frequency trading systems. As presented in this article, the proposed classification task could be taken to a regression task to estimate the velocity value and not leave it between a couple of intervals, allowing more accurate uses of velocity as a market indicator.

As mentioned, several papers perform predictions on market price signals and elaborate tests using different trading strategies, such as Fabbri. and Moro. (2018) or Yong'an et al. (2020). However, making a direct comparison with these or other studies

Table 11
Velocity estimation results (Accuracy) in EUR/USD. Accuracy for velocity estimation by varying the *train_period* and *synthetic_size* parameters in the test set on real data. *pct_syntetic* refers to the percentage representation of the synthetic data in the dataset built. The accuracy is always on real data.

train_period	real_size	synthetic_size	pct_synthetic	accuracy_train	accuracy_test
1h	720	0	0,0%	0,998	0,508
1h	720	14000	95,1%	0,852	0,438
1h	720	21000	96,6%	0,826	0,436
3h	2160	0	0,0%	0,963	0,470
3h	2160	14000	86,6%	0,688	0,428
3h	2160	21000	90,6%	0,660	0,427
10h	7200	0	0,0%	0,826	**0,406**
10h	7200	14000	66,0%	0,584	**0,405**
10h	7200	21000	74,4%	0,565	**0,415**

Table 12
Velocity estimation results (Accuracy per class) in EUR/USD. Accuracy per class for velocity estimation by varying the *train_period* and *synthetic_size* parameters in the test set on real data. With the increase of real and synthetic data, the accuracy per class increases while the overfitting decreases slightly. *pct_syntetic* refers to the percentage representation of the synthetic data in the dataset built. The accuracy is always on real data.

train period	real size	synthetic size	pct_synthetic	accuracy class −1	accuracy class 0	accuracy class 1
1h	720	0	0,0%	0,269	0,545	0,260
1h	720	14000	95,1%	0,263	0,545	0,261
1h	720	21000	96,6%	0,246	0,579	0,245
3h	2160	0	0,0%	0,334	0,600	0,340
3h	2160	14000	86,6%	0,432	0,423	0,434
3h	2160	21000	90,6%	0,407	0,443	0,413
10h	7200	0	0,0%	**0,454**	**0,550**	**0,475**
10h	7200	14000	66,0%	**0,566**	**0,311**	**0,573**
10h	7200	21000	74,4%	**0,545**	**0,322**	**0,561**

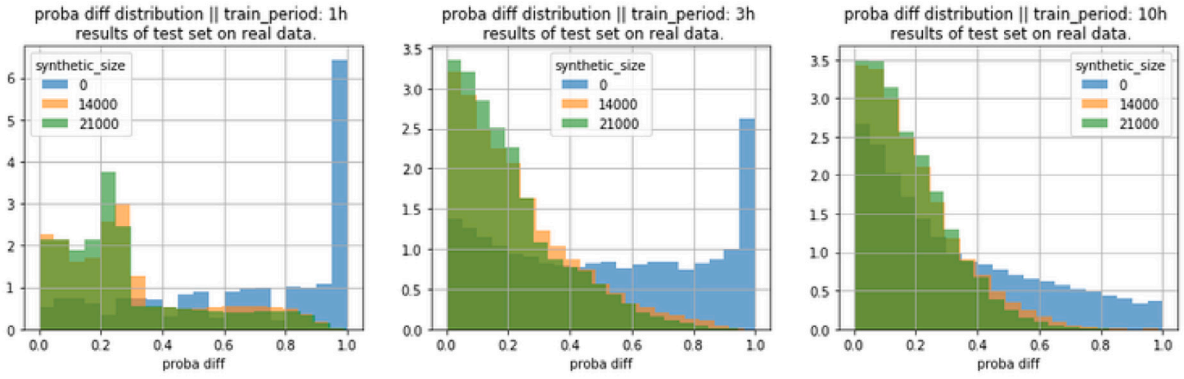


Fig. 14. *proba_diff* distribution in EUR/USD. Distribution of *proba_diff* for the different prediction experiments on the test set of real data for the EURUSD market. The three plots can be observed how the distributions of the predictions that do not make use of synthetic data have more population around 1. It means an improved certainty on those models. Target and prediction discrimination of 10 h experiments are detailed in Fig. 15.

of predictions in financial series is not straightforward, since trading instruments and frequencies vary largely. In our approach, we made design choices to avoid overengineering an already complex pipeline, following in Liu et al. (2019) or Nobre and Neves (2019). We obtain accuracy levels of around 59% with simple models that can be taken to production and work in an intraday trading environment. Other studies such as Wu et al. (2021) or Nti et al. (2021) present better performances, 82% and 98%, respectively, but use very complex models such as LSTM or CNN, which could represent a higher cost when taken to production environments, as their training processes are usually more complex. Apart from this, they do inter-day trading, which beholds a different nature than the higher frequency intra-day trading that we do. It is worth highlighting that we present better performance results than Moews et al. (2019) or Atkins et al. (2018). Again, these studies are performed on different predictive tasks than the one proposed in the present work, which consists of price velocity prediction. On the other hand, because our trading signal is different, we take advantage and design a trading strategy different from the classical buy-and-hold that is usually used in this type of study, as Masafumi et al. (2018) or Marian (2019). Other comparisons can be found in Table 13.

Table 13
Related works in similar tasks. We searched articles implementing generation of synthetic data and/or market prediction and/or trading strategies. Observe that most works address only one or two tasks (generation, prediction, trading). In our work we balance choices in all three tasks to get consistent end-to-end metrics end results. A dash (-) marks tasks not addressed by each reference.

Reference	Financial instrument	Generation synthetic data	Prediction method	Prediction performance	Prediction task	Time series cross-validation	Prediction spam	Trading strategy	Trading results	Trading
Takahashi et al. (2019)	S&P500	FIN-GAN	–	–	–	–	–	–	–	–
Fu et al. (2020)	JPM	cGAN	–	–	–	–	–	–	–	–
Zhang and Ci (2020)	XAU/USD	–	DBN	(MAE) 0.0460	Price prediction	No	1 month	–	–	–
Shao et al. (2021)	PJM electricity market	–	BiLSTM	(MAPE) 0.025	Price prediction	No	1 hour	–	–	–
Wei and Nguyen (2020)	CRSP database	–	BERT	(ACC) 58.4%	Price trend prediction	No	1 day	–	–	–
Elberawi and Belal (2021)	XAU/USD	–	LSTM	(MAPE) 1.86	Price prediction	No	1 day	–	–	–
Sanboon et al. (2019)	SET50	–	LSTM	(ACC) 99.31%	Buy–Sell recommendation prediction	No	90 days	–	–	–
Julisa (2021)	FB stock price	GAN	CNN + LSTM	(MAE) 2.2497	Price prediction	No	1 day	–	–	–
Vargas et al. (2018)	CVX stock price	–	SI-RCNN	(ACC) 56.84%	Price trend prediction	No	1 day	Buy when model say price growing up. Otherwise sell	(Return) 296.70USD	1 day
Chakole et al. (2021)	Indian stock markets	–	–	–	–	–	–	Reinforcement Learning	(%AR) 150.92	1 day
Fister et al. (2021)	DAX30	–	LSTM	–	Buy–Sell recommendation prediction	Yes	1 day	Buy when model say buy. Otherwise sell	(Portfolio value) 2450.76EUR	1 day
Koshiyama et al. (2021)	S&P 500	cGAN	MLP	–	returns prediction	Yes	–	–	(Sharpe ratio) 0.097576	–
Alberto et al. (2020)	NASDAQ	–	Genetic Algorithms	–	MACD estimation	No	0 min	Buy and hold	(Effective annual rate) 19.5%	1 day
Shintate and Pichl (2019)	Bitcoin	–	MLP	(ACC) 62.64%	Price trend prediction	No	30 min	Buy and hold	(log return) 1.3346	1 min
Our	XAU/USD	GAN	RandomForest	(ACC) 59%	Velocity estimation	Yes	0 min	Buy in high positive velocity values and sell in high negative velocity values	(ROI) 0.000119	5 s

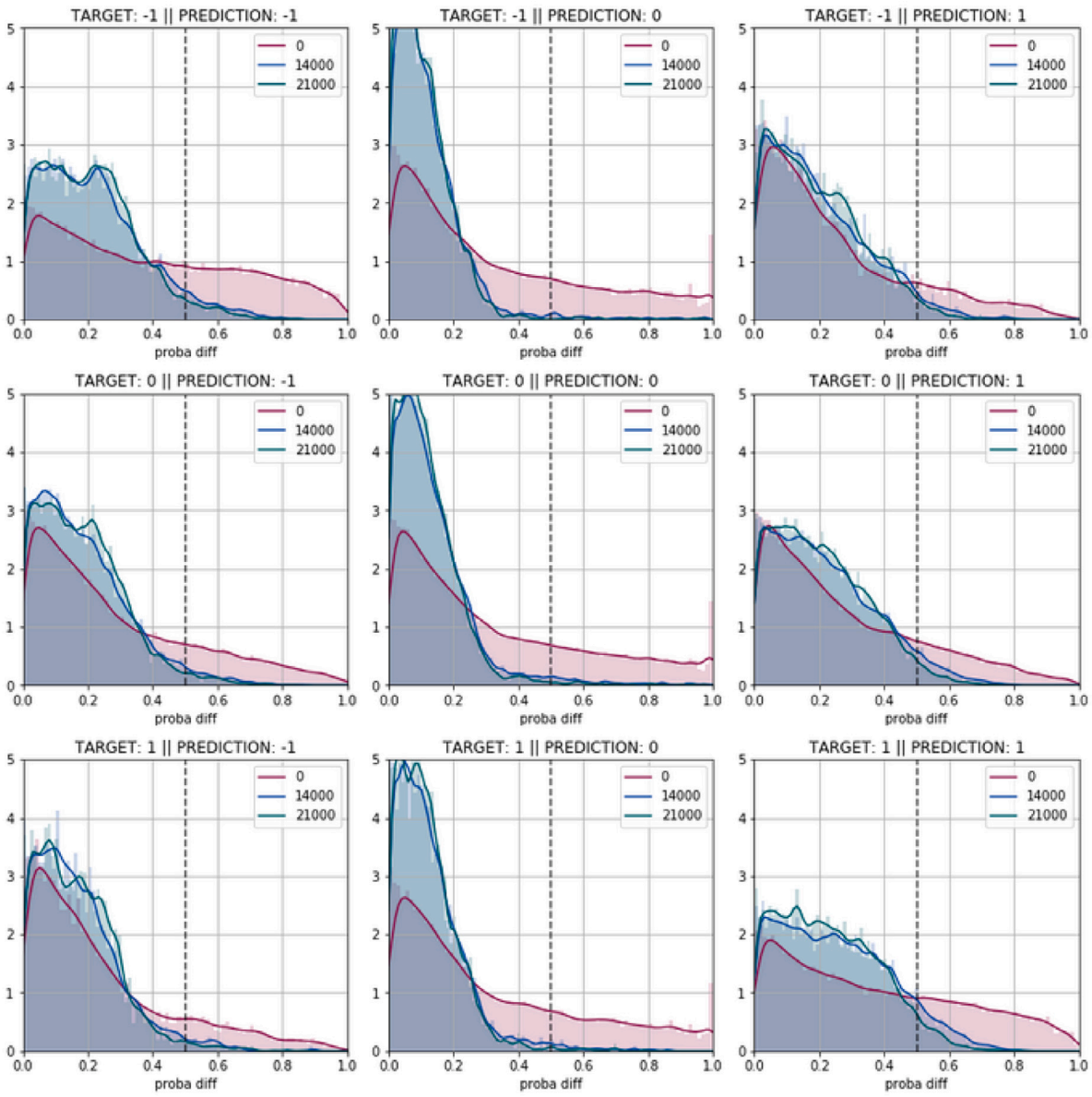


Fig. 15. Discriminated *proba_diff* distribution in EUR/USD. Distribution of *proba_diff* distinguished by the prediction and target combinations of each experiment of the test set on the real data for the EUR/USD market. The predictions with the same value show less population in erroneous estimates while the *proba_diff* value increases.

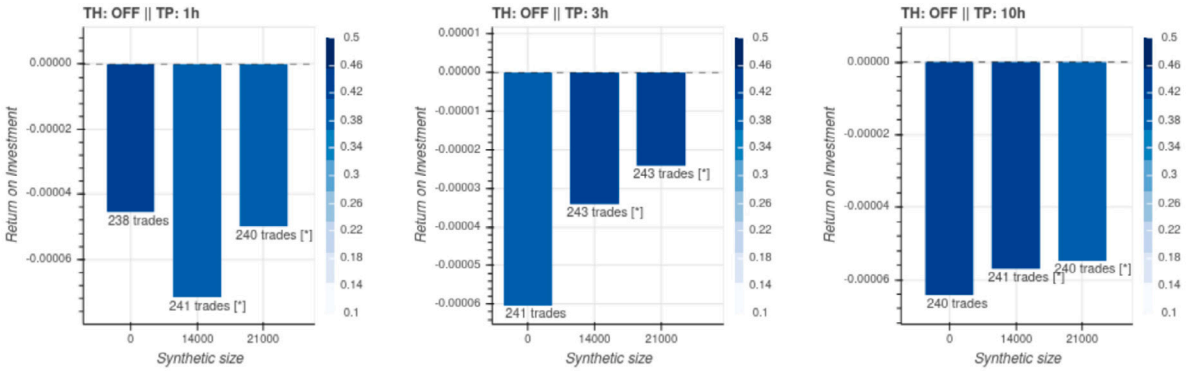
6. Conclusion

Making profits in trading strategies is a challenge faced by every participant in the different markets. Many of them opt for manual price monitoring and their experience in the market to perform trading operations. This article shows a trading methodology that makes use of AI and momentum-based trading notions. Price velocity estimates are produced to make buy and sell trades, which can generate profits in different periods and markets, as demonstrated by experimentation. Furthermore, this methodology can be implemented to work automatically in real-time, supporting high-frequency trading tasks.

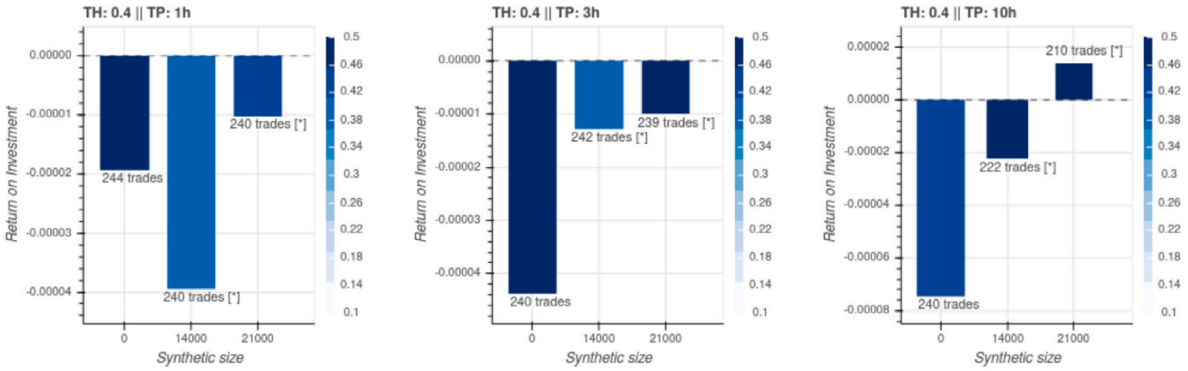
It is worth mentioning that the trading strategies that use the XAU/USD signal show better results than the strategies of the EUR/USD market, which could lead to the interpretation that it is better to trade with the XAU/USD signal than with the EUR/USD. However, the difference in performance could be due to using the same models and parametrization during the experimentation.

There are different points from where the development of the work could continue:

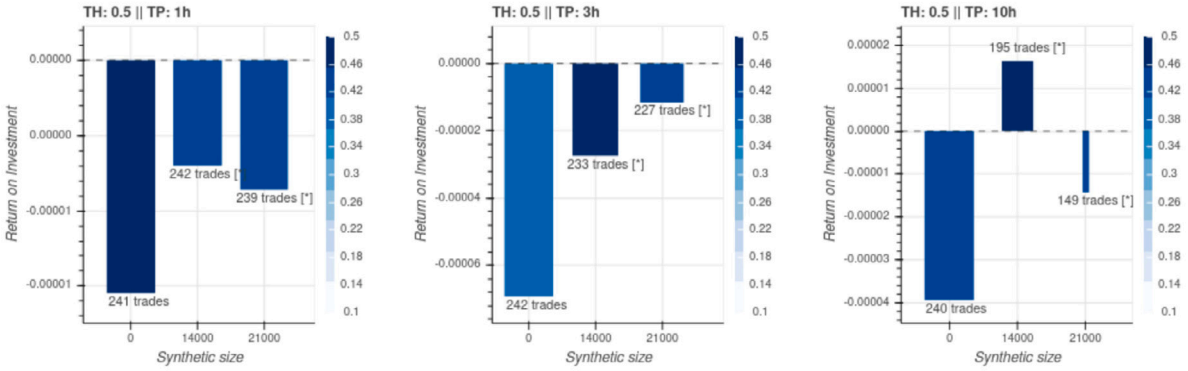
Trading on other market signals. We only tested the proposed methodology with two market signals. Having a notion of how this methodology works in various markets could have a significant impact.



(a) ROI for trading strategies that make use of a threshold of 0.



(b) ROI for trading strategies that make use of a threshold of 0.4.



(c) ROI for trading strategies that make use of a threshold of 0.5.

Fig. 16. Trading strategies result in EUR/USD. Results of trading strategy simulations using the Return on Investment metric for the EUR/USD market. The strategies make use of different amounts of synthetic data produced by deep generative models. Results for data between 2020-05-01 and 2021-04-01. The color of the bars represents the percentage of positive trades and the width, the number of trades. TP means *train_period*, and TH means threshold. [*] signals strategies using synthetic data. Observe how in the majority of the cases, synthetic data improves ROI, either by losing less money or by producing benefits (subplot (b), center and right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

More complex generative models. The predictive and generative models used in the present methodology are relatively simple. We could conduct further experimentation to determine a more robust model to improve the models' performance. For example, for the task of synthetic data generation in financial time series, one could try more recent models that have shown remarkable performance for this task, such as WGAN (Lou et al., 2018), TimeGANs (Jinsung et al., 2019), or novel models such as the one mentioned in Tetsuya et al.. For predictive tasks, one could use more robust models such as convolutional or recurrent neural networks, as presented in the works of Yong'an et al. (2020) and Saúl et al. (2020).

Balanced synthetic data: As observed in Tables 1 and 8, the dataset used for the velocity estimation is unbalanced. Future studies could be done where the aggregated synthetic data are used to balance the classes of the dataset and not only increase the number of samples. Studies could also be done modifying the intervals defined to identify the three categories of the classification task to find the intervals for which the best profit is generated in the trading strategies.

Other trading strategies: The proposed trading strategy and the mitigation mechanisms outlined in this paper are one way to use price velocity estimation as a trading signal. However, it is not the only way. For example, studies could be conducted to determine other forms of trading based on such estimates. Another possible work is that the velocity estimation task can be approached as a regression task rather than a classification task to find more diverse ways of trading.

Additional features and input signals: We trained predictive models with features of 5 signals derived from the price signal. Datasets with more features or more signals derived from the price, including external signals, could be created.

Interpretability: One additional aspect of future interest is understanding what kind of information generative models are able to extract that improves predictive models which, somehow cannot be directly exploited when trained with the original data.

CRediT authorship contribution statement

Daniel Carvajal-Patiño: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing – original draft. **Raul Ramos-Pollán:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing – review & editing, Supervision.

Data availability

The data source is described in the manuscript.

References

Adam, B., 2021. What is momentum trading? Investopedia. URL: https://www.investopedia.com/trading/introduction-to-momentum-trading/. (Accessed 09 September 2021).

Alberto, A., Ricardo, R., Néstor, D., 2020. Machine learning applied in the stock market through the moving average convergence divergence (MACD) indicator. Invest. Manag. Financial Innov. 17, 44–60. http://dx.doi.org/10.21511/imfi.17(4).2020.05.

Atkins, A., Niranjan, M., Gerding, E., 2018. Financial news predicts stock market volatility better than close price. J. Finance Data Sci. 4 (2), 120–137. http://dx.doi.org/10.1016/j.jfds.2018.02.002.

Aziz, S., Dowling, M., Hammami, H., Piepenbrink, A., 2021. Machine learning in finance: A topic modeling approach. Eur. Financial Manage. http://dx.doi.org/10.1111/eufm.12326.

Bergmeir, C., Hyndman, R.J., Koo, B., 2018. A note on the validity of cross-validation for evaluating autoregressive time series prediction. Comput. Statist. Data Anal. 120, 70–83. http://dx.doi.org/10.1016/j.csda.2017.11.003.

Braun, J., Hausler, J., Schäfers, W., 2020. Artificial intelligence, news sentiment, and property market liquidity. J. Prop. Invest. Finance 38 (4), 309–325. http://dx.doi.org/10.1108/JPIF-08-2019-0100.

Chakole, J.B., Kolhe, M.S., Mahapurush, G.D., Yadav, A., Kurhekar, M.P., 2021. A Q-learning agent for automated trading in equity stock markets. Expert Syst. Appl. 163, 113761. http://dx.doi.org/10.1016/j.eswa.2020.113761.

David, A.v.D., Xiao-Li, M., 2001. The art of data augmentation. J. Comput. Graph. Statist. 10 (1), 1–50. http://dx.doi.org/10.1198/10618600152418584.

Dev, S., Haruna, I., Farhana, Z., 2019. Stock market analysis: A review and taxonomy of prediction techniques. Int. J. Financial Stud. 7 (2), http://dx.doi.org/10.3390/ijfs7020026, URL: https://www.mdpi.com/2227-7072/7/2/26.

Diederik, P.K., Max, W., 2014. Auto-encoding variational Bayes. In: Proceedings of the 2nd International Conference on Learning Representations. ICLR, URL: http://arxiv.org/abs/1312.6114.

Duan, Y., Goodell, J.W., Li, H., Li, X., 2022. Assessing machine learning for forecasting economic risk: Evidence from an expanded Chinese financial information set. Finance Res. Lett. 46, 102273. http://dx.doi.org/10.1016/j.frl.2021.102273, URL: https://www.sciencedirect.com/science/article/pii/S1544612321003159.

Elberawi, A.S., Belal, M.P., 2021. A deep learning approach for forecasting global commodities prices. Future Comput. Inform. 6, http://dx.doi.org/10.54623/fue.fcij.6.1.4.

Fabbri., M., Moro., G., 2018. Dow jones trading with deep learning: The unreasonable effectiveness of recurrent neural networks. In: Proceedings of the 7th International Conference on Data Science, Technology and Applications - DATA. SciTePress, INSTICC, pp. 142–153. http://dx.doi.org/10.5220/0006922101420153.

Fernando, D.M.P., 2019. Enriching Financial Datasets with Generative Adversarial Networks. Delft University of Technology, URL: http://resolver.tudelft.nl/uuid:51d69925-fb7b-4e82-9ba6-f8295f96705c.

Fister, D., Perc, M., Jagrič, T., 2021. Two robust long short-term memory frameworks for trading stocks. Appl. Intell. 51, 7177–7195. http://dx.doi.org/10.1007/s10489-021-02249-x.

Fu, R., Chen, J., Zeng, S., Zhuang, Y., Sudjianto, A., 2020. Time series simulation by conditional generative adversarial net. Int. J. Mech. Indust. Eng. 14 (6), 458–471, https://publications.waset.org/vol/162.

Ghosh, P., Neufeld, A., Sahoo, J.K., 2022. Forecasting directional movements of stock prices for intraday trading using LSTM and random forests. Finance Res. Lett. 46, 102280. http://dx.doi.org/10.1016/j.frl.2021.102280.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems. Vol. 27. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Grace, L., 2020. Data quality problems troubling business and financial researchers: A literature review and synthetic analysis. J. Bus. Finance Librariansh. 25 (3–4), 315–371. http://dx.doi.org/10.1080/08963568.2020.1847555.

Gu, C.-S., Hsieh, H.-P., Wu, C.-S., Chang, R.-I., Ho, J.-M., 2019. A fund selection robo-advisor with deep-learning driven market prediction. In: 2019 IEEE International Conference on Systems, Man and Cybernetics. SMC, pp. 2845–2850. http://dx.doi.org/10.1109/SMC.2019.8914183.

Jiahang, Z., Chi, Z., 2022. Do cryptocurrency markets react to issuer sentiments? Evidence from Twitter. Res. Int. Bus. Finance 61, 101656. http://dx.doi.org/10.1016/j.ribaf.2022.101656.

Jinsung, Y., Daniel, J., Mihaela, v.d.S., 2019. Time-series generative adversarial networks. In: NeurIPS 2019 Reproducibility Challenge. URL: https://openreview.net/forum?id=rJeZq4reLS.

Jonathan, H., 2019. GAN - why it is so hard to train generative adversarial networks!. Medium. URL: https://jonathan-hui.medium.com/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b. (Accessed 09 September 2021).

Jørgensen, R.K., Igel, C., 2021. Machine learning for financial transaction classification across companies using character-level word embeddings of text fields. Intell. Syst. Account. Finance Manage. 28 (3), 159–172. http://dx.doi.org/10.1002/isaf.1500.

Julisa, B.A., 2021. Improving stock price prediction with GAN-based data augmentation. Indonesian J. Artif. Intell. Data Min. 4, 1–10. http://dx.doi.org/10.24014/ijaidm.v4i1.10740.

Jullum, M., Løland, R., Ånonsen, G., Lorentzen, J., 2020. Detecting money laundering transactions with machine learning. J. Money Laund. Control 23 (1), 173–186. http://dx.doi.org/10.1108/JMLC-07-2019-0055.

Kingma, D., Ba, J., 2014. Adam: A method for stochastic optimization. In: International Conference on Learning Representations.

Koshiyama, A., Firoozye, N., Treleaven, P., 2021. Generative adversarial networks for financial trading strategies fine-tuning and combination. Quant. Finance 21 (5), 797–813. http://dx.doi.org/10.1080/14697688.2020.1790635.

Lahmiri, S., 2020. A predictive system integrating intrinsic mode functions, artificial neural networks, and genetic algorithms for forecasting S&P500 intra-day data. Intell. Syst. Account. Finance Manage. 27 (2), 55–65. http://dx.doi.org/10.1002/isaf.1470.

Levina, E., Bickel, P., 2001. The earth mover's distance is the mallows distance: some insights from statistics. In: Proceedings Eighth IEEE International Conference on Computer Vision. Vol. 2. ICCV 2001, pp. 251–256. http://dx.doi.org/10.1109/ICCV.2001.937632.

Li, X., Yang, L., Xue, F., Zhou, H., 2017. Time series prediction of stock price using deep belief networks with intrinsic plasticity. In: 2017 29th Chinese Control and Decision Conference. CCDC, pp. 1237–1242. http://dx.doi.org/10.1109/CCDC.2017.7978707.

Liang, Q., Rong, W., Zhang, J., Liu, J., Xiong, Z., 2017. Restricted Boltzmann machine based stock market trend prediction. In: 2017 International Joint Conference on Neural Networks. IJCNN, pp. 1380–1387. http://dx.doi.org/10.1109/IJCNN.2017.7966014.

Liu, J., Lin, H., Liu, X., Xu, B., Ren, Y., Diao, Y., Yang, L., 2019. Transformer-based capsule network for stock movement prediction. In: Proceedings of the First Workshop on Financial Technology and Natural Language Processing. Macao, China, pp. 66–73, URL: https://aclanthology.org/W19-5511.

Liu, Q., Wang, C., Zhang, P., Zheng, K., 2021. Detecting stock market manipulation via machine learning: Evidence from China securities regulatory commission punishment cases. Int. Rev. Financ. Anal. 78, 101887. http://dx.doi.org/10.1016/j.irfa.2021.101887, URL: https://www.sciencedirect.com/science/article/pii/S1057521921002143.

Lou, H., Qi, Z., Li, J., 2018. One-dimensional data augmentation using a wasserstein generative adversarial network with supervised signal. In: 2018 Chinese Control and Decision Conference. CCDC, pp. 1896–1901. http://dx.doi.org/10.1109/CCDC.2018.8407436.

Luo, Z., Chen, J., Cai, X.J., Tanaka, K., Takiguchi, T., Kinkyo, T., Hamori, S., 2018. Oil price forecasting using supervised GANs with continuous wavelet transform features. In: 2018 24th International Conference on Pattern Recognition. ICPR, pp. 830–835. http://dx.doi.org/10.1109/ICPR.2018.8546240.

Magnus, W., Robert, K., Ralf, K., Peter, K., 2020. Quant GANs: deep generation of financial time series. Quant. Finance 20 (9), 1419–1440. http://dx.doi.org/10.1080/14697688.2020.1730426.

Marian, R., 2019. Combining wavelet decomposition with machine learning to forecast gold returns. Int. J. Forecast. 35 (2), 601–615. http://dx.doi.org/10.1016/j.ijforecast.2018.11.008.

Masafumi, N., Akihiko, T., Soichiro, T., 2018. Bitcoin technical trading with artificial neural network. Physica A 510, 587–609. http://dx.doi.org/10.1016/j.physa.2018.07.017.

Moews, B., Herrmann, J.M., Ibikunle, G., 2019. Lagged correlation-based deep learning for directional trend change prediction in financial time series. Expert Syst. Appl. 120, 197–206. http://dx.doi.org/10.1016/j.eswa.2018.11.027.

Nobre, J., Neves, R.F., 2019. Combining principal component analysis, discrete wavelet transform and XGBoost to trade in the financial markets. Expert Syst. Appl. 125, 181–194. http://dx.doi.org/10.1016/j.eswa.2019.01.083.

Novak, M.G., Velušček, D., 2016. Prediction of stock price movement based on daily high prices. Quant. Finance 16 (5), 793–826. http://dx.doi.org/10.1080/14697688.2015.1070960.

Nti, I.K., Adekoya, A.F., Weyori, B.A., 2021. A novel multi-source information-fusion predictive framework based on deep neural networks for accuracy enhancement in stock market prediction. J. Big Data 8, http://dx.doi.org/10.1186/s40537-020-00400-y.

Omid, R., Bahram, C., Abolhasan, F., Frederic, C., Elinaz, S., Himan, S., Eisa, M., John, T., Sabrina, C., Baharin, B.A., Dieu, T., 2019. Predicting uncertainty of machine learning models for modelling nitrate pollution of groundwater using quantile regression and UNEEC methods. Sci. Total Environ. 688, 855–866. http://dx.doi.org/10.1016/j.scitotenv.2019.06.320.

Pan, Z., Yu, W., Yi, X., Khan, A., Yuan, F., Zheng, Y., 2019. Recent progress on generative adversarial networks (GANs): A survey. IEEE Access 7, 36322–36333. http://dx.doi.org/10.1109/ACCESS.2019.2905015.

Panagiotis, T., Renatas, K., Bayasgalan, T.-A., 2020. Momentum trading in cryptocurrencies: Short-term returns and diversification benefits. Econom. Lett. 191, 108728. http://dx.doi.org/10.1016/j.econlet.2019.108728.

Phillips, P.P., Phillips, J.J., 2009. Return on investment. In: Handbook of Improving Performance in the Workplace. Vol. 1-3. John Wiley & Sons, Ltd, pp. 823–846. http://dx.doi.org/10.1002/9780470592663.ch53.

Rafiqul, I.M., Nguyet, N., 2020. Comparison of financial models for stock price prediction. J. Risk Financial Manage. 13, http://dx.doi.org/10.3390/jrfm13080181.

Salakhutdinov, R., Mnih, A., Hinton, G., 2007. Restricted Boltzmann machines for collaborative filtering. In: Proceedings of the 24th International Conference on Machine Learning. ICML '07, Association for Computing Machinery, New York, NY, USA, pp. 791–798. http://dx.doi.org/10.1145/1273496.1273596.

Sanboon, T., Keatruangkamala, K., Jaiyen, S., 2019. A deep learning model for predicting buy and sell recommendations in stock exchange of thailand using long short-term memory. In: 2019 IEEE 4th International Conference on Computer and Communication Systems. ICCCS, pp. 757–760. http://dx.doi.org/10.1109/CCOMS.2019.8821776.

Saúl, A.-M., Andrés, L.S.-C., Alejandro, C., David, Q., 2020. Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. Expert Syst. Appl. 149, 113250. http://dx.doi.org/10.1016/j.eswa.2020.113250.

Savitzky, A., Golay, M.J.E., 1964. Smoothing and differentiation of data by simplified least squares procedures. Anal. Chem. 36 (8), 1627–1639. http://dx.doi.org/10.1021/ac60214a047.

Shao, Z., Zheng, Q., Liu, C., Gao, S., Wang, G., Chu, Y., 2021. A feature extraction- and ranking-based framework for electricity spot price forecasting using a hybrid deep neural network. Electr. Power Syst. Res. 200, 107453. http://dx.doi.org/10.1016/j.epsr.2021.107453.

Shintate, T., Pichl, L., 2019. Trend prediction classification for high frequency bitcoin time series with deep learning. J. Risk Financial Manage. 12 (1), http://dx.doi.org/10.3390/jrfm12010017.

Such, F.P., Rawal, A., Lehman, J., Stanley, K., Clune, J., 2020. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In: III, H.D., Singh, A. (Eds.), Proceedings of the 37th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 119, PMLR, pp. 9206–9216.

Susan, A., Guido, W.I., Jonas, M., Evan, M., 2021. Using wasserstein generative adversarial networks for the design of Monte Carlo simulations. J. Econometrics http://dx.doi.org/10.1016/j.jeconom.2020.09.013.

Takahashi, S., Chen, Y., Tanaka-Ishii, K., 2019. Modeling financial time-series with generative adversarial networks. Physica A 527, 121261. http://dx.doi.org/10.1016/j.physa.2019.121261.

Tetsuya, K., Elena, M., Guillaume, P., An Adversarial Approach to Structural Estimation, Unpublished results, https://arxiv.org/abs/2007.06169.
Tizzano, A., 2018. Direct reinforcement learning for the DVA hedging through recurrent generative adversarial networks for dataset augmentation. Politecnico di Milano. URL: http://hdl.handle.net/10589/142609.
Uzunlu, B.Y., Hussain, S., 2020. Employing machine learning algorithms to build trading strategies with higher than risk-free returns. Int. Econometric Rev. 12, 112–138. http://dx.doi.org/10.33818/ier.805042.
Vargas, M.R., dos Anjos, C.E.M., Bichara, G.L.G., Evsukoff, A.G., 2018. Deep leaming for stock market prediction using technical indicators and financial news articles. In: 2018 International Joint Conference on Neural Networks. IJCNN, pp. 1–8. http://dx.doi.org/10.1109/IJCNN.2018.8489208.
Wei, F., Nguyen, U., 2020. Stock trend prediction using financial market news and BERT. In: Conference on Knowledge Discovery and Information Retrieval. pp. 325–332. http://dx.doi.org/10.5220/0010172103250332.
Wu, J.M.-T., Li, Z., Herencsar, N., Vo, B., Lin, J.C.-W., 2021. A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. Multimedia Syst. http://dx.doi.org/10.1007/s00530-021-00758-w.
Yong'an, Z., Binbin, Y., Memon, A., 2020. A novel deep learning framework: Prediction and analysis of financial time series using CEEMD and LSTM. Expert Syst. Appl. 159, 113609. http://dx.doi.org/10.1016/j.eswa.2020.113609.
Zhang, P., Ci, B., 2020. Deep belief network for gold price forecasting. Resour. Policy 69, 101806. http://dx.doi.org/10.1016/j.resourpol.2020.101806.