Walk-Forward Optimization Algorithm for Time-Series Models

CARLOS ANDRÉS CASTRO MARÍN

THESIS

Adviser

PAULA MARÍA ALMONACID HURTADO

UNIVERSIDAD EAFIT

ESCUELA DE ECONOMÍA Y FINANZAS

MAESTRÍA EN ADMINISTRACIÓN FINANCIERA - MAF

MEDELLÍN

2021

Dedication

This work is dedicated to my mother and my late father, whose efforts enabled me to pursue dreams that would otherwise have been unimaginable. I also dedicate this work to my nephews and nieces, my main motivation, in the hope that it will inspire them to work hard to achieve their dreams. Finally, it is dedicated to my brothers Juan, Rodrigo, and Fernando, who have always supported me in my chosen path.

Acknowledgments

Special appreciation is due to my adviser, former teacher, and coauthor Paula Almonacid, who supported me over this long process and has motivated me since I was an undergraduate. Special appreciation to my brother and co-author of the paper Rodrigo Alberto Castro (Rigo), who despite our contrarian opinions, always supports me and encourages me to question the world and to take the difficult path in order to achieve success and learn from experience.

Abstract

This paper presents a walk-forward optimization (WFO) algorithm to evaluate the predictive ability of time-series forecasting models. The algorithm is a series of steps to evaluate the performance of optimized models' performance over unseen observations to determine their predictive ability. Implementation examples are given for financial time series and the algorithm presented is used to measure the predictive ability of technical trading rules and machine learning–generated trading strategies after they are tested with three variations of the WFO process described in this paper. In recent years, WFO has been used in studies in different disciplines, but no proposals have been presented on how it should be implemented algorithmically. The algorithm is developed here, and our empirical results show that the WFO process is useful for evaluating the performance of time-series forecasting models.

Keywords

Financial markets; machine learning; predictive models; technical analysis; trading; walk-forward optimization; walk-forward validation

Table of Contents

Tables

Figures

1. Introduction

This paper describes a walk-forward optimization (WFO) algorithm and its different configurations to evaluate the predictive ability of time-series forecasting models. Its implementation and examples are given using traditional technical trading rules and trading strategies generated by machine learning algorithms. Moving average crossover and channel breakout are implemented, with an exhaustive specification search used as an optimization process to find the best rules to apply in each walk-forward step. A convolutional neural network (CNN) is used to train and generate a machine learning trading strategy to be tested in the WFO process.

This study is not intended to provide a framework for correctly optimizing the parameters or variables in a trading model, which could be key in the development of a model with predictive ability. The main goal of this paper is to give a framework for evaluating the predictive ability of time-series forecasting models. Different parameter configurations are given to set the walk-forward variations; however, an empirical study is necessary to determine the best walk-forward parameters for each variation presented in this study, which can depend on the stationarity, mean, and variance over time, as well as other factors. Many different forecasting models can be created from various data sources that might not be quantitative to evaluate its predictive ability through WFO; however, it is important to be able to express the model as an algorithm, leaving no space for subjective interpretations.

After the model's implementation, technical trading rules were found to have some degree of predictive ability, depending on the variation of WFO used, perhaps because market dynamics change over time or because the rules have a long- or short-term ability to capture predictable patterns over time, so some technical rules might be more suitable for quick, dynamic, and mean-reversal markets and others for long-term and trending markets. The CNN employed failed to show

predictive ability. This likely occurred because the one we used was a simple implementation of a CNN, which is likely to fail and worked as expected, providing good results over the training sets and poor results over validation sets, implying that the model might have been overfitted. CNNs are more suitable for classification, segmentation, and recognition of images, natural language processing problems, and other tasks (Collobert & Weston, 2008).

In recent years, WFO, sometimes called walk-forward validation, has been used in studies in different disciplines, yet to date no proposals have been raised as to how it should be described algorithmically. This paper proposes an algorithm with adjustable parameters to set different variations for different time-series phenomena.

The theoretical framework and concepts in the development of the algorithm are described in section 2. In section 3 the algorithm and its variations are presented. Then, trading rules and machine learning–generated trading strategies are implemented in the walk-forward algorithm, and the results are presented in section 4. Finally in section 5 conclusions and some considerations about the WFO algorithm are discussed.

2. Theoretical Framework

Because of the temporal order characteristics of observations over a time series, certain problems can arise when trading strategies or any other type of forecasting model are developed. In financial time-series forecasting, analysts look for patterns in the past that can occur again in the future, implying that the order of the past observations is important for spotting these patterns. To avoid altering the time dependence of the data, forecasting processes have traditionally employed out-of-sample (OOS) methods. As explained by Cerqueira et al. (2019), these methods designate a portion

of the time series as the out-of-sample period, which is adjacent to the in-sample period, in order to preserve the temporal order of the observations when developing and testing a forecasting model. The model is optimized over the in-sample period and then tested in the out-of-sample period, performance is compared between the two periods, and in this way, the model's predictive ability is measured. As mentioned by Cerqueira et al. (2019), OOS methods do not make the best use of the data available. The WFO process can be understood as a multiple OOS validation process over time to generate multiple in-sample and out-of-sample periods to train and test the models. Training and validation periods are interchangeable with in-sample and out-of-sample periods, respectively, for our purposes in this paper.

The main advantage of using WFO for testing a trading model or any other forecasting model is that it uses the data available, in the best possible way, without altering the temporal order of the series, ensuring that the model performance is evaluated over unseen observations, and then the predictive ability is determined. The walk-forward process is described by Pardo (2008) as "the closest possible simulation of an optimized trading strategy in real-time," which aims to optimize the model over many in-sample periods and test them in adjacent OOS periods. A similar approach is described by Zivot et al. (2006), who called this "rolling analysis," which is used to back-test the stability and predictive accuracy of statistical models with a time series. Bifet et al. (2009) describe the algorithm as "interleaved test-then-train or prequential," using a similar procedure by first testing over unseen sets and then training with new data, then moving over the full series and repeating the process. All of these descriptions are similar in concept and are used to construct the WFO algorithm.

One of the main model optimization issues that the WFO process helps to spot is data mining or snooping bias. This bias is created when the same data set is used several times for model inference.

The best model is selected by an extensive specification search or any other optimization process that can overfit the training sets, in which superior performance could be due to chance, rather than predictive ability (White, 2000). When this bias exists, superior performance is achieved over the training optimization period (in-sample) due to overfitting, which explains why performance deteriorates over unseen periods.

The WFO process can be used to avoid this issue and measure predictive ability before deploying a model. The algorithm developed here is based on descriptions provided by Pardo (2008), Zivot et al. (2006), and Bifet et al. (2009). The WFO algorithm creates different training and validation sets adjacent to them that are rolled over time, simulating real-time environments.

Suradhaniwar et al. (2021) used the WFO process for model evaluation in an agrometeorology time-series forecasting problem that was modeled with statistical and deep learning algorithms. They found walk-forward crucial for assessing the performance of time-series forecasting over a long forecast horizon. Short-term household electrical load data were evaluated using WFO and deep learning algorithms by Guo et al. (2021), who use mean absolute percentage error and root mean square deviation to evaluate the performance of the models in each walk-forward period. Mehtab et al. (2020) adopt walk-forward validation to evaluate a CNN and a long short-term memory (LSTM) algorithm with data on stock prices from the National Stock Exchange (NSE) of India. The study finds their models very accurate in forecasting stock prices after being evaluated using the WFO process. Ładyżyński et al. (2013) apply the WFO process with machine learning techniques to evaluate whether they can generate profitable trading strategies for the S&P 500 index, but had no positive results. They describe basic implementation of WFO in the study. The WFO process can be performed across different disciplines in time-series problems, with mixed results depending on the type of forecasting model and the time-series phenomena evaluated.

Various attempts have been made to predict financial markets, with different results. Brock et al. (BLL; 1992) carry out a study in which the use of simple trading rules such as moving averages and trading range breakouts led to profits, confirming the predictive ability over an OOS validation set. Ready (2002) reviews the BLL study and Allen and Karjalainen (1999), where the latter use genetic algorithms to create and select the best rules for making a profit on US stocks and short-term bonds. Ready (2002) finds that the patterns in historical data might look consistent, they do not necessarily continue in the future, indicating that "statistical arbitrage" might have made the models worse in the OOS period. Sullivan, Temmerman, and White (2002) find that the BLL results seem to be robust to data snooping bias, but the superior performance of the best rule was not replicated in the OOS period, for the next three reasons. First, the OOS period selected was not large enough. Second, data-mining bias might exist, meaning that they selected the best 7,846 rules from a larger tested universe. Third, the market has become more efficient, so profitable opportunities may have been traded away by cheaper computing power, lower transaction costs, and increased liquidity, eliminating short-term patterns. A study carried by Wang, Sun, Cao, and Wang (2018) employs CNN to decompose characteristics from traditional technical indicators to predict profitable returns over futures markets, achieving returns superior to those with other machine learning approaches. Tsantekidis et al. (2017) find that CNNs are more suitable than other machine learning algorithms at the moment of predicting stock market movements.

This study does not replicate any of the results by other authors but uses similar models through the WFO algorithm developed to determine models' predictive ability and to give examples of the algorithm's implementation. We use basic CNNs, moving average crossover and channel breakout, to implement the WFO algorithm and then test them for predictive ability, enabling any researcher to follow and implement variations and configurations depending on their prediction problems.

3. Methodology

In this section, the WFO process is presented, and the algorithm is described. Parameter configurations for each algorithm variation presented here are given with a brief explanation of use cases.

3.1 The Walk-Forward Optimization Algorithm for Time-Series Forecasting

The WFO is a two-step process that can be iterated over a time series. In the first step, the model is optimized (overfitting can occur in this process) over a training (in-sample) data set. In the second step, the model is tested over an adjacent validation (OOS) data set not seen before by the model, called the walk-forward window. After the model is tested over the validation set, a period re-optimization is performed, rolling over the training and validation data sets to perform the process from the first step through the full time series. At the end, the results of all the training and validation sets are recorded and compared to determine the models' predictive ability. The WFO algorithm presented here has three different variations, each of which can satisfy different forecasting needs, depending on the time series and models in question.

3.1.1 Moving Block Walk Forward

The moving block WFO algorithm is trained on *fixed-length list (block)* R of size r that is rolled each walk forward in k steps. Then, it is validated on *fixed length list* V of size v. When the iterative process is completed and the performance of all R and V sets is compared, the predictive ability can be assessed.

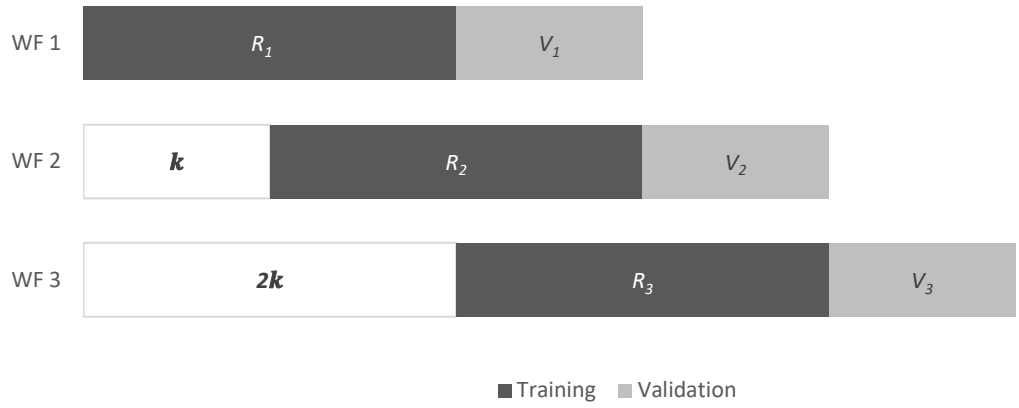

Fig 1. Moving Block Walk Forward

Given the nature of a moving block, which it forgets the past, it is more suitable for dynamic processes that are not fixed over time, such as nonstationary time series in highly variable environments.

3.1.2 Cumulative Walk Forward

The cumulative block WFO algorithm is trained on *constant growth length list* R of size r, which grows at Gr elements in each walk-forward step. Then, it is validated on *fixed or constant growth length list* V of size v, which can optionally grow at Gv elements in each walk-forward step. When the iterative process is completed and the performance of all R and V sets is compared, the predictive ability can be assessed.

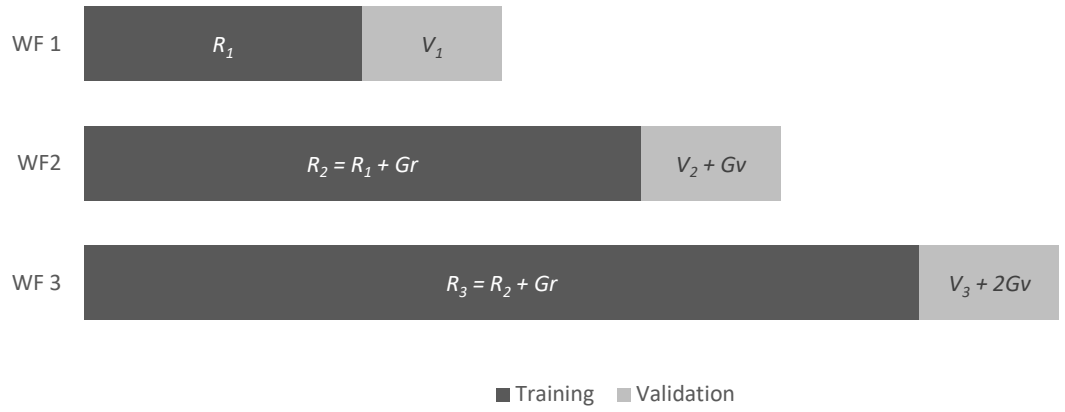

Fig 2. Cumulative Walk Forward (***Gv = 0***)

In the cumulative variation, the past is never forgotten, and all historical observations are included in the training sets for each walk-forward step. This is the classical use of WFO and may be more appropriate for time series that do not change much over time and have long-term trends.

3.1.3 Moving Block Growth Walk Forward

Moving block growth WFO is a variation of moving block WFO, with a cumulative characteristic in which R and V are blocks with size r and v and a constant growth of Gr and Gv, respectively, and each walk forward is rolled in k steps.

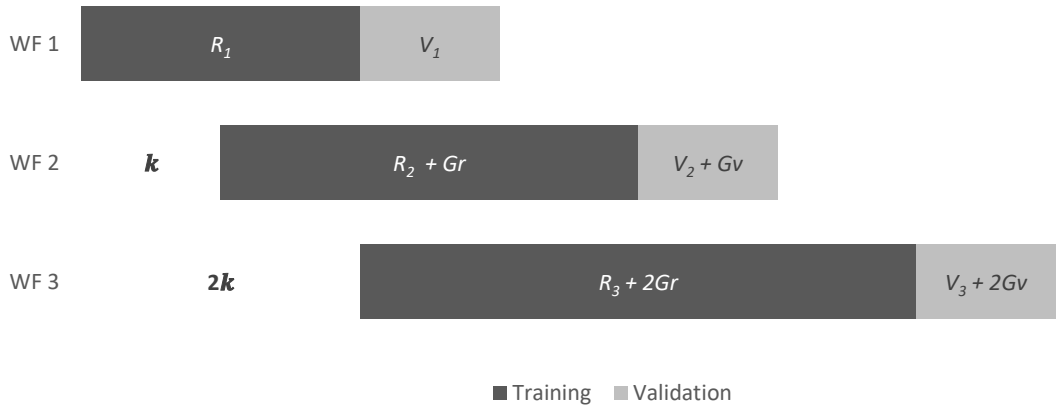

Fig 3. Moving Block Growth Walk Forward ***(Gr>0*** and ***Gv=0)***

Like moving block WFO, the growth variation is suitable for dynamic and changing time series. The cumulative nature could help models spot short- and medium-term patterns.

3.1.4 Using the Walk-Forward Optimization for Long-Term Horizon Evaluation

The walk-forward process can include as many elements as desired in the validation set. The best fit between training-validation observations is a matter of study and is generally chosen arbitrarily by the researcher. When more elements are added to the validation set, more steps ahead are evaluated in the forecasting process, which helps in evaluating multi-step forecasting methods. A parameter $\boldsymbol{u}$ is included in the algorithm to create an empty fixed space between the training and the validation set. In other words, the validation set can start at any observation in the future after the training set has completed each walk-forward step. Whether it is useful or not will depend on the time-series phenomena that are forecasted, and experimentation is required.

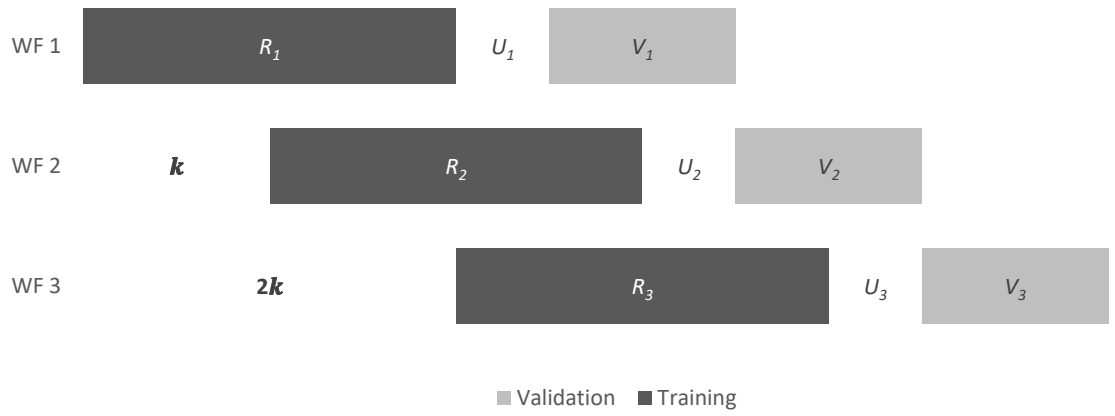

Fig 4. Moving Block Walk Forward (when $u>0$)

The parameter u is set with the default at 0, as seen in Figures 1, 2, and 3. Figure 4 illustrates the walk-forward behavior when $u > 0$. It can be applied to any of the variations previously described, but it does not suggest a new configuration. u observations are stored in U list if the observations need to be used.

3.2 Implementation of the Walk-Forward Algorithm

Here, we consider the model as a function for purposes of description.

Part 1 (Steps 1 and 2): Walk-Forward Sets and Optimization

Each walk-forward validation set depends on the number of observations that must be fed into the optimization function.

1. Let us define the seed series as $\boldsymbol{S} = \{\boldsymbol{x_0}, \boldsymbol{x_1}, \dots, \boldsymbol{x_n}\}$ where $\boldsymbol{x_i}$ is the observation of a phenomenon for a given time $\boldsymbol{i}$.
2. Let us define the function to be evaluated $\boldsymbol{f_i}\left(\boldsymbol{x_{n_i}}, \dots, \boldsymbol{x_{z_i}}\right) = \boldsymbol{w_i}$, which could depend on an arbitrarily high number of $\boldsymbol{z}$ observations that differ for each time $\boldsymbol{i}$ and where $\boldsymbol{w_i}$ represents the performance from evaluating $\boldsymbol{f_i}$ over a given time series.

3. Let us define constant $\boldsymbol{k}$, which is the size of each leap between walk forwards in the time series.
4. Let us define constant r, which is the cardinality of the training subset.
5. Let us define constant $\boldsymbol{Gr}$, which is the growth constant for the training subset.
6. Let us define constant v, which is the cardinality of validation subset.
7. Let us define constant $\boldsymbol{Gv}$, which is a growth constant for the validation subset.
8. Let us define $\boldsymbol{u}$ as the cardinality of the subset of empty observations between the training and validation subsets.
9. Let us define lists (walk forwards):

   Note that:

   1. $\boldsymbol{k}, \boldsymbol{r}, \boldsymbol{v}, \boldsymbol{u}$ are integers $< \boldsymbol{n}$.
   2. $\boldsymbol{z}$ is an integer $\leq \boldsymbol{r}$.

a) List 1, when $\boldsymbol{i} = \boldsymbol{1}$:

i. Training subset **1**,

$$\boldsymbol{R_1} = \{\boldsymbol{x_0}, \boldsymbol{x_1}, \dots, \boldsymbol{x_{r-1}}\}.$$

ii. Empty observations subset **1**,

$$\boldsymbol{U_1} = \{\boldsymbol{x_r}, \dots, \boldsymbol{x_{r+u-1}}\}, \text{ if } \boldsymbol{u} = \boldsymbol{0} \text{ then } \boldsymbol{U_1} = \boldsymbol{0}.$$

iii. Let us consider the optimized function $\boldsymbol{f_1^*}(\boldsymbol{x_{n_1}}, \dots, \boldsymbol{x_{z_1}}) = \boldsymbol{w_1}$ on the subset $\boldsymbol{R_1}$.

iv. Validation subset **1**,

$$\boldsymbol{V_1} = \{\boldsymbol{x_{r+u-z_1}}, \dots, \boldsymbol{x_{r+u+v-1}}\}.$$

v. $\boldsymbol{w_1^*}$ is obtained, as the result of evaluating the optimized function $\boldsymbol{f_1^*}$ on $\boldsymbol{V_1}$**.** That is, $\boldsymbol{w_1^*}$ is the forecast value.

b) List 2, when $\boldsymbol{i = 2}$:

i. Training subset **2**,

$$\boldsymbol{R_1 = \{x_{0+k}, x_1, \dots, x_{r+k+Gr-1}\}},$$

which has $\boldsymbol{r + Gr}$ elements (observations).

ii. Empty observations subset **2**,

$$\boldsymbol{U_2 = \{x_{r+k+G_r}, \dots, x_{r+k+G_r+u-1}\}}, \text{ if } \boldsymbol{u = 0} \text{ then } \boldsymbol{U_2 = 0}.$$

iii. Let us consider the optimized function $\boldsymbol{f_2^*(x_{n_2}, \dots, x_{z_2}) = w_2}$ on the subset $\boldsymbol{R_2}$**.**

iv. Validation subset **2**,

$$\boldsymbol{V_2 = \{x_{r+k+G_r+u-z_2}, \dots, x_{r+k+G_r+u+v+G_v-1}\}},$$

which has $\boldsymbol{v + Gv}$ elements.

v. $\boldsymbol{w_2^*}$ is obtained, as the result of evaluating the optimized function $\boldsymbol{f_2^*}$ over $\boldsymbol{V_2}$.

c) List $\boldsymbol{i}$-th:

i. Training subset $\boldsymbol{i}$-th,

$$\boldsymbol{R_i = \{x_{0+k(i-1)}, \dots, x_{r+(k+G_r)(i-1)-1}\}},$$

which has $\boldsymbol{r - (1 - i)G_r}$ elements.

ii. Empty observations subset $\boldsymbol{i}$-th,

$$\boldsymbol{U_i = \{x_{r+(k+G_r)(i-1)}, \dots, x_{r+(k+G_r)(i-1)+u-1}\}}, \text{ if } \boldsymbol{u = 0} \text{ then } \boldsymbol{U_i = 0}.$$

iii. Let us consider the optimized function $\boldsymbol{f_i^*(x_{n_i}, \dots, x_{z_i}) = w_i}$ on the subset $\boldsymbol{R_i}$**.**

iv. Validation subset $\boldsymbol{i}$-th,

$$\boldsymbol{V_i = \{x_{r+(k+G_r)(i-1)+u-z_i}, \dots, x_{r+(k+G_r+G_v)(i-1)+u+v-1}\}},$$

which has $\boldsymbol{v-(1-i)G_v}$ elements.

v. $\boldsymbol{w_i^*}$ is obtained, as the result of evaluating the optimized function $\boldsymbol{f_i^*}$ over $\boldsymbol{V_i}$.

10. When $\boldsymbol{r+(k+G_r+G_v)(i-1)+u+v-1>n}$ walk-forward lists stop being generated, and the last $\boldsymbol{i}$ is the total number of walk forwards, which is denoted as $\boldsymbol{m}$ lists. This is the stop criterion for the algorithm.

Part 2: Evaluation of the Models' Performance

1. The function to be studied was determined above and denoted $\boldsymbol{f}$**,** which might depend on an arbitrarily high number of observations (variables) and parameters.
2. The optimum parameters are found for $\boldsymbol{f}$ using the training subsets $\boldsymbol{R_i}$, so the performance value $\boldsymbol{w_i}$ is obtained from the optimal function $\boldsymbol{f_i^*}$ for $\boldsymbol{i=1,2,\dots m}$.
3. $\boldsymbol{f_i^*}$ is evaluated in the validation subset $\boldsymbol{V_i}$ to determine the forecast value $\boldsymbol{w_i^*}$**.**

$$\boldsymbol{f_i^*\left(x_{r+(k+G_r)(i-1)+u-z_1},\dots,x_{r+(k+G_r+G_v)(i-1)+u+v-1}\right)=w_i^*,\ \ i=1,2,\dots m.}$$

4. Finally, the predictive ability comparison function is calculated as $\boldsymbol{c(w_i^*,w_i)}$**,** which depends on the phenomena to study for $\boldsymbol{i=1,2,\dots m}$; in this way, the results are $\boldsymbol{m}$ walk forwards to be compared in order to determine whether the function $\boldsymbol{f}$ under the optimization process performed on the seed time series $\boldsymbol{S}$ has predictive ability over future observations of $\boldsymbol{S}$**.**

3.2.1 Variations for the Walk-Forward Optimization Algorithm

As mentioned in section 3.1, the WFO algorithm has three variations, each of which requires parameters to be **0** or greater than **0**; see Table 1 with all the parameters for each configuration.

	Parameters					
Configuration	***k***	***r***	***Gr***	***V***	***Gv***	***u***
Moving Block	>0	>0	=0	>0	=0	> or = 0
Moving Block Growth	>0	>0	>0	>0	> or = 0	> or = 0
Cumulative	=0	>0	>0	>0	> or = 0	> or = 0

Table 1. Parameters for walk-forward variations

4. Examples of Implementation and Results

All three walk-forward variations were implemented with $\boldsymbol{u} > \mathbf{0}$ and $\boldsymbol{u} = \mathbf{0}$, for a total of six different types of walk forwards to analyze the models. Traditional technical trading rules and a CNN were used to examine the predictive ability. The walk-forward algorithm was coded in python (see Appendix 1).

4.1 Dataset and Transformations

The dataset used in the model consists of S&P 500 index closing prices. Time-series data are available from January 4, 2006, to December 6, 2021, on a daily basis, yielding a total of 4,008 observations. In Figure 5, the S&P 500 index appears to be a nonstationary time series with a strong trend component. An augmented Dickey-Fuller test was performed, producing a p-value of 0.986, which fails to reject the null hypothesis; this implies that the series has a unit root and hence is nonstationary. In order for us to properly work with machine learning algorithms, we create a stationary and normalized series, to achieve this a differentiation of lag-1 is done to stabilize the mean and variance as shown in Figure 6 and then scaling the data between [-1, 1]. After the series is transformed to make it stationary, we perform an ADF test, which yields a *p*-value of 0.000115; therefore, the null hypothesis is rejected, and the time series does not have a unit root, confirming

that it is now stationary. Because of technical trading rules, the original series is used in order to emulate the decision-making process used by technical analysts in live markets.

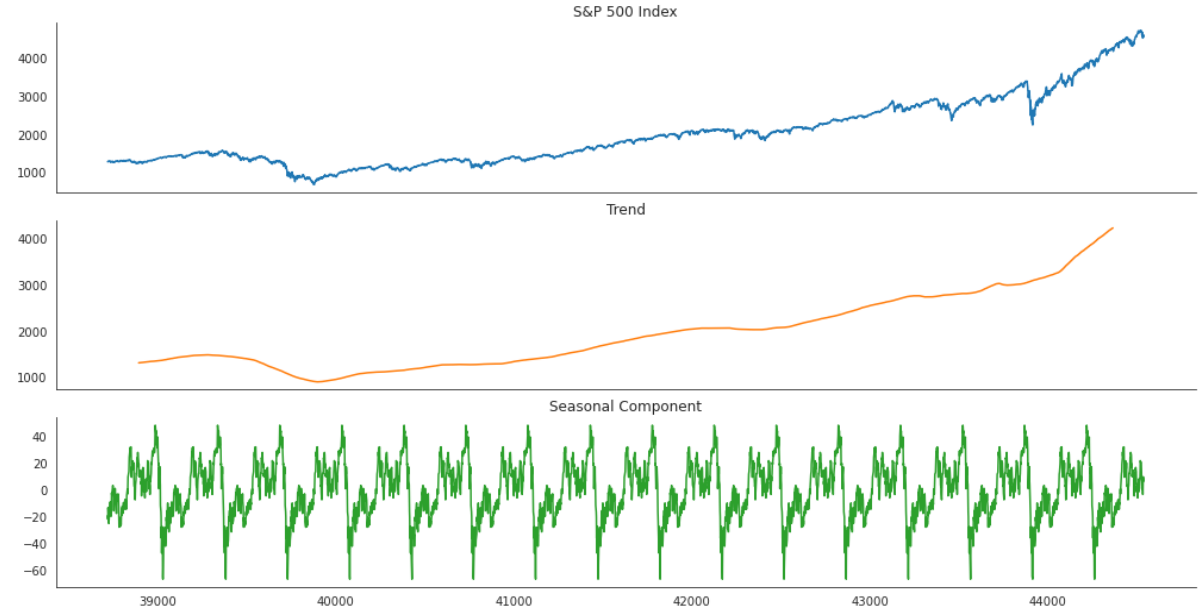

Figure 5. S&P 500 Trend and Seasonal Components

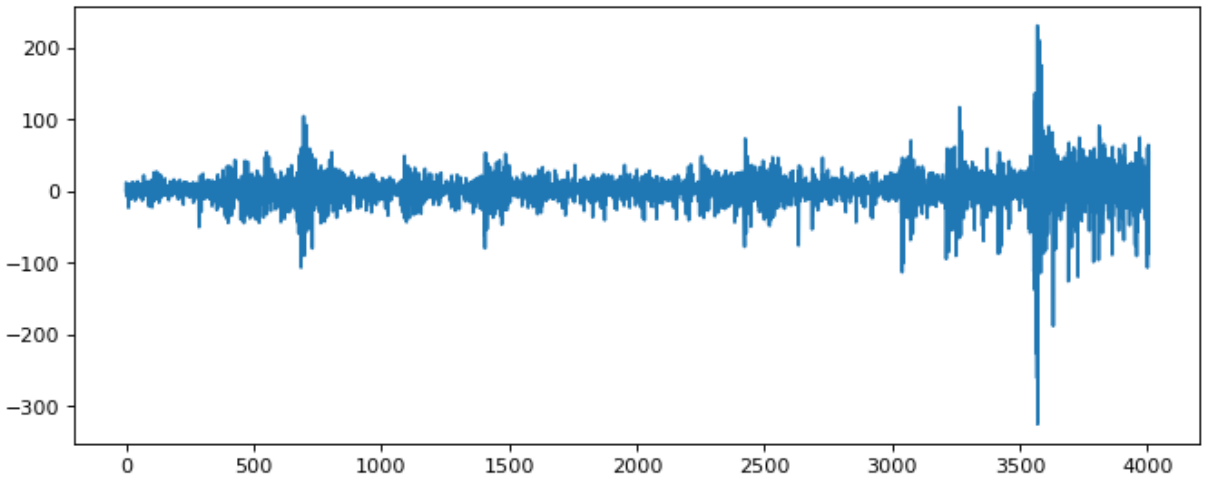

Figure 6. S&P 500 Index Differentiated with lag-1

4.2 Metrics and Performance Calculation

Performance and hence predictive ability can be measure in several different ways. In this study, the metrics used to measure predictive ability are mean squared error, mean absolute error, and root mean square error, all of which are calculated for each walk-forward variation. We use the annual total return for each walk-forward variation as prediction and performance metrics to compare the results.

4.2.1 Mean Squared Error

We construct the equation for the mean squared error (MSE) in the WFO algorithm as follows:

1. Let us denote $\boldsymbol{n}$ as the vector of predictions or walk-forward steps.
2. Let us denote $\boldsymbol{y_i}$ as the vector of values to predict (actual observations).
3. Let us denote $\boldsymbol{\hat{y}_i}$ as the vector of predictions.

$$\boldsymbol{MSE} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^{\mathbf{2}} .$$

The MSE is calculated for each walk-forward step $\boldsymbol{i} = \mathbf{1}, \mathbf{2}, \ldots \boldsymbol{m}.$, and the training performance $\boldsymbol{w_i}$ and the validation performance $\boldsymbol{w_i^*}$ are used as vectors for MSE, where $\boldsymbol{w_i} = \boldsymbol{y_i}$ and $\boldsymbol{w_i^*} = \boldsymbol{\hat{y}_i}$. Then, the MSE is calculated to determine the accuracy of the prediction in $\boldsymbol{c}(\boldsymbol{w_i^*}, \boldsymbol{w_i})$.

4.2.2 Mean Absolute Error

We construct the equation for the mean absolute error (MAE) for the WFO algorithm as follows:

1. Let us denote $\boldsymbol{n}$ as the vector of predictions or walk-forward steps.
2. Let us denote $\boldsymbol{y_i}$ as the vector of values to predict (actual observations).

3. Let us denote $\widehat{\boldsymbol{y}}_{\boldsymbol{i}}$ as the vector of predictions.

$$\boldsymbol{MAE} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} |\widehat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{y}_{\boldsymbol{i}}| \, .$$

The MAE is calculated for each walk-forward step $\boldsymbol{i} = \mathbf{1}, \mathbf{2}, \ldots \boldsymbol{m}.$, and the training performance $\boldsymbol{w}_{\boldsymbol{i}}$ and the validation performance $\boldsymbol{w}_{\boldsymbol{i}}^{*}$ are used as vectors for MSE, where $\boldsymbol{w}_{\boldsymbol{i}} = \boldsymbol{y}_{\boldsymbol{i}}$ and $\boldsymbol{w}_{\boldsymbol{i}}^{*} = \widehat{\boldsymbol{y}}_{\boldsymbol{i}}$. Then, the MAE is calculated to determine the accuracy of the prediction in $\boldsymbol{c}(\boldsymbol{w}_{\boldsymbol{i}}^{*}, \boldsymbol{w}_{\boldsymbol{i}})$.

4.2.3 Root Mean Square Error

We construct the equation for the root mean square error (RMSE) for the WFO algorithm as follows:

1. Let us denote $\boldsymbol{n}$ as the vector of predictions or walk-forward steps.
2. Let us denote $\boldsymbol{y}_{\boldsymbol{i}}$ as the vector of values to predict (actual observations).
3. Let us denote $\widehat{\boldsymbol{y}}_{\boldsymbol{i}}$ as the vector of predictions.

$$\boldsymbol{RMSE} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} |\widehat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{y}_{\boldsymbol{i}}| \, .}$$

The RMSE is calculated for each walk-forward step $\boldsymbol{i} = \mathbf{1}, \mathbf{2}, \ldots \boldsymbol{m}.$, and the training performance $\boldsymbol{w}_{\boldsymbol{i}}$ and the validation performance $\boldsymbol{w}_{\boldsymbol{i}}^{*}$ are used as vectors for MSE, where $\boldsymbol{w}_{\boldsymbol{i}} = \boldsymbol{y}_{\boldsymbol{i}}$ and $\boldsymbol{w}_{\boldsymbol{i}}^{*} = \widehat{\boldsymbol{y}}_{\boldsymbol{i}}$. Then, the RMSE is calculated to determine the accuracy of the prediction in $\boldsymbol{c}(\boldsymbol{w}_{\boldsymbol{i}}^{*}, \boldsymbol{w}_{\boldsymbol{i}})$.

4.2.4 Annual Total Return

To calculate the annual total return, we sum the logarithmic total return for the training and validation sets individually and for each walk-forward step, and then transform it into an annual return as follows:

1. Let us denote $\boldsymbol{dr}$ as a daily return.
2. Let us denote $\boldsymbol{tr}$ as the total return on the validation set.
3. Let us denote $\boldsymbol{tr}$ as the total return on the validation set.
4. Let us denote $\boldsymbol{dtr}$ as the number of days on the validation set.
5. Let us denote $\boldsymbol{ar}$ as the annual return.

$$\boldsymbol{dr} = (\mathbf{1} + \boldsymbol{tr})^{\mathbf{1}/(\boldsymbol{dtr})} - \mathbf{1},$$

$$\boldsymbol{ar} = (\mathbf{1} + \boldsymbol{dr})^{\mathbf{252}} - \mathbf{1},$$

Note that $(1 + \boldsymbol{dr})$ is powered to 252, which corresponds to the average number of trading days per year on the New York Stock Exchange (NYSE).

The annual return is calculated for each walk-forward step $\boldsymbol{i} = \mathbf{1}, \mathbf{2}, \ldots \boldsymbol{m}.$, the training performance $\boldsymbol{w_i}$ is the true value or value to be predicted, and validation performance $\boldsymbol{w_i^*}$ is the prediction. Then, the performance metric is used in $\boldsymbol{c}(\boldsymbol{w_i^*}, \boldsymbol{w_i})$ to evaluate the predictive ability of the model.

4.3 Parameters Used in the Walk-Forward Algorithm

In the examples given here, 540 observations and 160 observations are used for the training and validation sets, respectively, which is a ratio of about 77% for training and 23% for validation. In some variations, a step of 160 observations is set, which implies moving the series a full validation set ahead into the future, and in other variations, 0 steps are set, which implies a cumulative

variation of the walk-forward algorithm. Finally, some variations use a growth factor of 60 and 20 for the training and validation sets, respectively, on each walk-forward step, it implies that the ratio of training to validation changes constantly over time in favor of the training set, which obtains a higher share in each step. All the parameters given are subjective, and more research is recommended.

4.4 Technical Trading Rules

Moving average crossover and channel breakouts are used in this section. In order to perform the walk-forward optimization, we conduct an exhaustive specification search over 225 trading rules, which in this case are parameter configurations for both technical trading rules. An exhaustive specification search can lead to data snooping bias, but the WFO process can detect it. Trading rules generate signals for each trading day, and to calculate performance the list of signals is multiplied by the logarithmic daily market return one observation ahead of the date on which the signal was generated. If the signal is generated at t+0, performance will be calculated using the signal at t+0 and the market return at t+1.

4.4.1 Moving Average Crossover

A moving average crossover buy (sell) signal is generated when the short moving average crosses above (below) the long moving average. This model is always long or short, and the position is closed when a signal is generated, and then an opposite position is opened. The short moving averages are 1, 3, 5, 10, 15, 20, and 25. The long moving averages are 55, 65, 75, 100, 125, and 150, for a total of 56 rules.

1. Let us define $\boldsymbol{n}$ as the number of observations in the short period.
2. Let us define $\boldsymbol{m}$ as the number of observations in the long period.

3. Long period: $\boldsymbol{L_m = r_{i\,i=1}^{m} = r^1, r^2, \dots, r_{m-n+1}, \dots, r_m}$.
4. Short period: $\boldsymbol{S_n = r_{i\,i=m-n+1}^{m} = r_{m-n+1}, r_{m-n+2}, \dots, r_m}$, where $\boldsymbol{r_i}$ is the observation value on day $\boldsymbol{i, n < m}$.
5. Short moving average $\boldsymbol{P_{S_n}}$

$$\boldsymbol{P_{S_n} = \frac{1}{n} \sum_{k=m-n+1}^{m} r_k}.$$

6. Long moving average: $\boldsymbol{P_{L_m}}$

$$\boldsymbol{P_{L_m} = \frac{1}{m} \sum_{k=1}^{m} r_k}.$$

7. Signals generator: we denote the sign from $\boldsymbol{P_{S_n} - P_{L_m}}$ as $\boldsymbol{sig\left(P_{S_n} - P_{L_m}\right)}$ from

$$\boldsymbol{sig\left(P_{S_n} - P_{L_m}\right) = \frac{P_{S_n} - P_{L_m}}{\left|P_{S_n} - P_{L_m}\right|}}.$$

Note that $\boldsymbol{sig\left(P_{S_n} - P_{L_m}\right)}$ is **1** or **-1,** which is short and long, respectively.

4.4.1.1 Optimization and Results

We conduct an exhaustive specification search (data mining) to determine the combination of short and long moving averages that performed better in each walk-forward step. This does not guarantee the creation of a successful predictive model, and the risk of overfitting is rather high, however, this issue can be detected by the WFO process.

	Parameters						Metrics			Returns	
Algorithm	**k**	**r**	**gr**	**v**	**Gv**	**u**	**MSE**	**MAE**	**RMSE**	**Avg Ann. Training**	**Avg. Ann. Validation**
Moving Block	160	540	0	160	0	0	0.064	0.198	0.254	13.1%	8.0%
Moving Block+u	160	540	0	160	0	30	0.048	0.172	0.220	13.1%	8.7%
MB Growth	160	540	60	160	20	0	0.045	0.149	0.212	11.1%	8.7%
MB Growth+u	160	540	60	160	20	30	0.045	0.157	0.213	11.1%	6.8%
Cumulative	0	540	60	160	20	0	0.021	0.099	0.145	11.1%	6.8%
Cumulative+u	0	540	60	160	20	30	0.018	0.096	0.134	11.2%	7.0%

Table 2. Moving Average Crossover Results

Table 2 shows that, for all variations, the metrics are close to **0** and consistent, suggesting some degree of predictive ability. The average annual return, which is the performance metric, deteriorated when tested on validation sets for each walk-forward variation, and this behavior is expected in any type of forecasting model. Nevertheless, the returns were higher in the moving block variations, and the statistical metrics suggest that the cumulative variations are more suitable for this model, producing more consistent performance metrics.

4.4.2 Channel Breakout

A channel breakout is defined as a rule in which a channel is created by the high and low price of the last *n* period (3, 5, 10, 15, 20, 25, 30, 40, 50, 75, 100, 150, 200) over a moving average that ranges from 1 in the original series to 3, 5, 7, 10, 15, 20, 25, 30, 35, 40, 45, and 50. A buy (sell) signal is generated when the moving average is above (below) 70% (30%) of the channel value. When the channel value is within the range of 70% and 30% the signal is null, and the positions

are closed until a new signal is generated. The limits can be subject to data mining, and in this case, those limits are fixed. This setup yields a total of 169 trading rules.

1. Let us define the seed series as the order series

$$\boldsymbol{S} = \{\boldsymbol{x_1}, \boldsymbol{x_{2,\ldots,}} \boldsymbol{x_n}\}.$$

2. Let us define a constant that generates the transformed series $\boldsymbol{k}, \boldsymbol{k} < \boldsymbol{n}$.

3. Let us define a transformed series $\boldsymbol{S_k} = \{\boldsymbol{y_1}, \boldsymbol{y_{2,\ldots,}} \boldsymbol{y_{n-k+1}}\}$, where

$$\boldsymbol{y_1} = \frac{\boldsymbol{1}}{\boldsymbol{k}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{k}} \boldsymbol{x_i}\,, \boldsymbol{y_2} = \frac{\boldsymbol{1}}{\boldsymbol{k}} \sum_{\boldsymbol{i=2}}^{\boldsymbol{k+1}} \boldsymbol{x_i}\,, \ldots, \boldsymbol{y_{n-k+1}} = \frac{\boldsymbol{1}}{\boldsymbol{k}} \sum_{\boldsymbol{i=n-k+1}} \boldsymbol{x_i},$$

   then,

$$\boldsymbol{y_j} = \frac{\boldsymbol{1}}{\boldsymbol{k}} \sum_{\boldsymbol{i=j}}^{\boldsymbol{k+j-1}} \boldsymbol{x_i}\,, \boldsymbol{j} = \boldsymbol{1}, \ldots, \boldsymbol{n} - \boldsymbol{k} + \boldsymbol{1}.$$

4. Let us define $\boldsymbol{u}$ as a previously fixed observation and $\boldsymbol{v}$ as the observations to use after observation $\boldsymbol{u}$

5. Let us denote ${}_{\boldsymbol{u}}\boldsymbol{P_v}$ as the order subset from $\boldsymbol{S_k}$, defined by

$${}_{\boldsymbol{u}}\boldsymbol{P_v} = \{\boldsymbol{y_u}, \boldsymbol{y_{u+1,\ldots,}} \boldsymbol{y_{u+v-1}}\},$$

   where $\boldsymbol{1} \leq \boldsymbol{u} \leq \boldsymbol{v} \leq \boldsymbol{n} - \boldsymbol{k} + \boldsymbol{1}$.

6. Let us define ${}_{\boldsymbol{u}}\boldsymbol{m_v} = \boldsymbol{min}({}_{\boldsymbol{u}}\boldsymbol{P_v})$ *and* ${}_{\boldsymbol{u}}\boldsymbol{M_v} = \boldsymbol{max}({}_{\boldsymbol{u}}\boldsymbol{P_v})$.

7. Let us define channel value $\boldsymbol{V}$ as

$$\boldsymbol{V} = \left(\left(\boldsymbol{100}(\boldsymbol{y_{u+v-1}} - {}_{\boldsymbol{u}}\boldsymbol{m_v})\right)/({}_{\boldsymbol{u}}\boldsymbol{M_v} - {}_{\boldsymbol{u}}\boldsymbol{m_v}\right).$$

8. Define $\boldsymbol{L_w}$ as the lower percentage limit and $\boldsymbol{U_p}$ as the upper percentage limit, noting that $\boldsymbol{L_w} < \boldsymbol{U_p}$.

9. Let us define: $\boldsymbol{signal(V)}$ = **-1** when $\boldsymbol{V < L_w}$, $\boldsymbol{signal(V)}$ = **0** when $\boldsymbol{L_w < V < U_p}$, and $\boldsymbol{signal(V)}$ = **1** when $\boldsymbol{U_p < V}$.

4.4.2.1 Optimization and Results

We conducted an exhaustive specification search (data mining) to determine the combination of moving averages and channel averages that performed better in each walk-forward step. This does not guarantee the optimization of a successful predictive model, and the risk of overfitting is rather high, however, this issue can be detected by the WFO process.

	Parameters						Metrics			Returns	
Algorithm	**k**	**r**	**gr**	**v**	**gv**	**u**	**MSE**	**MAE**	**RMSE**	**Avg Ann. Training**	**Avg. Ann. Validation**
Moving Block	160	540	0	160	0	0	0.093	0.242	0.305	18.6%	2.6%
Moving Block+u	160	540	0	160	0	30	0.083	0.231	0.288	18.6%	4.0%
MB Growth	160	540	60	160	20	0	0.089	0.216	0.298	14.2%	3.9%
MB Growth+u	160	540	60	160	20	30	0.109	0.246	0.331	14.2%	1.1%
Cumulative	0	540	60	160	20	0	0.038	0.131	0.194	12.8%	8.3%
Cumulative+u	0	540	60	160	20	30	0.033	0.127	0.182	12.9%	8.2%

Table 3. Channel Breakout Results

Table 3 shows that for the cumulative variations, the metrics are close to 0, which suggests some degree of predictive ability. The performance metric (return) is more consistent in the cumulative variations, which supports the statistical metric results. As expected with any forecasting model, validation returns are lower than training returns. The results suggest that, for channel breakouts, cumulative variations have a better fit.

4.5 Machine Learning Algorithm

This section uses a specific type of machine learning algorithm, also called a deep learning algorithm, because it uses neural networks to learn and predict outcomes. Neural networks come in many types, with different purposes, depending on what is needed. The next section discusses a CNN adapted for time series.

4.5.1 Convolutional Neural Network Design

We set the parameters for the CNN in order to predict an observation in the time series as follows. The first step is creating a one-dimensional convolutional layer with max-pooling and a dense layer. An output layer is used to predict the next time step. The 1D CNN is implemented with 60 filters, 4 kernels, and activation function *ReLU*. An *Adam* optimizer is used as well as a loss function, *mean square error.* The input shape of the models is (60,1), using 60 lags or previous time steps and 1 feature, which is the univariate time series and 60 epochs. This is the basic setup of a 1D CNN for time series.

4.5.2 Convolutional Neural Network Results

To attain replicability of the results and easy comparison between walk forwards, we set a random seed as deterministic.

	Parameters						Metrics			Returns	
Algorithm	**k**	**r**	**gr**	**v**	**gv**	**u**	**MSE**	**MAE**	**RMSE**	**Avg Ann. Training**	**Avg. Ann. Validation**
Moving Block	160	540	0	160	0	0	0.412	0.581	0.642	64.2%	11.5%
Moving Block+u	160	540	0	160	0	30	0.399	0.580	0.632	64.7%	12.9%
MB Growth	160	540	60	160	20	0	0.292	0.481	0.541	42.4%	2.4%
MB Growth+u	160	540	60	160	20	30	0.325	0.491	0.570	42.4%	2.6%
Cumulative	0	540	60	160	20	0	0.129	0.291	0.359	28.0%	9.9%
Cumulative+u	0	540	60	160	20	30	0.124	0.293	0.352	28.4%	6.7%

Table 4. Convolutional Neural Networks Results

Table 4 shows that all attempts failed to make a prediction with a naïve CNN on all variations of the walk-forward, and the statistical metrics and performance metrics are consistent with this result. This does not confirm that CNNs are unsuitable for this task, because the results are probably due to the simplistic way in which the CNN was designed for this exercise in order to illustrate the implementation, and the WFO shows that using a trading model based on this design would have very poor performance, which is expected, given the nature of the design.

4.6 Overall Results

Technical trading rules performed better in some of the variations of the WFO process, which suggests that they may be more suitable for some types of time-series phenomena. The statistical metrics suggest that both technical models have some degree of predictive ability, with a better fit in cumulative variations of the walk-forward validation. Some models lack predictive ability when time-series phenomena are not studied with the appropriate walk-forward variation. The CNN fails to show predictive ability, as we would expect from a naïve model that is not designed for the task,

but rather only for illustrative purposes about the implementation of the WFO algorithm for machine learning predictive models.

5. Conclusions and Considerations

The WFO algorithm developed here and its variations appear to be helpful tools to measure predictive ability. Based on prior experiments done in this paper this suggests that some models have some degree of predictive ability, depending on the variation of walk forward used, but some of the models are not suitable for this kind of time-series forecasting. The forecasting models proposed and tested were not designed so as to properly predict future observations because they are naïve implementations; rather, the main goal of the paper is to provide examples of the implementation and use of the WFO algorithm proposed. No techniques were used to tune the hyperparameters on the CNN, therefore, the results could be enhanced by tuning them or by setting a more complex CNN design, but that is out of the scope of this paper. Other types of deep learning and machine learning algorithms can be used to enhance the results. Empirical study is necessary to determine the best walk-forward parameters for training and validation sets, depending on the configuration chosen, the time series studied, and the type of model evaluated.

6. References

Allen, F., & Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. *Journal of Financial Economics, 51*, 245-271.

Bifet, A., & Kirkby, R. (2009). *Data Stream Mining A Practical Approach.* Hamilton: Centre for Open Software Innovation.

Brock, W., Lakonishok, J., & LeBaron, B. (1992, December). Simple technical trading rules and the stochastic properties of stock returns. *Journal of Finance, 47*(5), 1731-1764.

Cerqueira, V., Torgo, L., & Mozetic, I. (2019). Evaluating time series forecasting. An empirical study on performance estimation methods. Retrieved from https://arxiv.org/abs/1905.11744

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: deep neural networks with multitask learning. *Proceedings of the 25th International Conference on Machine Learning* (pp. 160-167). ICML '08.

Guo, X., Gao, Y., Li, Y., Zheng, D., & Shan, D. (2021). Short-term household load forecasting based on long- and short-term time-series network. *2020 International Conference on Power Engineering (ICPE 2020),*, (pp. 58-64). Guangzhou.

Ładyżyński, P., Żbikowski, K., & Grzegorzewski, P. (2013). Stock trading with random forests, trend detection tests and force index volume indicators. *International Conference on Artificial Intelligence and Soft Computing ICAISC*, (pp. 441-452).

Mehtab, S., & Sen, J. (2020). Stock price prediction using CNN and LSTM-Based deep learning models. *2020 International Conference on Decision Aid Sciences and Application (DASA)*, (pp. 447-453).

Ready, M. (2002). Profits from technical trading rules. *Financial Management, 31*(3), 44-61.

Robert, P. (2008). *The Evaluation and Optimization of Trading Strategies.* New York: Wiley.

Sullivan, R., Timmermann, A., & White, H. (1999). Data-snooping, technical trading rule performance, and the bootstrap. *Journal of Finance 54*, 1647-1691.

Suradhaniwar, S., Kar, S., & Jagarlapudi, A. (2021). Time series forecasting of univariate agrometeorological data: A comparative performance evaluation via one-step and multi-step ahead forecasting strategies. *Sensors 21, 2430*.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Forecasting stock prices from the limit order book using convolutional neural networks. *IEEE 19th Conference on Business Informatics*, (pp. 7-12).

Wang, J., Sun, T., Liu, B., Cao, Y., & Wang, D. (2018). Financial markets prediction with deep learning. *17th IEEE International Conference on Machine Learning and Applications*, (pp. 97-104). Orlando, FL.

White, H. (2000, September). A reality check for data snooping. *Econometrica, 68*(5), 1097-1126.

Zivot, E., & Wang, J. (2006). *Modeling Financial Time Series With S-PLUS.* New York: Springer.

Appendix 1: Python Code for Walk-Forward Optimization

```
1.
2.  def walkforwardTraining(series,k,r,Gr,v,Gv,u):
3.    #Create the sets (lists) of training.
4.    R=[]
5.    U=[]
6.    n = len(series)
7.    #Iterate over the entire series given "u" and the stop criteria "r+(k+Gr+Gv)(i-
    1)+u+v-1>n". This could have been achieved with a for loop too.
8.
9.    i=1
10.   while r+(k+Gr+Gv)*(i-1)+u+v-1<n:
11.     #Important to note: "+1" was added at the end of the slicing process since python
    removes the last value when slicing lists, this is a library way of work.
12.     R.append(series[0+k*(i-1):r+(k+Gr)*(i-1)-1+1])
13.     U.append(series[r+(k+Gr)*(i-1):r+(k+Gr)*(i-1)+u-1+1])
14.     i += 1
15.
16.   return R,U
17.
18. def walkforwardValidation(series,k,r,Gr,v,Gv,u,z,i):
19.   n = len(series)
20.   #Important to note: "+1" was added at the end of the slicing process since python
    removes the last value when slicing lists, this is a library way of work.
21.   V=series[r+(k+Gr)*(i-1)+u-z:r+(k+Gr+Gv)*(i-1)+u+v-1+1]
22.   Vi=r+(k+Gr)*(i-1)+u-z
23.
24.   return V,Vi
```