UNIVERSITÉ LIBRE DE BRUXELLES

FACULTÉ DES SCIENCES

DÉPARTEMENT D'INFORMATIQUE

Machine learning approach for multiple financial time series forecasting

Jérôme Bastogne

Promoteur :

Prof. Gianluca Bontempi

Ir. Jacopo De Stefani

Mémoire présenté en vue de
l'obtention du grade de
Licencié en Informatique

Année académique 2016 - 2017

Acknowledgements

Contents

Abstract

Chapter 1

Introduction

Chapter 2

Background

In this chapter, I'll review the basics of the prediction of time series in order to have a brief introduction to machine learning. Therefore this chapter will be a quick reminder of fundamentals alongside some formal basic definitions.

2.1 Efficient Market Hypothesis

The efficient-market hypothesis (EMH) states that stock prices fully reflect all available information and that prices are immediately corrected by any news. This would mean that the market is unbeatable and therefore, forecasting would be impossible as the main assumption of forecasting is that any information of the past has some incidence on the future. Also some evidence argue against EMH proving that stock market is not efficient. Indeed it wrongly assumes that all investors are always rational. Also, sometimes only a few traders have knowledge of a particular new information jeopardising this theory. However, if this theory would hold, the best possible forecast would only be the addition of noise [6] :

$$Y_{n+1} = Y_n + \epsilon_n$$

The question if the market is truly unbeatable still didn't reach any consensus. So while it is still only a theory, it is important to have an overview of the subject.

2.2 Definitions

2.2.1 Machine learning

A very well known quote from *Tom Mitchell* defines machine learning as :

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" – *Tom Mitchell, Carnegie Mellon University.*

Machine learning is helpful when there is a need of understanding a time series behaviour to make predictions on it, even if nothing is known about the nature of the time series. There exists different types of learning algorithms and some of them will be discussed in section 3.2.

2.2.2 Time series

A time series is a finite sequence of n data points $Y_t \epsilon \Re$ observed over time. Points are often equally spaced by time intervals. All points are ordered by their timestamp and changing that order would change the meaning of the data. [8]

Examples

- Numbers of sales of a company each month of year 2016.
- Temperature measurements in a country over 20 years.
- Electricity consumption in Belgium measured every half-hour for 100 days, representing 144000 values.
- Unemployment rate in the United States (figure 2.1).

Figure 2.1: Unemployment rate in the United States

2.2.3 Forecasting

One of the biggest interests of studying time series is forecasting. Indeed, based on the previous observations of a time series $Y_1, ..., Y_n$, could we predict the future observations $Y_{n+1}, Y_{n+2}, ...$?

It is possible under the strong assumption that a dependecy between the past and the future exists. The challenge is to find the ***model*** that describes the best the pattern of the time series.

Forecasts can be required for short-term, medium-term and long-term purposes. Short-term forecasting is about close in time scheduling and decision making. Medium-term forecasting helps to determine future resource requirements. And long-term forecasting is used for strategic planning with a long-term horizon. [8]

2.3 Stochastic models

Choosing a model is an important step in time series forecasting. A model's principal function is to describe how the most important ***features*** of a time series can be combined to form a meaningful prediction. A ***feature*** is an individual measurable characteristic, a variable. A model should explain how the past interacts with the future and if handling multiple time series, a model should explain how they affect each other. Finally, a model

should accurately forecast future values of a time series. [8]

We will first have a look at traditional stochastic models, such as ARMA and ARCH, that were studied for a very long time. Later, we will compare them with some machine learning algorithms that, in comparison, are still booming.

2.3.1 ARMA

ARMA is the main statistical model for univariate linear time series forecasting. ARMA is a combination of auto regressive (AR[p]) and moving average (MA[q]) models. Its general form is :

$$Y_t = \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t \tag{2.1}$$

where ϕ_i and θ_i are the constant parameters and ϵ_t are independent errors also called ***white noise***.

However, the problem with ARMA is that it requires the time series to be stationary which isn't always the case. Otherwise, when the variance isn't constant, an ARCH model is the better choice. [13]

2.3.2 ARCH & GARCH

ARCH in an auto regressive (AR) conditional heteroscedastic (CH) model. It is useful when studying financial time series because it illustrates the evolution of high variance changes for short periods of time, also described as volatility clustering. For example, if a market incur a substantial drop, it can alert people to sell their stuff before it drops even more. Therefore we can observe conditional periods of increased variance followed by calmer periods. [13]

So, if we use ARCH as an auto regressive model of the variance of the series y_t with zero mean, we can write the ARCH(q) model as :

$$\begin{gathered} y_t = \sigma_t \epsilon_t \\ with\ \epsilon_t \sim iid(0,1) \\ \sigma_t^2 = Var(y_t \mid y_{t-1}, ..., y_{t-q}) = \alpha_0 + \alpha_1 y_{t-1}^2 + ... + \alpha_q y_{t-q}^2 \end{gathered} \tag{2.2}$$

From there, GARCH (generalised ARCH) was introduced by adding moving-average components of the conditional variance to the ARCH model. The GARCH(p,q) model

changes the definition of σ_t^2 in equation 2.2 as :

$$\sigma_t^2 = \alpha_0 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 + \sum_{k=1}^{q} \alpha_k y_{t-k}^2 \tag{2.3}$$

where $\beta_1, ..., \beta_p \geq 0$ and $\alpha_1, ...\alpha_q \geq 0$. [13]

2.4 Volatility

Volatility defines the amplitude of variations of a time series. It is a key feature for financial time series analysis. Volatility represents a risk measure which is therefore directly related to risk management, a primary task of investors. Financial individuals are vulnerable to time changing volatility that causes uncertainty in risk assessment.

Volatility is a vague notion and an unobservable variable. It can be interpreted in multiple ways and therefore it has to be calculated with a proxy of conditional variance. It can either be seen as the standard deviation computed over a moving window of a determined size, it can be computed as the variance between returns, or it can be one of some of the definitions of volatility given by Garman and Klass [25] which will be described later in section 3.1.1.

While GARCH models are known to be very well suited for volatility forecasting as we have seen, I'll try to show through this thesis the capabilities of machine learning in that domain.

Note that volatility forecasting is directly linked with multi-step-ahead forecasting here. Indeed, knowing the risk degree only one day ahead is not enough for traders to react and take actions on the market. Having an idea of the behaviour of volatility several days in advance is a big deal.

Chapter 3

Methodology

This chapter is dedicated to the *modus operandi* of machine learning and forecasting. One can have an overall overview of the machine learning procedure on figure 3.1; the main ***procedures*** are represented on the figure by black rectangles, and the blue ones represent what we have before the corresponding procedures, and what we get after each procedure. We will first have a look at the pre-processing phases. Then we will see the theory behind the models used in the tool. Finally, we will see how to choose a strategy with a model in order to make good predictions and how to quantify their performance.

3.1 Pre-processing

3.1.1 Structure of the data

One can choose to use the tool to forecast the original data as is, or can choose to forecast the continuously compounded returns or the volatility of the data instead.

Log returns

Log returns are often more interesting for financial time series because they show an unscaled value comparable with other returns. It is defined as :

$$r_{t_i} = \ln \frac{P_{t_i}}{P_{t_{i-1}}} \tag{3.1}$$

where P_{t_i} defines the price at time i.

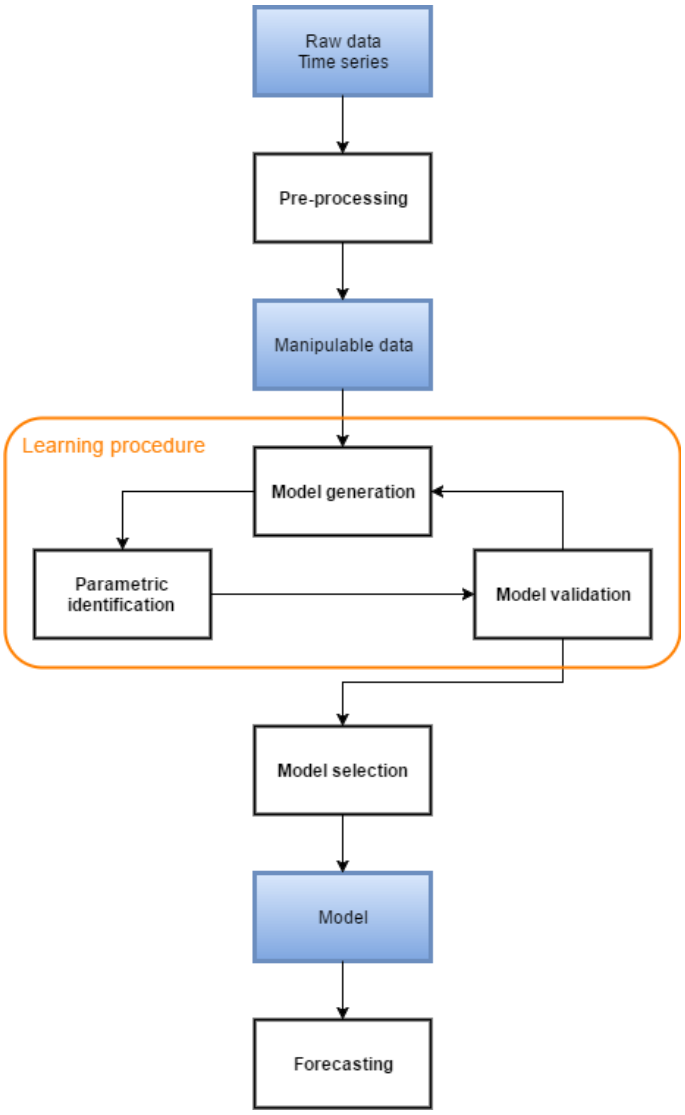


Figure 3.1: Time series learning procedure to forecasting.

Volatility

On the other side, volatility is the magnitude of changes in the prices of a financial asset. There are multiple ways of calculating the volatility of a financial time series, and one of the most suited should be one that considers historical OHLC prices (see 4.2). The estimator of volatility used in this tool is one of the most practical and efficient based on a study from Garman and Klass [25]:

$$\begin{gathered} u = \ln \frac{P_t^{(h)}}{P_t^{(o)}} \qquad d = \ln \frac{P_t^{(l)}}{P_t^{(o)}} \qquad c = \ln \frac{P_t^{(c)}}{P_t^{(o)}} \\ \hat{\sigma}\left(t\right) = 0.511\left(u-d\right)^2 - 0.019\left[c\left(u+d\right) - 2ud\right] - 0.383c^2 \end{gathered} \tag{3.2}$$

where $P_t^{(h)}$, $P_t^{(l)}$ and $P_t^{(c)}$ are respectively the high, low and opening prices at time t.

3.1.2 Normalisation

Normalisation is often required by some algorithms to ensure convergence and it is also a stable way of comparing multiple models and algorithms. More importantly, it prevents a feature from dominating the others by being much larger. Therefore, before using any machine learning algorithm, the data is normalised as follows :

$$x' = \frac{x - \mu}{\sigma} \tag{3.3}$$

so that we have a normal dataset. Note that μ, the mean of x, and σ, it's standard deviation, are calculated only on the training set with which the model will be trained. In this way, we avoid possible biases induced by not supposedly known information.

3.2 Models

3.2.1 Support vector machines

Definition

Support vector machines (SVM) is a supervised learning model that performs classification by constructing an multi-dimensional hyperplane that optimally separates the data into two categories.

SVM implements the structural risk minimisation principle which seeks to minimise an upper bound of the generalisation error rather than minimise the training error.

SVM uses linear models to implement nonlinear class boundaries through some nonlinear mapping of the input vectors into a higher dimensional feature space. A linear model constructed in the new space can represent a nonlinear decision boundary in the original space as seen on figure 3.2. In the new space, an optimal separating hyperplane is constructed. [19][20][23]

The support vectors are defined by the training examples that are closest to the maximum margin hyperplane and the support vector's number of coordinates are defined by the dimension space size. The maximum margin hyperplane gives the maximum separation between the decision classes as shown on figure 3.3.

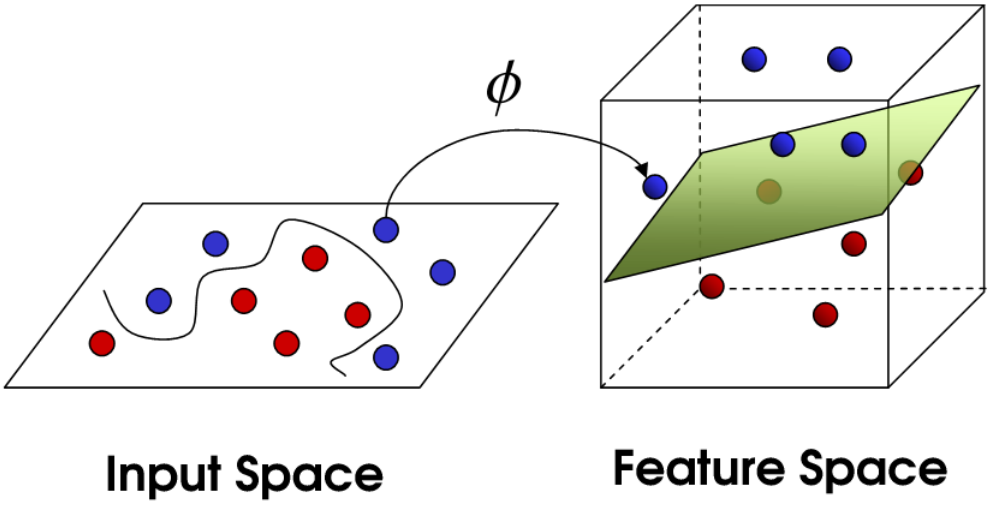


Figure 3.2: Transformation of the input space into a higher dimensional space with a kernel ϕ where the non-linearly separable data can be separated linearly by a hyperplane.

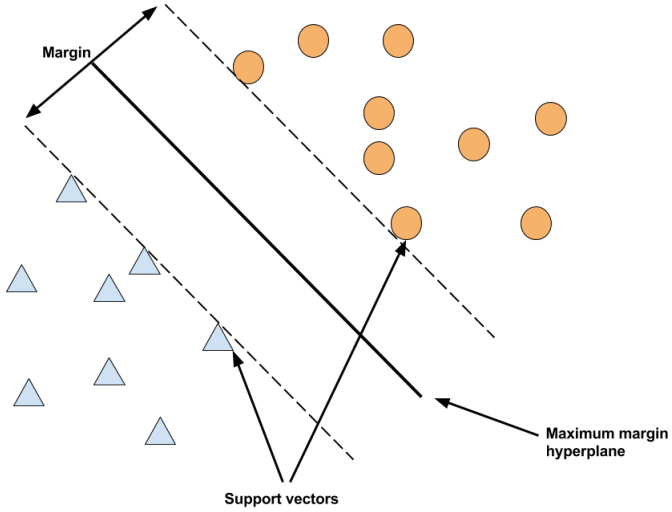


Figure 3.3: SVM linearly separable case.

Support vector machines can be applied to financial times series as a case of regression (SVR) by performing a linear regression in high dimensional feature spaces (instead of separating classes). They define the ***loss function*** that ignores errors which are situated within the certain distance of the true value.

The transformation of a space into a higher dimensional space is done via a ***Kernel***. Kernel functions have some tuning parameters. Overfitting can be avoided by allowing a small portion of the training data to lie outside the margin thanks to the slack parameter.

For financial time series forecasting, a combination of kernels may be used as information can behave from different sources. This introduces the importance of choosing good parameters and the good combination of kernels (kernel selection). [19][20][23]

Support vector regression theory

Support vector machines will be detailed more in depth as it is the most novel algorithm and also the most promising. Therefore, this subsection will describe the most important mathematical steps to understand the intrinsic working of SVMs.

Given a training set $\{(x_1, y_1), (x_2, y_2), ... (x_l, y_l)\}$ where y_i is the output for the vector input x_i and $x_i \in R^n, y_i \in R, \forall i \in \{1, 2, ..., l\}$, our goal is to find a function $f(x)$ as flat as possible to make predictions different from at most ε from those y_i values for all the training data. This is called ε-SVM regression. ε-SVM makes that small deviations in the input still leave to the same output, making the method more robust. The ε-sensitive loss function that allows errors inside the ε margin to be considered as zero is then given by :

$$L_\varepsilon = \begin{cases} 0 & if \ \ |y - f(x)| \leq \varepsilon \\ |y - f(x)| - \varepsilon & otherwise \end{cases} \tag{3.4}$$

[22][23]

The empirical risk (training error) is given by the means of the errors :

$$R_{emp} = \frac{1}{n} \sum_{i=1}^{n} L_\varepsilon (y_i, f(x_i)) \tag{3.5}$$

Let's see how it works with the linear case where our function f has the form :

$$f(x) = \langle w, x \rangle + b \text{ with } w \in R^n, b \in R \tag{3.6}$$

where $\langle ., . \rangle$ is the inner product.

To find a large-margin classifier, we want to minimise the norm of w. Therefore, by using regularisation our solution to the problem is :

$$\begin{gathered} \min \tfrac{1}{2} \|w\|^2 \\ \text{constrains: } |y_i - (\langle w, x_i \rangle + b)| \leq \varepsilon \end{gathered} \tag{3.7}$$

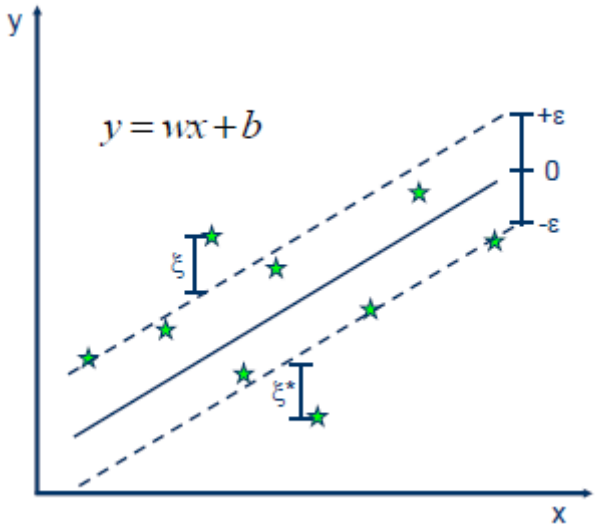


Figure 3.4: Soft margin with the ε bands and the ξ slack variables. [24]

We have to bear in mind the case where no such function $f(x)$ satisfying the constraints for all points exists. So we have a problem when the optimisation problem is infeasible. Therefore, similarly to the "soft margin" loss function, we add positive slack variables $\xi_i \xi_i^*$ to allow some regressions errors (see figure 3.4). The principle is to separate the training set with a minimal number of errors. The addition of slack variables gives us a new formulation of the solution as :

$$\begin{aligned} &\min \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ &\text{constrains} : \begin{cases} y_i - (\langle w, x_i\rangle + b) \leq \varepsilon + \xi_i \\ -(y_i - (\langle w, x_i\rangle + b)) \leq \varepsilon + \xi_i^* \\ \xi_i \geq 0 \\ \xi_i^* \geq 0 \end{cases} \end{aligned} \tag{3.8}$$

C is the constant parameter that represents the trade-off between the complexity of the model (i.e. the flatness of $f(x)$, regularisation) and the number of regression errors (i.e. values outside the ε margin, empirical risk). C is called the regularised constant and takes care of avoiding overfitting. The higher C is, the more we approach a hard-margin function and the less we will have misclassified individuals (overfitting).

By introducing Lagrange multipliers $\alpha_i \geq 0$, $\alpha_i^* \geq 0$ for each observation x_i, we can transform $f(x)$ and access it's dual form which is easier to solve, but also it will help us

for the nonlinear cases [22][23] :

$$\begin{aligned} \max -\tfrac{1}{2}\sum_{i,j=1}^{l}\left(\alpha_i-\alpha_i^*\right)\left\langle x_i,x_j\right\rangle-\varepsilon\sum_{i=1}^{l}\left(\alpha_i+\alpha_i^*\right)+\sum_{i=1}^{l}y_i\left(\alpha_i-\alpha_i^*\right) \\ \text{constrains}:\sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right)=0 \\ 0\leq\alpha_i\leq C \\ 0\leq\alpha_i^*\leq C \end{aligned} \tag{3.9}$$

This equation can be solved by Sequential Minimal Optimisation and the solution to the regression problem is finally given by [22][23]:

$$f\left(x\right)=\sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right)\left\langle x_i,x\right\rangle+b \tag{3.10}$$

For the nonlinear case, we don't search for the flattest function in the input space but rather in the feature space. The usefulness of the kernel comes from the fact that we don't have to explicitly compute the mapping of the input vector into the feature space $\phi(x)$. Instead we compute the inner products between the images of all pairs of data in the feature space very efficiently, this is known as the "kernel trick". The solution to the regression becomes :

$$f\left(x\right)=\sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right)K\left(x_i,x\right)+b \tag{3.11}$$

where $K\left(x_i,x\right)$ is the kernel function. The most popular kernel is named the Gaussian radial basis function kernel. The Gaussian RBF makes it the default kernel to use with no prior knowledge on the data and it is defined as :

$$K\left(x_i,x\right)=\exp\left(-\gamma\left\|x-x_i\right\|^2\right) \tag{3.12}$$

where γ is the parameter that defines how much influence a single training example can have. A small gamma means that the influence of x_i is more important (and so large variance, small bias) and vice versa.

As a final note, one should keep in mind that he can combine multiple kernels in order to better capture the feature's natures, especially if the input is made of different sources.

3.2.2 K-Nearest Neighbors

KNN is a local nonlinear classification model that can also be used for regression. KNNs are often referred to as one of the simplest of all machine learning algorithms. Its principle is to find in the training set the input that is the most similar because similar input would have similar outputs. Note that here the input is a vector of points, and the size of the vector is the order of the model.

Similarity between inputs can be computed with any distance function such as the Euclidean distance ($\sqrt{\sum_i (x_i - y_i)^2}$). The kNN model output is then given by the mean of the k-nearest neighbors output as follows :

$$f(x) = \frac{1}{k} \sum_{x' \in kNN(x)} f(x') \tag{3.13}$$

where $kNN(x)$ is the set of the k-nearest neighbors of x. There also exists a softer version of the k-nearest neighbors algorithm where instead of computing a mean, we compute a weighted average of each neighbor proportionally to their distance, the closer, the higher the weight is. The parameter k defines the bias-variance trade-off of the algorithm. The higher k is, the more samples we take from our training set, the lower the variance and the higher the bias are. In the other case, a small k will lead to overfitting. [27]

3.2.3 Naive model

Naive models can be computed in multiple ways. They generally are very simple and try to forecast the future as a mean of the past or a repetition of the last known value. In my case, I decided to implement the naive model by computing and repeating for each horizon, the mean of the last thirty values, i.e. a month more or less. So my naive model is defined as :

$$\hat{y}_t = \frac{1}{k} \sum_{i=0}^{k} y_{t-1-i} \tag{3.14}$$

where k is fixed to a month, so 30.

3.3 Forecasting strategies

Once we have a prediction model, we need a strategy in order to compute the actual predictions. If the model is our "regression function", then the strategy is how to apply it to the time series for forecasting. Predictions can be done in two ways :

- One-step-ahead prediction.
- Multi-step-ahead prediction.

Supervised learning approaches cover with success one-step-ahead prediction, which is the ability to predict the very next unknown value. However, multi-step-ahead prediction remains quite challenging. k-step-ahead prediction is the prediction of the next k unknown values of the time series.

3.3.1 Multi-step-ahead strategies

There exists some strategies for k-step-ahead prediction in the literature which the most common are [14][15]:

- **Recursive approaches** : (also called iterated approaches) we iterate k times with a one-step-ahead predictor. Each time a value ϕ_{t+1} is predicted, it is used as input for the next iteration and therefore, errors propagate consequently. Recursive strategies can therefore be biased.
- **Direct approaches** : we make k direct separate predictions ϕ_{t+h} $\vee h \in [1, k]$ for the next k unknown values, a different forecasting model being used for each horizon. Though, direct approaches makes a conditional independence assumption which may be incorrect to do. It also needs bigger time series than the iterated approach because of iterated generative's nature. Direct strategies have then less inputs in comparison. This leads to high variance for little time series or large horizon forecasting.
- **MIMO (multi-input multi-output) approaches** : The multi-input multi-output regression model returns a vector of k consecutive predictions $\{\phi_{t+1}, ..., \phi_{t+k}\}$ in one step. MIMO mimics the direct approach strategy but preserving the stochastic dependency of the time series. The problem involved with MIMO is that in

order to keep that stochastic dependency, all horizons have to be forecasted with the same model structure which can be restrictive. [14][15]

- **Rectify approach** : Rectify is a strategy that takes advantage of the strengths of both recursive and direct strategies. The principle is to first produce the predictions with the recursive strategy with a linear base model. As aforementioned, the results will be biased. Therefore, the second step of the strategy is to rectify the errors by adjusting the predictions with a nonlinear model with the direct strategy. This is supposed to reduce the bias of recursive linear systems. Rectify also proves to decrease variance's forecasts. [4]

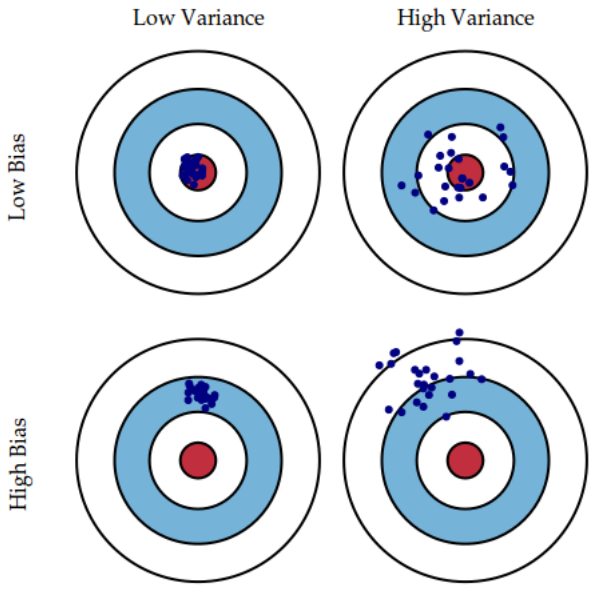


Figure 3.5: Fundamental bias-variance trade-off of forecasting strategies.

Dataset embeddings for multi-step forecasting

The strategies described in this section above require the data to be held into a certain data structure. The structures will be described for the strategies used in the tool: the recursive and direct ones.

Indeed, we want the model to learn how to predict a value Y at time t according to a number past values, so we need to train a model that is able to predict Y_t according to its k past values. The number of k past values, called the order of the model, is fixed beforehand.

E.g. , for the recursive approach, if the order of the model is 4, our model will try to predict Y_t according to $Y_{t-1}, Y_{t-2}, Y_{t-3}, Y_{t-4}$. Therefore our model is trained by passing a structure as shown on figure 3.6 where each input is a vector of k past values of each input Y_t in the training set, where k is the order of the model, and where the corresponding output is the value Y_t.

Recursive

- A single model for every horizon h.
- Forecast at step h based on forecast at step $h-1$.

input vector				**y**
y_1	y_2	y_3	y_4	y_5
y_2	y_3	y_4	y_5	y_6
...	...	...	...	...
y_{t-4}	y_{t-3}	y_{t-2}	y_{t-1}	y_t

Direct

- A single model for each horizon h.
- Forecast at h step is made using h^{th} model.

input vector				**y**
y_1	y_2	y_3	y_4	y_6
y_2	y_3	y_4	y_5	y_7
...	...	...	...	...
y_{t-7}	y_{t-6}	y_{t-5}	y_{t-4}	y_{t-2}

Figure 3.6: Data structures required for the recursive and direct approaches.

Therefore one additional step of the pre-processing is to embed the data into a similar structure than can be passed to a model according to the strategy used. Note that for the direct strategy, as explained, one model has to be built for each horizon h and therefore, each model requires the data to be re structured into a similar matrix but with different offsets of h.

3.4 Learning procedure

Now that we have covered the theory on models and strategies, we can have a look at the different phases of the learning procedure as shown on figure 3.1.

3.4.1 Model generation

We want to find a function that gives us the closest values to the output depending on the input, output being the forecasted values that we want to fit to the test data and the input being our training data. That function is often called the hypothesis. [4]

3.4.2 Parametric identification

The parametric identification selects the parameters that minimises the disparity between the forecasted values and the output. Note that not all models have parametric identification. In the case of the SVM, we search for the values of the cost and gamma parameters that reduce the most the gap between the forecasted values and the real output. For KNN, we have to find k as the number of instances that we take into account to determine similarities with classes. [4]

3.4.3 Model validation

The model validation phase is the process of examining the accuracy of the model. Estimating the errors is important, and if the evaluation is not satisfying enough, one should take the learning process back to model generation. A good way of measuring the performance of a model is by proceeding with a rolling window. Indeed, I chose a validation with a rolling window instead of a classical cross-validation because it ignores the temporal dependencies of time series. [4]

Rolling window forecasts

Rather than checking the performance of a model at one point in time, we would like to have a idea of how the model will generally perform. Sometimes we will have very accurate results, and other times the results will be worse, but we would like to have an idea of its general performance.

The principle is to use the model through the whole time series with a rolling window of a fixed size as shown on figure 3.7. For each window, the data is separated into a training and a testing set. The model is then trained and predictions are made. By comparing those with the actual values in the testing set, we can compute errors and by averaging those over the whole time series. In the end, we get an average of performance indicator.

In the web tool I designed, the minimum size of the window is taken as two-thirds of the dataset and the window is offset by the same value as the horizon.

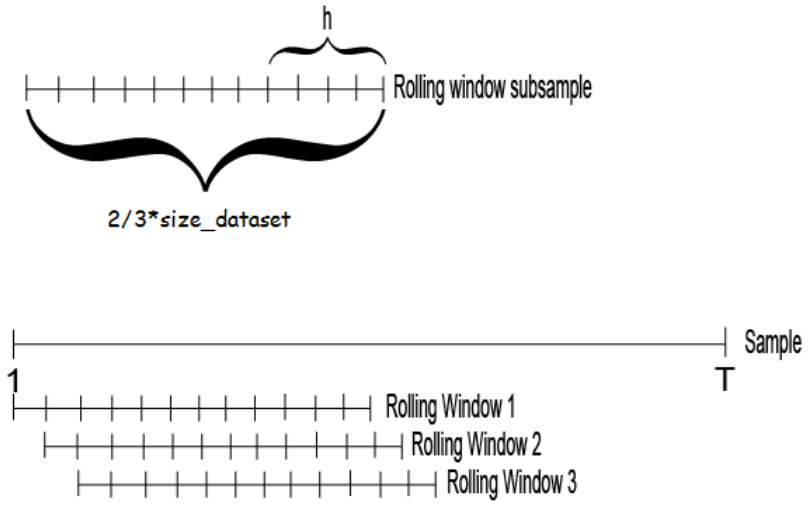


Figure 3.7: Rolling window principle[26].

Error measures and forecasting performance evaluation

This subsection will contain a brief description of all the errors measures that I show in my tool. I offer a large panel of choice regarding errors so that anyone can analyse the one that best suits its needs.

The errors shown in the tool are listed in the table below :

Full Name	Acronym	Formula
Mean Squared Error	MSE	$\frac{1}{n}\sum_{t=0}^{n}(y_t - \hat{y}_t)^2$
Root Mean Squared Error	RMSE	$\sqrt{\frac{1}{n}\sum_{t=0}^{n}(y_t - \hat{y}_t)^2}$
Mean Absolute Error	MAE	$\frac{1}{n}\sum_{t=0}^{n}\lvert y_t - \hat{y}_t\rvert$
Mean Absolute Percentage Error	MAPE	$\frac{1}{n}\sum_{t=0}^{n}\mid 100 \cdot \frac{y_t - \hat{y}_t}{y_t} \mid$
Scaled Mean Absolute Percentage Error	sMAPE	$\frac{100}{n}\sum_{t=0}^{n} \cdot \frac{\lvert y_t - \hat{y}_t\rvert}{\frac{y_t + \hat{y}_t}{2}}$
Mean Absolute Scaled Error	MASE	$\frac{1}{n}\sum_{t=1}^{n}\left(\frac{\lvert y_t - \hat{y}_t\rvert}{\frac{1}{n-1}\sum_{i=2}^{n}\lvert y_t - y_{t-1}\rvert}\right)$
Normalized Mean Squared Error	NMSE	$\frac{1}{n}\frac{\sum_{t=0}^{n}(y_t - \hat{y}_t)^2}{var(y_t)}$
Normalized Naive Mean Squared Error	NNMSE	$\frac{1}{n}\frac{\sum_{t=0}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=0}^{n}(y_t - \hat{y}_{Naive,t})^2}$

Remarkable notes :

- MAPE is generally used for reporting to outsiders, because it expresses an error in percentages that anyone can imagine and understand.
- MASE is often the best performing performance meter because it avoids most

of the problems induced by other error measures such as : scale independence, symmetry,...

- NNMSE is a relative error showing how a model is better performing than the naive method.

3.4.4 Model selection

Finally, model selection is the problem of choosing which method to use for a pool of possible methods. The best model should be chosen as the final model with the best related parametrisations. What is described as the best model is the model that performs the best on the validation set (i.e. testing set). That model should have the strongest capability of generalising, a good balance between fitting rightness and simplicity. [4]

Chapter 4

Forecasting tool

As a major contribution for my master's thesis, I developed a web application tool in *R* that offers an easy and fast way to compare different machine learning models, forecasting some financial time series. The tool runs online and allows anyone to choose a financial time series, modify it, apply some machine learning models on it and tuning their parameters.

The rest of this chapter includes a tutorial on how to use the tool, information on the data sources and the algorithms used in this tool and a section on how to include more models and data sources.

4.1 Presentation of the tool

This section describes how the tool works and what its components are. The web page is divided in three tabs : one for the data selection and visualisation, the second one to try and compare forecasting models and the last tab is for errors analysis.

Let's have a look at the first tab on figure 4.1 and its components descriptions.

4.1.1 Tab : Data inspection

1. This option allows one to choose a market. The corresponding data will be downloaded from *Yahoo*! *Finance*. The data is automatically refreshed when needed.

2. For each market there are different selectable options such as : the opening, closing, high and low prices and the volume.

Figure 4.1: View of the first tab of the tool.

3. Rather than using the raw data, a transformation can be selected among the volatility of the data and the compound returns.

4. The time series can be cut in time as desired by adjusting the starting date on the left and the end date on the right. This way, one can position his time series himself wherever he wants back in time.

5. A plot to visualise the data as chosen by all the parameters.

6. A summary of the data showing basic informations like the mean, first and third quartiles,...

4.1.2 Tab : Multi-step ahead forecasting

1. & 2. Those are the configurable parameters of the SVM : C, the cost 3.2.1 and γ, the Gaussian RBF kernel parameter.

3. The order of the model, i.e. the number of days which are used to predict a value Y at time t.

4. The horizon is the number of days we try to predict in multi-step ahead forecasting.

5. The strategies are defined in detail in section 3.3.

6. Those are some options to enable or disable the plotting of the models for a clearer visibility.

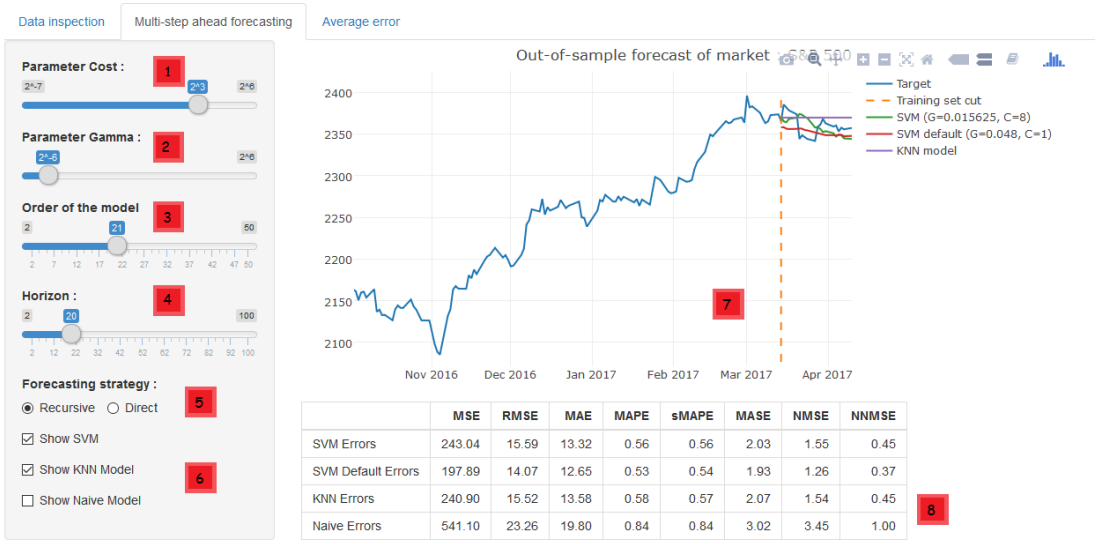


Figure 4.2: View of the second tab of the tool.

7. This is the resulting plot with its legend. The plot is made with the library **plotly** that gives a lot of options like zooming, downloading,... The orange dotted line shows the separation between the training set and the test set. Here, we are manually toying with parameters and therefore the line represents the horizon cut, which may be different during model validation.

8. This is a short table which resumes the errors of the models, as explained in section 3.4.3, for the current forecasting. For a more general idea of how a model is usually performing, one should refer to the third and last tab of the web page.

4.1.3 Tab : Average error

1. & 2. & 3. & 4. & 5. & 6. Those are the same parameters as those in the second tab.

7. This is a table of means of errors. Using the selected parameters, each model is applied through the whole dataset with a rolling window. The errors are averaged for each window over the whole dataset by applying the same model.

4.2 Datasets

The tool proposes some default datasets such as data from $CAC40$, $S\&P500$,... These datasets are directly downloaded from $Yahoo!\ Finance$ and are daily refreshed.

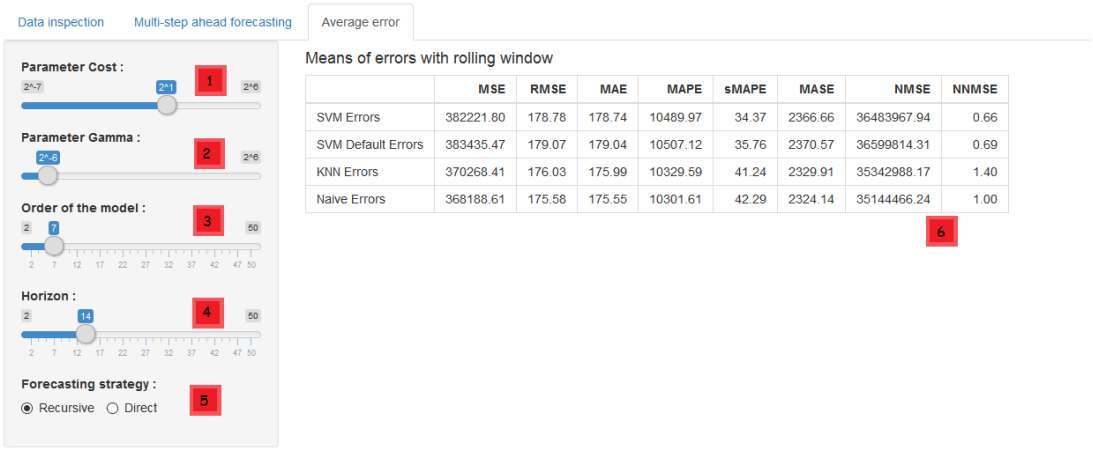


Figure 4.3: View of the third tab of the tool.

The data is publicly available and entirely free.

Each dataset is composed of : dates, opening, high, low and closing prices (OHLC) and the volumes of the stock for each weekday. The OHLC format is very often used to view stock movements and provides more information for volatility calculations as explained in section 3.2. The low and high prices are the lowest and highest prices indices for a day and the opening and closing are respectively, the price at the start of the day when the stock market opens and the price when the stock market closes.

The tool also ensures to refresh the data automatically when new data is available.

4.3 Libraries

The tool is written in *R* [28] and it's web client-server part is mostly handled by a library called **Shiny** [29].

Shiny is a library that proposes a web application framework that handles everything, including client-server, on itself. It allows one to only care about the functionalities of the code and the library will take care of how things are updated on the client part, how the client interacts with the server, ...

As for the rest, I use **Plotly** [30] for the nice visual appearance plots and their nice interaction with the user. I use **e1071** [31] for the SVM's models and **gbcode** [32] for the K-NNs models.

Finally, the **quantmod** [33] package is a quantitative financial modelling framework and can be used for modelling in finance. The function *getSymbols* of this library is

used to download the datasets from the Internet.

4.4 Code

Due to the **Shiny** library, the code of the tool is very loosely coupled and easy to read. The code is decomposed in one file for each of the client-server parts, one configuration file, one for the errors measurements and one file per machine learning algorithm.

There is no apparent links between the server and the client because **Shiny** handles it all. The client part contains a list of widgets to display accompanied by their positioning and a unique tag that refers to a variable from the server that they will be linked to. On the other side, the server contains all the logic of the program and stores its results in variables that the client will access thanks to their tag/variable name.

The configuration file, for its part, contains a list of datasets twinned with their respectively tag reference on $Yahoo!\ Finance$.

The rest of this section contains more information on how one could tweak the code and integrate new datasets and models.

4.4.1 Add new datasets

To add a new dataset, one has to find the tag reference of the market on $Yahoo!\ Finance$ and add it to the configuration file following the same pattern as those already added; see the content of the configration file below :

```
markets         <- c( "^FCHI",   "^IXIC",    "^GSPC")
names(markets) <- c("CAC 40", "NASDAQ", "S&P 500")
```

where the $markets$ are the tags, and the $names(markets)$ are the names displayed in the application.

4.4.2 Add new models

To add a new model, one can make a new function that takes in input a dataframe, a horizon and more parameters, and that returns the final predictions as a list of values. As an example, below is the header of the function that computes the predictions with SVM :

```
svmForecast <- function(df, horizon, window, gamma = 1/window,
                        cost = 1, strategy = "recursive"){
    ...
    return predictedY
}
```

On the client part, nothing has to be changed. Instead, there must be some inclusions of the new model computations on the server part.

The server has then to be updated by adding the new model in the following functions : *predPlot*, *predTable*, *errorTable*. They respectively correspond to the plotting function of the second tab, the predictions errors table on the second tab and the averaged errors of the rolling window table on the third tab of the tool. As an example, below is a piece of code of the function *predPlot* :

```
output$predPlot <- renderPlotly({
    # Initial plot
    p <- plot_ly(y = df$target, x = df$date, type = 'scatter',
                 mode = 'lines', name = 'Target')
    ...
    # Add forecasts for each model
    predictedY <- svmForecast(df, trainingcut(), window(), gamma(),
                              cost(), strategy())
    p <- p %>% add_trace(x = getTestDf()$date, y = predictedY,
                         mode = 'lines')
    ...
}
```

First, the plot is created with **plotly**, and then the traces of each model's predictions are added to the plot.

Another example is the function *predTable* that shows the errors table on the second tab of the tool :

```
output$predTable <- renderTable(rownames = TRUE, bordered = TRUE, {
    df <- getDf()
```

```
3       ...
4
5       # Get predictions for each model
6       svmPredY <- svmForecast(df, trainingcut(), modelOrder(), gamma(), cost
            (), strategy())
7       defaultSvmPredY <- svmForecast(df, trainingcut(), modelOrder(),
            strategy = strategy())
8       knnPredY <- modelSelectionKNN(df$target, trainingcut(), modelOrder(),
            strategy())$forecasts
9       naivePredY <- naiveForecast(df, trainingcut())
10
11      ...
12
13      # Show the means of all errors in table
14      as.data.frame(getErrors(svmPredY, defaultSvmPredY, knnPredY, naivePredY
            , getTestDf()$target))
15  })
```

As one can see, the code is just about forecasting values with each model and compute the errors with the actual values. Adding a new model here is about adding a call to the model and adding the predictions to the errors computations. The function *errorTable* is nearly the same. Note: the *getErrors*() function doesn't exists but it represents the actual computations of the table that in reality takes too much place as the table is very big.

A special note : the **Shiny** library handles the fact that if a same function is called for multiple outputs, it is only computed once and the same results are returned for each output. Therefore, we can call *svmForecast* for the plotting and the errors computations without doing over-computations because internally, **Shiny** handles it all and will update the results automatically if any of the parameters is changed.

So any new model can be included in those functions by following the same structure as the models already present. Nothing else has to be changed.

Chapter 5

Results

5.1 Description of my experiments

I conducted experiments to search for the best SVM parameters to perform forecasting with a search grid. I will compare the averaged MSE and MASE over a rolling window validation as explained in 3.4.3. First I will run the experiments on a reduced history of 6-12 months in order to select the most efficient model. After that, I will use the optimal models on a longer history (e.g. 10-15 years) to evaluate their performance over the long term and see if there is a model that is consistently performing better than the others.

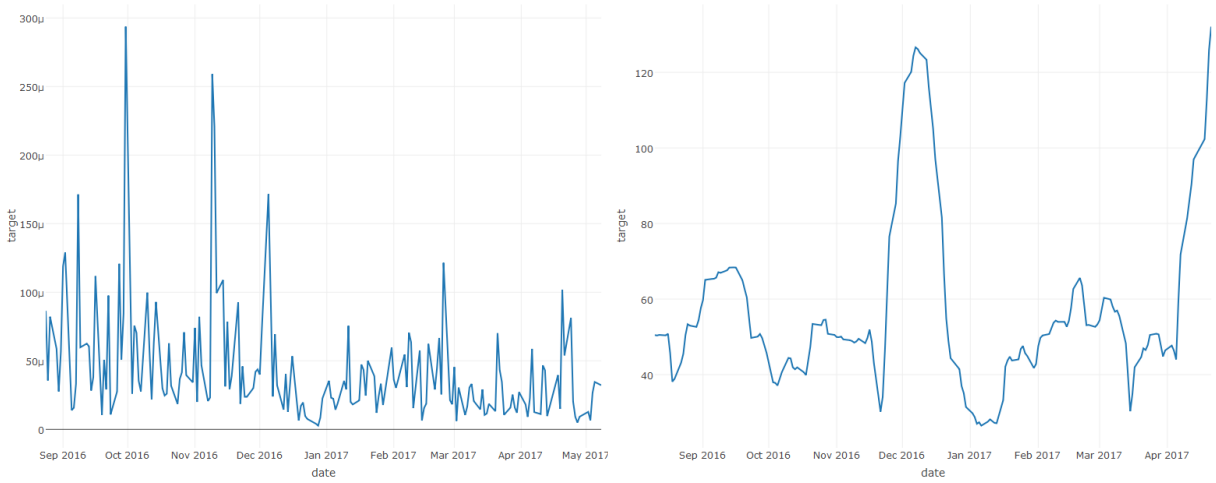
The experiments will be run on the volatility of the *CAC40* times series. I will use two different proxies for the volatility : the proxy described by equation 3.2 also used in the tool (proxy 1), and the rolling standard deviation with a window of 21 days (proxy 2). All the experiments are run with a window of 21 days and a horizon of 7 days which are values that I found to perform well while using my tool.

I made preliminary analysis by testing the SVM parameters with the tool I developed in order to better lead my grid searches. Therefore, not all results will be displayed with the same disparity.

Concerning the search grid process, searching through exponentially growing sequences of C and γ, such as powers of 2, is a practical method to identify good parameters. [34]

5.1.1 Reduced history

I first ran the experiment on a 9 months history. One will find on figure 5.1 the plots of the time series with the applied proxies.



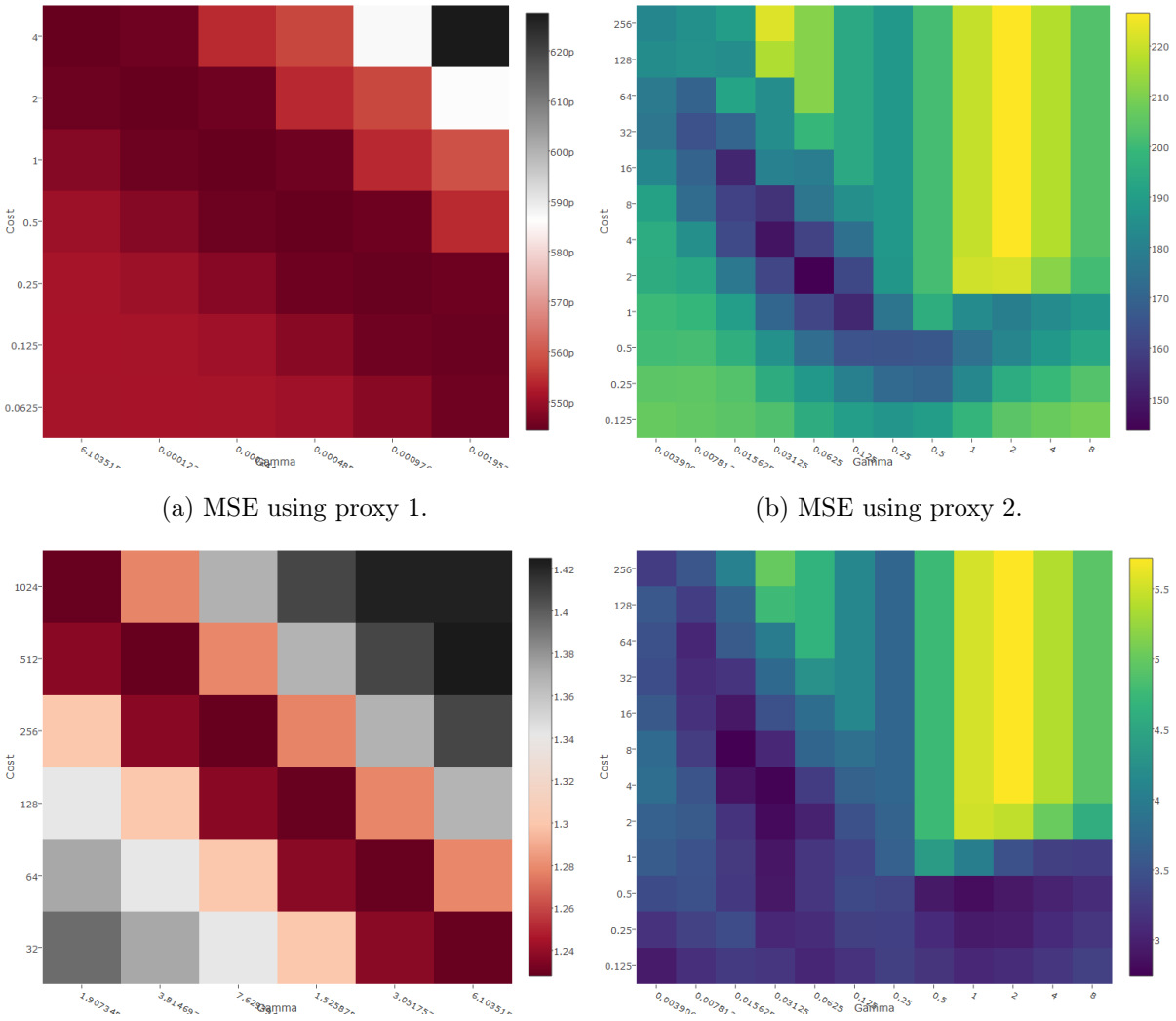
(a) Using proxy 1.

(b) Using proxy 2.

Figure 5.1: Plots of the reduced time series using the two proxies.

The results of the search grid can be visualised on the heat maps on figure 5.2 where the resulting errors are displayed according to the parameters. The complete numeric results of the search grid can be found in appendix A. As on can see on the heat map, the most optimal cost and gamma parameters seem to be inversely proportional because the darkest squares form a diagonal. This is an indication of how one should use those parameters.

Then on figure 5.3, one will find, for each proxy, the results of the experiments expressed as the cost and gamma that lead to the smallest related error measure.



(a) MSE using proxy 1.

(b) MSE using proxy 2.

(c) MASE using proxy 1.

(d) MASE using proxy 2.

Figure 5.2: Heat maps resulting of the search grid for each error using the two proxies.

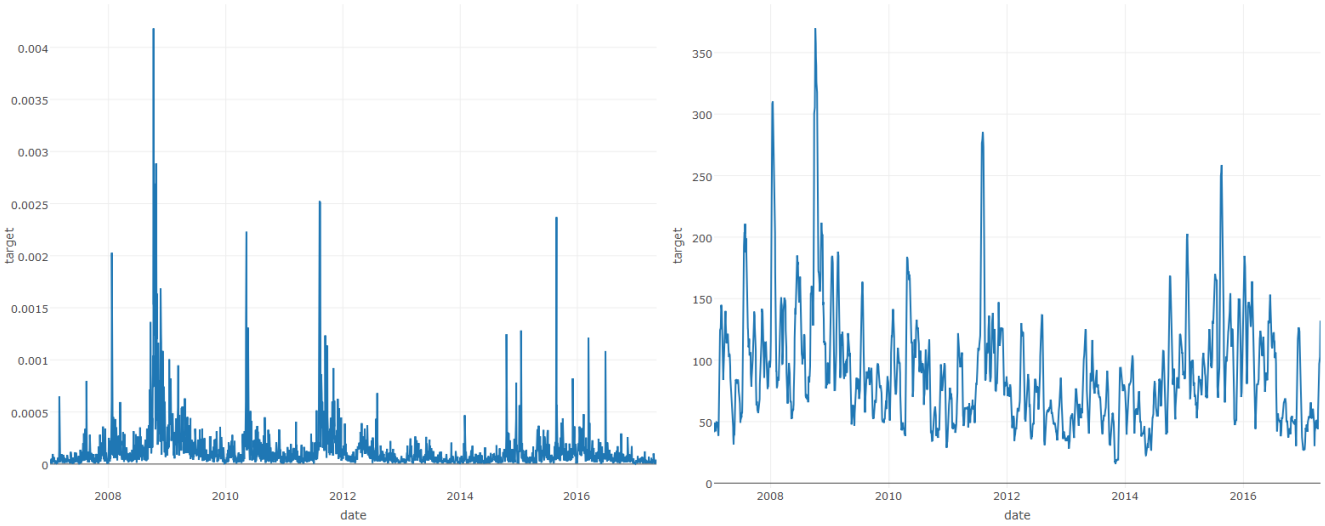
	Proxy 1	
	MSE	**MASE**
Cost	2	256
Gamma	0.0001220703	$7.629395e-06$
Error	$5.443549e-10$	1.227800

	Proxy 2	
	MSE	**MASE**
Cost	2	8
Gamma	0.0625	0.015625
Error	143.7454	2.742375

Figure 5.3: TODO

5.1.2 Longer history

Then, I ran the experiment on a 10 years history. One will find on figure 5.4 the plots of the time series with the applied proxies.



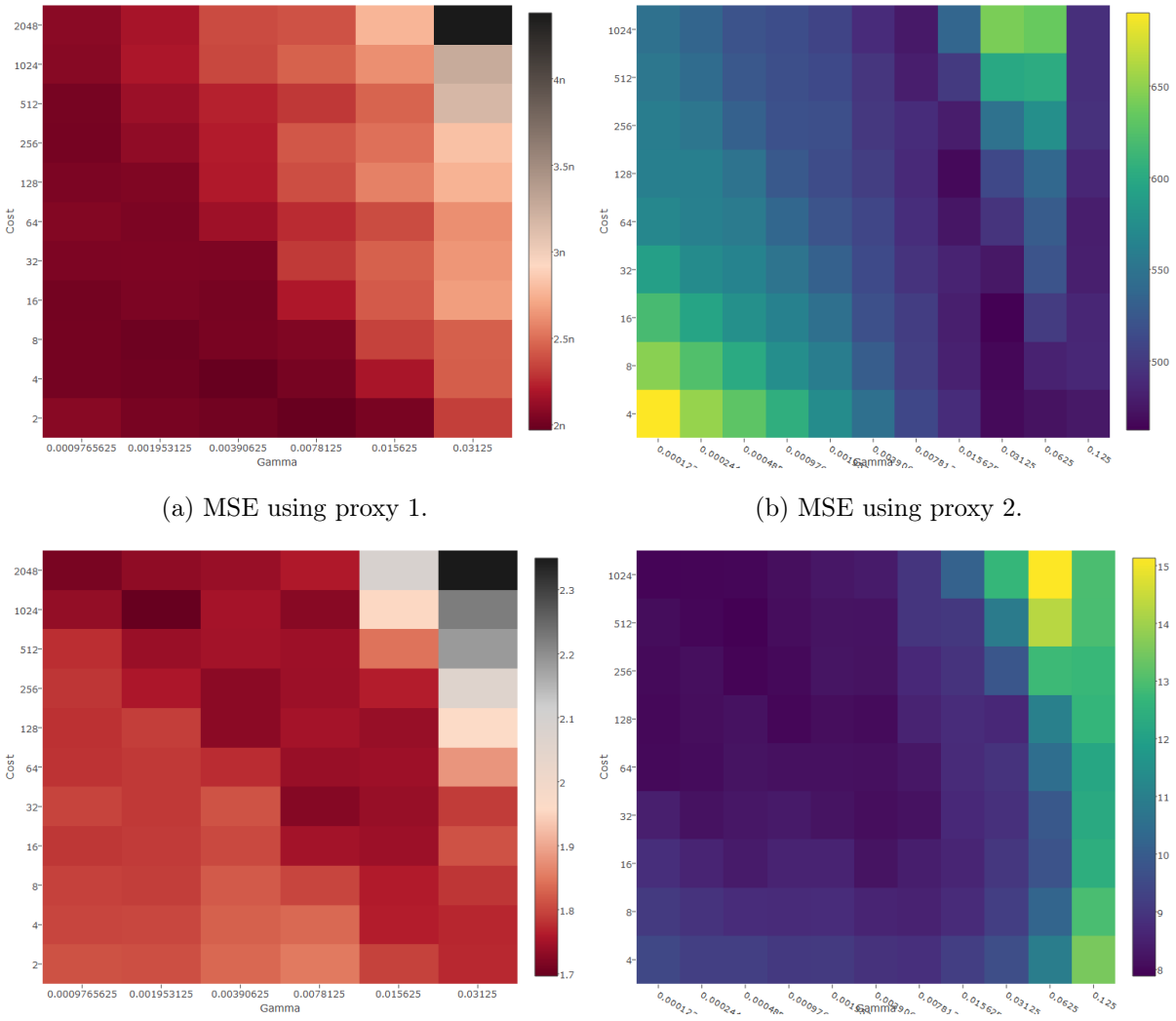
(a) Using proxy 1.

(b) Using proxy 2.

Figure 5.4: Plots of the longer time series using the two proxies.

The results of the search grid can be visualised on the heat maps on figure 5.5 where the resulting errors are displayed according to the parameters. The complete numeric results of the search grid can be found in appendix B.

Then on figure 5.6, one will find, for each proxy, the results of the experiments expressed as the cost and gamma that lead to the smallest related error measure.



(a) MSE using proxy 1.

(b) MSE using proxy 2.

(c) MASE using proxy 1.

(d) MASE using proxy 2.

Figure 5.5: Heat maps resulting of the search grid for each error using the two proxies.

	Proxy 1	
	MSE	**MASE**
Cost	4	1024
Gamma	0.00390625	0.001953125
Error	$1.969437e-09$	1.696371

	Proxy 2	
	MSE	**MASE**
Cost	16	512
Gamma	0.03125	0.0004882812
Error	462.2867	7.885050

Figure 5.6: TODO

Chapter 6

Conclusion & Future Work

6.1 Conclusion

6.2 Future work

Appendix A

Search grid on reduced history

		GAMMA					
	MSE	2^-14	2^-13	2^-12	2^-11	2^-10	2^-9
	2^-4	5.514144E-10	5.514746E-10	5.513528E-10	5.504903E-10	5.480246E-10	5.453717E-10
C	2^-3	5.514747E-10	5.513525E-10	5.504747E-10	5.479189E-10	5.452817E-10	5.447209E-10
O	2^-2	5.513524E-10	5.504670E-10	5.478657E-10	5.451697E-10	5.444985E-10	5.451554E-10
S	2^-1	5.504632E-10	5.478390E-10	5.450924E-10	5.444111E-10	5.451489E-10	5.541286E-10
T	2^0	5.478257E-10	5.450938E-10	5.443789E-10	5.452819E-10	5.540381E-10	5.590476E-10
	2^1	5.450847E-10	5.443549E-10	5.452804E-10	5.539904E-10	5.577905E-10	5.858172E-10
	2^2	5.443690E-10	5.453142E-10	5.540757E-10	5.574863E-10	5.870562E-10	6.275414E-10

Table A.1: Shortened results of the search grid for the SVM parameters for a period of 6 months with MSE using proxy 1.

		GAMMA					
	MASE	2^-19	2^-18	2^-17	2^-16	2^-15	2^-14
	2^5	1.394230E+00	1.372780E+00	1.341575E+00	1.299343E+00	1.237166E+00	1.228156E+00
C	2^6	1.372800E+00	1.341597E+00	1.299363E+00	1.237127E+00	1.228308E+00	1.278435E+00
O	2^7	1.341543E+00	1.299340E+00	1.236950E+00	1.228114E+00	1.277950E+00	1.367800E+00
S	2^8	1.299224E+00	1.236896E+00	1.227800E+00	1.277370E+00	1.368960E+00	1.407855E+00
T	2^9	1.236821E+00	1.227806E+00	1.278154E+00	1.368133E+00	1.407725E+00	1.424985E+00
	2^10	1.228114E+00	1.277693E+00	1.369221E+00	1.407579E+00	1.422009E+00	1.421993E+00

Table A.2: Results of the search grid for the SVM parameters for a period of 6 months with MASE using proxy 1.

		GAMMA											
	MSE	2^-8	2^-7	2^-6	2^-5	2^-4	2^-3	2^-2	2^-1	2^0	2^1	2^2	2^3
	2^-3	206.42	205.45	204.38	202.38	195.14	189.98	186.85	189.34	197.84	204.31	206.99	208.91
	2^-2	204.64	205.01	203.25	195.02	187.86	179.88	172.13	170.49	182.20	194.62	199.35	203.15
	2^-1	200.63	200.97	195.95	185.77	173.25	164.91	165.62	166.12	174.49	181.69	187.72	192.85
C	2^0	199.80	198.27	190.01	170.71	160.93	153.49	176.02	195.43	183.80	179.22	183.50	187.55
O	2^1	194.98	192.87	176.83	161.06	143.75	161.55	186.98	201.08	220.66	221.42	211.50	200.74
S	2^2	194.78	185.00	162.29	148.48	160.04	173.97	187.93	201.10	219.42	226.53	217.30	202.99
T	2^3	191.29	172.47	159.88	156.11	176.38	184.86	187.93	201.10	219.42	226.53	217.30	202.99
	2^4	181.92	170.10	152.84	180.04	178.63	193.97	187.93	201.10	219.42	226.53	217.30	202.99
	2^5	176.33	164.16	170.39	184.49	198.91	193.97	187.93	201.10	219.42	226.53	217.30	202.99
	2^6	177.71	169.77	191.88	184.24	211.09	193.97	187.93	201.10	219.42	226.53	217.30	202.99
	2^7	184.15	185.53	184.38	216.65	211.09	193.97	187.93	201.10	219.42	226.53	217.30	202.99
	2^8	181.79	185.21	189.34	222.38	211.09	193.97	187.93	201.10	219.42	226.53	217.30	202.99

Table A.3: Results of the search grid for the SVM parameters for a period of 6 months with MSE using proxy 2.

		GAMMA											
	MASE	2^-8	2^-7	2^-6	2^-5	2^-4	2^-3	2^-2	2^-1	2^0	2^1	2^2	2^3
	2^-3	2.94	3.13	3.24	3.20	3.05	3.17	3.29	3.18	3.06	3.11	3.21	3.30
	2^-2	3.16	3.32	3.43	3.06	3.11	3.29	3.31	3.09	2.94	2.96	3.09	3.20
	2^-1	3.42	3.48	3.24	2.91	3.20	3.41	3.35	2.93	2.83	2.92	3.02	3.10
C	2^0	3.59	3.50	3.23	2.90	3.20	3.34	3.64	4.38	4.02	3.48	3.29	3.28
O	2^1	3.65	3.58	3.17	2.81	3.01	3.46	3.69	4.76	5.51	5.44	5.02	4.59
S	2^2	3.81	3.51	2.88	2.75	3.26	3.66	3.73	4.76	5.53	5.72	5.38	4.92
T	2^3	3.77	3.28	2.74	3.06	3.71	3.85	3.73	4.76	5.53	5.72	5.38	4.92
	2^4	3.56	3.16	2.91	3.47	3.81	4.13	3.73	4.76	5.53	5.72	5.38	4.92
	2^5	3.43	3.11	3.19	3.76	4.25	4.13	3.73	4.76	5.53	5.72	5.38	4.92
	2^6	3.48	3.05	3.59	3.99	4.66	4.13	3.73	4.76	5.53	5.72	5.38	4.92
	2^7	3.55	3.28	3.68	4.76	4.66	4.13	3.73	4.76	5.53	5.72	5.38	4.92
	2^8	3.26	3.53	4.05	5.00	4.66	4.13	3.73	4.76	5.53	5.72	5.38	4.92

Table A.4: Results of the search grid for the SVM parameters for a period of 6 months with MASE using proxy 2.

Appendix B

Search grid on longer history

		GAMMA					
	MSE	2^-10	2^-9	2^-8	2^-7	2^-6	2^-5
	2^1	2.077E-09	2.027E-09	2.007E-09	1.974E-09	2.030E-09	2.324E-09
	2^2	2.016E-09	2.002E-09	1.969E-09	2.023E-09	2.182E-09	2.441E-09
	2^3	2.015E-09	1.992E-09	2.031E-09	2.051E-09	2.334E-09	2.449E-09
C	2^4	2.011E-09	2.040E-09	2.024E-09	2.198E-09	2.428E-09	2.671E-09
O	2^5	2.043E-09	2.045E-09	2.042E-09	2.305E-09	2.452E-09	2.641E-09
S	2^6	2.061E-09	2.037E-09	2.149E-09	2.261E-09	2.373E-09	2.616E-09
T	2^7	2.037E-09	2.051E-09	2.207E-09	2.382E-09	2.565E-09	2.756E-09
	2^8	2.019E-09	2.103E-09	2.214E-09	2.415E-09	2.506E-09	2.816E-09
	2^9	2.022E-09	2.140E-09	2.232E-09	2.299E-09	2.464E-09	3.171E-09
	2^10	2.075E-09	2.192E-09	2.358E-09	2.455E-09	2.616E-09	3.269E-09
	2^11	2.086E-09	2.177E-09	2.368E-09	2.397E-09	2.766E-09	4.381E-09

Table B.1: Results of the search grid for the SVM parameters for a period of 10 years with MSE using proxy 1.

	GAMMA						
	MASE	2^-10	2^-9	2^-8	2^-7	2^-6	2^-5
	2^1	1.81	1.81	1.83	1.85	1.79	1.77
	2^2	1.80	1.80	1.83	1.83	1.76	1.77
	2^3	1.79	1.79	1.82	1.80	1.76	1.78
C	2^4	1.78	1.79	1.80	1.75	1.74	1.81
O	2^5	1.80	1.79	1.81	1.72	1.74	1.79
S	2^6	1.78	1.79	1.78	1.74	1.74	1.88
T	2^7	1.78	1.79	1.73	1.75	1.74	1.96
	2^8	1.78	1.76	1.73	1.74	1.76	2.06
	2^9	1.78	1.74	1.75	1.74	1.84	2.19
	2^10	1.74	1.70	1.75	1.73	1.95	2.22
	2^11	1.71	1.73	1.74	1.76	2.09	2.35

Table B.2: Results of the search grid for the SVM parameters for a period of 10 years with MASE using proxy 1.

		GAMMA										
	MSE	2^-13	2^-12	2^-11	2^-10	2^-9	2^-8	2^-7	2^-6	2^-5	2^-4	2^-3
	2^2	689.84	652.30	629.70	605.13	572.33	545.67	510.68	490.46	466.88	471.65	475.91
	2^3	647.47	623.53	600.29	574.38	558.74	528.59	504.05	482.26	465.59	482.35	486.52
C	2^4	618.14	595.05	576.89	562.33	545.86	517.24	502.64	481.02	462.29	501.58	485.56
O	2^5	590.32	572.26	563.85	550.25	532.24	513.24	495.26	483.17	475.22	520.86	480.59
S	2^6	568.43	561.18	556.57	537.01	520.76	509.68	491.07	474.31	496.04	527.68	479.64
T	2^7	560.15	560.81	548.61	525.07	514.23	503.12	488.57	466.69	511.32	537.61	485.54
	2^8	558.71	551.63	532.41	518.77	516.82	498.86	490.47	478.86	547.95	575.43	493.55
	2^9	552.82	542.27	524.25	517.09	512.59	497.05	479.71	500.38	598.51	601.92	492.51
	2^10	546.50	535.82	520.20	515.10	508.30	489.82	476.60	536.28	642.16	635.38	492.12

Table B.3: Results of the search grid for the SVM parameters for a period of 10 years with MSE using proxy 2.

		GAMMA										
	MASE	2^-13	2^-12	2^-11	2^-10	2^-9	2^-8	2^-7	2^-6	2^-5	2^-4	2^-3
	2^2	9.45	9.21	9.20	9.07	9.09	8.91	8.85	9.19	9.60	10.93	13.52
	2^3	9.08	8.94	8.78	8.75	8.79	8.59	8.54	8.75	9.19	10.24	12.94
C	2^4	8.82	8.59	8.39	8.56	8.55	8.21	8.45	8.60	9.02	9.75	12.44
O	2^5	8.47	8.18	8.30	8.35	8.22	8.10	8.17	8.70	8.86	9.89	12.28
S	2^6	8.02	8.06	8.22	8.16	8.16	8.13	8.29	8.75	8.94	10.47	12.16
T	2^7	8.00	8.09	8.17	7.97	8.09	8.05	8.56	8.79	8.67	11.03	12.66
	2^8	8.03	8.11	7.94	8.03	8.23	8.19	8.70	8.94	9.83	12.79	12.73
	2^9	8.07	7.98	7.89	8.08	8.21	8.20	8.97	9.04	10.87	14.29	12.95
	2^10	7.94	7.95	7.97	8.13	8.28	8.38	8.98	10.15	12.70	15.13	12.96

Table B.4: Results of the search grid for the SVM parameters for a period of 10 years with MASE using proxy 2.

Bibliography

[1] E. Michael Azoff. *Neural Network Time Series Forecasting of Financial Markets*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[2] Francis E.H Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309 – 317, 2001.

[3] C.W.J. Granger, P. Newbold, and K. Shell. *Forecasting Economic Time Series*. Elsevier Science, 2014.

[4] Souhaib Ben Taieb. *Machine learning strategies for multi-step-ahead time series forecasting*. PhD thesis, Université Libre de Bruxelles, 2014.

[5] Souhaib Ben Taieb, Gianluca Bontempi, Amir F. Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert Syst. Appl.*, 39(8):7067–7083, June 2012.

[6] Rune Aamodt. *Using artificial neural networks to forecast financial time series*. Institutt for datateknikk og informasjonsvitenskap, 2010.

[7] Tristan Fletcher. Machine learning for financial market prediction. *UCL (University College London)*, 2012.

[8] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice:*. OTexts, 2014.

[9] Jeffrey Strickland. What is time series analysis?, 2015.

[10] R.A. Yaffee and M. McGee. *An Introduction to Time Series Analysis and Forecasting: With Applications of SAS® and SPSS®*. Elsevier Science, 2000.

[11] A. Azadeh, M. Sheikhalishahi, M. Tabesh, and A. Negahban. The effects of pre-processing methods on forecasting improvement of artificial neural networks. *Australian Journal of Basic and Applied Sciences*, 5(6):570–580, 2011.

[12] Manisha Gahirwal and M. Vijayalakshmi. Inter time series sales forecasting. *IJASCSE*, 2(1), 2013.

[13] Scott H. Holan, Robert Lund, and Ginger Davis. The arma alphabet soup: A tour of arma model variants. *Statistics Surveys*, 4:232–274, 12 2010.

[14] Gianluca Bontempi and Souhaib Ben Taieb. Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International Journal of Forecasting*, 27(3):689 – 699, 2011. Special Section 1: Forecasting with Artificial Neural Networks and Computational IntelligenceSpecial Section 2: Tourism Forecasting.

[15] S. B. Taieb, G. Bontempi, A. Sorjamaa, and A. Lendasse. Long-term prediction of time series by combining direct and mimo strategies. In *2009 International Joint Conference on Neural Networks*, pages 3054–3061, June 2009.

[16] R. Bharat Rao, Scott Rickard, and Frans Coetzee. Time series forecasting from high-dimensional data with multiple adaptive layers. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York City, New York, USA, August 27-31, 1998*, pages 319–323, 1998.

[17] Rob J Hyndman. Cran task view: Time series analysis., 2016.

[18] Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, 27(3):1–22, 3 2008.

[19] Kyoung-jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307–319, 2003.

[20] Li Wang and Ji Zhu. Financial market forecasting using a two-step kernel learning method for the support vector regression. *Annals of Operations Research*, 174(1):103–120, 2 2010.

[21] Brian Ripley. nnet: Feed-forward neural networks and multinomial log-linear models., 2016.

[22] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[23] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, August 2004.

[24] Saed Sayad. *Real Time Data Mining*. Self-Help Publishers, 2011.

[25] Mark B Garman and Michael J Klass. On the Estimation of Security Price Volatilities from Historical Data. *The Journal of Business*, 53(1):67–78, January 1980.

[26] Rolling-window analysis of time-series models, 2006.

[27] A. Navot, L. Shpigelman, N. Tishby, and Vaadia. Nearest neighbor based feature selection for regression and its application to neural activity. In *Proc. of NIPS*, 2006.

[28] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[29] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2016. R package version 0.14.2.

[30] Carson Sievert, Chris Parmer, Toby Hocking, Scott Chamberlain, Karthik Ram, Marianne Corvellec, and Pedro Despouy. *plotly: Create Interactive Web Graphics via 'plotly.js'*, 2016. R package version 4.5.6.

[31] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. R package version 1.6-7.

[32] Gianluca Bontempi. *gbcode: Code from the handbook "Statistical foundations of machine learning"*, 2016. R package version 1.0.

[33] Jeffrey A. Ryan. *quantmod: Quantitative Financial Modelling Framework*, 2016. R package version 0.4-7.

[34] Chih wei Hsu, Chih chung Chang, and Chih jen Lin. A practical guide to support vector classification, 2010.