Forecasting Equity Realized Volatility using Machine Learning Methods

Marc Mitri* Vincent Gaugé†

Abstract

Machine learning is today one of the most active research area related to impressive advances in artificial intelligence technics. A huge number of learning algorithms are successfully used in many domains including speach recognition, computer vision, text classification or time series forecasting. In this paper, we use several machine learning methods in order to forecast the equity realized volatility, a key quantity for many asset allocation and risk management models. In our approach we treat this problem as a classification task (i.e., pattern recognition) and we predict the sign of the difference between equity realized volatility and equity anticipated volatility. We found that machine learning algorithms have superior performances and are able to learn complex input-output relationships.

Keywords: Machine learning; Equity realized volatility; Forecasting; Binary classification; Financial markets; Time series; Statistics

1 Introduction

Over the last years, the world entered the Data Revolution era. This revolution is considered as important for humanity as the agricultural revolution (1775-1800) or the computer revolution (1960-2010). In fact, 2.5 quintillion bytes of data are generated every day and each year, enough data is already being created and stored to stack DVDs from Earth to the moon and backwith (some estimating this stack would reach halfway to Mars by the end of the decade). Every minute, people post 216 thousand photos on Instagram and 2.5 million posts on Facebook, publish 347 new blogs, create 570 new websites, upload 100 hours of videos on Youtube and send 204 million e-mails. The exploitation of this great amount of data permited beneficial projects in many sectors such as insurance, health care, defense, transportation or the world wide web.

In parallel to this phenomenon, a scientific field is receiving a great attention from both research and industrial communities called machine learning (or statistical learning). The birth of this field was driven by the need of models that don't rely on too much assumptions about the generating mechanisms of the data. In the paper "Statistical Modeling: The Two Cultures" (Breiman 2001) the reader can find a comprehensive comparison between:

- *The data modeling culture*: here we assume that the law of nature generating the data is a stochastic model. In the black box linking predictors x to the response variable y, we find linear or logistic regression models (parametric class of models). They are validated using goodness-of-fit tests and residuals examination.
- *The algorithmic modeling culture*: here we assume instead that the law of nature generating the data is complex and unknown. The black box contains learning algorithms like Random Forests, SVM and Neural Networks. The models are validated by measuring predictive accuracy through cross-validation (i.e., estimation of the generalization capacity of the model). However, developments in statistical learning theory provide mathematical foundations and bounds on the generalization error. Machine learning falls in this category.

Breiman pointed out the limitation of the parametric linear models and concluded that they are not always suitable for modeling complex systems involving unkown physical, chemical or biological mechanisms.

In this work, we tackle the problem of forecasting the sign of the equity realized volatility minus anticipated one-month volatility on the S&P500 index using a set of financial predictors. We test widely used machine learning methods, namely Random Forests (`RF`) and Extremely Randomized Trees algorithms (`ExtraTF`) belonging to more general ensemble learning methods, as well as Support Vector Machines (`SvmLin` as the soft-margin SVM with linear kernel, `SvmPoly` as the soft-margin SVM with polynomial kernel and `SvmRbf` as the soft-margin SVM with radial basis function (Gaussian) kernel). We compare them to the regularized `Logistic Regression` (used as a linear parametric benchmark).

The rest of the paper is organized as follows: section 2 gives the necessary background on statistical learning theory and supervised learning setting. Section 3 presents the proposed machine learning methods used for classification. Section 4 presents the experimental results and our observations, as well as possible explanations. Section 5 presents the conclusions.

*Hiram Finance, Paris, France. Email: mmitri@hiram-finance.com

†Hiram Finance, Paris, France. Email: vgauge@hiram-finance.com

2 Background in statistical learning theory

In this section, we present the background and some preliminaries related to the supervised learning setting. We also present some elements of statistical learning theory.

2.1 Introduction. The foundations of statistical learning theory, which provides the theoretical basis for many machine learning algorithms, are due to the seminal work of Vladimir Vapnik [1] [2] [3] [4] [6]. The need of a theory of inductive inference to formalize concepts like learning, overfitting or generalization in order to design better algorithms is the main motivation of this field.

Actually, inductive inference means to infer a general law or a pattern from an observed real world phenomenon and is characterized by the following approach : 1) phenomenon observation, 2) construction of an explanatory model and, 3) making predictions using the learnt model.

In general, the main learning problems are:

- **Supervised learning**: pattern recognition (we will focus on this task in this paper), regression estimation. In this type of learning, one is given a training dataset of input-output pairs (x_i, y_i) and the goal is to infer an output y from a new instance x. If $y_i \in \mathbb{N}$, it corresponds to a classification task (pattern recognition) and if $y_i \in \mathbb{R}$ it is a regression estimation.

- **Unsupervised learning**: density estimation, clustering. Here, instances x_i are given without a target value. The goal is to summarize the input space $\mathcal{X}$ and reveal the underlying structure.

Hereafter we will give a formal introduction to the learning model and its different components. For more details, see [5].

2.2 The learning model. In the supervised learning setting, we try to construct a model from observed data with the following components:

1. A **generator** of random vectors x drawn independently from a fixed but unknown distribution $P(x)$.

2. An **oracle** that associates to each input x an output $y \in Y$ according to a conditional distribution function $P(y|x)$ which is fixed but unknown.

3. A **machine** implementing a class of functions defining a hypothesis space $\mathcal{H} = \{f : X \to Y\}$. The hypothesis space is the space of functions the algorithm will search through. For example, if $Y = \mathbb{R}$ and $X = \mathbb{R}^n$ then: $\mathcal{H} = \{f_w : X \to Y, f_w(x) = w^T x + b, w \in \mathbb{R}^n, b \in \mathbb{R}\}$, is the set of possible hyperplanes.

Hence the goal of the learning process is to choose the function $f \in \mathcal{H}$ which predicts the oracle's response in order to minimize a specified loss function. The selection of f is based on a training set $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ of N i.i.d. (random independent indentically distributed) observations drawn according to the joint probability distribution $P(x, y) = P(y|x)P(x)$.

Note that in the standard setting of statistical learning there is no assumption on the probability distribution $P(x, y)$. From this point of view, it differs from standard statistics where one usually assumes that the probability distribution belongs to a certain family of distributions with parameters to be estimated.

Moreover, independent sampling is also an important and strong assumption that is not justified in all applications (like time series forecasting or drug discovery). Fields like active learning or transductive learning try to relax the independence postulate.

Also, we assume that the distribution $P(x, y)$ is fixed (or stationary) and hence does not change over time. Obviously, this assumption is not valid for financial time series where we have a well known phenomenon called concept drift (the relation between the input data and the target variable changes over time). The drift is caused by a change in the posterior probability $P(y|x)$ (called real drift) or by a virtual drift. The later is also called covariate shift (change in $P(x)$). Indeed, an assumption that one can make about the connection between the source (where we learn) and the target (where we predict) domains is that given the same observation $X = x$, the conditional distributions of Y are the same in the two domains. However, the marginal distributions of X may be different in the source and the target domains. Formally, we assume that $P_s(Y|X = x) = P_t(Y|X = x)$ for all $x \in \mathcal{X}$, but $P_s(X) \neq P_t(X)$. This difference between the two domains is called covariate shift.

2.3 Theoretical versus empirical risks. We previously said that the goal of the machine is to predict the oracle's response considering a loss function that we note $L(y, f(x))$. We define the true risk (or theoretical risk) as:

$$R(f) = \int_{X \times Y} L(y, f(x)) dP(x, y) \tag{2.1}$$

For example, $L(y, f(x)) = (y - f(x))^2$ is used for regression, $L(y, f(x)) = \mathbb{1}_{\{y \neq f(x)\}}$ for binary classification.

The goal of supervised learning is to find the function f_0 in $\mathcal{H}$ that minimizes the theoretical risk:

$$f_0 = \arg\min_{f \in \mathcal{H}} R(f) \tag{2.2}$$

under two constraints: f must belong to $\mathcal{H}$ and $P(x, y)$ is

unknown (we have only access to the information contained in the training dataset).

Using the following notations: $z = (x, y)$ and $Q(z, f) = L(y, f(x))$. We can rewrite the true risk as:

$$R(f) = \int Q(z, f) dP(z) \tag{2.3}$$

Since we don't know the probability distribution $P(x, y)$ we can not compute $R(f)$. Instead, we compute the empirical risk $R_{emp}(f)$ over the i.i.d. observations $\{z_1, \ldots, z_N\}$ drawn from $P(z)$:

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^{N} Q(z_i, f) \ , \ f_{emp} = \arg\min_{f \in \mathcal{H}} R_{emp}(f) \tag{2.4}$$

We also define R_{opt} as the optimal theoretical risk (i.e., best achievable risk without restricting f to the hypothesis space):

$$f_{opt} = \arg\min_{f} R(f) \ , \ R_{opt} = R(f_{opt}) \tag{2.5}$$

The main question is if f_{emp} and $R_{emp}(f_{emp})$ (computable quantities) are good approximations of f_0 and $R(f_0)$ and, more ambitious, of f_{opt} and $R(f_{opt})$.

2.4 The bias-variance tradeoff. The bias-variance tradeoff is a fundamental concept in machine learning that determines the generalization ability of a model (we say that a model generalizes well if the error on the training set roughly equals the error on the test set).

We write the difference between the true risk associated to the function minimizing the empirical risk $R(f_{emp})$ and the optimal risk R_{opt} as:

$$R(f_{emp}) - R_{opt} = [R(f_0) - R_{opt}] + [R(f_{emp}) - R(f_0)] \tag{2.6}$$

We have the sum of two terms:

1. The **approximation error** (first term): called the bias of an estimator in statistics, it measures the error we make by choosing the best function in a limited hypothesis space $\mathcal{H}$ instead of picking the best function in the entire space of all possible functions. In other words, to what extent $\mathcal{H}$ is able to approximate the target f_{opt}. This error depends on the choice of $\mathcal{H}$, not on the data.
2. The **estimation error** (second term): called the variance of an estimator in statistics, it's a random quantity (since it depends on the data) that measures to what extent f_{emp} is close to f_0. In other words, it deals with the uncertainty introduced by the random sampling process since we have to estimate f_0 (the best function in $\mathcal{H}$) using only the available data.

Statistical learning theory has mainly a focus on variance since it's difficult to estimate the bias (we have no information about f_{opt} - so the assumptions are on the value of R_{opt} rather than the optimal function). In practice, the researcher has to find the right balance between these two quantities in order to avoid overfitting (using cross-validation for instance).

2.5 Empirical risk minimization induction principle (ERM). We have seen that the true risk associated to f_{emp} (i.e., the expected loss) can not be minimized directly since $P(x, y)$ is unknown. However, training data are available and we can approximate the true risk $R(f_{emp})$ with the empirical risk $R_{emp}(f_{emp})$. This is the empirical risk minimization induction principle (ERM). This approach is straightforward and reasonable since we expect a convergence between the two types of risk towards $R(f_0)$ when $N \to \infty$ and thanks to the law of large numbers. We say that the ERM principle is consistent. In other words, the estimator is consistent if the error on the training set is a good estimate of the error on the unseen test set.

Hence the statistical learning theory gives mathematical answers (essentially bounds and guarantees) regarding some questions related to the consistency of the ERM principle:

- Under what conditions is the ERM principle consistent? In order to answer this question, we need to establish the following necessary and sufficient conditions:

  $$R(f_{emp}) \xrightarrow[N \to \infty]{P} R(f_0) \tag{2.7}$$

  means the convergence in probability of the solution returned by ERM toward the optimal one in $\mathcal{H}$.

  $$R_{emp}(f_{emp}) \xrightarrow[N \to \infty]{P} R(f_0) \tag{2.8}$$

  means the convergence in probability of the empirical risk toward the minimum of the theoretical risk.

- What is the rate of convergence of our learning machine (i.e., the generalization ability of the machine that implements the ERM principle)? Since we have a finite training set, it is necessary to obtain nonasymptotic bounds on the rate of convergence.
- How to control the ability of generalization of the learning machine? This is related to the structure of $\mathcal{H}$.

- How to built algorithms that are able to control the rate of convergence?

2.6 Theoretical results. Our goal in this section is to present the main results without giving details or proofs which is not the purpose of this paper. Interested reader can consult the references.

2.6.1 Law of large numbers. Suppose that $f \in \mathcal{H}$ is fixed. We recall that $Q(z_i, f)$ is a set of random i.i.d. variables since $\{z_i\}$ in the training set are i.i.d. random variables. Hence we can write: $R(f) = E[Q(z, f)]$. $R_{emp}(f)$ is also a random variable. The weak form of the law of large numbers gives us:

$$\frac{1}{N}\sum_{i=1}^{N} Q(z_i, f) \xrightarrow[N\to\infty]{P} \int Q(z, f)dP(z) \tag{2.9}$$

Using the Chernoff inequality we can derive, for a given function f, a bound for the probability of closeness between the empirical and the true risk:

$$\forall \epsilon > 0,\ P(|R_{emp}(f) - R(f)| \geq \epsilon) \leq 2\exp(-2N\epsilon^2) \tag{2.10}$$

This indicates that the probability of a large deviation between training and test errors converges toward 0 as $N \to \infty$.

However this result is not satisfactory (i.e., it can't prove the consistency of the ERM principle) for two reasons:

- The Chernoff bound gives an asymptotic result. The problem is that we have a finite observations set.
- The bound is valid for a fixed function f which doesn't depend on the training data. But in practice, the function f_{emp} is constructed from empirical data (here f is fixed whereas in the ERM principle we assume that f_{emp} is a random variable). So the necessary condition for consistency can not be derived.

Further developments use non-asymptotic concentration inequalities (like Hoeffding inequality) as tools to derive probabilistic bounds to prove the convergence of the empirical risk toward the true risk. However, these tools are not sufficient (because in practice f depends on the data, i.e., f_{emp} is a random variable) and Vapnik-Chervonenkis theory stipulates that in order to achieve consitency, we need to restrict the hypothesis space $\mathcal{H}$ (the idea is that if $\mathcal{H}$ contains functions with perfect fitting, the ERM principle cannot work). These restrictions of the admissible functions are related to the Vapnik-Chervonenkis dimension h that we will introduce later.

A key theorem of the learning theory (see section II.A in [1]) asserts that the analysis of the consistency of the ERM principle must be a worst case analysis. In other words, the consistency is determined by the worst case behavior over all functions $f \in \mathcal{H}$. This worst case is equivalent to a law of large numbers which is uniform over all functions in $\mathcal{H}$. This corresponds to a uniform one-sided convergence (necessary and sufficient condition).

So since we don't know the function that the algorithm will choose, we consider the following uniform bound:

$$R(f_{emp}) - R_{emp}(f_{emp}) \leq \sup_{f\in\mathcal{H}}(R(f) - R_{emp}(f)) \tag{2.11}$$

The uniform law of large numbers is then applied to bound the right hand side of the inequality and the consistency of the ERM principle is related to the convergence of the supremum $\sup_{f\in\mathcal{H}} |R(f) - R_{emp}(f)|$. It is proved that this convergence is related to the structure of $\mathcal{H}$.

So the main question at this stage is: how do we measure the richness of the structure of the hypothesis space $\mathcal{H}$ (i.e., the capacity or complexity of the learning machine)?

2.6.2 The three milestones. Let's introduce the milestones of the learning theory using a set of indicator functions $Q(z, f),\ f \in \mathcal{H}$ (i.e., pattern recognition setting which can be extended to real-valued functions):

1. **Milestone 1**: define the entropy for sets of indicator functions as:

   $$H^{\mathcal{H}}(N) = E[\ln N^{\mathcal{H}}(z_1, \ldots, z_N)] \tag{2.12}$$

   with

   $$N^{\mathcal{H}}(z_1, \ldots, z_N) = Card(\{(Q(z_1, f), \ldots, Q(z_N, f)),\ f \in \mathcal{H}\}) \tag{2.13}$$

   The necessary and sufficient condition for uniform convergence (strict consistency of the ERM principle) is:

   $$\frac{H^{\mathcal{H}}(N)}{N} \underset{N\to\infty}{\longrightarrow} 0 \tag{2.14}$$

2. **Milestone 2**: define the annealed entropy as:

   $$H^{\mathcal{H}}_{an}(N) = \ln(E[N^{\mathcal{H}}(z_1, \ldots, z_N)]) \tag{2.15}$$

   Then define the growth function as:

   $$G^{\mathcal{H}}(N) = \ln(\sup_{z_1,\ldots,z_N}(N^{\mathcal{H}}(z_1, \ldots, z_N))) \tag{2.16}$$

   Using the Jensen inequality we have:

   $$H^{\mathcal{H}}(N) \leq H^{\mathcal{H}}_{an}(N) \leq G^{\mathcal{H}}(N) \tag{2.17}$$

The second milestone of the learning theory, which gives us the necessary condition for a fast asymptotic rate of convergence, is:

$$\frac{H_{an}^{\mathcal{H}}(N)}{N} \underset{N\to\infty}{\longrightarrow} 0 \tag{2.18}$$

Considering the link between $H^{\mathcal{H}}(N)$ and $H_{an}^{\mathcal{H}}(N)$, this is also a sufficient condition for the uniform convergence and hence the strict consistency of the ERM principle.

3. **Milestone 3**: we notice that the two first milestones are valid only for a given probability measure $P(z)$ ($H^{\mathcal{H}}(N)$ and $H_{an}^{\mathcal{H}}(N)$ are distribution-dependent). However we want to construct learning machines that can solve a wide range of problems (i.e., we want to obtain a consistent and rapidly converging ERM principle independently of the probability measure). We use the growth function:

$$\frac{G^{\mathcal{H}}(N)}{N} \underset{N\to\infty}{\longrightarrow} 0 \tag{2.19}$$

This is the necessary and sufficient condition for consitency and fast convergence for any $P(z)$. The sufficient part for fast convergence comes from $H_{an}^{\mathcal{H}}(N) \leq G^{\mathcal{H}}(N)$.

We have defined some measures of the richness of the hypothesis space. However, $G^{\mathcal{H}}(N)$ depends on the size N of the training set. Also $H^{\mathcal{H}}(N)$ and $H_{an}^{\mathcal{H}}(N)$ depend on the probability distribution P. These results are not very usefull in practice and one wants to establish a more intuitive definition of the richness (or complexity) of the hypothesis space.

2.6.3 The Vapnik-Chervonenkis dimension (VC). This measure of the capacity of a statistical learning algorithm is derived using the growth function defined in the third milestone which has a particular structure.

From the theorem of Vapnik-Chervonenkis/Sauer/Shelah (see section III.A. in [1]), we have that the growth function $G^{\mathcal{H}}(N)$ is either linear (implying that the VC dimension is infinite) or bounded by a logarithmic function $h(1 + ln\frac{N}{h})$ with coefficient h (implying that the VC dimension is finite and equals h).

A very important result links the asymptotic behaviour of the growth function to the VC dimension:

$$\frac{G^{\mathcal{H}}(N)}{N} \underset{N\to\infty}{\longrightarrow} 0 \Longleftrightarrow h < \infty \tag{2.20}$$

Hence a necessary and sufficient condition for consistency and fast convergence of the ERM principle regardless the probability distribution P is the finiteness of the VC dimension h of the hypothesis space $\mathcal{H}$.

Formal definition: the VC dimension of a set of indicator functions $Q(z, f)$, $f \in \mathcal{H}$ is the maximum number h of points $z_1, \ldots, z_h$ which can be shattered (i.e., separated) in all 2^h possible ways using functions $Q(z, f)$. In other words, it is the maximum number of points that can be exactly learned (i.e., well classified) by a function of $\mathcal{H}$. If the shattering is possible $\forall N$ then the VC dimension is infinite. This definition can be extended to real-valued functions.
We give somes examples:

- Consider a set of linear indicator functions (set of hyperplanes in $\mathbb{R}^d$) such that: $\mathcal{H} = \{x \mapsto sign(w^T x + b),\ w \in \mathbb{R}^d,\ b \in \mathbb{R}\}$. Then the VC dimension of $\mathcal{H}$ is $h = d + 1$ (where d is the dimensionality of the input space). We could think that the VC dimension is always linked to the number of input variables. However, this is true only if the input-output relationship is linear.
- Consider the following: $\mathcal{H} = \{[(sin(tx))_+],\ t \in \mathbb{R}\}$. With this hypothesis space, and by tuning the parameter t, one can shatter every finite set of points in $\mathbb{R}$. Hence the VC dimension of this hypothesis space is infinite despite the fact that we have only one input.
- Consider a set of Δ-margin separating hyperplanes. The VC dimension h of this hypothesis space (considering large margins) is bounded (see section III.C. in [1]) and can be less then $d + 1$ (the larger the margins, the smaller is h). This result has a great practical implication since it motivates the developpment of classifers that aim at constructing decision boundaries by maximizing the margin of hyperplanes: the well known SVM algorithm.

Finally, the VC dimension plays an important role since it is used to define a bound on the true risk and guarantees on the generalization capacity of the learning machine.

2.6.4 Bounding the risk. Here we present the main theorem of Vapnik's theory. We consider the general case of bounded functions $0 \leq Q(z, f) \leq B$. For the set of indicator functions, B=1.

Let h be the VC dimension. With probability at least $1 - \eta$, we have:

$$R(f) \leq R_{emp}(f) + \frac{B\varepsilon}{2}\left(1 + \sqrt{1 + \frac{4R_{emp}(f)}{B\varepsilon}}\right) \tag{2.21}$$

with

$$\varepsilon = 4\frac{h(\ln\frac{2N}{h} + 1) - \ln\eta}{N} \tag{2.22}$$

This inequality justifies the use of the ERM principle.

2.6.5 Structural risk minimization induction principle (SRM). The previous inequality should be analysed regarding the N/h ratio. Actually, the ERM induction principle in justified when $N > h$: the second summand of the right hand side becomes small, and minimizing the empirical risk is a good approach to ensure generalization ability (i.e., it is a good estimate of the true risk)

However, if the VC dimension is relatively as large as the number of observations (N/h small), then minimizing $R_{emp}(f)$ does no more garantee a small value of the true risk. So we need a new minimization principle with two objective functions: the value of the empirical risk and a quantity related to the VC dimension of the set of functions. The second objective means controling the value of h (i.e., restricting the richness of the hypothesis space).

This simultaneous minimization is achieved via the structural risk minimization (SRM) principle.

Consider the set $S = \{Q(z, f),\ f \in \mathcal{H}\}$ composed of the nested subsets of functions:

$$S_k = \{Q(z, f),\ f \in \mathcal{H}_k\} \tag{2.23}$$

Such that:

$$S_1 \subset S_2 \subset \cdots \subset S_k \subset \cdots, and\ S = \bigcup_k S_j \tag{2.24}$$

Hence we have the associated growing VC dimensions :

$$h_1 \leq h_2 \leq \cdots \leq h_k \leq \cdots \tag{2.25}$$

Thus the SRM principle consists in choosing simultaneously $\mathcal{H}_k$ and $f_{emp} \in \mathcal{H}_k$ in order to minimize the bound of the true risk.

2.6.6 Regularization. The SRM principle consists in defining a sequence of models of increasing sizes and then minimizing the empirical risk (and computing bounds) for each model with a preference for models with low complexity.

The problem is that this approach is difficult to implement in practice. This led to a widely and successfully used method in machine learning: regularization. The idea is to introduce a regularization term of the model's complexity directly in the loss function.

$$f_{emp} = \arg\min_{f \in \mathcal{H}} R_{emp}(f) + \lambda \Omega(f) \tag{2.26}$$

Where λ controls the importance of the regularization term (and impacts the bias-variance tradeoff) and $\Omega()$ is typically a penalty on the complexity of f. Usually, λ is tuned using cross-validation procedure. Suppose the hypothesis space $\mathcal{H}$ such that f is parametrized with a p dimensional vector w. The most used regularizers are:

- L_1 **regularization:** it uses the L_1 norm of the parameters:

$$\Omega(f_w) = \|w\|_1 = \sum_{j=1}^{p} |w_j| \tag{2.27}$$

  This form of regularization leads to a sparse solution with many parameters set to 0. However, the loss function is no more continuously differentiable and need specific algorithms. It works well for problems where we have a lot of irrelevant features. Linear least squares regression with L_1 regularization is called Lasso regression.

- L_2 **regularization:** it uses the L_2 norm of the parameters:

$$\Omega(f_w) = \|w\|_2^2 = \sum_{j=1}^{p} w_j^2 \tag{2.28}$$

  Linear least squares regression with L_2 regularization is called Ridge regression (or more generally Tikhonov regularization). It reduces the magnitudes/values of the parameters and hence allows to avoid overfitting. This form of regularization converges faster than the L_1 penalty in practice.

Prior to Lasso, the most widely used method for choosing which covariates to include was stepwise selection, which only improves prediction accuracy in certain cases, such as when only a few covariates have an strong relationship with the outcome. However, in other cases, it can make prediction error worse. Also, at the time, Ridge regression was the most popular technique for improving prediction accuracy. Ridge regression improves prediction error by shrinking large regression coefficients in order to reduce overfitting, but it does not perform covariate selection and therefore does not help to make the model more interpretable.

Lasso is able to achieve both of these goals by forcing the sum of the absolute value of the regression coefficients to be less than a fixed value, which forces certain coefficients to be set to zero, effectively choosing a simpler model that does not include these coefficients.

As we will see it later, soft-margin SVM algorithm is an implementation of the SRM principle with regularization.

3 Proposed methodological approach

In this section, we will provide a description of the machine learning algorithms used to perform volatility forecasting.

3.1 Regularized logistic regression. Logistic regression (also known as logit model) is a widely used algorithm for classification tasks. It is a linear parametric model since the decision boundary is a hyperplane. Like linear regression, logistic regression is a conditional model which models the conditional probability of the output, given the input $p(Y|X)$ (contrary to generative models which model the joint distribution $p(X, Y)$).

We assume that we have a training dataset $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of i.i.d. random variables of inputs $X \in \mathbb{R}^p$ and output $Y \in \{0, 1\}$ which follows a Bernoulli distribution with parameter θ: the goal of the algorithm is to infer θ from the observed data. Logistic regression uses the well known logistic function [1] σ which is defined on $[0, 1]$ such that:

$$\forall z \in \mathbb{R},\ \sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.29}$$

We then assume that $Y|X = x \sim \mathcal{B}(\theta)$ with $\theta = \sigma(w^T x)$ [2] and w is the vector of weights to be estimated ($(p+1)$-dimensional vector if we add an intercept). We can write the conditional distribution as:

$$\begin{aligned} P(Y|X;\theta) &= \prod_{i=1}^{N} \theta^{y_i}(1-\theta)^{1-y_i} \\ &= \prod_{i=1}^{N} \sigma(w^T x_i)^{y_i}\sigma(-w^T x_i)^{1-y_i} \end{aligned} \tag{3.30}$$

We then compute the log-likelihood:

$$\begin{aligned} l(w) &= \sum_{i=1}^{N} [y_i \log \sigma(w^T x_i) + (1-y_i)\log \sigma(-w^T x_i)] \\ &= \sum_{i=1}^{N} [y_i w^T x_i + \log \sigma(-w^T x_i)] \end{aligned} \tag{3.31}$$

Since the log-likelihood is differentiable and concave, its global maxima are its stationary points. Let $\eta_i = \sigma(w^T x_i)$, we compute the gradient of $l(w)$:

$$\begin{aligned} \nabla_w l(w) &= \sum_{i=1}^{N} [y_i x_i - x_i \frac{\sigma(-w^T x_i)\sigma(w^T x_i)}{\sigma(-w^T x_i)}] \\ &= \sum_{i=1}^{N} x_i(y_i - \eta_i) \end{aligned} \tag{3.32}$$

Hence $\nabla_w l(w) = 0 \iff \sum_{i=1}^{N} x_i(y_i - \sigma(w^T x_i)) = 0$ has no closed form solution (this equation is nonlinear and we need an iterative optimization method to solve it). We use the second-order Taylor expansion of the log-likelihood function in order to use second order otimization algorithms (Newton's method).

We introduce the design matrix:

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{R}^{N \times p} \tag{3.33}$$

We write the Hessian matrix of $l(w)$ as:

$$\begin{aligned} Hl(w) &= \sum_{i=1}^{N} [0 - \sigma'(w^T x_i)\sigma'(-w^T x_i) x_i^T] \\ &= -X^T Diag(\eta(1-\eta))X \end{aligned} \tag{3.34}$$

Using the second-order Taylor expansion of the log-likelihood function at point $w = w^t$ we have:

$$l(w) = l(w^t) + (w - w^t)^T \nabla_w l(w^t) + \frac{1}{2}(w - w^t)^T Hl(w^t)(w - w^t) \tag{3.35}$$

We set $h = w - w^t$ and $D_\eta = Diag(\eta(1-\eta))$. The maximization problem is then:

$$\begin{aligned} &\max_h \{h^T X^T (y - \eta) - \frac{1}{2} h^T X^T D_\eta X h\} \\ &\iff \max_h \{h^T \nabla_w l(w) + \frac{1}{2} h^T Hl(w) h\} \end{aligned} \tag{3.36}$$

The optimization problem is solved with the IRLS algorithm (Iterative Reweighted Least Squares): $w^{t+1} \leftarrow w^t + (X^T D_{\eta^t} X)^{-1} X^T (y - \eta^t)$.

We know that linear classifiers over lower dimensional input spaces will have smaller VC dimension (actually, the number of training examples needed to learn well grows linearly with the VC dimension which grows about linearly in the number of parameters which typically grows at least linearly in the number of input features). Hence, in our work, we use a regularized form of logistic regression which

[1] Known as sigmoid function in the neural networks literature, it is used as an activation function in the neuron model.

[2] The key assumption behind this is that $\log \frac{P(Y=1|X=x)}{P(Y=0|X=x)} = w^T x$.

consists in finding the parameters θ that solve the following optimization problem:

$$\arg\max_{\theta} \sum_{i=1}^{N} \log p(y_i|x_i;\theta) - \lambda\Gamma(\theta) \tag{3.37}$$

where λ is the penalty coefficient and $\Gamma(\theta)$ is the regularization term. If $\Gamma(\theta) = \sum_{i=1}^{N} |\theta_i|$ then this is L_1 regularized logistic regression and if $\Gamma(\theta) = \sum_{i=1}^{N} \theta_i^2$, this is L_2 regularization. The above optimization program can be solved using quasi-Newton methods like LBFGS algorithm.

In [7] the author proved that using L1 regularization of the parameters, the sample complexity (i.e., the number of training examples required to learn well) grows only logarithmically in the number of irrelevant features. This logarithmic rate matches the best known bounds for feature selection, and indicates that L1 regularized logistic regression can be effective even if there are exponentially many irrelevant features as there are training examples.

In the rest of the paper we will note `LogRegL1` the L1 regularized logistic regression and `LogRegL2` the L2 regularized logistic regression.

3.2 Decision trees and Random Forests. The CART algorithm (Classification and Regression Tree) has been developed by Breiman et al. in 1986 (see [8]) and became one of most popular learning methods commonly used for data exploration. It is a non-parametric, greedy, top-down binary, recursive partitioning method that divides feature space into sets of disjoint rectangular and more homogeneous regions. Contrary to logistic regression, it produces a non-linear decision boundary.

Suppose we have a region R which is a subset of $\mathbb{R}^p$, we then note E_R to be the fraction of data points $x_i \in R$ misclassified by a majority vote in the region R (we consider a classification task). We note also $N_R = Card(\{i : x_i \in R\})$. Then the procedure for growing a classification tree (the regression case is similar) is relatively simple. Starting with the inital node containing all the training dataset, we chose a dimension j (i.e., the predictor x_j, with $j \in \{1, \ldots, p\}$) and the split s (i.e., a value in the range of values that variable x_j takes in the training set D) using a brute-force search in order to find the best binary split δ^* that minimizes the misclassification error in the child nodes:

$$\delta^* = \operatorname*{argmin}_{\delta \in \Delta} N_{R(j,s)} E_{R(j,s)} + N_{R'(j,s)} E_{R'(j,s)} \tag{3.38}$$

where Δ is the set of all possible splits, R and R' are the obtained child nodes (i.e., disjoint sub-regions of R) with $R(j,s) = \{x_i : x_{ij} > s\}$ and $R'(j,s) = \{x_i : x_{ij} \leq s\}$. In other words, the goal is to find the splitting point δ that gives child nodes that are more pure (i.e., contain a larger proportion of one class in each node). In our definition, we used the accuracy as a measure of purity, however, it is a bit misleading since the measure's focus is on partitioning the data in a way that minimizes misclassification rather than a focus on partitioning the data in a way that places samples primarily in one class. In practice and in all the available packages, we use two alternative measures: the Gini index or cross-entropy (also known as deviance) for classification, squared error for regression.

Hence, the CART algorithm repeats this procedure recursively for each newly created sub-region a sufficient number of times (therefore increasing the depth of the tree) until we have pure terminal nodes (i.e., nodes containing observations with the same label), or until another stopping criteria is met (such as the minimum number of samples in a terminal node [3] or the maximum tree depth). Trees with maximum depth are known to overfit (since they build very complex decision boundaries) and hence pruning strategies can be used to enhance their generalization ability. For more details see [9] [10] and [11].

The CART algorithm has many advantages such as natural handling of "mixed" data type (continuous, discrete or categorical) as well as missing values, robustness to outliers in the input space, computational scalability and ability to deal with irrelevant inputs (i.e., automatic features selection). However, this algorithm has an unstable structure due to the recursive construction principle and is very sensitive to the data distribution (a little change in the training dataset leads to a completely different classifier). Hence this model has a high variance (i.e., estimation error) and low prediction accuracy. This fact has led to the development of ensemble learning methods that reduces the variance of the final aggregated classifer obtained by combining several weak classifiers such as CART.

3.3 Ensemble learning methods. To overcome the lack of robustness of weak learners like decisions trees, ensemble learning have been developped. The idea is to built a set of such learners by introducing a given quantity of randomness and then average the set of decisions given by each base classifier (final prediction is the mean prediction for regression or class with maximum votes for classification).

3.3.1 Bagging. In 1996, Breiman [12] introduced the bootstrap aggregating meta-algorithm (called bagging) to enhance the accuracy of CART algorithm and reduce the variance. Recall that we have a training dataset $D =$

[3] If the terminal nodes contain too few samples, the prediction is not statistically significant.

$\{(x_1, y_1), \ldots, (x_N, y_N)\}$. The procedure draws B bootstrap samples (random sampling with replacement) noted $\{D_b\}_{b=1,\ldots,B}$ and builts a model ϕ_{D_b} on each of them. The final model depends on the supervised learning setting:

- **Regression:**

$$\hat{\phi}_B(\cdot) = \frac{1}{B}\sum_{b=1}^{B}\hat{\phi}_{D_b}(\cdot) \tag{3.39}$$

- **Classification:**

$$\hat{\phi}_B(\cdot) = \underset{j}{\text{argmax}}\ Card\{b|\hat{\phi}_{D_b}(\cdot) = j\} \tag{3.40}$$

Breiman used CART algorithm as family of models $\phi(\cdot)$ and experimental results suggest using completly grown trees without pruning (thus each tree has a low bias but a high variance). Since each bootstrap sample (and by consequence each tree) generated by bagging is identically distributed (i.d.), the bias of the ensemble is the same as that of the individual trees, and the only hope lies in variance reduction.

3.3.2 Random Forests. In 2001, Breiman [13] introduced the Random Forests (RF) algorithm which is an improvement of the bagging procedure (experimental results show generally a great enhancement). In bagging we have a first source of randomness (bootsrapping). In Random Forests, we go one step further by adding a second source of randomness: during the construction of a single tree and for each split, we select a random subset of m features as candidate for choosing the best split δ^* with $m < p$. In the literature, researchers report the following best values for m: $\sqrt{p}$ for classification and $p/3$ for regression. We see that bagging is a special case of RF where $m = p$.

The goal of the additional randomness is to increase the independence between the trees. In fact, de-correlation gives better accuracy. Why? Let $\rho(x)$ be the sampling correlation between any pair of trees used in the averaging [4] and $\sigma^2(x)$ is the sampling variance of any single randomly drawn tree, then the variance of the ensemble is:

$$\begin{aligned} Var\left(\frac{1}{B}\sum_{i=1}^{B}\hat{\phi}_{D_i}(x)\right) &= \frac{1}{B^2}\sum_{i=1}^{B}\sum_{j=1}^{B} Cov\left(\hat{\phi}_{D_i}(x), \hat{\phi}_{D_j}(x)\right) \\ &= \frac{1}{B^2}\sum_{i=1}^{B}\left(\sum_{j\neq i}^{B} Cov(\hat{\phi}_{D_i}(x), \hat{\phi}_{D_j}(x)) + Var(\hat{\phi}_{D_i}(x))\right) \\ &= \frac{1}{B^2}\sum_{i=1}^{B}((B-1)\rho\sigma^2 + \sigma^2) \\ &= \frac{B(B-1)\rho\sigma^2 + B\sigma^2}{B^2} \\ &= \rho\sigma^2 + \sigma^2\frac{1-\rho}{B} \end{aligned} \tag{3.41}$$

We can see that the second term can converges to 0 if we increase the number B of trees in the forest. The first term is directly dependent on the pairwise correlation ρ, which is reduced by the injection of randomness at the tree construction level. That's why Random Forests works. In general, this aggregation procedure yields greater accuracy and generalization and this seems particulary true for high dimensional data (where $p \gg N$). This beneficial effect of randomization can be seen in Figure 1.

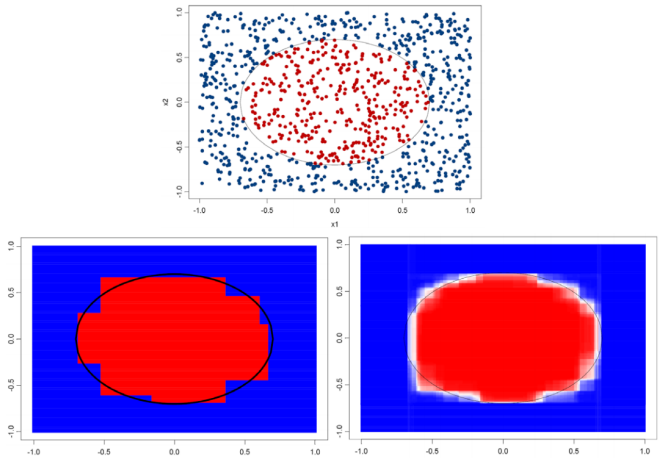

Figure 1: Variance reduction using bagging. On the top, we have two classes (blue and red dots) with a clear ellipsoid separation. On the bottom left, the single tree decision boundary, and on the right, the decision boundary of 100 bagged and aggregated trees. Bagging leads to a more accurate classification. See [14].

In the rest of the paper, we will note the Random Forests algorithm as `RF`.

In the same vein, Geurts et al. [15] derived the Extremely Randomized Trees (ERT) algorithm similar to Random Forests except that: ERT does not apply the bagging

[4] Do not confuse this quantity with the average correlation between fitted trees in a given Random Forests ensemble. See [10] section 15.4.1 for more details.

procedure to construct a subset of the training samples for each tree and ERT picks a node split very extremely (both a variable index and variable splitting value are chosen randomly), whereas Random Forests finds the best split (optimal one by variable index and variable splitting value) among random subsets of variables. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias.

In the rest of the paper, we will note the Extremely Randomized Trees algorithm as `ExtraTF`.

3.3.3 Variable importance scores. An interesting feature of Random Forests concerns variable selection. In fact, using this algorithm we can compute variable importances measures that allow us to have an insight into the set of relevant and irrelevant variables. A recent work of Louppe et al. [16] gives a theoretical study of the Mean Decrease Impurity (MDI) measure [5]. This is particulary useful in financial prediction since it gives us the financial market drivers.

To measure the importance of a predictor X_m, the MDI measure computes the average, over all B trees in the forest, of the sum of weighted impurity decreases for all nodes where X_m is used. Then this measure is normalized over the set of predictors to sum up to 1. Using the Gini index as impurity function, this measure is known as the Gini importance or Mean Decrease Gini. Empirical studies show that this measure is biased towards predictors with a large number of values (this bias stems from an unfair advantage given by the usual impurity functions). However, we are not affected in our financial application since the predictors have roughly the same entropy.

From the theoretical point of view, they proved that MDI importances as computed by totally randomized trees exhibit desirable properties for assessing the relevance of a variable: it is equal to zero if and only if the variable is irrelevant and it depends only on the relevant variables. However, these results are asymptotic and do no longer hold for RF or ERT algorithms.

Other measures exist: counting the number of times a predictor is used to split the root node (too simple) or a more sophisticed one known as Mean Decrease Accuracy (MDA or permutation importance), which measures the average variation of accuracy on the out-of-bag (OOB) samples where the values of X_m are randomly permuted.

3.3.4 Theoretical results. Random Forests is a widely used algorithm in many scientific areas with very successful empirical results. However, the theoretical and mathematical properties of this method are still not well understood due to the fact that this method is a combination of several components (bootstrapping procedure, randomness in the splitting step, fixed number of observations in each terminal node) which makes the algorithm difficult to analyze with rigorous mathematics. Some work has been done focusing on the consistency of this estimator and using very simplified versions of the original framework to make the analysis more tractable.

We can notice the following important works: the early study by Breiman 2004 [17], a more rigorous formalization by Biau et al. 2008 [18] and Biau 2012 [19] as well as Genuer 2012 [20]. Denil at al. 2013 [20] recently proved the consistency of an online version of Random Forests.

The most recent works are: Denil et al. 2014 [22] and Scornet et al. 2015 [23]. In the former, the authors proved the consistency of a variant of random regression forests (unlike the original RF of Breiman, they do not perform bootsrapping). An important contribution of this work is the empirical comparision of their theoretical model to the theoretical models developped by Biau et al. 2008 and Biau 2012 as well as the widely used RF algorithm of Breiman 2001 on several datasets. They found that their model has the closest performance to RF. Their work gives basics to more analyses such as complexity bounds and asymptotic convergence rates. In the later, the authors study the asymptotic properties of Breiman's algorithm in the context of additive regression models and proved the consistency of random forests which gives a first basic theoretical guarantee of efficiency for this algorithm (this is the first consistency result for Breiman's original procedure).

3.4 Support Vector Machines. The Support Vector Machines (SVM) is a state-of-the-art classification and regression method introduced by the work of Boser et al. 1992 [24] and Cortes and Vapnik 1995 [25].

This algorithm has solid theoretical foundations and is a practical implementation of the results from statistical learning theory developed by Vapnik. The SVM algorithm is attractive for the following reasons: robustness to very large number of variables and small samples, ability to learn both simple and highly complex classification models (via the kernel trick, i.e., by mapping the data into high dimensional feature space) and usage of sophisticated mathematical principles to avoid overfitting.

Finally, as we will see, this algorithm consists in solving a convex optimization problem using quadratic programming methods with the guarantee of a global optimum rather than local optimum achieved by many machine learning algorithms (for instance k-Nearest Neighbors and Neural Networks). Excellent empirical performances have been noticed in many application like bioinformatics, text categorization, image recognition, genes prediction, stock picking in finance, etc..

[5] G. Louppe implemented this measure used in the Random Forests implementation of Scikit-learn package in Python.

In the following developments, we will consider the pattern recognition case with binary classification.

3.4.1 The main idea. Recall that we have a training dataset $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of random variables of inputs $X \in \mathbb{R}^p$ and output $Y \in \{-1, +1\}$. Hence, our pattern or instance x_i belongs either to the negative class -1 (cancer patient for example) or to the positive one +1 (normal patient). We define a linear classifer to be a linear discriminant function of the form:

$$f(x) = w^T x + b \tag{3.42}$$

where $w \in \mathbb{R}^p$ is the weight vector and $b \in \mathbb{R}$ is the bias term. If we consider $b = 0$ then the set $H(w, b) = \{x \in \mathbb{R}^p | f(x) = w^T x = 0\}$ corresponds to all points (i.e., vectors in the input space) that are perpendicular to w and go through the origin (a line in two dimensions, a plane in three dimensions, and more generally, a hyperplane). So the bias b translates the hyperplane away from the origin and the hyperplane of equation $H(w, b) = \{x \in \mathbb{R}^p | f(x) = w^T x + b = 0\}$ divides the space in two regions (i.e., the data points are separated in two classes depending on the sign of $sign(f(x)) = sign(w^T x + b)$).

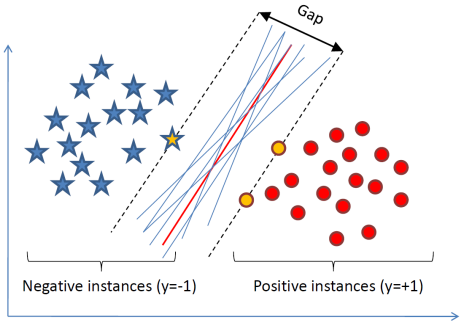

Figure 2: All possible hyperplanes. See [28].

As we can see in Figure 2, in the separable case (where there exists a linear decision boundary that separates the two classes perfectly), an infinite number of hyperplanes $H(w, b)$ exist that can discriminate our training dataset. Hence our goal is to find the hyperplane that maximizes the gap (or the margin) between data points on the boundaries.

Before going further, we point out that regarding the ERM principle, the best hyperplane is the one that minimizes the misclassification error on the training set (this criterion leads to the Rosenblatt's preceptron algorithm for instance). The contribution of Boser et al. 1992 [24] was to consider a more robust criterion comming from the statistical learning theory: this led to the large margin concept.

In fact, they showed (and we noticed it in Section 2) that the VC dimension of the hypothesis space formed by the set of separating hyperplanes decreases as the margin increases. As a consequence, this increases the generalization capacity of the learning machine.

3.4.2 Case 1: Linearly separable data; Hard-margin linear SVM. Here, we suppose that our observtions are linearly separable in the input space. Lets derive an expression for the margin.

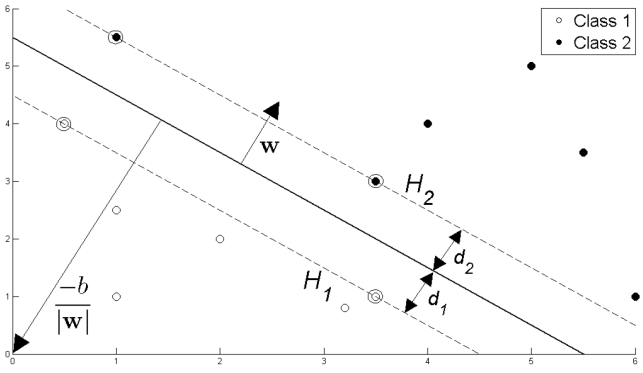

Figure 3: Hard-margin SVM. See [27].

Consider Figure 3 where we have points (our dataset) in 2D and the hyperplane $H(w, b) = \{x \in \mathbb{R}^p | w^T x + b = 0\}$ with two equidistant parallel hyperplanes $H_1(w, b) = \{x \in \mathbb{R}^p | w^T x + b = -1\}$ and $H_2(w, b) = \{x \in \mathbb{R}^p | w^T x + b = +1\}$. The points that verify these equation (i.e., that lie on the hyperplanes H_1 and H_2) are called Support Vectors (encircled points).

Geometrically, w is normal to the hyperplane and $\frac{b}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin. From geometry, we know that the distance between two parallel hyperplanes H_1 and H_2 is $D = \frac{2}{\|w\|}$. Since the hyperplanes are equidistant $d_1 = d_2 = \frac{1}{\|w\|}$, a quantity known as the SVM's geometric margin. We note that maximizing the margin is equivalent to minimizing $\|w\|$ (we will consider $\frac{1}{2}\|w\|^2$ to be able to perform quadratic programming optimization). So the goal of SVM is to find the hyperplane with the optimal orientation (optimizing with respect to w) and the optimal translation (optimizing with respect to b) in order to maximize the margin and simultaneously classify the observations perfectly (i.e., $w^T x_i + b \geq +1$ if $y_i = +1$ and $w^T x_i + b \leq -1$ if $y_i = -1$).

This leads to the following convex quadratic programming optimization problem:

$$\begin{cases} \min_{w,b} \frac{1}{2}\|w\|^2 \\ s.t. \quad y_i(w^T x_i + b) \geq 1, \forall i \in \{1, \ldots, N\} \end{cases}$$

This is called the primal formulation of the problem and

has a unique solution with $(p+1)$ parameters. The numerical resolution of the primal problem in the separable case can be done using Gauss-Seidel or interior point methods as well as Newton or conjugate gradient algorithms (see the paper of Chapelle 2007 [26] for more details on the resolution of the primal).

However, the primal formulation has an equivalent dual formulation [6] (which is also a convex quadratic programming problem) and historically SVM has been mainly implemented in the dual form. This has been motivated by the following interesting points: the dual problem is a quadratic program of size N which we will see, can be easier to solve. In fact, the primal involves $(p+1)$ parameters which is tractable when p is small but very costly if p is more than few hundreds. The process will reveal some interesting and important properties of SVM related to high dimensional input spaces. Moreover, in the dual formulation appears the Gramian matrix XX^T which will allow us to introduce non linearity using kernels [7].

To write the dual formulation, we introduce the Lagrangian which is the sum of the objective function and a linear combination of the constraints with coefficients $\alpha_i \geq 0$ (called Lagrange multipliers or dual variables):

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i [y_i(w^T x_i + b) - 1] \tag{3.43}$$

Then, we minimize the Lagrangian function $\mathcal{L}(w,b,\alpha)$ with respect to the primal variables w and b (for fixed α) and then we substitute the primal variables for dual variables in the Lagrangian. After that, we maximize the Lagrangian with respect to the dual variables $\alpha_i \geq 0$. Finally, we recover the solution (for the primal variables) from the dual variables.

By setting the derivatives of the Lagrangian with respect to w and b to 0 (stationarity conditions [8] verified at the saddle point of the Lagrangian) we have:

$$\begin{aligned} \frac{\partial}{\partial w}\mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \\ \Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i \end{aligned} \tag{3.44}$$

And,

$$\frac{\partial}{\partial b}\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.45}$$

Substituting the above equations in the Lagrangian gives us a new formulation that we seek to maximize depending only on the dual variables $\alpha_i \geq 0$:

$$\begin{aligned} \mathcal{L}(w,b,\alpha) &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^T x_j - b\sum_{i=1}^{N} \alpha_i y_i \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^T x_j \end{aligned} \tag{3.46}$$

Here we see the benefits of using the dual formulation: we have only inner products between input feature vectors (this is important for the kernel trick) and the number of free parameters is bounded by the number of support vectors and not by the number of variables (beneficial for high dimensional problems). In fact, the later is explained by the Karush-Kuhn-Tucker dual complementarity condition which tells us that we will have $\alpha_i > 0$ at the optimal solution only for the training examples that have functional margin exactly equal to one (we say that the corresponding constraint is active, meaning it holds with equality rather than with inequality at the optimum): these training examples are the so called support vectors which determine the position of the separating hyperplane (they lie on the hyperplanes H_1 and H_2). The fact that the number of support vectors is generally much smaller than the size the training set gives us a sparse solution and is interesting for very high dimensional problems.

As said before, we now seek to maximize the Lagrangian with respect to the dual variables α_i and obtain the following dual convex quadratic programming optimization problem:

$$\begin{cases} \max_\alpha \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ s.t. \quad \alpha_i \geq 0, \ \forall i \\ \quad \sum_{i=1}^{N} \alpha_i y_i = 0, \ \forall i \end{cases}$$

[6] From optimization theory, when working with constrained optimization problems with inequality constraints, we can write down primal and dual problems. The dual solution is always a lower bound on the primal solution (weak duality). The duality gap equals 0 under certain conditions (strong duality), and in such cases we can either solve the primal or dual problem since they have the same solution. Strong duality holds for the SVM problem, and in particular the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient for the optimal solution.

[7] The key idea here is that in the dual formulation is based on inner products $x_i^T x_j$.

[8] At the optimum, the Karush-Kuhn-Tucker (KKT) conditions include also the dual complementarity condition $\alpha_i(y_i(w^T x_i + b) - 1) = 0, \forall i$, the primal admissibility condition $y_i(w^T x_i + b) \geq 1, \forall i$ and the dual admissibility condition $\alpha_i \geq 0$.

Once we have the optimal values α^* by solving the optimization problem above (using the sequential minimal optimization (SMO) algorithm for instance [9]) we can find the optimal value of the weights vector w^* using $w^* = \sum_{i=1}^{N} \alpha_i^* y_i x_i$. Finally, considering the primal problem, we write the optimal value for the intercept using the fact that the constraint for a support vector x_{SV} is active (i.e., $y_{SV}(x_{SV}^T w^* + b) = 1$) at the optimum:

$$b^* = \frac{1}{N_{SV}} \sum_{j \in SV} (y_j - \sum_{i \in SV} y_i \alpha_i x_i^T x_j) \tag{3.47}$$

where SV is the set of indices of support vectors and $N_{SV} = Card(SV)$.

The optimal hyperplan (i.e., the trained classifier) that will allow us to classify a new observation x is:

$$H^*(x) = sign(\sum_{i=1}^{N} \alpha_i^* y_i x_i^T x + b^*) \tag{3.48}$$

Note that $\alpha_i^* y_i x_i^T x + b^*$ will be different from 0 only for the x_i's belonging to the set of support vectors.

3.4.3 Case 2: Not linearly separable data; Kernel trick.

We supposed in the previous section that the training data points are linearly separable in the input space. However, this is rarely the case and we want to construct non-linear decision boundaries. To do so, we introduce the well known kernel trick (and will explain why we call it a trick later) by mapping our data from the input space $\mathcal{X}$ to a higher dimentional feature space $\mathcal{F}$ using a non-linear function $\phi : \mathcal{X} \to \mathcal{F}$ such that the new hyperplane equation is $H(w, b) = \{x \in \mathbb{R}^p | w^T \phi(x) + b = 0\}$.

Consider the feature mapping $\phi(x) = (x, x^2, x^3)$ of a given attribute x (here an attribute means one column of X). If we want to use SVM algorithm with these new features, we can simply take the previous formulation and replace x everywhere with $\phi(x)$ (since the dual formulation is written in terms of inner products $\langle x, z \rangle$, we replace them with $\langle \phi(x), \phi(z) \rangle$). We define for each feature mapping ϕ the corresponding kernel [10]:

$$K(x, z) = \langle \phi(x), \phi(z) \rangle = \phi(x)^T \phi(z) \tag{3.49}$$

However explicitly computing non-linear features has a huge time and space complexity which grows with the dimentionality of the input space. In fact, suppose $x, z \in \mathbb{R}^p$ and consider:

$$K(x, z) = (x^T z)^2 = (\sum_{i=1}^{p} x_i z_i)(\sum_{j=1}^{p} x_i z_i) = \sum_{i,j=1}^{p} (x_i x_j)(z_i z_j) \tag{3.50}$$

This corresponds to $K(x, z) = \phi(x)^T \phi(z)$ where the feature mapping ϕ is given by (for example for $p = 3$):

$$\begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix} \tag{3.51}$$

As we can see it, the high-dimensional feature vector $\phi(x)$ is computed in $\mathcal{O}(p^2)$ time, whereas $K(x, z)$ takes only $\mathcal{O}(p)$ time (i.e., linear in the dimension of the input space). We never need to explicitly represent the vectors in the high-dimensional feature space $\mathcal{F}$ and instead we use a valid kernel [11] function $K(\cdot, \cdot)$. This is the kernel trick.

The most commonly used kernels are:

- Linear kernel:

$$K(x_i, x_j) = x_i^T x_j \tag{3.52}$$

- Polynomial kernel (of degree q and intercept r):

$$K(x_i, x_j) = (x_i^T x_j + r)^q \tag{3.53}$$

- Gaussian kernel (with γ controling the width of the Gaussian):

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \tag{3.54}$$

[9] This algorithm, due to John Platt, is the main implementation of the dual formulation of SVM that we find in the most important libraries like LIBSVM and Scikit-learn package. See [29] for more details.

[10] Definition: let $\mathcal{X}$ be a non-empty set. Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function. K is a positive definite kernel if and only if for any finite set $\{x_1, \ldots, x_N\}$ of $\mathcal{X}$ and the column vector c of $\mathbb{R}^N$, $c^T K c = \sum_{i,j} c_i c_j K(x_i, x_j) \geq 0$.

[11] Not every function $K(\cdot, \cdot) : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ can be a valid kernel, it has to satisfy the so-called Mercer condition. Otherwise, the underlying quadratic program may not be solvable. Mercer theorem: let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function. K is a positive definite kernel if and only if there exists a Hilbert space $\mathcal{F}$ and a feature map: $\phi : \mathcal{X} \to \mathcal{F}$ such that: $K(x, x') = \langle \phi(x), \phi(x') \rangle$.

Regarding the solution of the optimization problem, the formulation of the optimal hyperplane remains the same:

$$H^*(x) = sign(\sum_{i=1}^{N} \alpha_i^* y_i K(x_i, x) + b^*) \tag{3.55}$$

Hence, the original data points are projected in a high-dimensional feature space where a linear separating hyperplane can be found. This hyperplane corresponds to a non-linear decision boundary in the original input space. We say that that a linear decision boundary is "kernelized". Since the kernel $K(\cdot, \cdot)$ is associated to a scalar product in the feature space $\mathcal{F}$, it can be intrepreted as a sort of correlation or similarity measure between the points $\{(\phi(x_1), y_1), \ldots, (\phi(x_N), y_N)\}$ in $\mathcal{F} \times \{\pm 1\}$. That's why the choice of the kernel can be guided by the type of similarity that we expect in the data (prior knowledge).

Besides the introduction of non-linearity, the use of kernels is very useful in practice: the definition of SVMs over complex objects like images, graphs or proteins. In fact, in [30] the authors use image kernels that generally compute statistics or features of two images and then compare them.

3.4.4 Case 3: Not linearly separable data; Soft-margin linear SVM. In the previous section, we introduced the kernel trick and showed that mapping the data into a high dimensional feature space allows us to construct non-linear decision boundaries in the input space and we wish that this will allow us to perfectly classify the observations. However, this is not guaranteed. Moreover, constructing a separating hyperplane can be very sensitive to outliers or noisy measurements and can lead us to overfitting.

To have a more flexible decision boundary, we introduce the soft-margin SVM which is very similar to the hard-margin formulation except that we allow the classifier to misclassify some points by introducing a positive slack variable $\xi_i \geq 0, \forall i$ for each observation (x_i, y_i), which can be thought of distance from the separating hyperplane if the observation is misclassified. The new constaints are $y_i(w^T x_i + b) \geq 1 - \xi_i, \ \forall i$ and we have:

$$\begin{cases} \xi_i = 0 \quad if\ x_i\ is\ correctly\ classified \\ 0 \leq \xi_i \leq 1 \quad if\ x_i\ is\ in\ the\ margin \\ \xi_i > 1 \quad if\ x_i\ is\ misclassified \end{cases}$$

So in soft-margin SVM, data points on the incorrect side of the margin boundary have a penalty that increases with the distance from it (see Figure 4).

This leads to an optimization problem similar to the hard-margin case but with an aditional penalization term $C \sum_i \xi_i$ to penalize misclassification and margin errors :

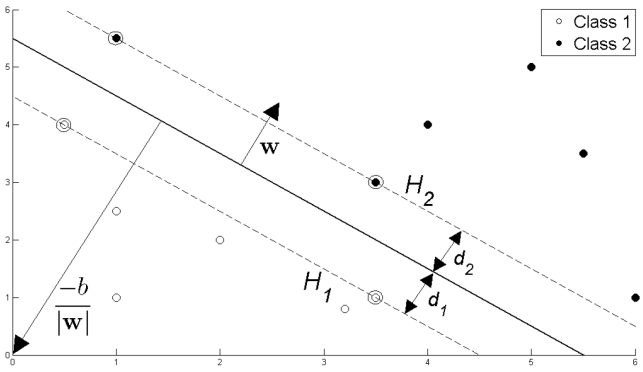

Figure 4: Soft-margin SVM. See [27].

$$\begin{cases} \min_{w,b} \ \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} \xi_i \\ s.t. \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \forall i \in \{1, \ldots, N\} \\ \qquad \xi_i \geq 0, \forall i \in \{1, \ldots, N\} \end{cases}$$

As we noticed it at the beginning, soft-margin SVM is a regularized version with the use of L_1 regularization [12] here. The parameter C controls the trade-off between the two objectives: maximizing the size of the margin and minimizing the number of misclassifications (this directly controls the over/under-fitting of the classifier).

This primal problem involves $(p+1+N)$ parameters and can be solved with optimization algorithms noticed above. However, we will derive the dual formulation for the same reasons as in hard-margin SVM.

As before, we minimize the Lagrangian function $\mathcal{L}(w, b, \xi, \alpha, \mu)$ with respect to the primal variables w, b and ξ and then we maximize the Lagrangian with respect to the dual variable α_i (actually $\alpha_i \geq 0, \ \mu_i \geq 0, \ \forall i$ are the Lagrange multipliers but we will see that the later can be written as a function of the former).

$$\begin{aligned} \mathcal{L}(w, b, \xi, \alpha, \mu) &= \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \\ &- \sum_{i=1}^{N} \alpha_i [y_i(w^T x_i + b) - 1 + \xi_i] - \sum_{i=1}^{N} \mu_i \xi_i \end{aligned} \tag{3.56}$$

We will only give the derivation with respect to ξ_i which give us the additional KKT stationarity condition [13]

[12] From a minimize(loss + λ penalty) view, soft-margin SVM primal formulation is similar to using the hinge loss $\sum_{i=1}^{N} [1 - y_i f(x_i^T)]_+$ with $\lambda \|w\|_2^2$ as penalty function.

[13] At the optimum, the Karush-Kuhn-Tucker (KKT) conditions include

(compared to hard-margin case): $\partial_{\xi_i}\mathcal{L}(w,b,\alpha) = C - \alpha_i - \mu_i = 0$. Knowing the $\mu_i \geq 0,\ \forall i$ we have: $0 \leq \alpha_i \leq C$.

$$\begin{cases} \max_\alpha \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ s.t. \quad 0 \leq \alpha_i \leq C,\ \forall i \\ \quad \sum_{i=1}^{N} \alpha_i y_i = 0,\ \forall i \end{cases}$$

We can see that the new dual problem differs from the one in the hard-margin case only regarding the constraint on the α_i's. We have now $0 \leq \alpha_i \leq C$ instead of $0 \leq \alpha_i$. Solving the dual problem with SMO-type algorithm allows us to find the optimal hyperplane. Finally, we note that in practice we use the "kernelized" version of the soft-margin SVM algorithm.

In the rest of the paper we will note `SvmLin` as the soft-margin SVM with linear kernel, `SvmPoly` as the soft-margin SVM with polynomial kernel and `SvmRbf` as the soft-margin SVM with radial basis function (Gaussian) kernel.

4 Experimental results

In this section we present detailed experimental results, applying the algorithms presented in Sec. 3 to forecast the sign of realized volatility minus anticipated volatility on the S&P500 index. We note this target variable as VD (volatility difference) where :

$$VD_t = VIX_{(t-21)} - VOL_{(t-21)\to t} \tag{4.57}$$

where VIX_t is the value of the VIX index at time t (a popular measure of the implied volatility of S&P500 index options calculated by the Chicago Board Options Exchange (CBOE)) and $VOL_{(t-21)\to t}$ is the realized volatility of the S&P500 index (computed as the annualized standard deviation of daily returns). We turn this problem into a binary classification task by taking the sign of the difference.

4.1 Experimental setup. We consider a set of 21 predictors presented in Table 1. Time series have been downloaded from Bloomberg at a daily frequency.

The entire dataset contains 3529 observations. We fix the size of the sliding window at 70% of the number of observations (i.e., 2470 samples from 10/1/2002 to 28/10/2011) and keep the remaining 30% (i.e., 1059 samples from 31/10/2011 to 15/1/2016) for the out-of-sample test set. We use a sliding window approach (known as walk forward analysis) as a simple solution to deal with concept drift and non-stationarity. We perform one-step-ahead predictions (i.e., classification) and all the models are relearned every 20 days. At each relearning, the parameters of each model are optimized using a grid search with 5-fold cross-validation adapted for time series[14] and with the values in Table 2.

After relearning, the optimized models are used to predict the 20 subsequent observations. Finally, we measure the performance of each model using the ROC curve which is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. It plots the true positive rate (TPR, called also sensitivity or recall) as a function of false positive rate (FPR, called also false alarm ratio). Before defining these quantities we present the confusion matrix in Figure 5. In the field of machine learning, a confusion matrix, also known as a contingency table or an error matrix, is a specific table layout that allows visualization of the performance of an algorithm.

Hence, we have that: TPR = TP/(TP+FN) and FPR = FP/(FP+TN) = 1-specificity.

Table 1: Set of predictors.

Code	Description
VOL_21_USGG3M Index	21 days ann. vol. on US generic govt. 3 months yield
VOL_21_RX1 Comdty	21 days ann. vol. on Euro Bund (fut. gen. 1st) returns
VOL_21_CCMP Index	21 days ann. vol. on Nasdaq 100 E-mini (fut. gen. 1st) returns
VOL_21_MXEF Index	21 days ann. vol. on MSCI Emerging Markets index returns
VOL_21_XAU Curncy	21 days ann. vol. on Gold spot returns
VOL_21_DXY Curncy	21 days ann. vol. on US dollar index spot rate returns
VOL_21_SPX Index	21 days ann. vol. on S&P500 E-mini (fut. gen. 1st) returns
VOL_21_CO1 Comdty	21 days ann. vol. on Crude Oil (fut. gen. 1st) returns
VOL_JPMVXYEM Index	J.P.Morgan Emerging Markets volatility index
VOL_21_EUR Curncy	21 days ann. vol. on EUR/USD spot returns
VOL_EURUSDV1M Index	EURUSD 1 month at-the-money implied volatility
VOL_21_NKY Index	21 days ann. vol. on Nikkei 225 index returns
VOL_V2X Index	Euro Stoxx 50 volatility index
VOL_21_USGG2YR Index	21 days ann. vol. on US generic govt. 2 years yield
VOL_JPMVXYGL Index	J.P.Morgan Global FX volatility index
VOL_21_SX5E Index	21 days ann. vol. on Euro Stoxx 50 index returns
VOL_21_CRY Index	21 days ann. vol. on CRB commodities index returns
R_SP_t_1	One day delayed S&P500 index returns
R_VIX_t_1	One day delayed VIX index first order difference
VIX_t_1	One day delayed VIX index
Spread_t_1	One day delayed difference realized minus anticipated volatility

Table 2: Values used for parameters optimization.

Parameter	Values	Description
C	linspace(0.001,10,50)	Regularization parameter
nb_tree	[1000,2000]	Number of classification trees in the forest
tree_depth	[2,3,4,5]	Maximum depth of each classification tree
γ	linspace(0.001,10,50)	Width of the Gaussian kernel
q	[2,3,4,5]	Degree of the polynomial kernel
r	linspace(-5,5,50)	Independent term in the polynomial kernel

also the dual complementarity condition $\alpha_i(y_i(w^T x_i + b) - 1 + \xi_i) = 0, \forall i$ and $\mu_i \xi_i = 0, \forall i$, the primal admissibility condition $y_i(w^T x_i + b) + \xi_i \geq 1, \forall i$ and $\xi_i \geq 0, \forall i$ and the dual admissibility condition $\alpha_i \geq 0$ and $\mu_i \geq 0$.

[14] In fact, classical cross-validation is not valid for times series forecasting since it implies predicting the past using the future after the first iteration.

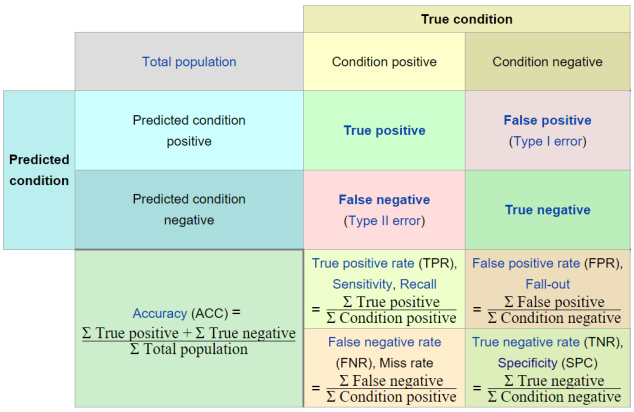

Figure 5: Confusion matrix.

In addition to the ROC curves we also compute, as performance measures, the precision (PRE), recall (REC) and F_1-score ratios:

$$PRE = \frac{TP}{TP+FP}$$
$$REC = \frac{TP}{TP+FN} \tag{4.58}$$
$$F_1 - score = 2\frac{PRE \times REC}{PRE + REC}$$

In pattern recognition with binary classification, the PRE ratio is intuitively the ability of the classifier not to label as positive (resp. negative) a sample that is negative (resp. positive). The recall is intuitively the ability of the classifier to find all the positive (resp. negative) samples. Precision can be seen as a measure of exactness or quality, whereas recall is a measure of completeness or quantity. In simple terms, high precision means that an algorithm returned substantially more relevant results than irrelevant, while high recall means that an algorithm returned most of the relevant results.

PRE and REC ratios are hence computed for each class. The F_1-score can be interpreted as a weighted harmonic mean of the precision and recall.

4.2 Results. In Figure 6 we present the ROC curves on the test set. We can make several observations:

- `LogRegL1` and `LogRegL2` methods as well as `SvmLin` have an Area Under the Curve (AUC) scores lower than the score achieved by a random guess (AUC=0.5). It is interesting to notice that logistic regression gives better accuracy than linear SVM.
- Non-linear machine learning algorithms have systematically superior performances with best score (AUC=0.67) achieved by `SvmPoly`.

Table 3: Precision (PRE), recall (REC) and F_1-score ratios achieved by the classifiers on the test set. Results are presented in the following format: value for class "-1" / value for class "+1".

	Precision	Recall	F_1-score
`LogRegL1`	0.12/0.84	0.28/0.65	0.17/0.73
`LogRegL2`	0.06/0.84	0.05/0.86	0.06/0.85
`RF`	0.3/0.86	0.13/0.95	0.18/0.9
`ExtraTF`	0/0.85	0/0.99	0/0.92
`SvmLin`	0.04/0.85	0.01/0.97	0.01/0.91
`SvmPoly`	0.3/0.89	0.41/0.83	0.35/0.86
`SvmRbf`	0.58/0.86	0.09/0.99	0.16/0.92

In Table 3 we present the results on the test set for the previously mentioned ratios for both classes "-1" and "+1". It allows us to have a deeper insight in the behavior of the algorithms. First of all, we can see that all used methods struggle in predicting correctly negative instances (low Precision scores) except for `SvmRbf` (PRE=0.58). However, `SvmRbf` is not able to find all the negative samples since its Recall score is very low (REC=0.09). This means that this algorithm "votes" rarely "-1" (less than the natural frequency we find in the dataset) but when the output is "-1", the confidence is high.

Hence, Table 3 shows that the superior performance of `SvmPoly` observed in Figure 6 is due to its ability to predict correctly negative samples with a relatively higher confidence compared to other algorithms. However, its performance on class "-1" remains not satisfactory.

5 Conclusions

In this paper, we gave some elements of statistical learning theory in order to understand the foundations and motivations of this field. We introduced machine learning algorithms giving the detailed derivations and theoretical justifications. We used the described methods to forecast the equity realized volatility (S&P500 index). Obviously, taking into account non-linear input-output relationships using machine learning methods gives better accuracy. In future work, we will treat the non-stationary nature of financial time series by looking at the literature related to concept drift and experts aggregation.

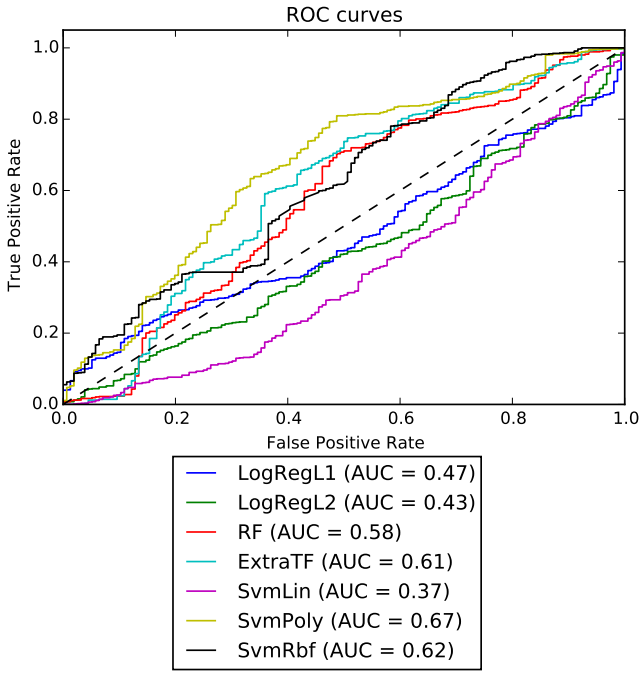

Figure 6: ROC curves on test set.

References

[1] V.N. Vapnik. An Overview of Statistical Learning Theory. IEEE Transactions on Neural Networks, 10(5), pp. 988-999, 1999.

[2] V.N. Vapnik. Statistical Learning Theory. Wiley-Interscience, 1998.

[3] V.N. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag New York, 1995.

[4] V.N. Vapnik and A.Y. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to their Probabilities. Theory of Probability and Its Applications, 16(2), pp. 264-280, 1971.

[5] M. Geist. Précis introductif à l'apprentissage statistique. Course at Supélec, 2014.

[6] O. Bousquet, S. Boucheron and G. Lugosi. Introduction to Statistical Learning Theory. Advanced Lectures on Machine Learning, pp. 169-207, 2004.

[7] A.Y. Ng. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. Proceedings of the Twenty-first International Conference on Machine Learning, pp. 78, 2004.

[8] L. Breiman, J. Friedman, R. Olshen and C. Stone. Classification and Regression Trees. Wadsworth International Group, 93(99), 1984.

[9] M. Kuhn and K. Johnson. Applied Predictive Modeling. Springer-Verlag, 2013.

[10] J. Friedman, T. Hastie and R. Tibshirani. The Elements of Statistical Learning. Springer Berlin, 2001.

[11] A. Criminisi, J. Shotton and E. Konukoglu. Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. Foundations and Trends in Computer Graphics and Vision, 2-3(7), pp. 81-227, 2012.

[12] L. Breiman. Bagging Predictors. Machine Learning, 24(2), pp. 123-140, 1996.

[13] L. Breiman. Random Forests. Machine Learning, 45(1), pp. 5-32, 2001.

[14] A. Montillo. Random Forests. Guest lecture: Statistical Foundations of Data Analysis, Temple University, 2009.

[15] P. Geurts, D. Ernst and L. Wehenkel. Extremely Randomized Trees. Machine Learning, 63(1), pp. 3-42, 2006.

[16] G. Louppe, L. Wehenkel, A. Sutera and P. Geurts. Understanding Variable Importances in Forests of Randomized Trees. Advances in Neural Information Processing Systems, pp. 431-439, 2013.

[17] L. Breiman. Consistency for a Simple Model of Random Forests. Technical Report, Statistical Department, University of California at Berkeley, 670, 2004.

[18] G. Biau, L. Devroye and G. Lugosi. Consistency of Random Forests and Other Averaging Classifiers. The Journal of Machine Learning Research, 9, pp. 2015-2033, 2008.

[19] G. Biau. Analysis of a Random Forests Model. The Journal of Machine Learning Research, 13(1), pp. 1063-1095, 2012.

[20] R. Genuer. Variance Reduction in Purely Random Forests. Journal of Nonparametric Statistics, 24(3), pp. 543-562, 2012.

[21] M. Denil, D. Matheson and N. De Freitas. Consistency of Online Random Forests. The Journal of Machine Learning Research, 28(3), pp. 1256-1264, 2013.

[22] M. Denil, D. Matheson and N. De Freitas. Narrowing the Gap: Random Forests in Theory and in Practice. Proceedings of The 31st International Conference on Machine Learning, pp. 665-673, 2014.

[23] E. Scornet, G. Biau and J.P. Vert. Consistency of Random Forests. Annals of Statistics, 43(4), pp. 1716-1741, 2015.

[24] B.E. Boser, I. Guyon and V. Vapnik. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144-152, 1992.

[25] C. Cortes and V. Vapnik. Support-Vector Networks. Machine Learning, 20(3), pp. 273-297, 1995.

[26] O. Chapelle. Training a Support Vector Machine in the Primal. Neural Computation, 19(5), pp. 1155-1178, 2007.

[27] T. Fletcher. Support Vector Machines Explained. Course at University College London, 2009.

[28] A. Statnikov, D. Hardin, I. Guyon and C.F. Aliferis. A Gentle Introduction to Support Vector Machines in Biomedicine. The American Medical Informatics Association (AMIA) Annual Symposium, 2009.

[29] R.E. Fan, P.H. Chen and C.J. Lin. Working Set Selection Using Second Order Information for Training SVM. Journal of Machine Learning Research, 6, pp. 1889-1918, 2005.

[30] M.B. Blaschko and C.H. Lampert. Learning to Localize Objects with Structured Output Regression. Computer Vision-ECCV, Springer Berlin Heidelberg, pp. 2-15, 2008.