UNIVERSITÉ DE NANTES

IAE NANTES
ÉCONOMIE & MANAGEMENT

Variable Selection for Equity Markets Forecasts

Aurouet Lucas, Meynier Thibaud

Presented for the Master's degree of
Applied Econometrics, under the direction of Pr. O.DARNE

Master EKAP
IAE Nantes
01/12/2020

Contents

1 Introduction

With big data, the amount of information available in econometrics has grown exponentially. From this point on, the objective is to efficiently filter information. The inclusion of variables irrelevant to the problem may cause multiple issues. Our goal here is to solve some of the concerns raised by the availability of irrelevant information. We are trying to forecast the S&P 500 returns using standard OLS, the data available to fit contains 24 variables. All of them are potentially helpful in determining S&P 500 returns. All of them potentially also are irrelevant. We then need to reduce the number of available variables, to the truly necessary ones, using statistical methods and machine learning algorithms. Linear regression is not efficient if variables are correlated, it tends to become harder to interpret as the number of parameters increases. Also, overfitting increases the variance of future forecasts, linear regression is less exposed to overfitting than other models as the model fit a linear function to the data. Overfitting further penalizes non-linear models, but to a certain extent we also want to avoid including to much variables, to balance the bias/variance trade-off. By reducing the number of parameters, we induce bias, but we also reduce variance. Low bias makes the parameters closer to their true value as long as the data stays exactly the same, low variance makes the parameters less precise, but more adjustable to changing data. When forecasting, the goal is to produce reliable estimates of future values, based on current information with confidence that future data will behave as it did when the model was estimated. If stock markets experience a significant change, a model with numerous parameters and therefore low bias, will be less accurate than a model with low variance. Which is why we choose high bias over high variance is this problem. To decide on a reasonable trade-off between the two, different methods exists. We present an exhaustive list of models and machine learning algorithms designed to perform variable selection, to help us filter the information, avoid overfitting and unnecessary complexity, to produce quality forecasts on the S&P 500 returns. The following sections are ordered as such: the first section describes the dataset and data processing, the second section is dedicated to variable selection with penalized regression, GETS modelling and non-parametric algorithms (random trees and random forests). In the last section we forecast the S&P 500 returns with OLS regression and the selected variables from section 2.

2 Data analysis

2.1 Stationarity

Spurious regression is a very well-known phenomenon in time series analysis. Granger & Newbold showed that using non-stationary series for regression can bias the estimates. Non-stationary data can also induce invalid hypothesis testing. In other terms, to estimate a valid model using time series, we need to ensure that our data is stationary. To study the stationarity of our series, we programmed an R function that iterates through the entire data frame. For each column (each variable), four stationarity tests are applied: Augmented Dickey-Fuller (ADF), Kwiatkowski-Phillips-Schmidt-Shin (KPSS), AR(1) coefficient and Ljung-Box for significant auto-correlation at lag k[1]. If two of those four tests are showing significance statistical evidence that the series is not stationary[2], the function will automatically differentiate the variable in question once. The table 1 bellow represents our 28 variables, as well as the result of the previous function.

[1] k = 1 in this study

[2] Either $p_{value} ADF > 0.05$, $p_{value} KPSS < 0.05$, $AR(1) coefficient > 0.7$ or $p_{value} Ljung - Box < 0.05$

Table 1: Testing for stationarity

Variable	I(0) variable stationarity	I(1) variable stationarity
Returns	Yes	-
D12	No	No
D.P	No	Yes
DY	No	Yes
EP	No	Yes
DE	No	Yes
E12	No	Yes
b.m	No	Yes
tbl	No	Yes
AAA	No	Yes
BAA	No	Yes
lty	No	Yes
ntis	No	Yes
Rfree	No	Yes
infl	No	Yes
ltr	No	Yes
corpr	No	Yes
svar	No	Yes
$CRSP_{SPvw}$	Yes	-
$CRSP_{SPvwx}$	Yes	-
IP	No	Yes
IPG	No	Yes
gap	No	Yes
tms	No	Yes
dfr	No	Yes
dfy	No	Yes
epu	No	Yes

Almost every variable in our set of predictors is non-stationary at first. Apart from $CRSP_{SPvw}$ and $CRSP_{SPvwx}$, all of the 26 remaining variables are showing either a significant auto correlation at lag = 1 (AR(1) and Ljung-Box tests) and/or a unit root process (ADF and KPSS tests). We differentiate these series once and we run the same tests on the newly differentiated variables. The results are shown in the last column of the previous table. Every variable expect for **D12** is now stationary, and can be used in a model without carrying the risk of spurious regression. One question still arises: should we differentiate **D12** once again, to stationarize the series ? Lee, Shi and Gao argued that standard LASSO regressions where biased if the predictors displayed different integration order[3], which would be the case considering we differentiate the variable a second time. The LASSO estimators lose the oracle property in such a framework and might not be efficient. However, they also showed that Twin Adaptive LASSO (TaLASSO) should remedy the bias introduced by the different integration orders in predictors. We do not plan on fitting a TaLASSO, but rather standard and adaptive LASSO[4]. For that reason, we might be forced to remove **D12** from the set of available predictors.

Another precision has to be made here. The variable **Index**, which represents the US stock prices had to be differentiated. The resulting values represent the returns between two time stamps for that particular Index.The logarithmic returns are often preferred in the literature. Therefore, we will now use the **returns** variable which expresses the log-returns of the US stock between two periods. hence, our models will focus on estimating and forecasting the **returns** instead of the prices, encompassed by the **Index** variable.

2.2 Outliers

Outliers can also cause different issues when running regression models. Most of the time the researcher is concerned about the influential point dragging the regression line towards it, pushing it away from the rest of the representative data. Hence, we obtain an equation that represents a fallacious correlation between our predictors and dependent variable. We therefore need to detect outliers and treat them before passing any variable into the model. We will use the methodology by Boudt and Al. which consist in a threshold technique for outlier correction based on the idea of winsorisation[5] of the series. We apply this technique on every series, and we compute the corrected analogous series, where outliers have been replaced. We will work with this new corrected data from now on. We present each and every series (raw and corrected), in the Appendix.

2.3 Descriptive Statistics

In this section we give a general overview of the data once all series have been made stationary and cleaned of outliers. We are mostly interested in Mean, Standard Deviation

[3] LASSO estimates are biased by nature, but unequal integration orders among predictors introduces additional bias

[4] adaptive LASSO will now be referred to as aLASSO

[5] Extreme values are replaced by a chosen quantile, the 5^{th}, or 95^{th} for example, where extreme negatives values are replaced by the value 5^{th} quantile and extreme positive values by the 95^{th}

(sd), Skewness and excess kurtosis. We represent each of the four moments in the following table.

Variable	Mean	Standard deviation	Skewness	Excess kurtosis
D12	0.0519	0.1214	0.5610	6.6202
D.P	-0.0013	0.0441	0.2379	0.9015
DY	-0.0013	0.0439	0.2752	1.0217
EP	-0.0006	0.0505	0.1992	2.5357
DE	-0.0004	0.0286	0.0873	19.4717
E12	0.1192	0.8218	-0.1884	10.5608
b.m	-0.0004	0.0282	0.2817	3.3273
tbl	0.0000	0.0030	-0.0969	7.9584
AAA	0.0000	0.0017	-0.0405	4.0298
BAA	0.0000	0.0017	-0.0260	2.5526
lty	0.0000	0.0025	0.0700	3.3919
ntis	0.0000	0.0035	-0.1700	2.9495
Rfree	0.0000	0.0002	-0.0969	7.9585
infl	0.0000	0.0044	-0.0700	3.0511
ltr	0.0047	0.0241	0.1947	1.3338
corpr	0.0000	0.0273	-0.0801	1.8197
svar	0.0020	0.0027	3.7682	16.7989
$CRSP_{SPvw}$	0.0095	0.0426	-0.2640	0.7883
$CRSP_{SPvwx}$	0.0065	0.0427	-0.2657	0.7818
IP	0.1109	0.4070	-0.3209	1.6777
IPG	0.0029	0.0125	-0.0644	5.9339
gap	-0.0016	0.0521	-0.4368	3.0567
tms	0.0000	0.0031	0.3949	3.0301
dfr	0.0001	0.0125	0.0135	3.4124
dfy	0.0000	0.0009	0.2666	6.3930
epu	-0.1161	22.2539	0.2372	1.8300
returns	0.0056	0.0425	-0.3801	0.8418

Table 2: Summary statistics

Must variables are zero-mean, which tells us variables are centered around 0. This is often the case with financial series, especially when we differentiate the variables. Skewness is below 0 again in most cases, indicating series are approximately symmetric. **svar** and **D12, D.P, DY, EP** have a positive significant skewness, meaning values are concentrated on the left of the mean. We notice very high kurtosis for some of the series. **DE**, **svar** and **E12** display heavy tails and more values concentrated around the mean than in the normal distribution. **E12** also is the series with highest variance along with **IP**. We looking at descriptive statistics, **returns** appears normally distributed with mean = 0.006 and small value for skewness and kurtosis. To better illustrate the phenomenon. We plot the comparison between the **returns** and a normal density function with corresponding mean and variance.

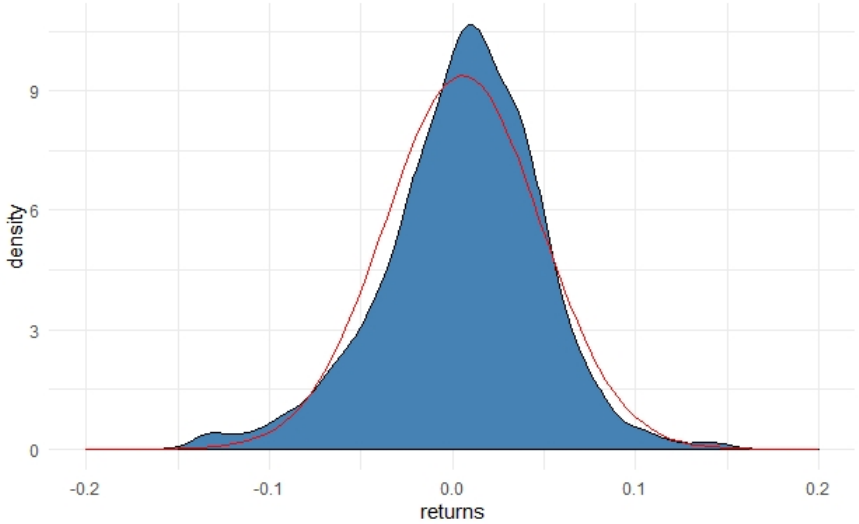
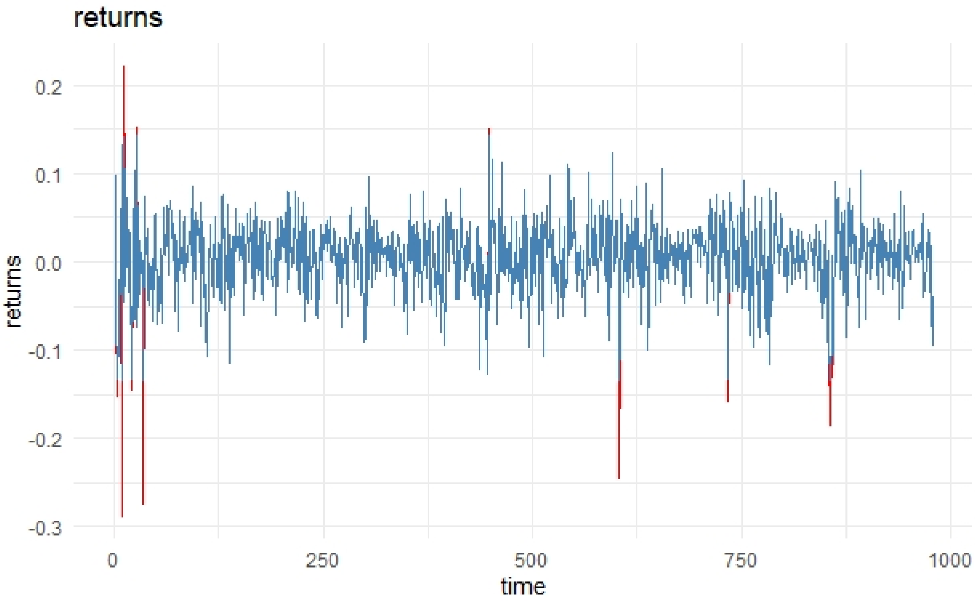


Figure 1: Fig. 1 - Returns kernel density against Normal density function

The figure 1 above shows that the **returns** distribution (blue) does not exactly match the Normal (red) curve. We can visually assess the positive excess kurtosis which results in heavier tails and more values concentrated around the mean, compared to a normal distribution. Although this result will not be compromising our estimations, it is an interesting detail. Financial series such as this one have been empirically associated with heavy tails and high kurtosis, the previous plot show this assumption verifies once again with US stock returns. Althought with this series we are a lot closer from a normal distribution than it is usually the case with other financial time series. The figure 2 is showing the returns cleaned from outliers in a temporal evolution

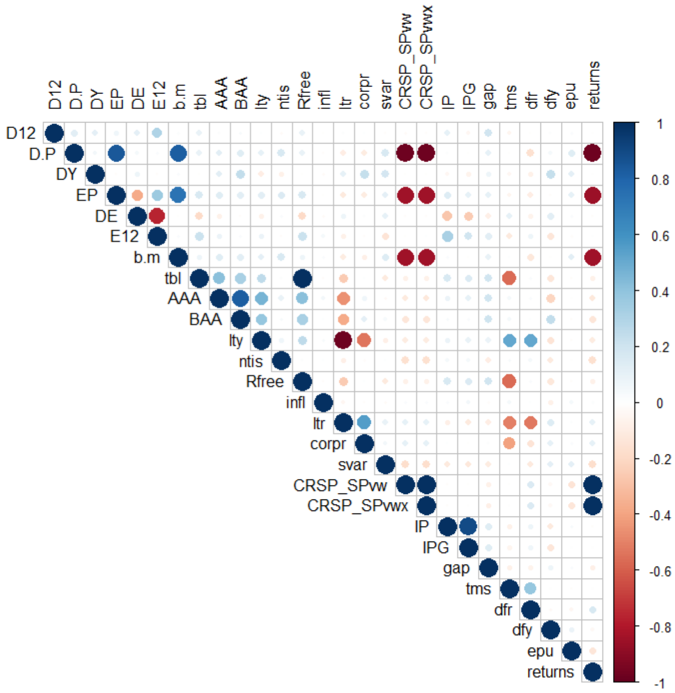
Figure 2: S&P 500 returns clean from outliers



The beginning of the series at index = 0 is 06/01/1937. In previous steps, some values have been overwritten by Boudt & Al. outliers detection method, we will now try to understand what caused the appearance of such extreme values in our series. USA suffered from a recession in 1937-1938, although economists disagree on what caused the markets to crash, some suggest it was due to to cut on public spending, other argue the recession is a consequence of the newly adopted regulation on federal reserves, raising interest rates. We can also spot the "black monday" of october 1987, where the S&P 500 suddenly lost 20.5% in one day. Outliers at the the end of the series (index = 850) may be imputed to the Asian crisis in 1997, but this is a rather difficult hypothesis to prove.

2.4 Correlations & Dependencies

Figure 3: Correlations between Returns and potential regressors



According to figure 3, we can notice that 3 potential regressors are extremely correlated with **returns**. These regressors are in fact equivalent to **returns**. They do not exactly represent logarithmic returns, but they are market indicators built on that very variable. This problem could drive us to model our response variable, Y by a transformed Y variable and give us false results at the end. Hence we drop $CRSP_{SPvw}$, $CRSP_{SPvwx}$ and **D.P** from the dataset.

Predictors are sometimes corraleted with each other. As discussed before, multicollinearity can lead to biased results, correlation between explanatory variable as to be addressed. Hopefully, penalized regressions and other variable selection methods are supposedly robust to multicollinearity. Those models are used, partly, because when facing correlated predictors, they are capable of eliminating variables correlated with other. The final model will retain the most significant one and drop other with high degree of multicollinearity. Variables kept in the final model should then not be correlated with each other (or at least no too much: $\rho < |0.5|$). Non-parametric models and GETS should also be able to eliminate this concern. To analyze further the relation of our datas, we made a PCA in 3 dimmension (see Appendix). The PCA shows that returns is linked with DE, E12, svar, b.m, ntis, dfy, BAA, etc, relative to the projection of variables in a 2D graph (1,2 or 1,3). We expect that some of these variables will be retained by some variables selection methods use after.

3 Variable Selection

3.1 GETS Modelling

The first variable selection technique we are going to employ is GETS modelling (or GEneral To Specific). The goal with this technique is to avoid the recurrent path dependency. Issue known to appear with other variable selection methods such as Stepwise regression. Here, we apply the GETS methodology to our entire set of potential regressors. We have 26 variables to consider at this point, some of them might be irrelevant to forecast US stock returns, some of them are multicolinear with each other, and overall, the less variables we keep in the final model the easier it is to interpret and make economic sense of the results. With GETS we apply an iterative selection process based on statistical tests. At each point of the process the model is tested and will be kept if and only if the statistical tests suggest to do so. However we ran into a computational problem when trying to apply GETS. The matrix of predictors appears to be singular and non-inversible. For the code to run, this matrix needs this desirable property. One solution that was suggested to us was to pre-screen the variables using the SIS method, already reducing the set of predictors and then apply the GETS algorithm. SIS stands for Sure Independent Screening and is a technique for variable pre-selection, often used is ultra-high dimension frameworks. It is usually presented in a two step process; where the first step is designed to output a reduced set of variables based on their marginal correlation with the response variable. The higher the correlation between one predictor and the response, the higher the chances for that predictor to be selected. The second step consists in applying a penalized regression to the already pre-selected set of predictors, further reducing dimensionality. In our case, the second step is embodied by the GETS method. The overall process then consists in pre-screening the variables to find the ones most correlated with our response and then apply the GETS algorithm to these remaining predictors. The following table show the results.

Method	Variables	SIS-selected predictors	Final set of predictors
SIS + GETS	D12	-	-
	DY	-	-
	DE	DE	-
	E12	E12	E12
	b.m	b.m	b.m
	tbl	tbl	-
	AAA	AAA	-
	BAA	BAA	-
	lty	lty	lty
	ltr	ltr	-
	ntis	ntis	-
	Rfree	Rfree	-
	infl	infl	infl
	corpr	corpr	-
	svar	svar	svar
	IP	-	-
	IPG	-	-
	gap	-	-
	tms	tms	-
	dfr	dfr	dfr
	dfy	dfy	-
	epu	epu	-

Table 3: GETS selection

SIS and GETS alltogether retained 6 variables among the available ones. SIS decided to remove **D12**, **DY**, **IP**, **IPG**, **IP** and **gap**. We then applied GETS modelling and 11 more variables were eliminated. We will later add these variables in a linear regression to try and forecast S&P 500 returns.

3.2 Penalized Regressions

In this section we pursue our search of useful variables among a large set of potential candidates. This search will be achieved by the use of penalized regressions. The difference with the GETS modelling aforementioned resides in the fact that, in penalized regressions, the search is implemented in the regression itself, and does not rely on statistical variable removal. In penalized regression we modify the loss function to optimize in such a way that the coefficients will be voluntarily biased towards 0. Many penalized regression with different loss functions are known to this day, the most famous ones are LASSO and Ridge. And all of them rely on the introduction of a constraint on the coefficients' values. Here we will apply 11 different methods and detail the constraint applied to the coefficients in each one.

- **LASSO regression**: probably the most common form of penalized regression. The constraint is written as: $\sum_{i=1}^{p}||\beta_i||^1 \leq \tau$[6]. In that configuration, β's can be constrained to 0. This allows for variable removal (once the associated coefficient attains 0, the variable in removed from the model). If we change the norm in the constraint to 2, the constraint becomes quadratic, and β's can no longer equal 0. This brings us to Ridge regression.

- **Ridge regression**: as said before, ridge regression is very similar to LASSO, the difference lies in the norm of the constraint: $\sum_{i=1}^{p}||\beta_i||^2 \leq \tau$. Here we set the sum of squared values of the coefficients β smaller than a predetermined τ constraint, meanwhile in LASSO, we set the sum of absolute values of the coefficients β smaller than τ. In the ridge regression, coefficients might be reduced in magnitude by the constraint effect, but they cannot be set equal to 0. Hence, the ridge regression does not automatically remove variables from the models but rather makes the coefficients associated to variable with less explanatory power very small.

- **Elastic-Net regression**: Elastic-Net regression is a weighted sum of a LASSO and ridge regression. the constraint is expressed as: $\sum_{i=1}^{p}\alpha||\beta_i||^2 + (1-\alpha)||\beta_i||^1 \leq \tau$. This kind of regression takes one additional parameter α[7] that measure the partition between LASSO and ridge constraints, if α is equal to 1 (resp.0) Elastic-Net is the same as ridge (resp. LASSO). It allows for a lesser bias for large coefficients (ridge constraint) and variable selection (LASSO constraint) at the same time.

- **Bridge regression**: Bridge regression allows the norm q, to vary between 0 and 2. Whereas in LASSO and ridge, this parameter was strictly equal to either 1 (LASSO) or 2 (ridge). In the same purpose as Elastic-Net, when q varies, $\sum_{i=1}^{p}||\beta_i||^q \leq \tau$ switches from a LASSO to a ridge, and is able to attain an in-between, profiting from both the LASSO and ridge properties of variable selection and low bias.

- **SCAD regression**: SCAD aims at reducing the bias/variance trade-off. Bias is a

[6] with p the number of parameters, β the coefficient associated to the p^{th} regressor, τ measure how strong the constraint is, the smaller τ the more the coefficient will be dragged to 0

[7] $0 < \alpha < 1$

direct implication of penalized regression as coefficients estimated will be different from those estimated by OLS. This difference (the bias) depends on the constraint used. The more restrictive the constraint, the bigger the difference between the true (OLS[8]) estimator and the penalized estimator, the larger the bias. As we've already seen, LASSO has a more restraining constraint than ridge thus, can perform variable selection but subsequently introduces more bias than ridge. Elastic-Net and Bridge try to find the best combination of LASSO and ridge to obtain the best bias/variance trade-off. SCAD uses the LASSO constraint but applies cutoff values that, when exceeded, change the penalty applied to the coefficient:

$$\sum_{i=1}^{p} p_{\lambda}^{SCAD}(||\beta_i||^1),\ with,\ p_{\lambda}^{SCAD} = \begin{cases} \lambda|\beta|,\ si\ |\beta| \leq \lambda \\ \frac{2a\lambda|\beta|-\beta^2-\lambda^2}{2(a+1)},\ si\ \lambda \leq |\beta| \leq a\lambda^{9} \\ \frac{\lambda^2(a^2+1)}{2},\ si\ |\beta| \geq a\lambda \end{cases} \tag{1}$$

The motive beind SCAD, and its advantage compared to Bridge or Elastic-Net, is the use of threshold values allowing for non-linear penalties. Large coefficients are not penalized, small coefficient are linearly penalized and in-between coefficients are penalized with a quadratic penalty. Whereas in aforementioned techniques all coefficients are penalized the same way, that is a coefficient without explanatory power will be brought to 0, but an interesting coefficient will also diminish because the same constraint is applied. SCAD avoids unjustifiably penalizing coefficients that should remain untouched while still performing variable selection among the less useful (and/or correlated regressors.)

- **Adaptive LASSO**: Adaptive LASSO also aims to reduce the bias introduced by the penalty. It does so by adding weights to the coefficients, the larger coefficient have lesser weights, therefore will be less penalized than small coefficients. We first estimate a standard regression (OLS, LASSO, Elastic-Net, etc...), we then extract the coefficient associated to every regressor. We introduce the estimation into the constraint of a second regression model with an l2 normed constraint (LASSO) and apply a weight, inversely proportional to the estimated coefficient value. In the same fashion, SCAD and aLASSO allow for lesser constraints on large coefficients

[8] bearing the assumption that MCO estimators are BLUEs

and larger penalty for small coefficients. $\lambda \sum_{i=1}^{p} \hat{\omega}_i |\beta_i|$[10]

- **Weighted Fusion**: Weighted Fusion specifically addresses the issue of multicollinearity in high dimensional data. it applies weights to each coefficient relative to the correlation between each pair of explanatory variables. The stronger the correlation, the bigger the weights. It implies that the coefficient associated to two very colinear variable will bear greater weights and will therefore be more constrained than the coefficients of two non correlated variables.

- **Multi-Step adaptive Elastic-Net**: Multi-Step adaptive Elastic-Net and Multi-Step adaptive SCAD are iterative processes that rely on the more general adaptive framework. Each coefficient has a weight associated, and at each step of the process, the weights are updated the following way: $\omega_i^k = \frac{1}{|\hat{\beta}^{(k-1)}(\lambda^{(k-1)})|}$. Where k is the current step in the process $\hat{\beta}^{(k-1)}$ is the coefficient estimated at the previous step. The number of steps is determined by the user. The idea closely relate to aLASSO, expect it extends it to both Elastic-Net and SCAD but also allows for more than one step (weights are computed as many times as their are steps involved).

The previous methods require to determine optimal values for various parameters $(\lambda, \alpha, \gamma)$. Usually, the best value for each parameter is attained with cross-validation, where different values or grid of values are tested and the cross-validation error for each values are compared. the optimal parameters are those which produce the smallest error. Here, λ and other parameters have been optimised with 10-folds cross-validation. We can plot the variations in cross-validation MSE for each value of λ[11].

[10] with $\hat{\omega_i} = \frac{1}{|\hat{\beta_i}|^\gamma}$, where $|\hat{\beta_i}|$ is the coefficient obtained in the first stage regression and $\gamma > 0$

[11] Ridge is top left, bridge is top right LASSO ans a LASSO are bottom left and bottom right respectively

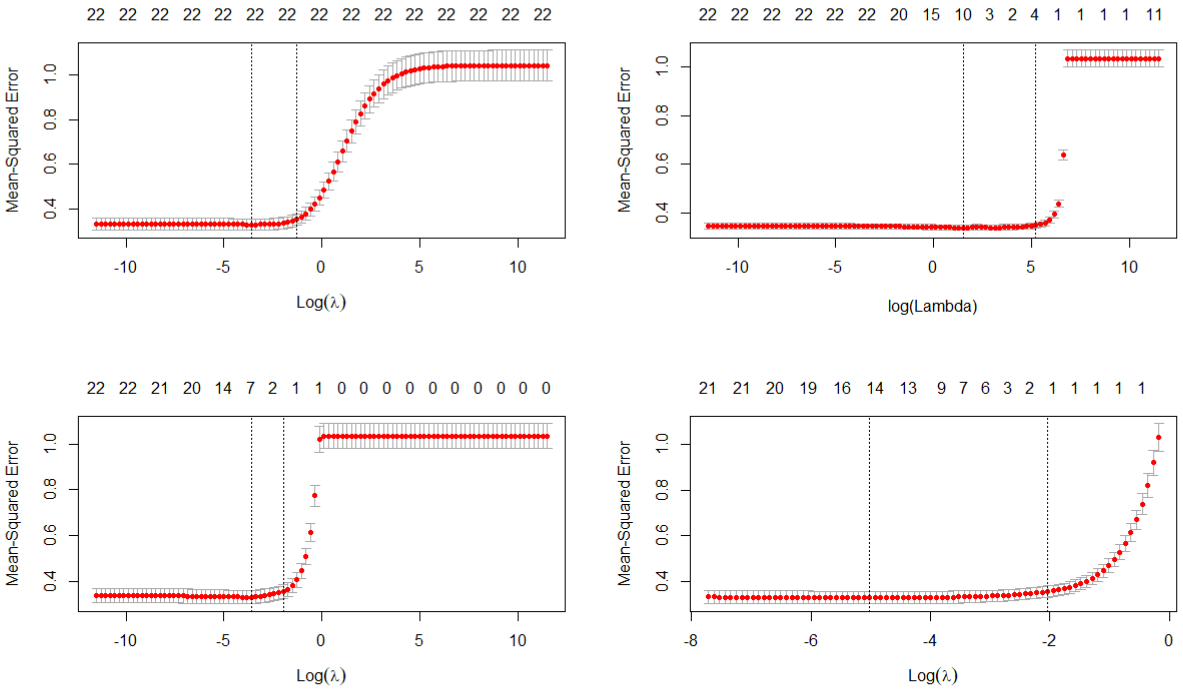


Figure 4: Lambda paths for Ridge, Bridge, LASSO and aLASSO regressions

We can visualize how the precision of our model increases or decreases for different values of λ. For a particular value, the cross-validation MSE is smallest, we then extract λ which gives the best accuracy and choose it as the penalization coefficient for the considered model. We repeat this procedure for every model. Each are specified in a different manner with different constraints, and thus will produce different MSEs for the same value of lambda. Some methods might apply heavier penalization than others, hence we should see higher λ for the methods with heaviest penalization. Models with high λ should eliminate more variables than models with small λ. The following table presents the optimal λ selected for each model.

Model	λ
LASSO	0.0067
Ridge	0.0215
Bridge	3.594
Elastic-Net	0.1385
SCAD	0.0262
aLASSO	0.0085
WF	0.0062
aEN	0.0102
aSCAD	0.0116
MSaEN	0.0101
MSaSCAD	0.0262

Table 4: Optimal values for λ

Exept for Bridge and Elastic-net, λ remains in the same magnitude for every model between 0.006 and 0.0262 Which means every model applies a similarly heavy penalty on the number of parameters. We should then see the same number of variables retained in all models, as they all start from the same set of predictors and apply similar penalties. The variables might differ, as other phenomenons dictate the variable selection process such as correlation between regressors, correlation between regressors and response variable, the form of the penalty etc. We can notice three levels of penalization. For LASSO, aLASSO and WF λ is the smallest, aEN, aSCAD, MSaEN have slightly higher λ, all in the [0.010 : 0.012] interval. Finally, Ridge, Bridge, Elastic-net, SCAD and MSaSCAD have the highest λ. Bridge set aside, we expect more parsimonious variable selection for higher λ. However, because the models are specified in a variety of ways, it is likely to observe models with higher λ retaining more variables than others.

The variables selected by all the different methods are presented below[12].

[12] aLASSO stands for Adaptive LASSO, WF for Weighted Fusion, MS for Multi-Step

Table 5: Summary of penalized regression

Variables	Lasso	Ridge	Brdige	Elastic Net	SCAD	aLASSO	WF	aEN	aSCAD	Ms-aEN	Ms-aSCAD
D12	D12	D12	D12								
DY	DY	DY	DY		DY			DY	DY	DY	
DE	DE	DE		DE				DE			
E12	E12	E12	E12	E12	E12	E12	E12	E12	E12	E12	E12
b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m
tbl		tbl									
AAA	AAA	AAA									
BAA		BAA									
lty	lty	lty	lty	lty	lty			lty	lty	lty	lty
ntis	ntis	ntis	ntis	ntis	ntis			ntis		ntis	
Rfree		Rfree									
infl	infl	infl	infl	infl	infl			infl	infl	infl	
ltr		ltr									
corpr		copr									
svar	svar	svar	svar	svar	svar	svar	svar	svar	svar	svar	svar
IP	IP	IP			IP			IP		IP	
IPG		IPG									
gap	gap	gap						gap			
tms	tms	tms									
dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr
dfy	dfy	dfy	dfy		dfy			dfy		dfy	
epu	epu	epu	epu	epu	epu			epu		epu	

3.3 Random Forest

This section will be dedicated to Random Forests (RF). RF, discovered by L.Breiman in 2001 is an ensemble method, meaning it takes advantage of multiple models fitted on the same data and the final output is computed via a sum of all models previously fitted. In RF, the model used is a Classification And Regression Tree[13], therefore the random forest is an ensemble of N trees. Let us first define the rationale behind CARTs. In our example, each tree is a regression tree (as opposed to classification trees), designed to predict the value of the **returns** variable based on the values of the 26 remaining variables. Althought classification and regression trees follow a similar logic, we will focus on the regression case as it is what concerns our data. A tree is composed of nodes, at each node, the dataset is divided in two sub-samples [14] according to one of the predictors and a threshold value for that predictor. The predictor and cutoff values are chosen so the variance within the two sub-samples is minimal and the variance between the two sub-samples is maximal. In other terms, at each step we find the predictor, and the value for that predictor that best divide the dataset. We repeat the process until either the number of iterations (nodes) attains a limit fixed by the user, or the nodes are small enough, i.e. they contain a sample that cannot be further divided[15]. The stopping rule has to be decided prior to fitting the model. Using a single regression tree often bears the risk of instability, CARTs are very sensible to the data and the hyperparameters values. To solve this issue, Breiman proposed to aggregate and average the results from multiple trees. After choosing a number of trees in the forest, each CART is fitted on a portion of the original data randomly (and without replacement) chosen among the entire dataset. At each node of each tree, a subset of k random predictors among the p predictors available is tested and the best one (along with its threshold value) is computed. The idea being testing only a subset of the available predictors is to make each tree as different as possible from the next one. By exploring as many paths as possible, the average, final model will be as reliable as possible. Each individual tree maximum growth is limited by a maximum nodesize of 5[16], i.e. when a node contains 5 or less observations, it cannot be divided any further, therefore stopping the growth of that particular branch.

[13] CART

[14] of similar or different sizes

[15] because every observation has the same value, or each node contains only one observation

[16] most R package choose nodesize = 5 by default

In our RF, we hence have to choose a number of trees, the size of the subset of regressor to evaluate at each node. The optimal values for these parameters are chosen with 10-folds cross validation, the values that give the smallest cross-validation RMSE are selected as the parameters values. The following figure shows the decrease in RMSE relative to the number of trees in the forest, when mtry, the size of the subset of regressor to evaluate at each node, is fixed[17].

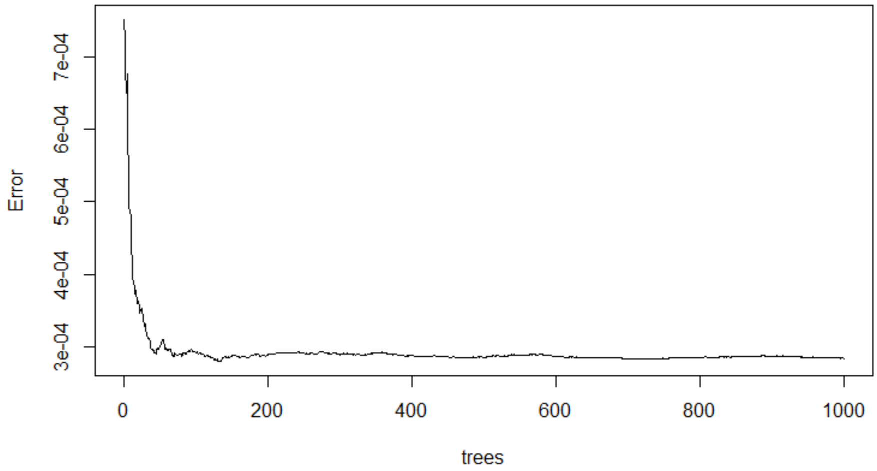


Figure 5: RMSE & number of trees

The RMSE generally decreases as the number of trees grows. When the number of trees exceeds 300, when can no longer observe noticeable decrease in RMSE, suggesting that ntree = 300 is large enough to obtain meaningful results. In addition, when ntree increases, the computation time required to fit the model increases. 200 is a relatively small number of trees, allowing us for less costly computations and shorter execution time in Rstudio. We then choose ntree = 200 for the following sections.

mtry also has to be found. Once again, we can plot the parameter against different cross-validation RMSE and pick mtry that outputs the smallest value.

[17] mtry is fixed to 16 here, as further results suggest

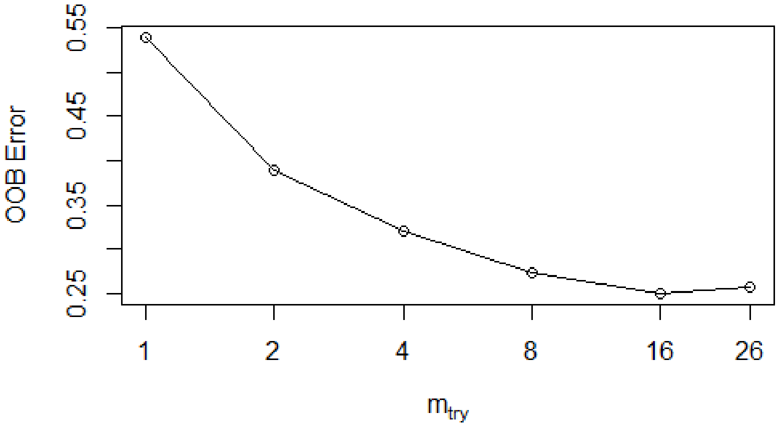


Figure 6: RMSE & mtry

The error seems to be the smallest when mtry = 16, suggesting that 16 out of 26 predictors should randomly be chosen to divide the dataset in optimal groups. The metric used in this case is the Out Of Bag (OOB) RMSE. OOB is closely related to out-of-sample (OOS), the random forest is fitted on a subset of the orginal dataset chosen randomly without replacement. The remaining, unused observations are treated as a testing set. OOB RMSE is the cross-validation RMSE on observations the model has not been trained on. 'Bagging' is used in RF and other machine learning models as a way to verifiy model's stability, but it also allows one to assess the importance of each variable. Machine learning models are often hard to interpret, and the impact of each variable on the response may be unclear. To remedy to this well known issue, bagging sheds lights on the role of each variable in the model. Bagging works as follows:

1. randomly assign a certain percentage of the original dataset to training.
2. the remaining observations constitute the OOB sample.
3. fit the trained model on the OOB sample, compute the OOB RMSE.
4. randomly shuffle the observations of one regressor.
5. fit the fitted model of the shuffled OOB sample.
6. compute the difference $d = RMSE^{OOB}_{ordered} - RMSE^{OOB}_{shuffled}$

7. if the regressor for which values have been shuffled is important, it should highly impact the prediction and $RMSE^{OOB}$, therefore $RMSE^{OOB}_{shuffled} > RMSE^{OOB}_{ordered}$ and d decreases.

8. if the regressor for which values have been shuffled is negligible, the prediction should not be impacted, therefore $RMSE^{OOB}_{shuffled} \approx RMSE^{OOB}_{ordered}$ and d remains the same.

9. shuffle one variable at a time, compute d, which indicates which variables pay a major role in increasing the model's accuracy.

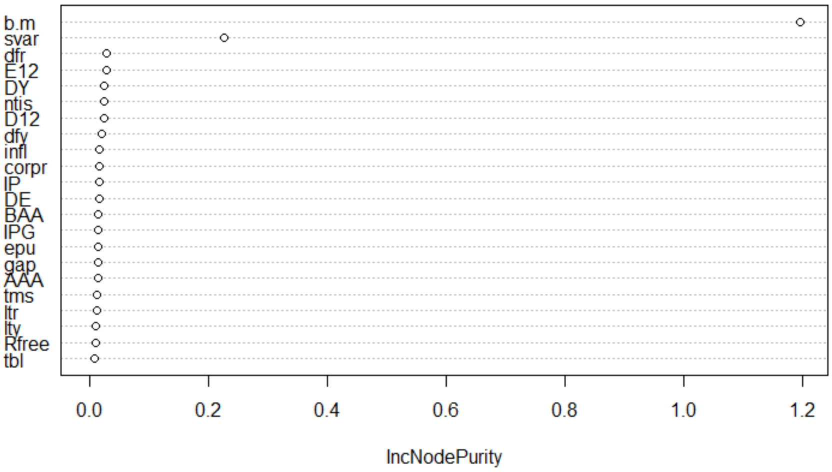


Figure 7: Increase in node purity

Bagging gives an understandable metric to measure each variable's importance. However, it is not an algorithm designed to select the most important one, but simply a measure of the variation in accuracy when a certain variable is added to the model or removed from the model. To chose which variable are worth keeping, we can set a threshold value; every variable which has a lower importance than the threshold will be removed. This method is subject to interpretation, and may not perfectly reflect the reality, as the cutoff value is arbitrarily fixed. More advanced techniques such as Regularized Random Forest are available, and perform iterative feature selection. Here, the results suggest **svar** and **b.m** are the only relevant variables.

If we want to add variables in the model (appart from **svar** and **b.m**), we need to scale the previous figure to detect gaps in increase in node purity between variables that currently appear irrelevant due to scale. To do so, we repeat the bagging process without the **b.m** variable.

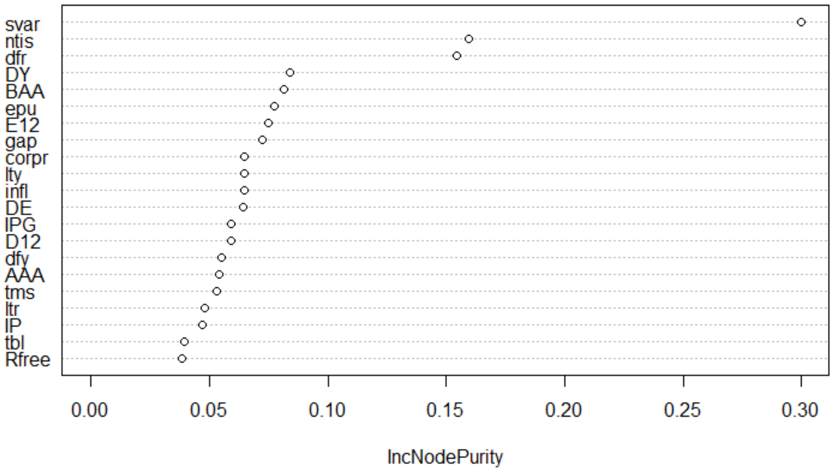


Figure 8: Increase in node purity

ntis, **dfr**, **DY**, **BAA**, **epu** and **E12** are the most important variables once **b.m** has been removed. We can try to produce forecasts with these additional variables and asses wether or not adding these supplementary variables is relevant in forecasting S&P 500. **b.m** and **svar** might carry sufficient information to produce reliable forecasts, but we are trying to explore the impact of variable selection on the model's quality, hence we found interesting testing both scenarios.

4 Results

We have estimated a total of 14 models. Each with the desire of reducing the number of features in a high dimensional framework. Variable selection was achieved in three different ways; algorithms (GETS modelling), penalized regression (LASSO, ridge, EN, SCAD, adaptive regressions, multi-step regressions), non-parametric algorithms (RF). Starting from 27 features, each model has selected a subset of variables, to reduce dimensionality, multicolinearity, and risk of over-fitting. The models should select a parsimonious, but useful subset of variables, i.e. the variables kept during the selection process should allow for smaller errors, whereas the ones which were removed do not help increasing accuracy. In the following section we summarize every model and present the results of OLS estimated regressions. We fit one model for every features combination selected by the 12 methods mentioned above, for a total of 12 models. Each model also contains the variable of interest, **returns**, lagged one unit.

4.1 Model comparison

Table 6: Variable selection

Variables	Lasso	Ridge	Bridge	Elastic Net	SCAD	aLASSO	WF	aEN	aSCAD	Ms-aEN	Ms-aSCAD	GETS	RF
D12	D12	D12	D12										
DY	DY	DY	DY		DY			DY	DY	DY			DY
DE	DE	DE	DE	DE				DE					
E12	E12	E12	E12	E12	E12	E12	E12	E12	E12	E12	E12	E12	E12
b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m	b.m
tbl		tbl											
AAA	AAA	AAA											
BAA		BAA											BAA
lty	lty	lty	lty	lty	lty			lty	lty	lty	lty	lty	
ntis	ntis	ntis	ntis	ntis	ntis			ntis		ntis			ntis
Rfree		Rfree											
infl	infl	infl	infl	infl	infl			infl	infl	infl		infl	
ltr		ltr											
corpr		copr											
svar	svar	svar	svar	svar	svar	svar	svar	svar	svar	svar	svar	svar	svar
IP	IP	IP			IP			IP		IP			
IPG		IPG											
gap	gap	gap						gap					
tms	tms	tms											
dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr	dfr
dfy	dfy	dfy	dfy		dfy			dfy		dfy			
epu	epu	epu		epu	epu			epu		epu			epu

We can notice a few variables are systematically retained in the models. **E12**, **b.m**, **svar** and **dfr** appear in almost every model. We can assume that these variables in particular are highly significant in determining the S&P 500 returns. Other variables such as **epu**, **dfy**, **infl**, **DY**, also appear in various models. **b.m** is the book-to-market ratio[18], which is well known to be significant when studying returns. It was previously mentioned in the Fama-French three factors model, which explains 90% of stocks returns. Hence, it was expected to see this variable being selected 100% of the time. Another interesting result is the Random Forest selecting **BAA** as one of the top 5 most significant variables, whereas **BAA** does not appear in any other model. Further investigation on the predictability of S&P 500 returns will tell us if including **BAA** is really pertinent. Except for **BAA**, all models seem to give similar importance to each variable. This is encouraging, when different methods yield identical results, we have stronger confidence in the information delivered by those methods. All methods worked as expected, selecting approximately half of the available variables, reducing dimensionality by roughly 50%. GETS(+SIS), aLASSO, WF and MSaSCAD are the most parsimonious models with the less predictors. This result may indicate different things. First, WF is specially designed to deal with high multicollinearity, if only one predictor is retained, maybe the other ones are too correlated and WF chooses to eliminate them. Adaptive and Multi step adaptive are iterative processes that apply the same filter multiple times, which may explain why aLASSO and MSaSCAD eliminate more variable than other methods. Maybe the shape of our data requires at least two steps to perform fully efficient variable selection. GETS is implemented with pre-screening performed by SIS, which makes it a two-steps variable selection process overalll. We will test these hypothesis by assessing the accuracy and predictive power of all the different collections of regressors. By fitting a model with OLS, and examining which is best, we verify if the more parsimonious models are justified or if more complex models might be as accurate, considering probable multicollinearity.

[18] Ratio between the accounting value of a firm and the share price observed on financial markets

Table 7: Accuracy and p-values for hypothesis testing on the OLS regeressions

Model	Adj R^2	Shapiro test	Jarque Berra test	Breusch Pagan test	VIF
OLS Lasso	0.6767	0.0	0.0	0.0	5
OLS Ridge	0.676	0.0	0.0	0.0	aliased
OLS Bridge	0.6772	0.0	0.0	0.0	5
OLS EN	0.6746	0.0	0.0	0.0	5
OLS SCAD	0.6774	0.0	0.0	0.0	5
OLS aLasso	0.6667	0.0	0.0	0.0	5
OLS WF	0.6771	0.0	0.0	0.0	5
OLS aEN	0.6625	0.0	0.0	0.0	5
OLS aSCAD	0.6755	0.0	0.0	0.0	5
OLS Ms-aEN	0.6774	0.0	0.0	0.0	5
OLS Ms-aSCAD	0.6723	0.0	0.0	0.0	5
OLS Gets	0.6741	0.0	0.0	0.0	5
OLS RF	0.6527	0.0	0.0	0.0	5
OLS RF pruned	0.6705	0.0	0.0	0.0	5

Table 6[19] summarizes important metrics of OLS models fitted with selected variables. As we can see, none of the models respect assumptions to apply OLS. the Shapiro-Wilk test suggests to reject the null hypothesis of normal distribution of the residuals, results hold with the Jarques-Bera test. Brush-Pagan suggests heteroskedasticity of the residuals at a 1% confidence level. Hence, the non normality of residuals, and heteroskedasticity mean that coefficients are biaised. We can not tell the sens of the bias and his magnitude. VIF (Variance Inflation Factor) are all under 5 expected for ridge OLS, indicating high multicolinearity in the regressors for this model[20]. The variable selection did not allow us to completely get rid of recurrent problems in linear regression. Nevertheless, a good way to see the real quality of the model is to test it on a test sample. There are several methods to test our models, we gonna use two: we will produce rolling forecasts, i.e we fit a model on N data points and forecast the N+1 observation, then we mode the window on which the model is estimated, re-fit it and get predictions for the next value. This way the model is estimated at each step, on the N points prior to the forecasted one. Afterward we use recursive forecasts, where the model is also updated but the start point of the window remains the same, hence we just add more and more data points into the model. In the recursive scheme the model is fitted on N data points, then N+1 data points, then N+2 etc. Whereas is rolling, the number of data points used to fit the model is N at all

[19] We notice that the total explained variance with RF pruned as a model is to 76.54%, which is 9 points more than OLS

[20] 5 is often used as the cutoff value to detect multicollinearity, by consensus

time, but N is constituted of the most recently available observations and the oldest ones are dropped at each step.

This allows us to avoid using dated observations to fit the model. Indeed, the relation between S&P 500 returns and regressors is not necessarily the same at each period of time. Therefrom, the rolling forecast method allows to estimate coefficients with more recent observations, and less outdated, possibly irrelevant observations.

In terms of in-sample adjustement, adjusted R^2 are unexpectedly high with values around 0.65, meaning with explain approximately 65% of the S&P 500 returns. The best variable selection methods seem to be SCAD or MS-aEN. SCAD allows for lesser penalization of highly significant regressors and MS-aEN allows for iterative selection and more flexible variable elimination by combining LASSO and Ridge penalties. We then proceed to forecasts the S&P 500 returns using the two forecasting scheme mentioned above.

4.2 Forecasts

Table 8: Forecast quality of different OLS model

Model	MSE recursive forecast	MSE rolling forecast
OLS Lasso	0.0006603	0.0006559
OLS Ridge	0.0006614	0.0006525
OLS Bridge	0.0006532	0.0006532
OLS EN	0.0006602	0.0006521
OLS SCAD	0.0006423	0.0006379
OLS aLasso	0.0006454	0.0006433
OLS WF	0.0006713	0.0006612
OLS aEN	0.0006501	0.0006435
OLS aSCAD	0.0006395	0.0006376
OLS Ms-aEN	0.0006423	0.0006379
OLS Ms-aSCAD	0.0006380	0.0006350
OLS Gets	0.0006404	0.0006370
OLS RF	0.0007590	0.0007362
OLS RF pruned	0.0006501	0.0006459

Table 7 summarizes the quality (measured with MSE) of different OLS models fitted before. As expected, MSE with rolling forecast of all models are lower than MSE obtained with recursive forecast. According to the results we can say that OLS RF is the worst model in both forecasting schemes, indicating the variables selected by the random forest are not optimal. We noticed earlier that random forest kept the **BAA** variable, the previous results indicates this choice may have been unjustified. The best model is OLS Ms-aSCAD. However, We notice that all MSE are similar. Therefore, we used the Diebold & Mariano test to measure if it exist a significant difference between the 14 models. Due to coercion, multitest of Diebold & Mariano could not be done on all 14 models. We chosed the 5 best models according to MSE to test if it exist a statistical difference of forecast. The multitest DM says that it exists a statistical difference between all models (p-value < 0.05). Table 8 shows the univariate test of each models. According to these resultats, finnaly it is not existing any difference between each models as MSE let thought.

Table 9: Univariate Diebold & Mariano test

	SCAD	aSCAD	Ms-aSCAD	GETS	Ms-aEN
SCAD	1				
aSCAD	0,9819	1			
Ms-aSCAD	0,8392	0,7477	1		
GETS	0,9505	0,9341	0,6453	1	
MS-aEN	1	0,9819	0,8392	0,9506	1

The Diebold & Mariano test does not suggest statistical difference in prediction accuracy between the models. The p-values are close to one, we can not reject the null hypothesis of similar forecasting accuracy between models. Between our five most precise models none excels, indicating that SCAD, aSCAD, MS-aSCAD, GETS and MS-aEN yielded subsets of variable similar enough to produce identically precise forecasts. This results is in line with previously made observations that all models retained similar subsets of variables. We can visually explore the previous results with cumMSE (cumulative MSE) figures. cumMSE is the cumulative sum of the difference between the error of two different models. $cumMSE = \sum_{h=1}^{H} MSE_h^{model1} - MSE_h^{model2}$. If the model 1 is more accurate, the difference in MSE is positive, if the model 1 is **consistently** more accurate, the sum of differences will also be positive. cumMSE help visualize which models are constantly better based on multiple forecasts. h is the forecast horizon, H is the total number of h-step-ahead forecasts considered. The next figure shows cumMSE for the 5 best models. Because models are so similar, we expect cumMSE, to be 0. The model used as a benchmark here is the SCAD OLS. Therefore, a positive cumMSE indicates the model M is more precise than SCAD OLS, a negative cumMSE indicates model M is less precise than SCAD OLS. We will use the rolling forecasts as they are slightly more precise than the recursive forecasts.

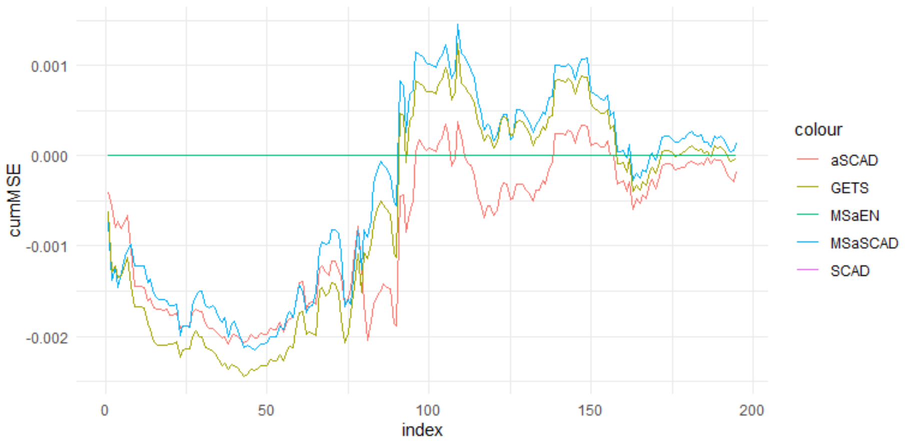
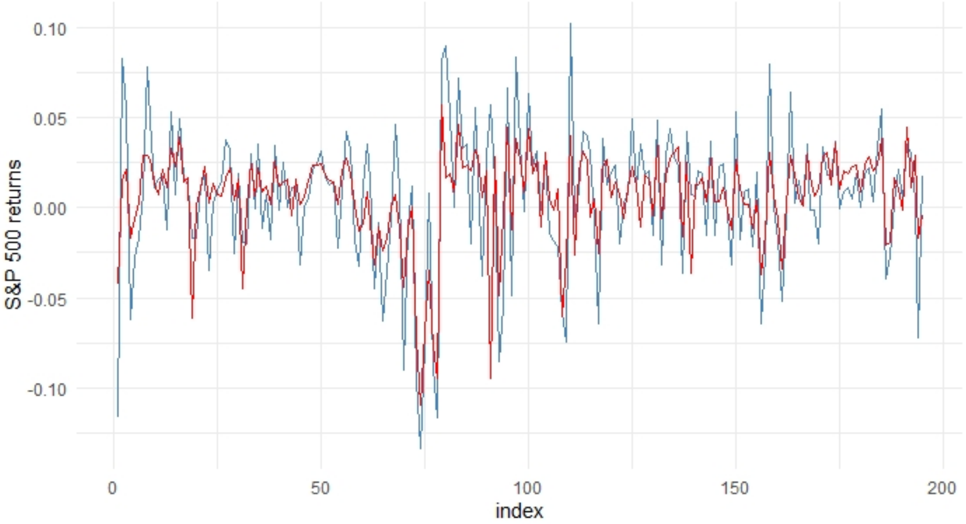


Figure 9: Cumulative MSE

cumMSE are not stricly positive or negative, indicating forecasts from the variables selected by aSCAD, GETS and MSaSCAD are not consistently better or worse compared to SCAD. Obviously cumMSE between SCAD and itself is 0, however, cumMSE for MSaEN also is 0, which tells us the forecats from the two models are strictly identical. This was expected as the variables elected by these two methods are the same. Althought cumMSE is different from 0 for three models, the Diebold & Mariano test suggests that the difference in forecasting accuracy is not significant. The similar behaviours of the models is visualized on the previous figure with all the series displaying almost identical patterns. Multi Steps adaptive SCAD seems to be consistently better than aSCAD and GETS model even thought the increase in accuracy is not significant. Finally these results are obtain on a test sample which was cleaned from outliers before. Therefore, our results are in reallity a bit worser if we took real values from S& P 500 as the test sample. Nevertheless, in real values, there is only a couple of outliers detected by boudt method. Ultimately, the forecast accuracy of OLS regression is really good, considering that all OLS regression does not hold every assumptions on residuals, which gave us biased coefficient.

In the following figure, we can observe the real S&P 500 returns and forecast values:

Figure 10: Forecast of S&P 500 with OLS Ms-aSCAD

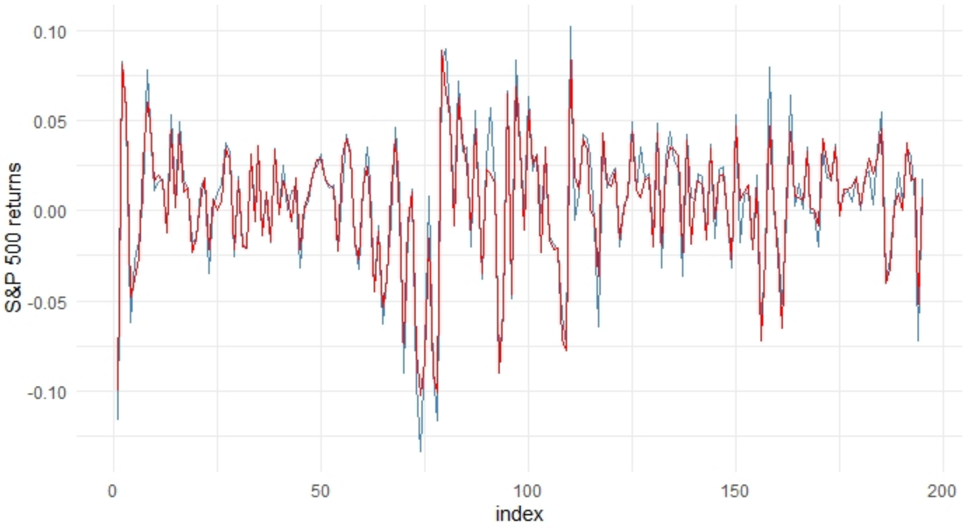


We used before an algorythm of random forest to select some potential usefull regressor to fit the best possible model on S&P 500. We will use it to produce forecast too, using only the rolling scheme, and compare its performance *versus* the best OLS model. *A prioris* due to the fact that random forest has no assumptions to follow on residuals, we should obtain a better accuracy in forecast in term of MSE.

According to our data and the rolling forecast method, with a mtry parameter equals to 8 (the number of variables choosed randomly among all possible to fit a tree), a nodesize parameter equal to 20 i.e the minimal size of terminal node of a branch; we obtain a MSE_{OOS} of 0.00010 which is 6 time inferior to the best OLS regression fitted. In addition, we can notice that the RF pruned model is robust to compute forecasting, i.e that the difference between results fitted in sample and out of sample are relativelly close, in comparison of OLS models where we loose in accurracy. As we can see on the figure 11, the forecast are way better than ones made with OLS Ms-aSCAD[21].

[21] the p-value of DM test is 0.00 which mean that the difference in forecasting between RF pruned and OLS Ms-aSCAD is very significant

Figure 11: S&P 500 forecast with RF pruned

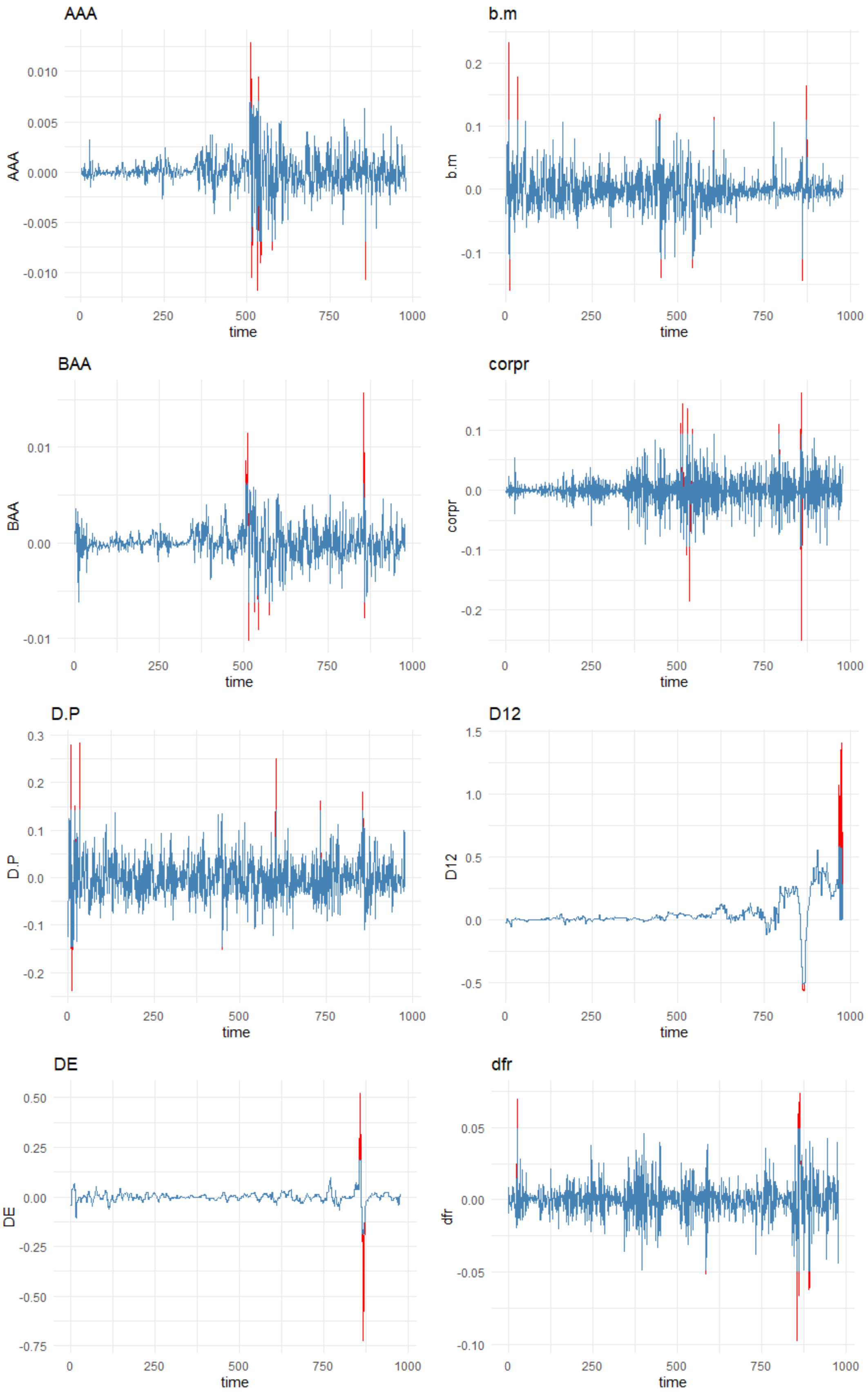


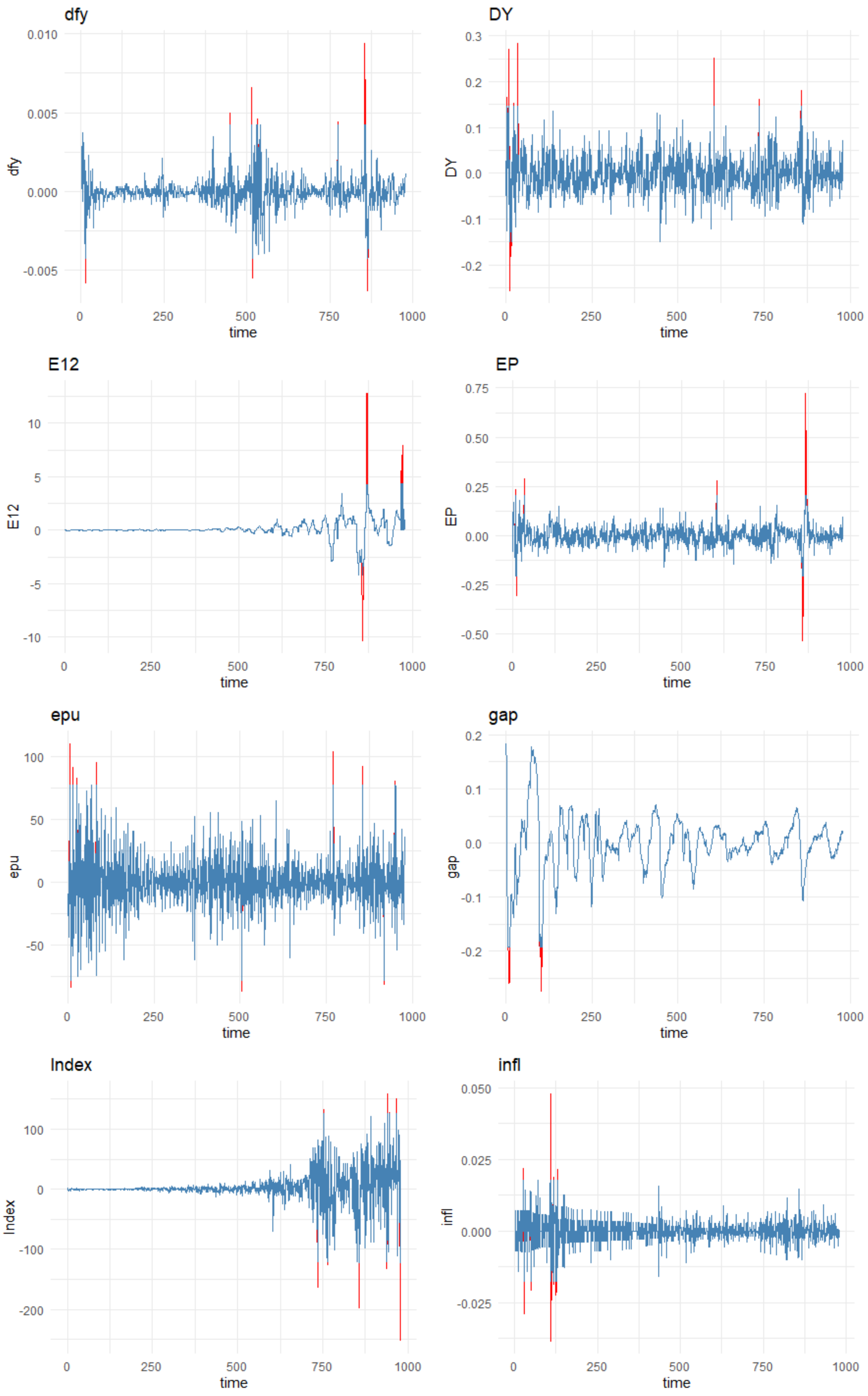
5 Conclusion

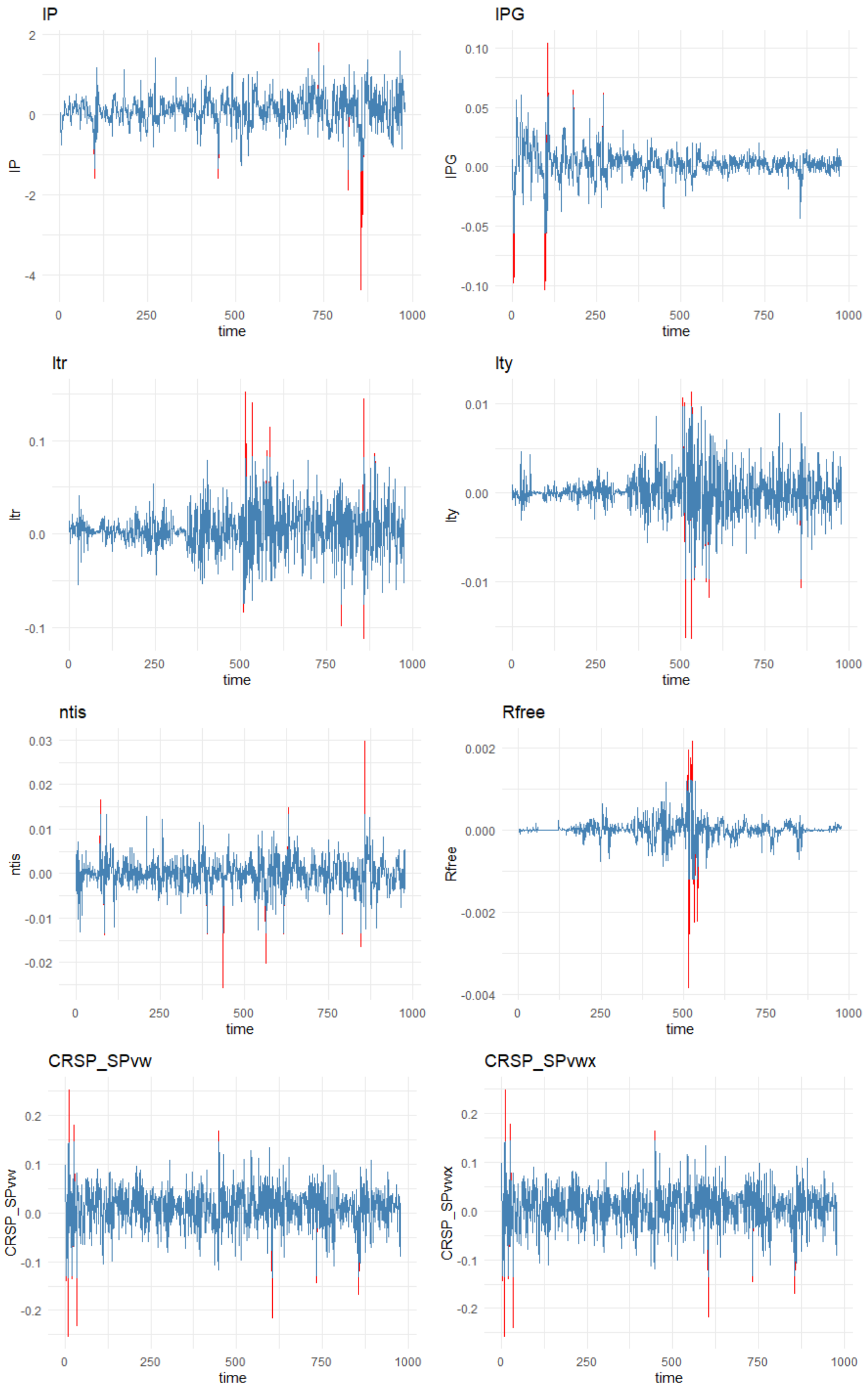
The main objective of this work was to experiment with different variable selection methods to build models designed to accurately forecast S&P 500 returns. We started out with 26 available regressors, on which we performed variable selection with GETS, penalized regression and random forests. The 14 methods yielded very similar subsets of variable which we then used to fit an OLS on S&P 500 returns. The similarities between each subsets gave similar forecasts, and accuracy. Although we showed some models were slightly more accurate than others, Diebold & Marinao tests proved the differences were not significant. the variable selected by the random forest semm to be less significant than other models, with the introduction of **BAA** as the 5^{th} most important variable, while it does not appear in other models, MS-aSCAD seems to yield the subset which fits the data best but forecasts from a standard aSCAD appear more reliable. The iterative nature of multi-step penalized regression might induce over-fitting, which would explain better fit but less accurate forecasts.

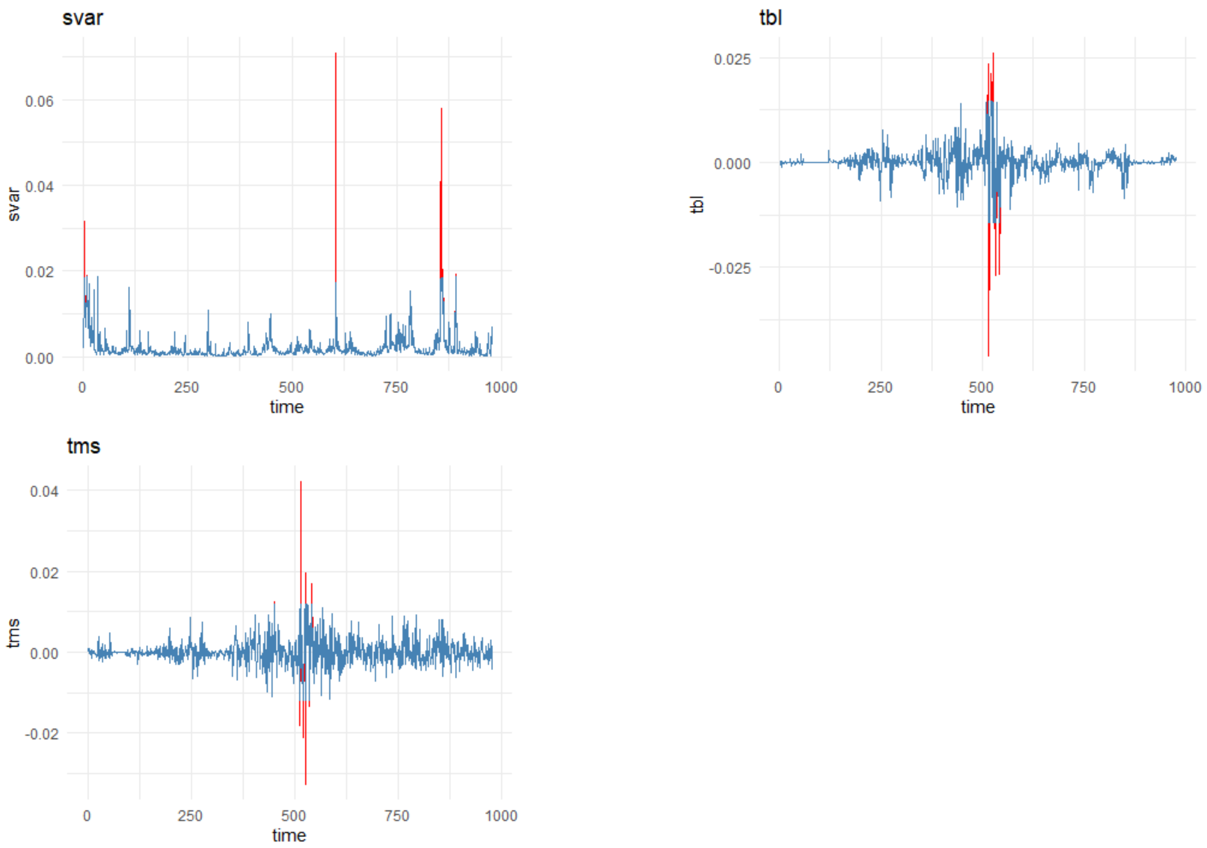
Appendix

Appendix 1: Series cleaned with boudt method vs raw series

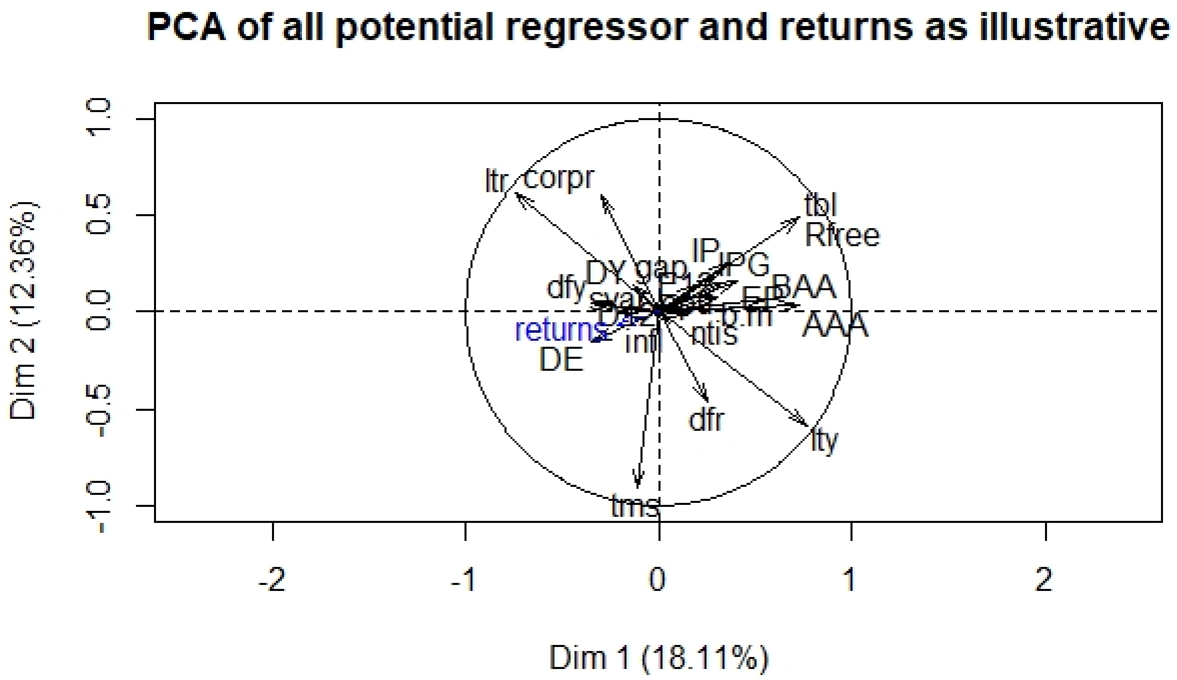



dfy
0.010
0.005
0.000
-0.005
dfy
0
250
500
750
1000
time
DY
0.3
0.2
0.1
0.0
-0.1
-0.2
DY
0
250
500
750
1000
time
E12
10
5
0
-5
-10
E12
0
250
500
750
1000
time
EP
0.75
0.50
0.25
0.00
-0.25
-0.50
EP
0
250
500
750
1000
time
epu
100
50
0
-50
epu
0
250
500
750
1000
time
gap
0.2
0.1
0.0
-0.1
-0.2
gap
0
250
500
750
1000
time
Index
100
0
-100
-200
Index
0
250
500
750
1000
time
infl
0.050
0.025
0.000
-0.025
infl
0
250
500
750
1000
time

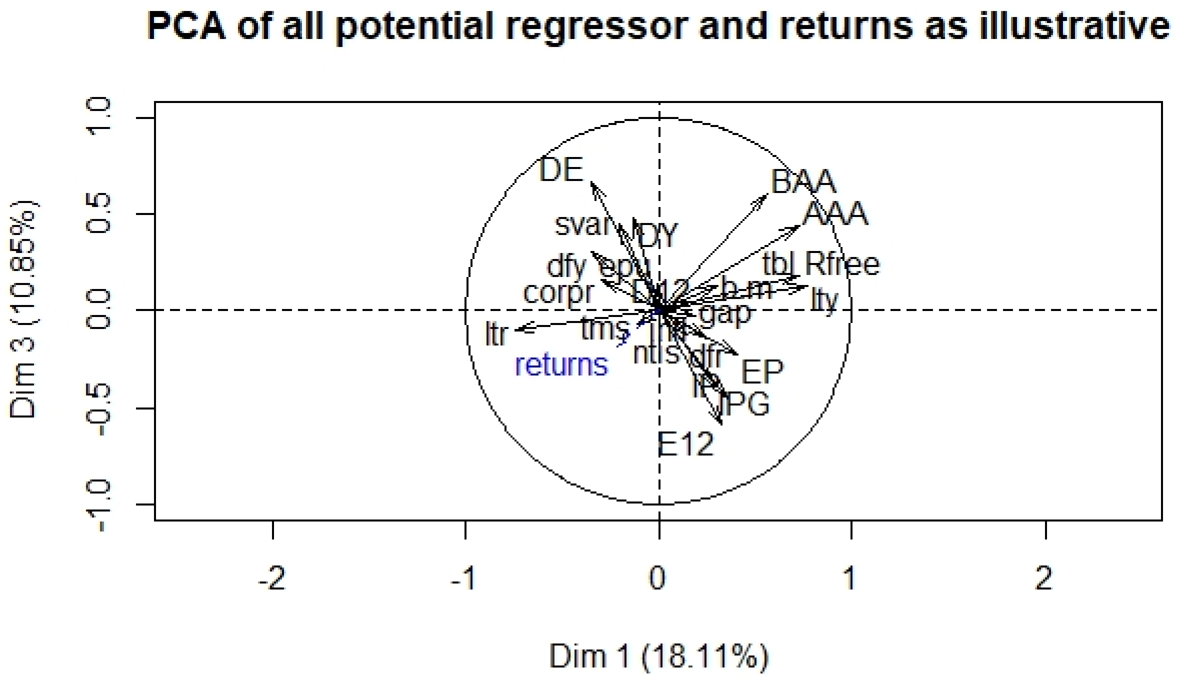

IP
IP
2
0
-2
-4
0
250
500
750
1000
time
IPG
IPG
0.10
0.05
0.00
-0.05
-0.10
0
250
500
750
1000
time
ltr
ltr
0.1
0.0
-0.1
0
250
500
750
1000
time
lty
lty
0.01
0.00
-0.01
0
250
500
750
1000
time
ntis
ntis
0.03
0.02
0.01
0.00
-0.01
-0.02
0
250
500
750
1000
time
Rfree
Rfree
0.002
0.000
-0.002
-0.004
0
250
500
750
1000
time
CRSP_SPvw
CRSP_SPvw
0.2
0.1
0.0
-0.1
-0.2
0
250
500
750
1000
time
CRSP_SPvwx
CRSP_SPvwx
0.2
0.1
0.0
-0.1
-0.2
0
250
500
750
1000
time


svar
0.06
0.04
0.02
0.00
svar
0
250
500
750
1000
time
tbl
0.025
0.000
-0.025
tbl
0
250
500
750
1000
time
tms
0.04
0.02
0.00
-0.02
tms
0
250
500
750
1000
time

Appendix 2: PCA of potential regressor and returns



(a) PCA axes 1 and 2



(b) PCA axes 1 and 3