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

Contents

Abstract

Chapter 1

Introduction

Chapter 2

Background

Chapter 3

Related work

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 permits anyone to choose a financial time-series, modify it, apply some machine learning models on it and toy with their parameters. The tool is very intuitive to use and the science behind it will be described in the following sections.

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.

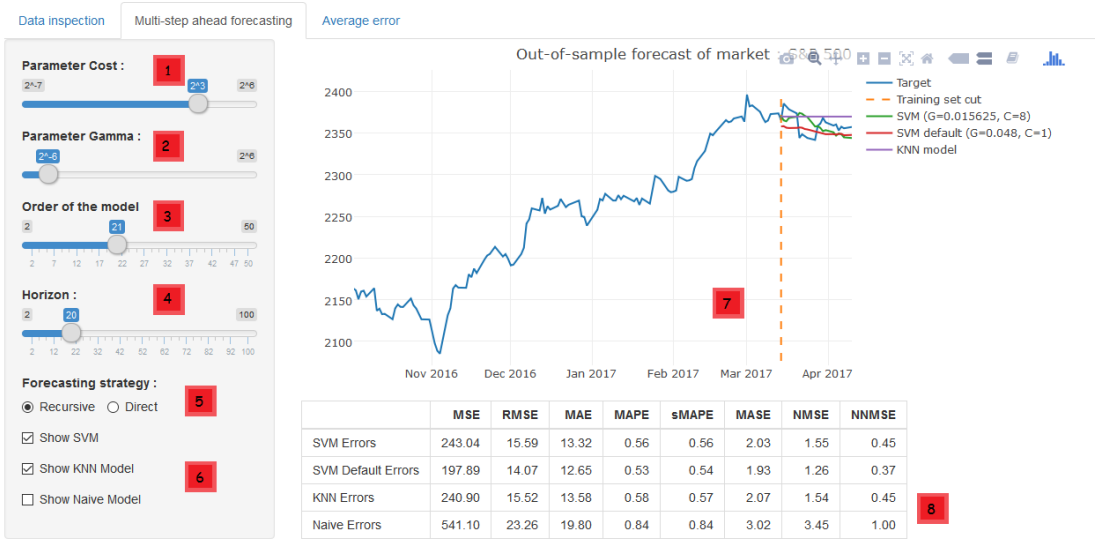
3. Rather than using the raw data, a transformation can be selected among the volatility of the data and the compound returns.

4. The time series can be cut in time as desired.

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

6. A summarise of the data.

4.1.2 Tab : Multi-step ahead forecasting



	MSE	RMSE	MAE	MAPE	sMAPE	MASE	NMSE	NNMSE
SVM Errors	243.04	15.59	13.32	0.56	0.56	2.03	1.55	0.45
SVM Default Errors	197.89	14.07	12.65	0.53	0.54	1.93	1.26	0.37
KNN Errors	240.90	15.52	13.58	0.58	0.57	2.07	1.54	0.45
Naive Errors	541.10	23.26	19.80	0.84	0.84	3.02	3.45	1.00

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

1. & 2. Those are the configurable parameters of the SVM.

3. The order of the model, i.e. the number of days which are used to predict a value Y at time t. E.g. if the order of the model is 4, our model tries to predict Y_t according to $Y_{t-1}, Y_{t-2}, Y_{t-3}, Y_{t-4}$.

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

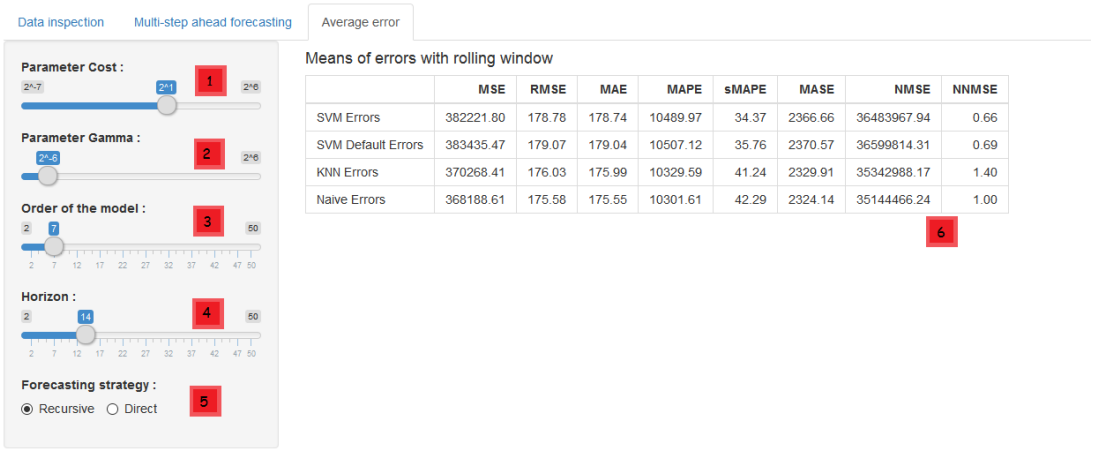
5. The strategies are defined in detail in section 5.3.

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

7. This is the resulting plot with it's 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 model 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



	MSE	RMSE	MAE	MAPE	sMAPE	MASE	NMSE	NNMSE
SVM Errors	382221.80	178.78	178.74	10489.97	34.37	2366.66	36483967.94	0.66
SVM Default Errors	383435.47	179.07	179.04	10507.12	35.76	2370.57	36599814.31	0.69
KNN Errors	370268.41	176.03	175.99	10329.59	41.24	2329.91	35342988.17	1.40
Naive Errors	368188.61	175.58	175.55	10301.61	42.29	2324.14	35144466.24	1.00

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

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 data set with a rolling window. The errors are averaged for each window over the whole data set by applying the same model.

4.2 Data sets

The tool proposes some default data sets such as data from $CAC40$, $S\&P500$,... These data sets are directly downloaded from $Yahoo!\ Finance$ and are daily refreshed. The data is publicly available and entirely free.

Each data set is composed of : dates, opening, closing, high and low prices and the volumes of the stock. The tool also ensures to refresh the data automatically when new data is available.

4.3 Libraries

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

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** for the nice visual appearance plots and their nice interaction with the user. I use **e1071** for the SVM's models and **gbcode** for the K-NNs models. Finally, the **quantmod** library is used to download the data sets from the Internet.

4.4 Code

The code of the tool is very loosely coupled, mostly due to the **Shiny** library. 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 data sets twinned with their respectively tag reference on $Yahoo!\ Finance$.

4.4.1 Add new data sets

To add a new data set, 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. That's it.

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. 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 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. The 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

Scientific Research

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 5.1; 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.

5.1 Pre-processing

5.1.1 Structure of the data

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

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

$$r_{t_i} = \ln \frac{P_{t_i}}{P_{t_{i-1}}} \tag{5.1}$$

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

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 opening, closing, high, and

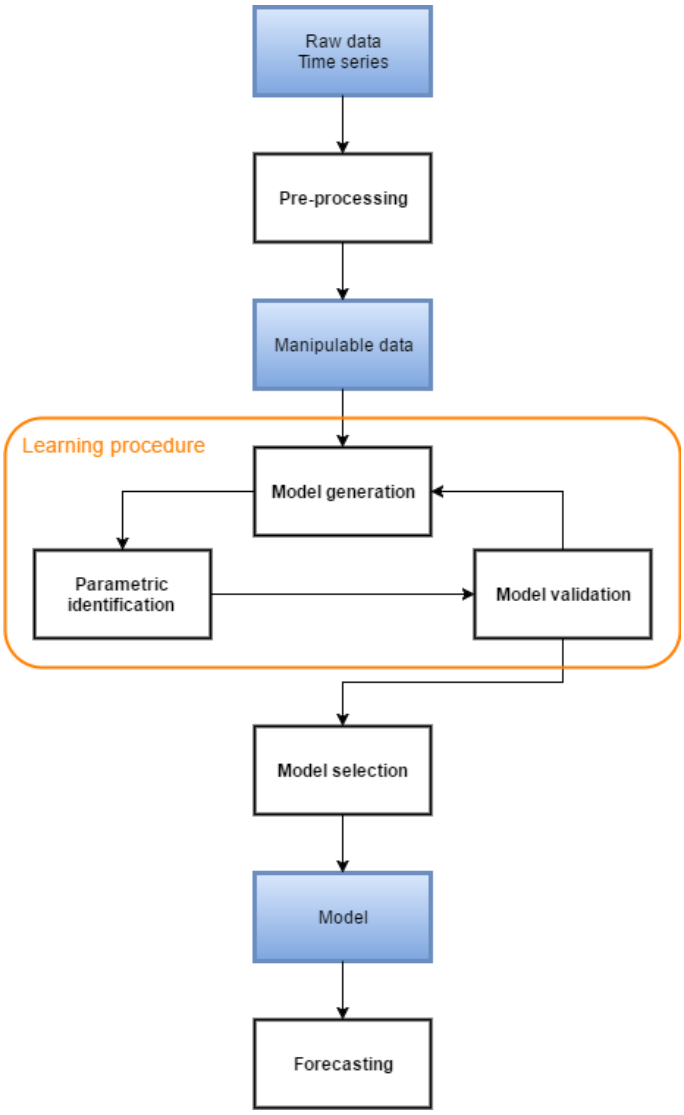


Figure 5.1: Time series learning procedure to forecasting.

low prices and the volume. The estimator of volatility used in this tool is one of the most practical and efficient based on a study from Garman and Klass [34]:

$$\begin{gathered} u = \ln \frac{P_t^{(h)}}{P_t^{(o)}} \qquad d = \ln \frac{P_t^{(l)}}{P_t^{(o)}} \qquad c = \ln \frac{P_t^{(c)}}{P_t^{(o)}} \\ \hat{\sigma}\left(t\right) = 0.511\left(u-d\right)^2 - \left(2\ln 2 - 1\right)c^2 \end{gathered} \tag{5.2}$$

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

5.1.2 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 :

$$x' = \frac{x - \mu}{\sigma} \tag{5.3}$$

so that we have a normal data set. 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.

5.2 Models

5.2.1 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 5.2. In the new space, an optimal separating hyperplane is constructed. [28][29][32]

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 5.3.

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

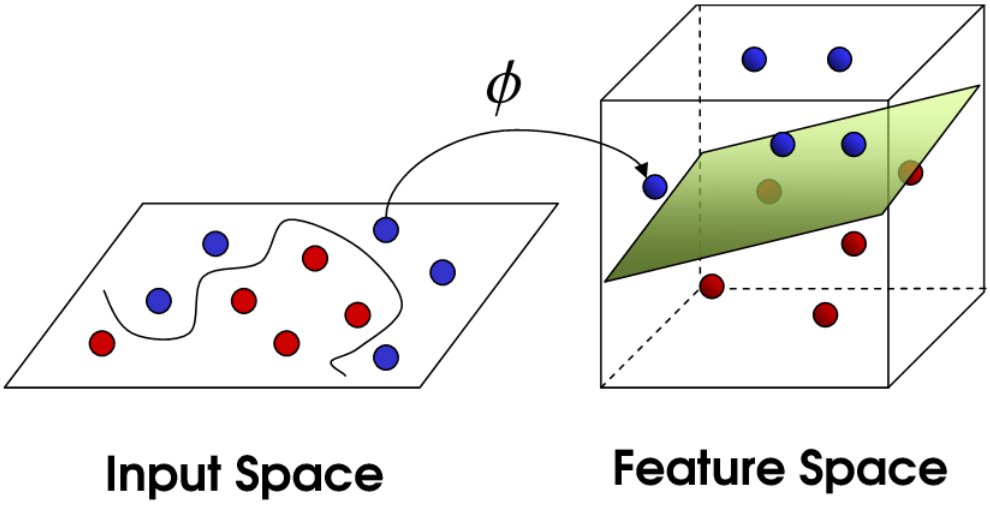


Figure 5.2: Separation by higher dimensional hyperplanes can be easier.

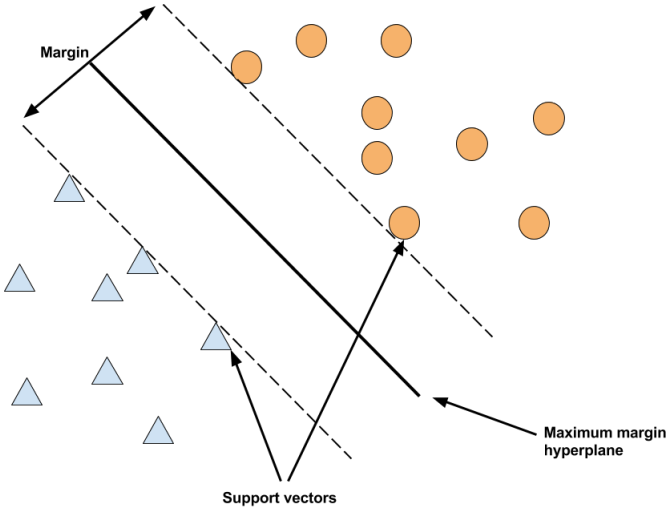


Figure 5.3: SVM linearly separable case.

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). [28][29][32]

Support vector regression theory

Support vector machines deserves a more in depth analysis as it is the algorithm I studied the most for my thesis. 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 :

$$L_\varepsilon = \begin{cases} 0 & if \ \ |y - f(x)| \leq \varepsilon \\ |y - f(x)| - \varepsilon & otherwise \end{cases} \tag{5.4}$$

[31][32]

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

$$R_{emp} = \frac{1}{n} \sum_{i=1}^{n} L_\varepsilon (y_i, f(x_i)) \tag{5.5}$$

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

$$f(x) = \langle w, x \rangle + b \text{ with } w \in R^n, b \in R \tag{5.6}$$

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 :

$$\begin{gathered} \min \tfrac{1}{2} \|w\|^2 \\ \text{constrains: } |y_i - (\langle w, x_i \rangle + b)| \leq \varepsilon \end{gathered} \tag{5.7}$$

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 5.4). The principle is to sepa-

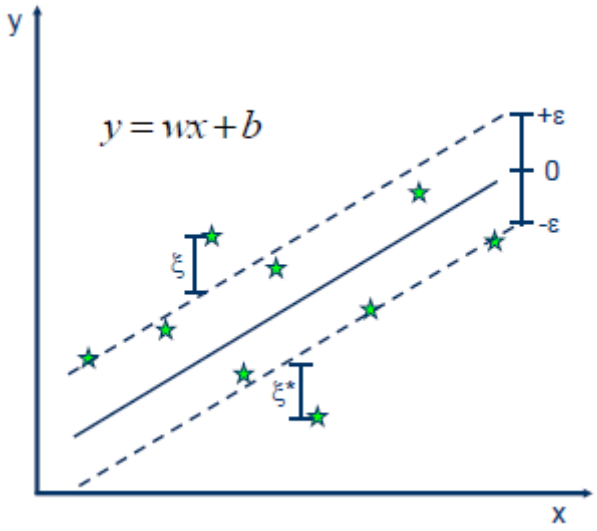


Figure 5.4: Soft margin with the ε bands and the ξ slack variables. [33]

rate the training set with a minimal number of errors. The addition of slack variables gives us a new formulation of the solution as :

$$\begin{aligned} &\min \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ &\text{constrains} : \begin{cases} y_i - (\langle w, x_i\rangle + b) \leq \varepsilon + \xi_i \\ -(y_i - (\langle w, x_i\rangle + b)) \leq \varepsilon + \xi_i^* \\ \xi_i \geq 0 \\ \xi_i^* \geq 0 \end{cases} \end{aligned} \tag{5.8}$$

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 [31][32] :

$$\begin{aligned} \max -\tfrac{1}{2}\sum_{i,j=1}^{l}(\alpha_i-\alpha_i^*)\langle x_i,x_j\rangle-\varepsilon\sum_{i=1}^{l}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{l}y_i(\alpha_i-\alpha_i^*) \\ \text{constrains}:\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)=0 \\ 0\le\alpha_i\le C \\ 0\le\alpha_i^*\le C \end{aligned} \tag{5.9}$$

This equation can be solved by Sequential Minimal Optimisation and the solution to the regression problem is finally given by [31][32]:

$$f(x)=\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)\langle x_i,x\rangle+b \tag{5.10}$$

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 :

$$f(x)=\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)K(x_i,x)+b \tag{5.11}$$

where $K(x_i,x)$ is the kernel function. 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.

5.2.2 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. 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 :

$$f(x)=\frac{1}{k}\sum_{x'\in kNN(x)}f(x') \tag{5.12}$$

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. [36]

5.2.3 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.

5.3 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.

5.3.1 Multi-step-ahead strategies

There exists some strategies for k-step-ahead prediction in the literature which the most common are [15][16]:

- **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 $\phi_{t+h} \vee h \in [1, k]$ 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. [15][16]

- **Rectify approach** : Rectify is a strategy that takes advantage of the strengths of both recursive and direct strategies. The principle is to produce the predictions with the recursive strategy with a linear base model. As aforementioned, the results will be biased. Therefore we 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. [5]

5.4 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 5.1.

5.4.1 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. [5]

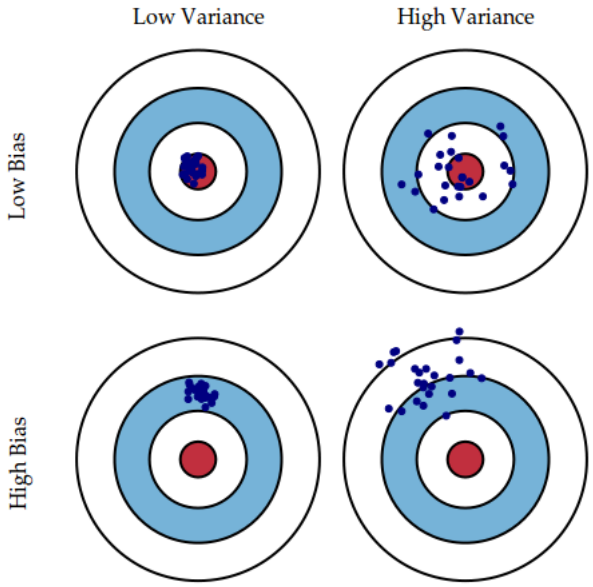


Figure 5.5: Fundamental bias-variance trade-off of forecasting strategies.

5.4.2 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. [5]

5.4.3 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. [5]

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 5.6. 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 data set and the window is offset by the same value as the horizon.

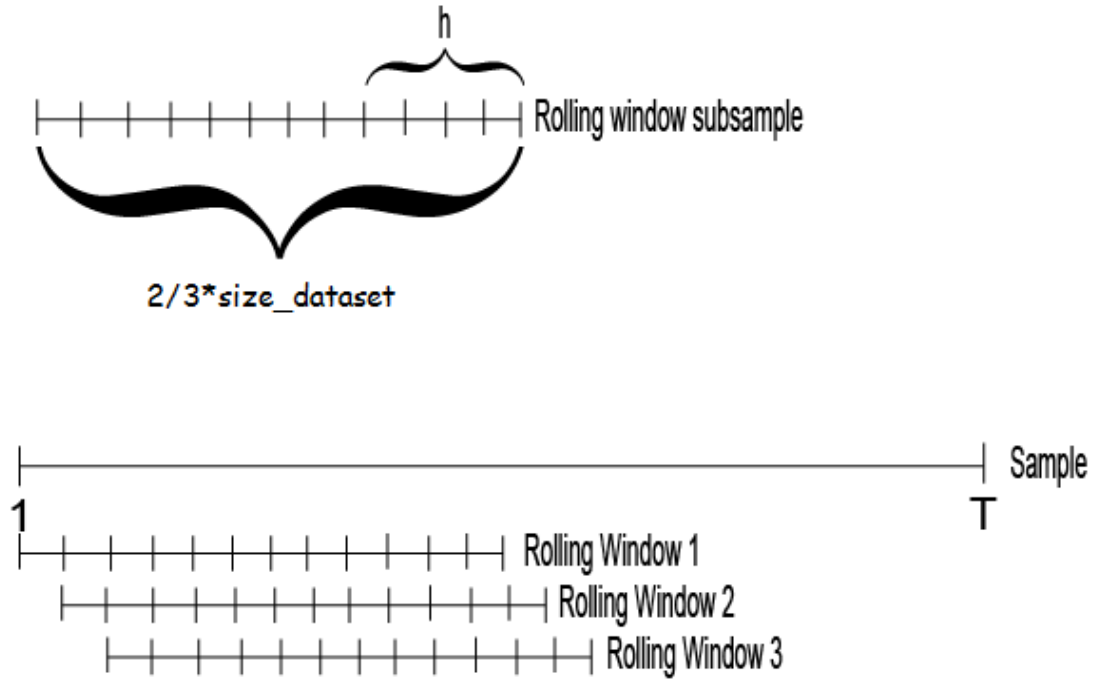


Figure 5.6: Rolling window principle[35].

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 showed are : MSE - Mean Squared Error, RMSE - Root Mean Squared Error, MAE - Mean Absolute Error, MAPE - Mean Absolute Percentage Error, sMAPE - Scaled Mean Absolute Percentage Error, MASE - Mean Absolute Scaled Error, NMSE - Normalized Mean Squared Error and the NNMSE - Normalized Naive Mean Squared Error and they are defined as follows :

$$
\begin{aligned}
MSE &= \frac{1}{n}\sum_{t=0}^{n}(y_t - \hat{y}_t)^2 \\
RMSE &= \sqrt{\frac{1}{n}\sum_{t=0}^{n}(y_t - \hat{y}_t)^2} \\
MAE &= \frac{1}{n}\sum_{t=0}^{n}|y_t - \hat{y}_t| \\
MAPE &= \frac{1}{n}\sum_{t=0}^{n} \mid 100 \cdot \frac{y_t - \hat{y}_t}{y_t} \mid \\
sMAPE &= \frac{100}{n}\sum_{t=0}^{n} \cdot \frac{|y_t - \hat{y}_t|}{\frac{y_t + \hat{y}_t}{2}} \\
MASE &= \frac{1}{n}\sum_{t=1}^{n}\left(\frac{|y_t - \hat{y}_t|}{\frac{1}{n-1}\sum_{i=2}^{n}|y_t - y_{t-1}|}\right) \\
NMSE &= \frac{1}{n}\frac{\sum_{t=0}^{n}(y_t - \hat{y}_t)^2}{var(y_t)} \\
NNMSE &= \frac{1}{n}\frac{\sum_{t=0}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=0}^{n}(y_t - \hat{y}_{Naive,t})^2}
\end{aligned}
\tag{5.13}
$$

Remarkable notes :

- MAPE is generally used for reporting to outsiders, because it expresses an error in percentages that anyone can imagine.
- MASE is often the best performing performance meter because it avoids most of the problems induced by other error measures such as : scale independent, symmetry,...
- NNMSE is a relative error showing how a model is better performing than the naive method.

5.4.4 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 a good balance between fitting rightness and simplicity. [5]

Chapter 6

Results

Chapter 7

Conclusion & Future Work

7.1 Conclusion

7.2 Future work

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] Andy Doyle, Graham Katz, Kristen Summers, Chris Ackermann, Ilya Zavorin, Zunsik Lim, Sathappan Muthiah, Patrick Butler, Nathan Self, Liang Zhao, Chang-Tien Lu, Rupinder Paul Khandpur, Youssef Fayed, and Naren Ramakrishnan. Forecasting significant societal events using the embers streaming predictive analytics system. *Big Data*, 2(4):185 – 195, 12/2014 2014.

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

[6] 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.

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

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

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

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

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

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 2015 full year financial results, 2016.

[18] George C. S. How much data is generated every minute on social media?, 2015.

[19] 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.

[20] Spark overview, 2016.

[21] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.

[22] Grace Huang. Towards benchmarking modern distributed streaming systems, 2015.

[23] Cloudera. Time series for spark, 2016.

[24] Storm. Documentation., 2016.

[25] Flink. Apache flink is an open source platform for distributed stream and batch data processing., 2016.

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

[27] 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.

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

[29] 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.

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

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

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

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

[34] 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.

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

[36] 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.