

# An Evaluation of Equity Premium Prediction using Multiple Kernel Learning with Financial Features

Argimiro Arratia · Lluís A. Belanche · Luis Fábregues

Received: date / Accepted: date

**Abstract** This paper introduces and extensively explores a forecasting procedure based on multivariate dynamic kernels to re-examine –under a non-linear framework– the experimental tests reported by Welch and Goyal (*Review of Financial Studies* 21(4),1455-1508, 2008) showing that several variables proposed in the finance literature are of no use as exogenous information to predict the equity premium under linear regressions. For this non-linear approach to equity premium forecasting, kernel functions for time series are used with multiple kernel learning (MKL) in order to represent the relative importance of each of the variables. We find that, in general, the predictive capabilities of the MKL models do not improve consistently with the use of some or all of the variables, nor does the predictability by single kernels, as determined by different resampling procedures that we implement and compare. This fact tends to corroborate the instability already observed by Welch and Goyal for the predictive power of exogenous variables, now in a non-linear modelling framework.

**Keywords** Support vector classification · Support vector regression · Financial time series · Multiple kernel learning · Kernel functions for time series

## 1 Introduction

There is a long history of attempts to predict stock market returns by specifying regression models based on lagged predictor variables independent of the stock market returns. Shiller [27], Campbell and Shiller [6], and Cochrane [10], among others, have studied the forecasting of future excess returns using the dividend price ratio as predictor. Other popular predictor variables explored in the literature are the dividend yield, earnings price ratio, dividend-to-earnings ratio, volatility,

---

A. Arratia research is supported by grant TIN2017-89244-R from MINECO (Ministerio de Economía, Industria y Competitividad) and the recognition 2017SGR-856 (MACDA) from AGAUR (Generalitat de Catalunya)

A. Arratia, Ll. Belanche, L. Fábregues  
Computer Science Department, Universitat Politècnica de Catalunya, Jordi Girona, 1-3 08034, Barcelona, Spain. E-mail: {argimiro,belanche}@cs.upc.edu, luis.fabregues@est.fib.upc.edu

interest rates, exchange rates, consumption indices and inflation rates (see, e.g., [17, 20, 22, 23], and [12] for a general discussion). Indeed, the list of valuation ratios sought of as possible forecasters of expected excess returns is much longer and show “... a pervasive pattern of predictability across markets wherein the cashflow or price change one may have expected is not what is forecast.” [12]. In view of this and further evidence showing the spurious nature of predictor models (mostly linear regressions on the aforementioned valuation ratios), several authors have conducted extensive studies on the forecasting performance of various economic variables and different models (to mention a few, [2, 7, 11, 28]). The latter, work by Welch and Goyal [28], is of particular interest since the authors carry out a comprehensive revision of the empirical performance of the most widely accepted variables as predictors of equity premium –under *linear* regression models– and conclude that these models have poor predictive capabilities, both in-sample and out-of-sample.

In this work we extend the stock return predictability tests of Goyal and Welch to non-linear (and semi-parametric) models on the different valuation ratios. We are thus considering the possible non-linear relationship between the stock returns and the predictor variables and expanding the framework of predictability from linear to more complex models, in terms of both classification and regression settings. The models we consider come from kernel-based statistical learning adapted to the goal of time series forecasting. We use  $\nu$ -Support Vector Machines for classification and regression [8], coupled with kernel functions able to analyze multivariate temporal structures, and Multiple Kernel Learning (MKL) [3] to integrate the different financial information and the different kernels.

The rest of the paper is structured as follows. In Section 2, we present the multivariate dynamic kernels used in this study, including the construction of new kernels. In Section 3, we present the non-linear forecasting methodology and the formal definition of the variables used as features for the kernels. Section 4 follows with the results of the experimental design from a trading perspective and discusses the performance of the kernels methods. Section 5 presents results of a larger experiment –set up following a classical factorial design– that compares performance of each individual kernel with combinations of the economic variables against the linear composition of kernels given by MKL. Finally, we summarize the major findings of the study and give indications for future research.<sup>1</sup>

## 2 Multivariate Dynamic Kernels for Time Series

Kernels are two-place symmetric functions that evaluate an inner product of the arguments in some suitable feature space, thereby inducing an implicit mapping that creates an image of the inputs into this feature space. More formally, a kernel function  $k$  implicitly defines a map  $\phi : \mathcal{X} \rightarrow \mathcal{H}$  from an input space of objects  $\mathcal{X}$  into some Reproducing Kernel Hilbert Space (RKHS)  $\mathcal{H}$  (called the *feature space*). The “kernel trick” consists in performing the mapping and the inner product simultaneously as in:

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}, \quad \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \quad (1)$$

---

<sup>1</sup> A preliminary report of this work has been presented at the IWAN 2017 meeting [16].

where  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$  denotes inner product in  $\mathcal{H}$ . It is known that a function is a valid kernel function if and only if it induces positive semi-definite (p.s.d.) matrices  $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]$ . This property can be expressed as

$$\mathbf{c}^\top \mathbf{K} \mathbf{c} = \sum_{i=1}^N \sum_{j=1}^N c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0, \quad (2)$$

for all  $N \in \mathbb{N}$ ,  $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathcal{X}$  and  $\mathbf{c} \in \mathbb{R}^N$ . Kernels for time series can be constructed using two approaches: structural similarity and model similarity. *Structural* similarity employs methods to find an alignment of the data that makes the comparison between series possible. *Model* similarity changes the structure of the data by constructing a high-level representation (e.g., a model) and the comparison is performed using this new representation [21].

The general goal is to define p.s.d. kernels between two time series (not necessarily of the same length),  $\mathbf{s}_1 = (s_1(1), \dots, s_1(N_1))$  and  $\mathbf{s}_2 = (s_2(1), \dots, s_2(N_2))$  where the pairwise comparisons  $(s_1(i), s_2(j))$  are reasonable.<sup>2</sup> One of the main difficulties is that the commonly used Euclidean distance disregards the temporal dependency among the observations of the two series. In the next section the main kernel functions used in this work will be formally described.

## 2.1 Vector Auto-Regression Kernel

The Vector Auto-Regression (VAR) is a classical econometric model that finds correlations between a set of variables given a certain number of lags. Each variable is defined by a function that represents its changes over a defined period of time using past information of the series and other variables. The VAR model relates the observation  $\mathbf{x}(t)$  at time  $t$  with a linear combination of lagged values of the observation. In order to fit a model, a lag parameter  $L$  is provided, which defines how many time steps the function will be looking at in the past to assess the linear combination parameters, as in:

$$\mathbf{x}(t) = \sum_{l=1}^L \mathbf{A}_l \mathbf{x}(t-l) + \mathbf{b} + \boldsymbol{\varepsilon}_t, \quad (3)$$

where  $L$  is the number of lags of the model,  $\mathbf{A}_1, \dots, \mathbf{A}_L$  are the transition matrices (each a square matrix with as many dimensions as features the data has),  $\mathbf{b}$  is the intercept (a vector of dimension equal to the number of features) and  $\boldsymbol{\varepsilon}_t$  is the Gaussian noise at time  $t$ . The VAR is a model by itself, capable of predicting values for new data, but the information generated can be used to create a kernel. VAR kernels can be built using three steps [24]:

1. Build two VAR models, one for each time series using a certain (common) number of lags.
2. Bind the values of the transition matrices and intercepts of each series and calculate the Frobenius norm over the difference of those values.

<sup>2</sup> In financial series, the length of the different time series is variable since it is a function of the number of business days of each month, among other causes.

3. Apply a Radial Basis Function (RBF) transform to the Frobenius distance to convert it to a similarity measure and a p.s.d. kernel.

Formally, given an obtained VAR model for a series  $\mathbf{s}$ , we consider the binding:

$\hat{B}(\mathbf{s}) = [\mathbf{A}_1 | \mathbf{A}_2 | \dots | \mathbf{A}_L | b]$ . The distance between series  $\mathbf{s}_1$  and  $\mathbf{s}_2$  is then defined as the Frobenius norm of the difference between the respective bindings:

$$d_F(\mathbf{s}_1, \mathbf{s}_2) = \sqrt{\text{Tr} \left\{ (\hat{B}(\mathbf{s}_1) - \hat{B}(\mathbf{s}_2))(\hat{B}(\mathbf{s}_1) - \hat{B}(\mathbf{s}_2))^T \right\}}. \quad (4)$$

Once this Frobenius distance is calculated, it can be transformed into a valid kernel using a Laplacian RBF transform:

$$k_{\text{VAR}}(\mathbf{s}_1, \mathbf{s}_2) = \exp \left\{ \frac{-d_F(\mathbf{s}_1, \mathbf{s}_2)}{2\sigma} \right\} \quad (5)$$

The parameters of this kernel methodology are the number of lags  $L$  and  $\sigma > 0$ . In this work, the value of  $L$  will be fixed to 5, whereas  $\sigma$  will be set to the median Frobenius distance between the time series being compared. Both parameters are set following previous recommendations [14] and the authors' own experience on this type of data [24].

## 2.2 Global Alignment Kernel

Sakoe and Chiba proposed the now classic *dynamic time warping* algorithm (DTW), to find a good alignment between two series before computing any Euclidean distance [25]. DTW is still quite popular for the classification of time series (see, e.g., [19]). An *alignment* (or *warping function*)  $\pi$  between two time series  $\mathbf{s}_1$  and  $\mathbf{s}_2$  is a pair of increasing tuples  $(\pi_1, \pi_2)$  of length  $P \leq N_1 + N_2 - 1$  such that  $1 = \pi_1(1) \leq \dots \leq \pi_1(P) = N_1$  and  $1 = \pi_2(1) \leq \dots \leq \pi_2(P) = N_2$ , with unitary increments and no simultaneous repetitions. Intuitively, an alignment is a series of connecting lines associating each time point of  $\mathbf{s}_1$  to one or more time points in  $\mathbf{s}_2$ , and *vice versa*. The associations are:  $\mathbf{s}_1(t)$  with  $\mathbf{s}_2(t)$  denoted by  $\rightarrow$ ,  $\mathbf{s}_1(t)$  with  $\mathbf{s}_1(t+1)$  denoted by  $\uparrow$  and  $\mathbf{s}_1(t)$  with  $\mathbf{s}_2(t+1)$  denoted by  $\nearrow$ . In practice, these can be represented as two integer vectors  $\pi_1, \pi_2$  of the same length with binary increments:

$$\begin{pmatrix} \pi_1(i+1) - \pi_1(i) \\ \pi_2(i+1) - \pi_2(i) \end{pmatrix} \in \left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$$

The length of these vectors is always equal or lower than the length of the smallest series. For the sake of simplicity the two vectors that represent the alignment will be denoted as  $\pi$ . Note that, by definition, the alignments only consider values of zero or one lag in both series.

After obtaining a satisfying alignment, the distance between the two series can be obtained as:  $D_\pi(\mathbf{s}_1, \mathbf{s}_2) = \sum_{i=1}^{|\pi|} d(x_{\pi_1(i)}, y_{\pi_2(i)})$ , where the function  $d$  can be any metric –most often an Euclidean one– or an arbitrary conditionally negative

definite kernel. The optimal alignment will be the one that minimizes the distance between the series, as given by:

$$\text{MDTW}(\mathbf{s}_1, \mathbf{s}_2) = \min_{\pi \in \mathcal{A}(\mathbf{s}_1, \mathbf{s}_2)} \frac{D_\pi(\mathbf{s}_1, \mathbf{s}_2)}{|\pi|}, \quad (6)$$

where  $\mathcal{A}(\mathbf{s}_1, \mathbf{s}_2)$  is the set of all possible alignments between  $\mathbf{s}_1$  and  $\mathbf{s}_2$ . This distance measurement does not fulfill the p.s.d.-ness requirement to form a kernel, even after applying an RBF transformation. Global Alignment (GA) [13] is a generalization of DTW which enables the creation of a structural similarity kernel. The method follows the same computational steps as MDTW but, instead of selecting the alignment with minimum distance, it considers all the alignments, and combines them with an exponentiated *soft-minimum*,<sup>3</sup> based on the idea that all alignments provide valuable information about the similarities between the series.

The formulation of this kernel function can be expressed using several distance metrics. The following formula is the one used in the context of this work:

$$k_{\text{GA}}(\mathbf{s}_1, \mathbf{s}_2) = \sum_{\pi \in \mathcal{A}(\mathbf{s}_1, \mathbf{s}_2)} \exp(-D_\pi(\mathbf{s}_1, \mathbf{s}_2)) \quad (7)$$

An improvement over GA was introduced under the name of Fast Global Alignment [15], aiming at reducing the computational time of the procedure. This is accomplished by using an extra parameter  $T$  that restricts the number of alignments taken into account during the final calculation of  $k_{\text{GA}}$ .

### 2.3 Multivariate Dynamic Euclidean Distance Kernel

In the same lines as the GA kernel, the Multivariate Dynamic Euclidean Distance Kernel (MDED) is a structural similarity model that creates an alignment between two series of different size in order to be able to compute the distance measure. The MDED opts for a simpler approach, as it removes the first elements of the longest series until it matches the size of the shortest time series. Even if this alignment is potentially worse in most of the cases with respect to MDTW, MDED is computationally much less expensive. The approach is backed by financial theory: observations generated in later time stamps already contain information from older ones. Having that  $N_1 \leq N_2$ , where  $N_1$  and  $N_2$  are the lengths  $\mathbf{s}_1$  and  $\mathbf{s}_2$  respectively, we define this alignment in the notation of DTW as  $\pi_1 = [0, 1, 2, \dots, N_1 - 1, N_1]$  and  $\pi_2 = [N_2 - (N_1 - 1), N_2 - (N_1 - 2), \dots, N_2 - 1, N_2]$ . Using this alignment, the distance between the series is computed as in eq. (5). The basic metric employed is again the Euclidean distance; in a way analogous to the  $k_{\text{VAR}}$ , the final expression for the MDED kernel is obtained as in [24]:

$$k_{\text{MDED}} = \exp \left\{ \frac{-D_\pi(\mathbf{s}_1, \mathbf{s}_2)}{2\sigma} \right\} \quad (8)$$

The  $\sigma$  parameter of this kernel function is again estimated following the authors' past experience [24], using the median of all  $D_\pi$  of the available data.

<sup>3</sup> For a vector of positive scalars  $\mathbf{z} = (z_1, z_2, \dots, z_n)^\top$ , the softmin is defined as  $\log \sum e^{-z_i}$ .

## 2.4 Multivariate Dynamic Arc-Cosine Kernel

The Arc-Cosine kernel relies on an integral representation, and provides interesting connections with neural networks [9]. Specifically, given two vectors  $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ , the kernel is defined as:

$$k_{\text{ARC}}^n(\mathbf{x}, \mathbf{x}') = 2 \int d\mathbf{w} \frac{\exp(-\frac{\|\mathbf{w}\|^2}{2})}{(2\pi)^{d/2}} \Theta(\mathbf{w}^\top \mathbf{x}) \Theta(\mathbf{w}^\top \mathbf{x}') (\mathbf{w}^\top \mathbf{x})^n (\mathbf{w}^\top \mathbf{x}')^n \quad (9)$$

where  $n$  is a ‘‘degree’’ parameter and  $\Theta(z) = \frac{1}{2}(1 + \text{sgn}(z))$ . The solution of the previous integral results in:

$$k_{\text{ARC}}^n(\mathbf{x}, \mathbf{x}') = \pi^{-1} \|\mathbf{x}\|^n \|\mathbf{x}'\|^n J_n(\theta) \quad (10)$$

which shows a rather trivial dependence on the lengths of  $\mathbf{x}, \mathbf{x}'$ , but a more complex relation via the angle ( $\theta$ ) between them:

$$\theta = \cos^{-1} \left( \frac{\mathbf{x}^\top \mathbf{x}'}{\|\mathbf{x}\| \cdot \|\mathbf{x}'\|} \right) \quad (11)$$

The function  $J_n$  is expressed as follows:

$$J_n(\theta) = (-1)^n (\sin \theta)^{2n+1} \left( \frac{1}{\sin \theta} \frac{\partial}{\partial \theta} \right)^n \left( \frac{\pi - \theta}{\sin \theta} \right) \quad (12)$$

where  $(-1)^n \left( \frac{1}{\sin \theta} \frac{\partial}{\partial \theta} \right)^n \left( \frac{\pi - \theta}{\sin \theta} \right)$  is the  $n$ -th partial derivative of  $\left( \frac{\pi - \theta}{\sin \theta} \right)$  with respect to  $\cos \theta$ . In particular,  $J_0(\theta) = \pi - \theta$  and  $J_1(\theta) = \sin(\theta) + (\pi - \theta) \cos(\theta)$ . A structural similarity model for time series can be built using this formulation [16], making the kernel mainly depend on the angle between the series, given by  $k_{\text{ARC}}^n(\mathbf{s}_1, \mathbf{s}_2)$ . In the context of this work, only  $n = 0$  and  $n = 1$  will be considered.

## 2.5 $\nu$ -Support Vector Machines

In classical support vector regression (SVR), the proper value for the  $\epsilon$  parameter –which determines the specific error function being used– is difficult to determine beforehand. This problem is partially addressed in  $\nu$ -SVR, in which  $\epsilon$  itself is a variable in the optimization process and is controlled by another parameter  $\nu \in (0, 1)$ , being an upper bound on the fraction of training errors and a lower bound on the fraction of points inside the  $\epsilon$ -insensitive tube, making it a more convenient parameter. Suitable descriptions of the  $\nu$ -SVR can be found elsewhere [26], both for classification and regression tasks.

## 2.6 Multiple Kernel Learning

Multiple Kernel Learning (MKL) [3] aims at finding the best combination of kernels to solve a task. It is possible for a problem to have several kernel functions that cover different characteristics of the data, or different representations of the same data. Those procedures create different kernel matrices that can be used to train

a predictive model. Although it is possible to combine the information of those matrices into a single combined kernel matrix, MKL makes possible to combine in the same predictive model information obtained using  $P$  different techniques. The mathematical formulation of this process is the following:

$$k_\eta(\mathbf{x}_i, \mathbf{x}_j) = f\left(\{k_m(x_i^m, x_j^m)\}_{m=1}^P; \eta\right), \quad (13)$$

where  $k_\eta$  represents the combined kernel,  $f$  is the (linear or may be non-linear) combination function,  $k_m$  represents the  $m$ -th kernel function and  $\eta$  parametrizes the combination. In this paper, the algorithm of choice is EasyMKL [1], which obtains the combination parameters  $\eta$  using an optimization approach that starts with defining the convex combination:

$$k_\eta = \sum_{m=1}^P \eta_m k_m, 0 \leq \eta_m \leq 1, \sum_{m=1}^P \eta_m = 1, \quad (14)$$

where, in this case,  $\eta_m$  is the assigned weight to each kernel matrix. Specifically, a max-min problem is solved involving the  $\eta$  parameters and the probability distribution  $\gamma$  of each class. After the  $\eta$  weights are obtained they are combined convexly (the optimization restrictions ensuring that the weights are positive and sum to one). The  $L_1$  norm is used as a structural risk function to guide the process. As base learner EasyMKL uses KOMD, a kernel classifier that performs direct optimization of the margin distribution. The initial optimization equation is:

$$\max_{\eta: \|\eta\|=1} \min_{\gamma \in T} Q(\eta, \gamma) = \max_{\eta: \|\eta\|=1} \min_{\gamma \in T} (1 - \lambda) \gamma^T \hat{y} \left( \sum_{m=1}^P \eta_m \widehat{\mathbf{K}}_m \right) \hat{y} \gamma + \lambda \|\gamma\|^2 \quad (15)$$

where  $\lambda$  is an exogenous parameter of the optimization process,  $\hat{y}$  is the vector of training set classes and  $\widehat{\mathbf{K}}_m$  is the  $m$ -th kernel matrix of the training set. The use of MKL in computational finance is not new (see, *e.g.*, [18]). Common implementations include the capability of predicting values using the KOMD classifier. In the context of this paper, the chosen procedure is to extract the parameters yielded by EasyMKL, build the combined kernel matrix and use it to train a  $\nu$ -SVM. This makes the method capable of more accurate predictions as the  $\nu$  parameter is of capital importance for these models. In this work, the explored values are in  $\{0.2, 0.4, 0.6, 0.8, 1\}$ .

### 3 A non-linear predictability framework

As stated in the introduction, the aim of this work is to replicate the experiments performed by Welch and Goyal in [28] with kernel methods and using multiple kernel learning to weight each feature in order to determine its relative influence in the prediction of the equity premium. The experimental work uses the same data as in [28] for a partial set of the variables considered by the authors. It is a data set that comprises several financial variables measured monthly, quarterly and yearly in the range of years between 1871 and 2005, and compiled from several sources.<sup>4</sup>

<sup>4</sup> Publicly available from <http://www.hec.unil.ch/agoyal/>.

The experimental process follows two phases: the evaluation of each combination of kernels using only endogenous variables and the comparison between exogenous variables using multiple kernel learning.

### 3.1 Considered Variables

The dependent variable or target for our predictors is, as in [28], the S&P 500 index equity premium, formally computed as follows. Let  $SPX$  denote the price series of the S&P 500 index, and  $r$  a short term risk-free interest rate. In practice  $r$  is obtained as the interest rate of the three months U.S. Treasury bill. Let  $D12$  be the 12-month moving sums of dividends paid on the S&P 500 index.<sup>5</sup> The *equity premium* for the S&P 500 index, or  $SPXeqp$ , is the difference of the total rate of returns, including dividends, and the risk-free interest rate:

$$SPXeqp_t = \log\left(\frac{SPX_t + D12_t}{SPX_{t-1}}\right) - \log(r_t + 1) \quad (16)$$

Specifications for each of the variables that we use as predictors follow. These conform a subset of the variables considered by Welch and Goyal [28].

*Dividend Price ratio (DP)*. Both this variable and the next are dependent on the variable  $D12$ . The formula for Dividend Price Ratio at time  $t$ , is

$$DP_t = \log(D12_t) - \log(SPX_t). \quad (17)$$

*Dividend Yield ratio (DY)*. Analogous to the Dividend Price, but considering past values of  $SPX$ :

$$DY_t = \log(D12_t) - \log(SPX_{t-1}). \quad (18)$$

*Earnings-to-Price (EP)*. Let earning price ( $E12$ ) be the moving sum of earnings from the S&P 500 index in a window of 12 months.<sup>6</sup> The Earning Price is then calculated as

$$EP_t = \log(E12_t) - \log(SPX_t). \quad (19)$$

*Stock Variance (SV)*. The sum of squared daily returns of the  $SPX$ .

*Book-to-Market ratio (BM)*: The ratio of book value to market value for the Dow Jones Industrial Average. From March to December, this is computed by dividing book value at the end of the previous year by the price at the end of the current month; for January and February, it is computed by dividing book value at the end of two years ago by the price at the end of the current month.

<sup>5</sup> The dividends data is obtained from R. Shiller's website and the S&P Corporation.

<sup>6</sup> Part of this data is extracted from R. Shiller's website and part is the result from an interpolation process by Welch and Goyal.



### 3.2 Experimental Methodology

The available data has to be grouped in blocks of a given size, as the described kernel functions need a sequence of events to extract their similarity. A simple but effective approach is to group monthly data into yearly blocks, creating data structures that contain 12 months, and then build moving windows of a certain number of years. The target variable is the equity premium of the first month (January) of the next year. The input variables are constructed as described in the previous section. Included in the data frames are lagged values of each variable in order to increase the information of each data entry and help the models form better temporal dependencies.

Our approach is then divided in two steps. The first is to determine which is the best kernel function or combination thereof to perform the predictive task (classification and regression, see later). The second is to use the selected method to determine the relative importance of each exogenous variable using the weights of the trained Multiple Kernel Learning (MKL) model.

The first step is in accordance with the standard validation-based methodology to evaluate several models: perform a predictive task using common data and compare the results using a predefined metric. In this step, the different kernel functions considered will be used both individually and in conjunction using MKL. Each of these models will have its parameters  $\nu$ ,  $\sigma$  and  $\lambda$  individually tuned.

Our implementation considers two types of validation: Out-of-Sample (OOS) and 5-fold Cross-Validation (CV). The inclusion of two validation procedures is due to the historically controversial use of CV for the case of time series. By the nature of this approach the folds do not respect any particular order (*e.g.*, the validation part could be the oldest, using a model trained with “future” data). This can be considered unrealistic although, truly, all the data used has already been observed at model construction time. However, it has been recently argued that CV *can* be used with stationary time series provided that the predictor values are lagged versions of the response value [4].

In the case of the single kernel functions, the creation and evaluation of the model is straightforward: a  $\nu$ -SVM is fitted using the training data and tested or validated with the rest of the data. In the case of MKL, all kernel functions introduced in section 2 will create a corresponding kernel matrix. Unfortunately, the implementations of EasyMKL do not admit any parameter, reducing the adaptability of the model. To correct this, the training procedure will fit an EasyMKL model, extract the weights, build the combined matrix of kernels and then train a standard  $\nu$ -SVM. The second step tries to determine the relative importance of each exogenous variable by using the MKL weights. To this end, one data frame for each exogenous variable is created, each containing the exogenous variable and *SPX*, with different lags. In the list of data frames for the MKL are also included the matrix with the endogenous variable (*SPX*) and a data frame with all the variables. Using this representation, the input to each experiment will be a list containing seven matrices encapsulating the following features sets:

1. *SPX* only
2. *SPX* plus Dividend-Price Ratio
3. *SPX* plus Earning Price
4. *SPX* plus Dividend Yield

5. SPX plus Stock Variance
6. SPX plus Book-to-Market Ratio
7. All of the above

## 4 Results and discussion

### 4.1 Equity Premium classification experiments

In this first set of experiments we will consider as target variable not the equity premium but its sign, or *direction*. A positive value indicates a good precondition to hold a share and a negative value indicates a proper time to sell the stock shares. The main performance metric is the accuracy, the fraction of correctly classified samples, reported as training, validation and test accuracies.<sup>7</sup> A second metric is given by the weights of the learned MKL models, which can be helpful to determine the relative importance of each kernel function.

Table 1 shows the evaluation of the different kernel methods and multiple kernel learning using EasyMKL. These results are obtained using the same data for each method but adjusting the parameters individually. All the methods are compared using both OOS and CV.

Out Of Sample Validation							
	$k_{\text{VAR}}$	$k_{\text{GA}}$	$k_{\text{MDED}}$	$k_{\text{ARC}}^0$	$k_{\text{ARC}}^1$	MKL	MKL (norm.)
Train	0.588	0.739	0.722	0.674	0.734	0.982	0.777
Validation	0.872	0.859	0.740	0.491	0.452	0.529	0.763
Test	0.631	0.640	0.640	0.640	0.658	0.640	0.694
Cross-Validation							
	$k_{\text{VAR}}$	$k_{\text{GA}}$	$k_{\text{MDED}}$	$k_{\text{ARC}}^0$	$k_{\text{ARC}}^1$	MKL	MKL (norm.)
Train	0.610	0.762	0.742	0.675	0.729	1.000	0.898
Validation	0.751	0.691	0.674	0.559	0.445	0.417	0.568
Test	0.568	0.649	0.631	0.640	0.640	0.667	0.658

**Table 1** Accuracy results for all kernel combinations. MKL (norm.) refers to MKL using normalized kernel matrices.

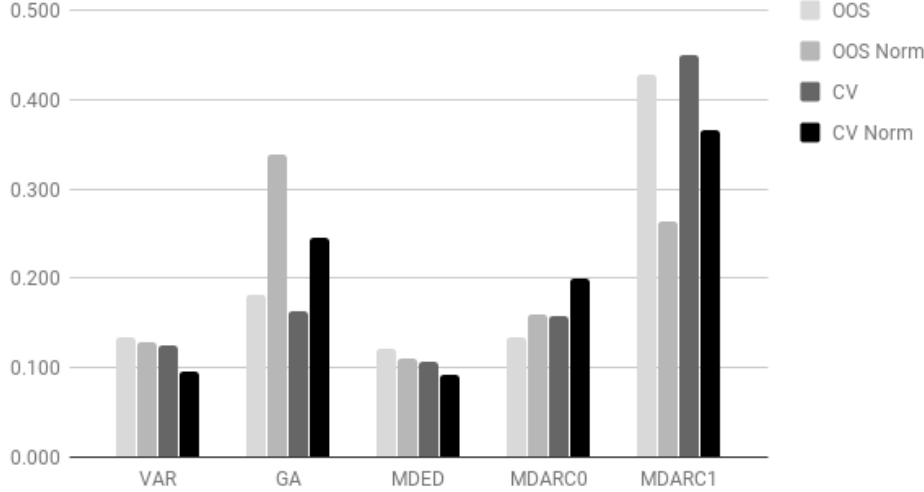
All test accuracies fall within the range from 55% to 70%, which indicates the general capacity of Multivariate Dynamic kernels for this task. Individual kernels share similar test accuracies (around the 64% mark) including the simpler kernels like  $k_{\text{MDED}}$  and  $k_{\text{ARC}}^0$ . It is also worth to mention that the kernels based on the arc-cosine perform as well if not better than other more common kernel functions. In particular,  $k_{\text{ARC}}^1$  is the best performing individual kernel, surpassing even  $k_{\text{GA}}$  in the version using OOS validation.

By a considerable margin, the best performing method is the combination of kernels created with EasyMKL, surpassing all the individual kernels. This technique tends to over-fit, as can be observed in the difference of accuracy between

<sup>7</sup> The training results are computed refitting the model using the best set of parameters.

training, validation and testing, and most prominent in the case of CV, were training accuracy is much higher than testing and validation figures. Kernel normalization has remarkable consequences on the results: it reduces training accuracy and increases validation accuracy, reducing over-fitting. The results of the normalization technique seem to differ depending on the validation technique. In this vein, there is no clear pattern to determine if CV is better or worse than OOS since all kernels react differently. In the cases of  $k_{GA}$  and standard MKL the results improve, but the rest of kernel functions have similar or worse test accuracies.

### Weights of kernel methods with different MKL procedures for classification



**Fig. 1** Weights of the kernels with different MKL procedures for the classification experiments under Out of Sample (OOS) and 5-fold Cross-Validation (CV), both for normalized and unnormalized matrices.

Figure 1 displays the resulting weights of the MKL process. It can be observed that  $k_{ARC}^1$  usually is the kernel with most weight, further supporting the fact that this kernel function is possibly the best performing one for the problem. However, in the case of MKL with OOS validation and normalization,  $k_{GA}$  is the kernel with highest weight and this combination is also the one with higher test accuracy.  $k_{ARC}^0$  also receives weights higher than their mean, signaling that it is also important in the construction of the model and provides additional information. Finally, it is worth to comment the impact of the normalization in the weights: in all the cases it decreases the weights of  $k_{VAR}$  and  $k_{MDED}$ , the worst performing kernel functions.

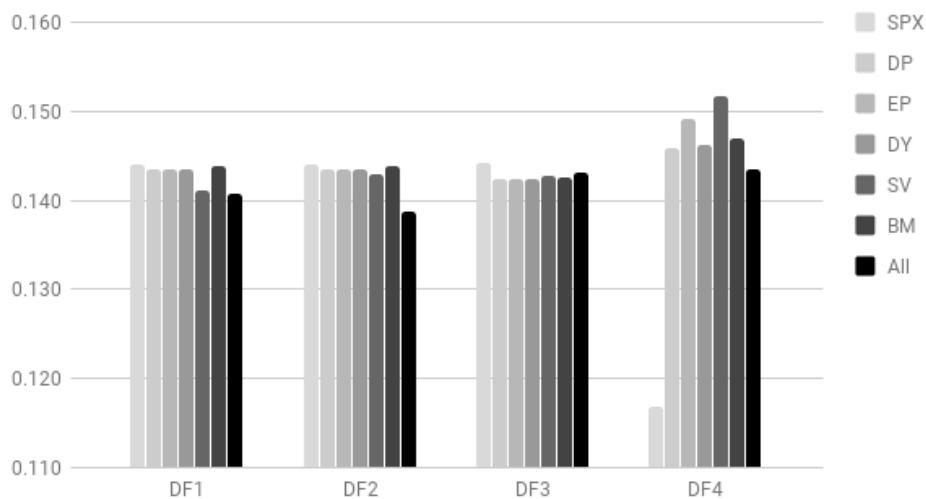
Table 2 and Fig. 2 display the results obtained using exogenous variables. These experiments are carried out using the best performing kernel method: normalized MKL using OOS validation. Each variable is contained in a data frame that will be transformed into a kernel matrix using the MKL procedure. This creates four data frames (one for each exogenous variable) plus one containing only the endogenous variable (the S&P 500 index) and an additional data frame containing all the variables. In order to extract more information from these variables, four different

methods of building the data frames were tested. DF1 only considers the exogenous variables without lags and SPX with four lags is included in each data frame. DF2 adds to the information of DF1 each exogenous variable with a lag of 3; this is motivated by the fact that the resulting data frame will contain more information but without adding too much redundancy. DF3 includes four lags of each exogenous variable. DF4 contains the same information as DF3 for the exogenous variables, but only includes the SPX without lags.

	Accuracy		
	Train	Validation	Test
DF1	0.767	0.757	0.660
DF2	0.768	0.757	0.660
DF3	0.759	0.744	0.680
DF4	0.744	0.787	0.649

**Table 2** Accuracy results for the data configurations with exogenous variables (see text).

### Weights of variables with different experimental settings using MKL



**Fig. 2** Weights of variables with different experimental settings using MKL.

The best performing method is DF3 in terms of test accuracy. DF1 and DF2 are fairly similar, indicating that the inclusion of the third lag does not affect the model too much. DF4 is the worst, which means that the model heavily relies on the lags of SPX for its predictions. The weights of DF1 and DF2 are near the mean, with slightly higher weight for the endogenous variable. This result does not imply that the rest of variables are not important to predict the result

(that would be represented by weights near zero) but that they are mostly equally important. DF3 shows a shift in the weights, as SV (Stock Variance) is near SPX. The data frame containing all the variables also increases in weight, indicating a better predictive capability of all the variables by including all their lags until 4. Further experimental research shows that the fourth lag of the stock variance provides relevant information that helps the predictive process. Finally, DF4 has its weights shifted towards the exogenous variables. As those data frames lack lags of the SPX, it is possible that the information contained in the exogenous variables takes a more active role in the prediction procedure; however, the model performs worse in comparison to the rest. It is also worth noting that stock variance is still one of the most relevant variables in this case.

## 4.2 Regression experiments

In order to provide robustness to the experimental framework constructed, the regression task is also explored. In this case the target values change from a categorical label to the Equity Premium itself. The structures of data compression and frame grouping are kept the same as in the classification experiment. The  $\nu$ -SVM for classification is replaced by its regression counterpart and the performance metric is now the mean squared error.

Tables 3 and 4 show the evaluation of the different kernel methods and MKL. These results are obtained using the same data for each method but adjusting the parameters individually. Again, the impact of using OOS validation and CV is also tested. All the methods are validated in the same range of hyper-parameters.

	$k_{\text{VAR}}$	$k_{\text{GA}}$	$k_{\text{MDED}}$	$k_{\text{ARC}}^0$	$k_{\text{ARC}}^1$	MKL	MKL (norm.)
Train	0.0140	0.0029	0.0039	0.0045	0.0015	0.0032	0.0132
Validation	0.0269	0.0119	0.0171	0.0137	0.0355	0.0091	0.0140
Test	0.0472	0.0295	0.0326	0.0304	0.0645	0.0252	0.0351

**Table 3** Results of kernel combinations using OOS validation. MKL (norm.) refers to MKL using normalized kernel matrices.

	$k_{\text{VAR}}$	$k_{\text{GA}}$	$k_{\text{MDED}}$	$k_{\text{ARC}}^0$	$k_{\text{ARC}}^1$	MKL	MKL (norm.)
Train	0.0122	0.0014	0.0015	0.0030	0.0009	0.0017	0.0109
Validation	0.0305	0.0141	0.0184	0.0172	0.0326	0.0106	0.0184
Test	0.0434	0.0265	0.0297	0.0275	0.0618	0.0251	0.0313

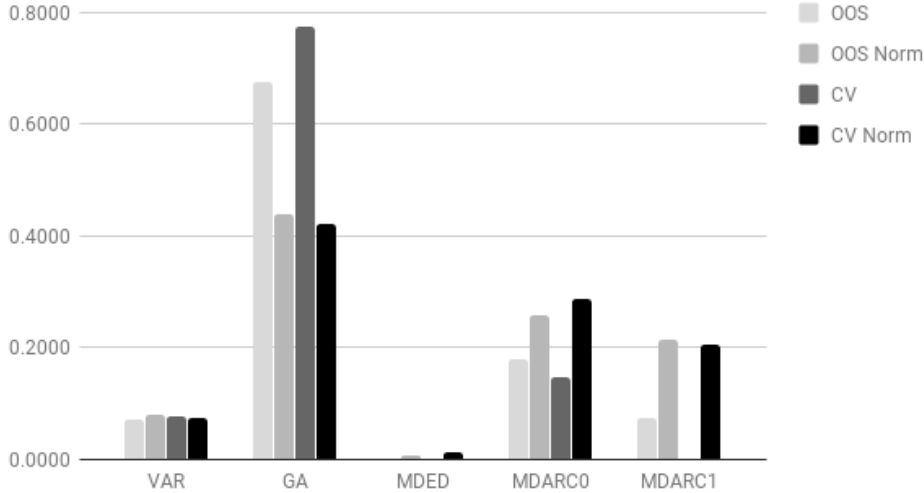
**Table 4** Results of kernel combinations using CV. MKL (norm.) refers to MKL using normalized kernel matrices.

The results fall in a determined range that states the capabilities of these methods in financial prediction tasks. This time, however, larger differences can be observed between the different methods, some methods being nearly thrice more accurate than others. Simpler kernels like  $k_{\text{MDED}}$  and  $k_{\text{ARC}}^0$  perform relatively well;  $k_{\text{ARC}}^1$  has worse results, probably due to an over-fitting effect.

The best performing technique is MKL, as it was in the classification case. A clear case of over-fitting can also be observed in this case, considering the difference between the training and test errors. The normalization only increases the error of the methods, indicating a loss of information during this process.

The different validation techniques have a revealing impact on the regression task. As it can be observed, the results obtained using CV as a parameter validation technique are always better than those using OOS. These results support [4], contradicting the conclusions reached in the classification task.

**Weights of kernel methods with different MKL procedures for regression**



**Fig. 3** Weights of kernel methods with different MKL procedures for regression.

Figure 3 contains the weights of the MKL process. It can be observed that  $k_{GA}$  obtains the higher percentage of weight in all of the variations of the experiment, with even higher values in the best performing settings of MKL. This fact is reinforced by the better results obtained by this kernel in the individual tests. The normalization procedure impacts the weights by making them closer to their mean and at the same time assigning a much higher weight to  $k_{ARC}^1$ . These results for  $k_{ARC}^1$  are opposite to the ones in the classification task, which indicates how much these tasks are different.  $k_{ARC}^0$  has a significant impact on the resulting vector of weights, which further mirrors the individual results.

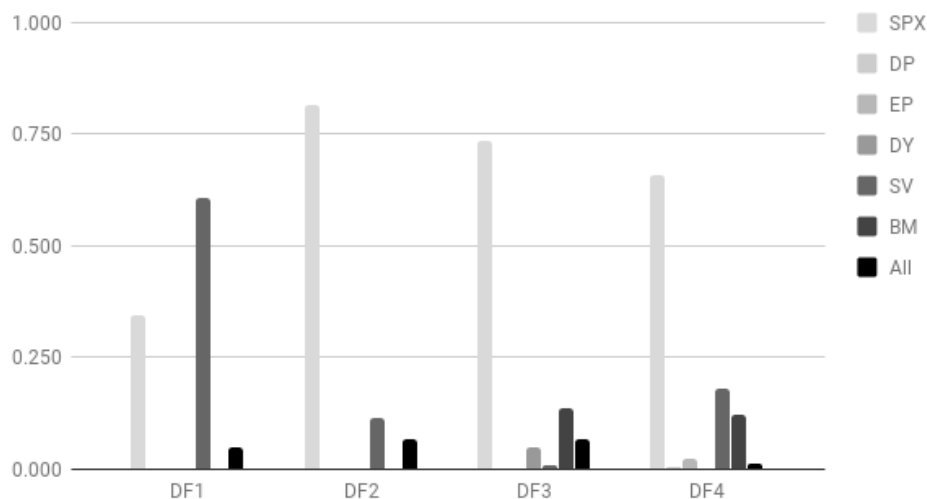
Table 5 and Fig. 4 display the experiments performed with exogenous variables. Again, these experiments are run using the best performing method for the task: non-normalized MKL using CV for validation.

The best performing method is DF1 in terms of test MSE, albeit there is not much difference between the four experiments. It is noteworthy that all the results are worse than the versions without the exogenous variables. On the other hand, the weights are somewhat different. The SPX index is clearly the feature that gets more weight in most of the variations, ranging between 60% and almost 80%. DF1 is an interesting result, as it gives more weight to the Stock Variance than to the

	Train MSE	Validation MSE	Test MSE
DF1	0.0025	0.0129	0.0321
DF2	0.0024	0.0131	0.0332
DF3	0.0024	0.0131	0.0332
DF4	0.0024	0.0135	0.0344

**Table 5** MSE results for the data configurations with exogenous variables (see text).

### Weights of variables with different experimental settings using MKL



**Fig. 4** Weights of variables with different experimental settings using exogenous variables.

SPX. This experiment also has the best results in the test set, however those results are bad in comparison to the errors of the endogenous variable. It would appear that the regression does not give any relevant weight to the exogenous variables in most cases, but relies mostly on the SPX itself. It is conceptually possible that the stock variance has an impact on the predictions, but the results contradict this fact.

The instability of these results was clearly reported in [28]. Not only the results are worse with the introduction of exogenous variables, but the weights also fluctuate with each different configuration of lags and variables. This can also be attributed to the capabilities of MKL as a learning model. The problem of overfitting was present in both tasks, reporting low training error values contrasted by high errors in the validation and test sets.

In summary, in this experimental setting, the exogenous variables have a questionable importance in the predictive models. The results also support this fact given the high weights assigned to the endogenous variable. The overarching conclusion, including the results obtained from the classification task, is that exogenous variables do not seem to increase the predictive capabilities of the model. Actually, in some cases, they worsen it.

## 5 An exhaustive experiment

We present in this last section results of an exhaustive comparative analysis of the predictability of equity premium by the MKL method against each of the kernels considered, using the financial exogenous variables as features. The purpose of these experiments is to ascertain if MKL is *hindering* the predictive capabilities of the exogenous variables when learning the best combination of kernels, regardless of the used variables. In other words, is MKL an overkill, may a single kernel suffice to best exploit the information content of the variables?

The experimental procedure is analogous to the one used in the previous regression and classification tasks. The only changes to the procedure are:

- Only regression is considered: the results from classification may be useful for investment tasks but the classes are created using a threshold, which makes the resulting binary division unstable, hard to predict, and hard to evaluate.
- Normalization is not used: as seen in the previous results, normalizing the kernels in regression tasks does not yield to an improvement.
- The data is not structured in experiments: instead all the data frames created for those experiments is used individually to test each algorithm.
- The capabilities of the kernels for time series will be also tested against a non-time dependant model in the form of a standard RBF kernel, named  $k_{\text{RBF}}$ .

In order to arrange for all the involved experiments and to obtain more solid conclusions, a factorial experiment design has been created, as described next.

### 5.1 Factorial Design settings

The Factorial Design [5] is a methodology to define tests and to compare results in problems that have several decisions to be made and several possible combinations. This procedure determines if a decision is statistically better than other and if there are relevant interactions between decisions. In our case, the decisions (factors) of the factorial experiment were as follows:

- Use MKL or any of the *single* kernel functions
- Use a combination of exogenous variables or only the endogenous variable
- Use Cross-Validation (CV) or not

Even though the decisions are clear, the number of factors on each decision are not that straightforward. In order to decide this, an exhaustive experimentation procedure has been performed with all the combinations of single kernels and exogenous variables. In the lines of the decisions presented in the experimental definition, the following a priori conclusions were obtained:

- The best result of the MKL method is surpassed by the best results of  $k_{\text{RBF}}$  and  $k_{\text{ARC}}^0$  kernels by a very slim margin (0.02505 vs 0.02448 and 0.02497)
- All kernel methods except MKL perform better with the inclusion of exogenous variables, but with different degree of improvement.
- Cross-Validation seems to produce small but general improvements applied with any kernel function.



Factor Levels	-	+
1 Kernel Method(KM)	RBF	MKL
2 Variables(Var)	Endogenous	Exogenous
3 Validation Method(VM)	OOS	CV

**Table 6** The different factor levels for each decision of the Factorial Experiment.

Formulation	1	2	3	Result	Variance
1	-	-	-	0.0293	0.0121
2	+	-	-	0.0252	0.0104
3	-	+	-	0.0249	0.0090
4	+	+	-	0.0292	0.0120
5	-	-	+	0.0284	0.0123
6	+	-	+	0.0251	0.0116
7	-	+	+	0.0250	0.0098
8	+	+	+	0.0273	0.0115

**Table 7** The results of each experiment of the Factorial Experiment.

In terms of the decision of using MKL or not, the results of  $k_{\text{RBF}}$  are surprising, as it surpasses the much more complex MKL algorithm.  $k_{\text{ARC}}^0$  also gets an improvement by using the exogenous variables and it becomes the single best performing, time-dependent kernel function. The inclusion of exogenous variables has an impact on the result: most notably  $k_{\text{RBF}}$  improves the most compared with other methods shifting its results from 0.0293 using only endogenous variables to 0.0245. Other improvements near the 15% performance increase can be appreciated in the VAR kernel.  $k_{\text{ARC}}^0$  also improves in about 10% its test MSE. The rest of kernels only increase their results by 5% or less. In the methods with increased performance by the introduction of exogenous variables, the endogenous variable has a reduced impact contrary to the methods that saw little or no impact in the use of exogenous variables. Looking at the results it is possible that models like MKL and  $k_{\text{ARC}}^1$  over-fit towards the SPX index as it is the most similar to the output variable, but the models like  $k_{\text{RBF}}$  and  $k_{\text{ARC}}^0$  unravel correlations between the output and the exogenous variables that the other methods apparently do not do.

## 5.2 Results and Discussion

Each of the decisions were reduced to two levels (options). The first decision was reduced to use MKL or the best performing single kernel method, which is  $k_{\text{RBF}}$ . The second decision boiled down to consider the best performing exogenous variable combination, which is SPX with zero lags and Dividend Yield with lags from zero to four, for each method. The third decision remained unaltered, testing both validation techniques.

The different levels of each decision are represented in Table 6, and the different formulations (experiments) in Table 7, with the test MSE of each combination along with the variance of the results.

Table 8 shows the calculated effects. The effects of these decisions are very small, even including the interactions between them. This indicates that the decisions taken do not change the capabilities of the model by any significant factor,

Name	Effect $\pm$ standard error
Kernel Methods(KM)	$-0.000117 \pm 0.052615$
Variables(Var)	$-0.000500 \pm 0.052615$
Validation Method(VM)	$0.000605 \pm 0.052615$
KM x Var	$0.003613 \pm 0.052615$
KM x VM	$-0.000412 \pm 0.052615$
Var x VM	$-0.0000797 \pm 0.052615$
KM x Var x VM	$-0.000790 \pm 0.052615$

**Table 8** Calculated effects of the Factorial Experiment.

which is an expected result looking at the very small differences between the compared methods. Without looking at the standard errors it can be seen that: MKL yields worst results than RBF on average, the same can be said for the use of exogenous variables over endogenous variables, and the cross-validation technique is better on average than out-of-sample validation. These results, however, are strongly influenced by the noise in the model reflected in the high values for the variance. Nonetheless, the interaction between the kernel methods and the selected economic variables is notable and this effect can appear in other versions of the problem.

The results of the factorial experiment indicate that most of the possible improvements that can be observed in the MSE of these decisions are ultimately not reliable due to the high amount of noise in the samples. This unreliability in the models was reported previously in [28]. Some interesting issues, however, do stand out: MKL integrating five kernels for time-series is matched by a simple RBF kernel and, to a lesser extent, by  $k_{ARC}^0$ . Furthermore, those simple models achieve the predictive capabilities of MKL using exogenous variables. The relation between the kernel methods and the exogenous variables can be also observed in the relatively high interaction between those decisions. CV points itself as a possibly better validation method than OOS.

## 6 Conclusions

In this article, a non-linear approach has been applied to the Welch and Goyal experiments [28] to measure the influence of several economic indicators in the prediction of the equity premium. The experiment is designed around several kernel functions for time series that aimed to extract relevant information from these variables and create a predictive model. The experimental procedure was based on selecting the best kernel or combination thereof for this problem, applying the selected model to each of the variables creating several kernel matrices and then obtaining the multiple kernel learning weights of each matrix. These weights indicate the relative importance of each variable.

The results indicate that, in this experimental procedure, the exogenous variables have a relative importance comparable with the S&P 500 index. However, the predictive capabilities of the model are not improved upon the introduction of these variables and the weights are not consistent across the different experiments. This instability was previously reported in Welch and Goyal's work for linear models, and it is now confirmed in our work for a family of non linear models based on single or linear compositions of machine learning kernels.

As future works, the influence of each individual variable and the number of lags included could be further explored to find deeper relationships. More powerful kernels could be introduced and studied to enrich the descriptive capabilities of the multiple kernel learning models. In this sense, a finer tuning of the parameters of each kernel should be performed to extract the most of the approach.

## References

1. Aioli, F., Donini, M. (2015). EasyMKL: a scalable multiple kernel learning algorithm. *Neuro computing* 169, 215–224.
2. Ang, A., and Bekaert, G. (2007). Stock return predictability: Is it there?. *Review of Financial studies*, 20(3), 651–707.
3. Bach, F. R., Lanckriet, G. R., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the Twenty-First International Conference on Machine learning*, p. 6, ACM.
4. Bergmeir, C., Hyndman, R., Koo, B. (2015). A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction. Department of Econometrics and Business Statistics, Working Paper, ISSN 1440-771X.
5. Box, George E. P., Hunter, J. Stuart, and Hunter, William G.: *Statistics for Experimenters: Design, Innovation, and Discovery*, 2nd Edition ISBN: 978-0-471-71813-0, 664 pages. (2005)
6. Campbell, John Y., and Shiller, R. J. (1988). The dividend-price ratio and expectations of future dividends and discount factors, *Review of Financial Studies* 1, 195–228.
7. Campbell, J. Y., and Thompson, S. B. (2008). Predicting excess stock returns out of sample: Can anything beat the historical average?. *Review of Financial Studies*, 21(4), 1509-1531.
8. Chang, C., Lin, C. (2001) Training  $\nu$ -Support Vector Classifiers: Theory and Algorithms. *Neural Comput.* 13 (9), 2119–2147.
9. Cho, Y., Saul, L. (2009). Kernel Methods for Deep Learning. *Advances in Neural Information Processing Systems* 22, 342–350.
10. Cochrane, John H. (1992). Explaining the variance of price-dividend ratios, *Review of Financial Studies* 5, 243–280
11. Cochrane, John H. (2006). The dog that did not bark: A defense of return predictability, *Review of Financial Studies* 21, 1533–1575.
12. Cochrane, John H. (2011). Presidential Address: Discount Rates, *The Journal of Finance* 56 (4), 1047–1108.
13. Cuturi, M., Vert, J.-P., Birkenes, Ø., Matsui, T. (2007). A kernel for time series based on global alignments. In *IEEE Int. Conf. ICASSP 2007*, pages II–413. IEEE.
14. Cuturi, M., Doucet, A. (2011). Autoregressive kernels for time series. Technical Report arXiv:1101.0673.
15. Cuturi, M. (2011). Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 929–936.
16. Fábregues, L., Arratia, A., and Belanche, L. A. (2017). Forecasting Financial Time Series with Multiple Kernel Learning. *Advances in Computational Intelligence: 14th International Work-Conference on Artificial Neural Networks, IWANN 2017, Cádiz*.
17. Fama, E. F., French, K. R. (1988). Dividend yields and expected stock returns, *Journal of Financial Economics* 22, 3–25.
18. Fletcher, Hussain, T. Z., Shawe-Taylor, J. (2010). Currency Forecasting using Multiple Kernel Learning with Financially Motivated Features. In *NIPS 2010 Workshop: New Directions in Multiple Kernel Learning*.
19. Geler Z., Kurbalija V., Radovanovi M., Ivanovi M. (2014) Impact of the Sakoe-Chiba Band on the DTW Time Series Distance Measure for kNN Classification. In: Buchmann R., Kifor C.V., Yu J. (eds) *Knowledge Science, Engineering and Management. KSEM 2014 (LNCS vol. 8793)*, Springer.
20. Hansen, Lars Peter, and Hodrick, Robert J. (1980). Forward exchange rates as optimal predictors of future spot rates: An econometric analysis, *Journal of Political Economy* 88, 829–853.
21. Kale, D. C., Gong, D., Che, Z., Liu, Y., Medioni, G., Wetzal, R., and Ross, P. (2014). An examination of multivariate time series hashing with applications to health care.

22. Kothari, S. P., and Shanken, J. (1997). Book-to-market, dividend yield, and expected market returns: A time-series analysis. *Journal of Financial Economics*, 44(2), 169-203.
23. Lettau, M., and Ludvigson, S. (2001). Consumption, aggregate wealth, and expected stock returns. *the Journal of Finance*, 56(3), 815-849.
24. Peña, M., Arratia, A., and Belanche, L. A. (2016). Multivariate Dynamic Kernels for Financial Time Series Forecasting. In 25th International Conference on Artificial Neural Networks, Springer LNCS 9887, 336-344.
25. Sakoe, H.; Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*. 26 (1): 43-49
26. Schölkopf, B., Smola, A. J., Williamson, R. C. and Bartlett, P. L. (2000). New Support Vector Algorithms. *Neural Computation* 12:5, 1207-1245.
27. Shiller, Robert J. (1981). Do stock prices move too much to be justified by subsequent changes in dividends? *American Economic Review* 71, 421-436.
28. Welch, I., and Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. *Review of Financial Studies*, 21(4), 1455-1508.