ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis Quantitative Finance

Evaluating the differences between elastic net, ridge and lasso regression on empirical asset pricing

Name student: Igor Lipper
Student ID number: 451299

Supervisor: X. Xiao
Second assessor: S.H.L.C.G. Vermeulen

Date final version: July 5th, 2020

Abstract

This paper looks at different penalization models in the context of empirical asset pricing. The first part consists of replicating the elastic net model used in (Gu, Kelly, & Xiu, 2020). We use the predictive power of this elastic net model as a benchmark. The second part consists of extending the research in (Gu, Kelly, & Xiu, 2020) by looking at the ridge and lasso regression models and comparing their predictive power to our elastic net benchmark model. In our research we see that the lasso model outperforms the elastic net model, and the ridge model performs more poorly than the elastic net model. This is no surprise, as we are dealing with a large set of independent variables that are highly correlated. We compare these models on a number of comparative statistics, including out of sample R-squared, that measure predictive power. In the end we also look at what variables are selected in each of the three models.

Contents

1 Introduction

There has been conducted a lot of research on empirical asset pricing in the past, particularly in the paper (Gu, Kelly, & Xiu, 2020). In that paper multiple machine learning methods are used to predict excess stock return, among them the elastic net model. However, other penalization methods aren't tested in that paper.

Empirical asset pricing is about predicting return of equity (in our case stocks) in excess of the risk free rate. This is done by using a large number of covariates that can predict the risk free rate one month ahead by using different machine learning models. Being able to predict the excess return relatively accurately is very interesting to investors that want to maximize their return. Looking for more accurate models is important research as this can directly lead to a better prediction and thus larger monetary gains.

Using machine learning models over other non machine learning methods has a number of advantages. First, in the case of empirical asset pricing there are a lot of predictors available to predict the excess stock returns. This means that in some cases the amount of predictors approaches the total number of observations, which often leads to overfitting the data. Second, the predictors that are available are more often than not closely related which means that they have a high correlation. But by using penalization models these two problems go away. This is because the models are not only able to perform dimension reduction, but also variable selection.

In this paper, we build on the research in (Gu, Kelly, & Xiu, 2020) on empirical asset pricing. First, we replicate their findings of the elastic net model as a benchmark. Secondly, we then look to improve on this benchmark by using the ridge and lasso models to predict the one month ahead excess stock returns and subsequently comparing their results to the elastic net benchmark. The ridge and lasso model haven't been evaluated yet in the context of empirical asset pricing. We look at predicting the return of individual stocks in excess of the risk free rate, also called the excess stock return. We are comparing the existing elastic net model used in (Gu, Kelly, & Xiu, 2020) with our own ridge and lasso models. We compare these models using a number of different statistics. The first and most important one is the out of sample R^2, from now on denoted as R^2_{OOS}. Along with the R^2_{OOS} we also look at the Mean Squared Prediction Error and the Root Mean Square Error. We also go into the different scenarios when one of these models would be expected to outperform the others. We do this by looking at previous research articles.

After calculating the R^2_{OOS} of the elastic net model, we find that we weren't able to exactly replicate the elastic net model used in (Gu, Kelly, & Xiu, 2020) for a number of reasons. Part of the reason is that our data set is around 5% of the data set that is used by (Gu, Kelly, & Xiu, 2020).

The other part of the reason is that our elastic net model is different on a few mathematical fronts.

For the extension of introducing the ridge and lasso model and then comparing these results to our elastic net model, we are also able to draw a few conclusions. We note that the lasso model is able to outperform the elastic net model with an R^2_{OOS} that is three times as large. However, our ridge model has a negative R^2_{OOS} value which consequently makes it perform more poorly than the elastic net model.

Since the existing paper (Gu, Kelly, & Xiu, 2020) only covers the results of elastic net, it is a great contribution to dive deeper into the ridge and lasso models. Also, that paper doesn't use other statistics like the MSPE and RMSE that are a good way of determining the predictive power of these models. Our research is necessary because an extensive comparison between these three models predicting excess return hasn't been done before to our knowledge. Finding that either ridge or lasso is clearly superior in predicting the one month ahead excess return than elastic net, then this is useful knowledge for further research into empirical asset pricing using machine learning techniques. This means that it is of interest for practical applications. But it is also interesting in terms of scientific relevance since we look at three models that are very similar at first glance but have a fundamental difference.

This bachelor thesis addresses this by comparing elastic net, ridge, and lasso regression side to side on the same data set. This way we can compare the results in a clear and easy to understand way.

2 Literature Review

Since this research is extending on the paper (Gu, Kelly, & Xiu, 2020), it is vital to discuss what their findings relevant to this extension are. In their paper, they note that the elastic net model is able to achieve a positive R^2_{OOS} value when looking at the 1 month ahead excess return predictions. The elastic net model is able to outperform the ordinary least squares model using all of the independent variables. However, the more complex machine learning models like random forest and neural networks are able to perform better than the elastic net model. Also, it is important to note that the OLS model using only three of the independent variables is able to outperform the elastic net model. On the other hand, they don't look at lasso and ridge regression. Comparing the elastic net model to the lasso and ridge model is the core of our research paper.

There has already been previous research that compares lasso and ridge regression for data analysis. (Melkumova & Shatskikh, 2017) show that regression models with regularization techniques outperform linear regression when the independent variables are heavily correlated and/or when the

data set contains a lot of independent variables. Also, they note that since lasso regression can set the parameter coefficients to 0 whereas ridge regression cannot (ridge regression can only set the parameter coefficients close to 0, but not equal to 0), lasso is expected to outperform ridge when we are dealing with a large number of independent variables.

3 Methodology

We will now look at the way that the excess stock return is constructed. The easiest way to understand this is by looking at the prediction error model in Equation 1:

$$r_{i,t+1} = \mathrm{E}_t(r_{i,t+1}) + \epsilon_{i,t+1}. \tag{1}$$

Here $r_{i,t+1}$ denotes the observed excess stock return for stock i in period $t+1$, and $\mathrm{E}_t(r_{i,t+1})$ denotes the value for $r_{i,t+1}$ that the 'true' model would predict it to have. The error term $\epsilon_{i,t+1}$ simply denotes the error term associated with $\mathrm{E}_t(r_{i,t+1})$. This expected return $\mathrm{E}_t(r_{i,t+1})$ is constructed as

$$\mathrm{E}_t(r_{i,t+1}) = g^*(z_{i,t}), \tag{2}$$

where $z_{i,t}$ is a vector of dimension P that contains all of the predictors for stock i at time t. The function $g^*(\cdot)$ transforms this P-dimensional vector into the expected return at time $t+1$. It is important to note that $g^*(\cdot)$ doesn't depend on stock i or time period t, but only on the input of vector $z_{i,t}$. This means that this function is easy to work with since it doesn't vary over time or company stock. Also, the expected return $\mathrm{E}_t(r_{i,t+1})$ only depends on $g^*(\cdot)$ via $z_{i,t}$ which means that in predicting the excess return of stock i at $t+1$ we do not take any other stock $j \neq i$ or time interval $t-k$ where $k \geq 1$.

All three of the machine learning methods that we discuss are built up in the same way. Since we don't know the function of the expected excess stock return $g^*(\cdot)$, we approximate this by creating the linear function $g(\cdot)$ that can be seen as

$$g(z_{i,t}; \theta) = z'_{i,t}\theta, \tag{3}$$

where $z_{i,t}$ is once again the P-dimensional vector containing the predictors. The vector θ is the parameter vector that contains all of the coefficients for the predictors in $z_{i,t}$. This model doesn't allow any interaction between predictors or other nonlinear operations.

Now that we have the function $g(\cdot)$ that approximates $g^*(\cdot)$, we can construct a least squares

function that minimizes the total sum of squared residuals. We start by constructing the following least squares function that can for example be used for pooled OLS:

$$\mathcal{L}(\theta) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}(r_{i,t+1} - g(z_{i,t};\theta))^2. \tag{4}$$

The stocks go from $i = 1, ..., N$ and the time periods go from $t = 1, ..., T$. Minimizing this cost function $\mathcal{L}(\theta)$ gives us the optimal parameter vector θ for the case of pooled OLS.

3.1 Splitting the Sample and Tuning the Hyperparameters

We will split the complete data set into three sub samples: a training sample, a validation sample, and a testing sample. The training sample is used to calculate the parameter vector of the model subject to the value of a specific hyperparameter. We then use this model to calculate the predicted values of the excess return in the validation sample. This gives us a more realistic predicted value since we are not using the training sample that is used to fit the model. We then compute the R^2_{OOS} to see how well the fit of the model with this specific hyperparameter value is. We do this for multiple different hyperparameter values so that we can compare the R^2_{OOS} of each different hyperparameter value. The hyperparameter value that causes the higest R^2_{OOS} value in the validation sample is the one that we will use for our model in the testing sample since it has performed the best.

The next step is to take the fitted model that corresponds to the optimal hyperparameter so that we can calculate the R^2_{OOS} in the testing sample. This gives us an even better idea of the predictive power of this model since the testing sample is 'truly' out of sample. This is the case because of the fact that our hyperparameter value is determined by the validation sample. Therefore the validation sample is not truly out sample anymore when we use the optimal hyperparameter to fit our model.

The split between the training, validation and testing sample is the following: First we split the data 80/20 and set the last 20% of the observations aside to use as the testing sample. We then split the first 80% of observations again in a 80/20 split so that we now have three sub samples. The first sub sample is the training sample and contains the first 64% of observations (80% * 80% = 64%). The second sub sample is the validation sample and includes the 16% of observations following the training set (20% * 80% = 16%). The third and last sub sample is the testing sample and includes the last 20% of observations.

3.2 Machine Learning Models

In this research we look at three penalization models that are closely related but still differ on a fundamental level. They are all based on the same principle of adding a regularization penalty to the original loss function. This is made clear by Equation 5.

$$\underbrace{\mathcal{L}(\theta;\cdot)}_{\text{total loss function}} = \underbrace{\mathcal{L}(\theta)}_{\text{original loss function}} + \underbrace{\phi(\theta;\cdot)}_{\text{regularization penalty}} \tag{5}$$

The most basic penalization model of the three is the ridge regression. Using the ridge regression model boils down to adding the sum of squared parameters to the original loss function. This has the effect of making the predictors less dependent to the testing set which has the effect of reducing the overfit to the training set. Equation 6 shows what the regularization penalty is in case of the ridge regression.

$$\phi_{Ridge}(\theta;\lambda) = \frac{1}{2}\lambda\sum_{j=1}^{P}\theta_j^2 \tag{6}$$

Here λ is the magnitude of the regularization penalty, and θ represents the parameter vector. P corresponds to the total number of independent variables, so that θ_j contains the coefficient for independent variable number j.

The lasso regression case differs from the ridge regression case by using the sum of the absolute values of the parameters instead of the sum of squared parameters. Equation 7 shows the regularization penalty of the lasso regression.

$$\phi_{Lasso}(\theta;\lambda) = \lambda\sum_{j=1}^{P}|\theta_j| \tag{7}$$

The elastic net regression is a combination of both the ridge and lasso regression. For this case we have to introduce another hyperparameter α that chooses the weight of the ridge regression and the weight of the lasso regression. This hyperparameter α can take any value from 0 to 1. The case where $\alpha = 1$ corresponds to the ridge regression and the case where $\alpha = 0$ corresponds to the lasso regression. Equation 8 shows the regularization penalty of the elastic net regression.

$$\phi_{ElasticNet}(\theta;\lambda,\alpha) = \lambda(1-\alpha)\sum_{j=1}^{P}|\theta_j| + \frac{1}{2}\lambda\alpha\sum_{j=1}^{P}\theta_j^2 \tag{8}$$

For all three models, too low values of λ lead to overfitting and too high values of λ lead to underfitting (Melkumova & Shatskikh, 2017).

3.3 Comparative statistics

We compare the three penalization regression models that are explained above in a number of different ways. The first one is by comparing the R^2_{OOS} values for excess return of individual stocks. The formula for the R^2_{OOS} is given in equation 9.

$$R^2_{OOS} = 1 - \frac{\sum_{(i,t)\in\mathcal{T}_3}(r_{i,t+1} - \hat{r}_{i,t+1})^2}{\sum_{(i,t)\in\mathcal{T}_3} r^2_{i,t+1}} \tag{9}$$

Here $\mathcal{T}_3$ is the testing sample. That means that the R^2_{OOS} is truly out of sample since only the training and validation sample are used to determine the parameters and hyperparameters respectively. This means that the R^2_{OOS} is a good way to compare the predictive power of the three models. The $r_{i,t+1}$ denotes the true value of the excess return for stock i at time $t+1$, and $\hat{r}_{i,t+1}$ denotes the predicted value of the excess return for stock i at time $t+1$ using the machine learning model that is trained and validated on the training and validation sample respectively. A subtle aspect of this R^2_{OOS} is that we construct it in the same way as (Gu, Kelly, & Xiu, 2020) construct the R^2_{OOS} in their paper. The denominator is the sum of squared excess stock returns without demeaning. The reasoning behind this is that the historical average stock returns are extremely noisy, which makes it seem that our predictions are worse than they actually are.

We also look at the Mean Squared Prediction Error (MSPE) of these three models and compare them. The MSPE is a simple statistic that will take the expected value of the squared difference between the predicted and the actual value of the excess return. Equation 10 shows the formula of the MSPE. In this equation the matrix L represents the projection matrix that can be understood as $\hat{r}_{i,t+1} = Lr_{i,t+1}$. This projection matrix turns the observed values of excess return into the predicted values of excess return.

$$\text{MSPE}(L) = \frac{1}{NT}\sum_{(i,t)\in\mathcal{T}_3}\left[(r_{i,t+1} - \hat{r}_{i,t+1})^2\right] \tag{10}$$

The last statistic that we use to compare the three models is the Root-Mean-Square Error (RMSE). This statistic can be seen as taking the root of the MSPE. The formula of the RSME is give in equation 11.

$$\text{RMSE} = \sqrt{\frac{1}{NT}\sum_{(i,t)\in\mathcal{T}_3}[(r_{i,t+1} - \hat{r}_{i,t+1})^2]} \tag{11}$$

Previous research tells us that the lasso regression performs best to predict the value of the dependent variable when only a handful of independent variables are significant and the majority is insignificant. This is because the lasso model sets the parameter value of all of the insignificant independent variables equal to 0.

On the other hand, it is better to use ridge regression when we know that a vast majority of our independent variables is significant. This is because ridge regression cannot set the parameter value of an independent variable equal to 0, but can only set this equal to a value that is close to zero when it finds that an independent variable is insignificant.

Elastic net uses both characteristics and can be seen as a model that is in between the two and thus generates a parameter vector that isn't as extreme as either ridge or lasso.

3.4 Determining Optimal Hyperparameters

For each of the three machine learning models that we are looking at, we have to determine the optimal hyperparameters. For the ridge and lasso case this includes only λ and for the elastic net case this includes both λ and α. However, we choose $\alpha = 0.5$ for the elastic net case since this is a nice equilibrium between ridge and lasso. That leaves us with determining what the best λ would be for each of the three machine learning models.

We do this by comparing the R^2_{OOS} of different values for λ in the validation sample. We choose to pick values for λ on the $[10^{-4}, 10^{-1}]$ interval for comparing the R^2_{OOS}. In total we compare 13 values of λ where $\lambda = 10^x$, and x increases by 0.25 each step, so that we get $x = -4, -3.75, -3.5,$ ... $, -1.5, -1.25, -1$.

As stated before, we then proceed to pick the λ with the highest R^2_{OOS} in the validation sample to test the model in the testing sample.

3.5 Coordinate Descent

We use coordinate descent for finding the optimal parameter vector θ in all of the three machine learning models. A big advantage of coordinate descent is that the loss function doesn't have to be smooth in each separable part for it to find a minimum. In the case of lasso regression and elastic net regression, we have $\sum_{j=1}^{P} |\theta_j|$ as part of the loss function. Because taking the absolute value of $|\theta_j|$ result is a non differentiable function, we have to use an optimization algorithm to overcome this

hurdle. Coordinate descent is an optimization algorithm that is excellent at performing this task. It minimizes the loss function by only moving along one coordinate direction per iteration.

4 Data Collection and Construction

4.1 Collecting the Data

We use the same data set as is used in (Gu, Kelly, & Xiu, 2020). This data consists of the excess monthly stock returns from March 1957 up to December 2016 as the dependent variable. The independent variables are 94 stock-level predictive characteristics, 74 industry dummies that correspond to the first two digits of the SIC (Standard Industrial Classification) codes, and 8 macroeconomic predictors. Along with using the 94 stock-level predictive characteristics by themselves as independent variables, we also multiply these 94 stock-level predictive characteristics by the 8 macroeconomic predictors. In total this brings us to 94*(8+1) + 74 = 920 independent variables.

Since the excess monthly stock return is the difference between the monthly stock return and the 1-month risk free rate, we need to obtain both to calculate the excess monthly stock return of each individual stock at each month. We use the monthly holding return from CRSP as the monthly stock return. As for the 1-month risk free rate, we use the monthly risk free rate that is used in (Welch & Goyal, 2008).

The 94 monthly stock-level predictive characteristics can be found on the website of Dacheng Xiu along with the SIC codes. For constructing the macroeconomic predictors, we use the same data that is used in (Welch & Goyal, 2008).

4.2 Constructing the Data

We choose to remove all observations that have missing values, since a lot of observations are missing close to half of the 94 stock-level predictive characteristics. This results in a usable data set with only around 175000 observations instead of using a data set with around 3 to 4 million observations that has many missing values. Another setback is that this usable data set starts in 1985 instead of 1957 like in (Gu, Kelly, Xiu, 2020). Also, after removing all the observations with missing values, we are left with only 907 independent variables since 13 of the 74 industry dummy variables don't include any information for the observations in our usable data set. The last thing that we do is removing the last dummy variable from the set of independent variables so that we won't fall into the dummy variable trap.

5 Results

We now present the R^2_{OOS} in the validation sample for 13 different λ values per machine learning model. These findings are displayed in Figures 1 to 3 where the λ value that produces the highest R^2_{OOS} in the validation sample is the one that we choose for calculating comparative statistics in the testing sample.

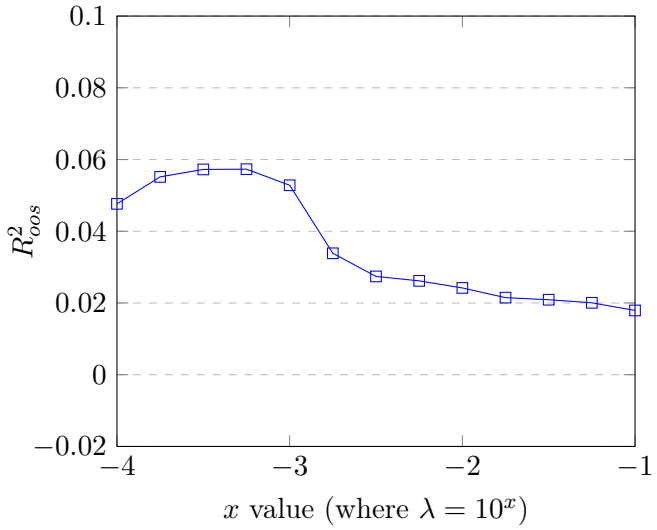
We start with looking at the elastic net model in section 5.1. What we are doing here is trying to replicate the findings of (Gu, Kelly, & Xiu, 2020). Therefore we hope to get a R^2_{OOS} value that is as close as possible to the one that they find in their research. Along with the R^2_{OOS}, we also look at the MSPE and RMSE of the elastic net model so that we can get more insight. This is something that isn't done in (Gu, Kelly, & Xiu, 2020), but we feel that it adds value to also look at these two comparative statistics.

After replicating the R^2_{OOS} value of the elastic net model, we look at the ridge and lasso model. We hope that either one of these two models is able to outperform the elastic net model, since this would mean that we are able to improve on the elastic net model that is used in (Gu, Kelly, & Xiu, 2020).

5.1 Elastic Net

In the case of the elastic net model, we obtain the following validation sample R^2_{OOS} values for each of the following λ values displayed in the graph below:

Figure 1: R^2_{OOS} value for determining λ in the elastic net model



It is clear from this graph that for the elastic net model we should take $\lambda = 10^{-3.25}$ as our hyperparameter value since the R^2_{OOS} is the largest. Also, the MSPE and RMSE are the smallest for this λ (this isn't shown in Figure 1).

Taking the λ value corresponding to the highest R^2_{OOS} in Figure 1 to calculate the R^2_{OOS}, MSPE and RMSE for the testing sample gives us the following results displayed in Table 1. Here we also add the R^2_{OOS} that is obtained in (Gu, Kelly, & Xiu, 2020) for comparison.

Table 1: R^2_{OOS}, MSPE and RMSE values for the testing sample using elastic net

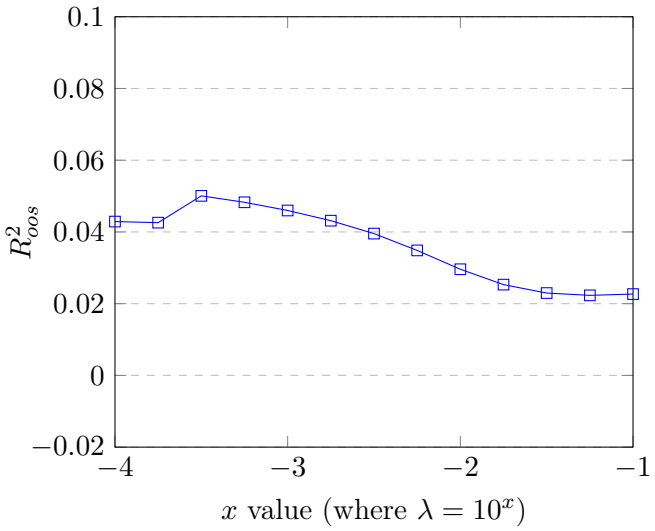
	Elastic Net	(Gu, Kelly, & Xiu, 2020)
R^2_{OOS}	0.00064	0.00110
MSPE	0.01767	NA
RMSE	0.13295	NA

It is important to note that (Gu, Kelly, & Xiu, 2020) use the Huber loss function whereas we use the sum of least squares loss function. Also, we use coordinate descent for fitting the model while (Gu, Kelly, & Xiu, 2020) use accelerated proximal gradient descent.

5.2 Ridge and Lasso

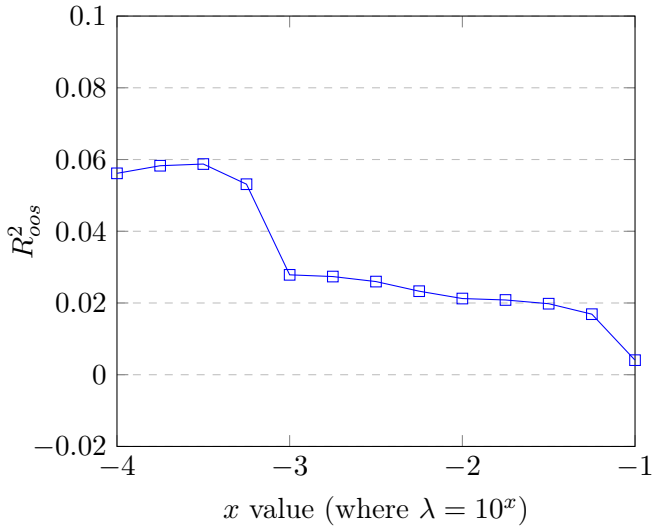
We now follow the same process that we have used for fitting the elastic net model to fit the ridge and lasso regression model. We obtain the following validation sample R^2_{OOS} values for each of the following λ values displayed in Figures 2 and 3.

Figure 2: R^2_{OOS} value for determining λ in the ridge model



It is clear from this graph that for the ridge model we should take $\lambda = 10^{-3.5}$ as our hyperparameter value since the R^2_{OOS} is the largest. Also, the MSPE and RMSE are the smallest for this λ (this isn't shown in Figure 2).

Figure 3: R^2_{OOS} value for determining λ in the lasso model



It is clear from this graph that for the lasso model we should take $\lambda = 10^{-3.5}$ as our hyperparameter value since the R^2_{OOS} is the largest. Also, the MSPE and RMSE are the smallest for this λ (this isn't shown in Figure 3).

Taking the λ values corresponding to the highest R^2_{OOS} in Figures 2 and 3 for ridge and lasso respectively to calculate the R^2_{OOS}, MSPE and RMSE for the testing sample gives us the following results displayed in Table 2.

Table 2: R^2_{OOS}, MSPE and RMSE values for the testing sample using ridge and lasso

	Ridge	Lasso
R^2_{OOS}	-0.02897	0.00238
MSPE	0.01820	0.01764
RMSE	0.13490	0.13283

The R^2_{OOS} for ridge is negative, which means that a 'straight line' is better at predicting the excess stock return than our ridge model. However, this is not the case for the lasso model since that R^2_{OOS} is positive.

5.3 Comparing the Three Models

We can compare the three machine learning models by looking at the comparative statistics that we calculate using the trained model on the testing sample. Table 3 summarizes the results that

we have obtained and presented in Tables 1 and 2. Along with the elastic net R^2_{OOS} found by (Gu, Kelly, & Xiu, 2020), we also include the λ values that are used for each of the models.

Table 3: R^2_{OOS}, MSPE and RMSE values for the three machine learning models

	(Gu, Kelly, & Xiu, 2020)	Ridge	Lasso	Elastic Net
λ	NA	$10^{-3.5}$	$10^{-3.5}$	$10^{-3.25}$
R^2_{OOS}	0.00110	-0.02897	0.00238	0.00064
MSPE	NA	0.01820	0.01764	0.01767
RMSE	NA	0.13490	0.13283	0.13295

Once again, it is important to note that (Gu, Kelly, & Xiu, 2020) use the Huber loss function whereas we use the sum of least squares loss function. Also, we use coordinate descent for fitting the model while (Gu, Kelly, & Xiu, 2020) use accelerated proximal gradient descent. Furthermore, (Gu, Kelly, & Xiu, 2020) don't present their λ value, and don't include the MSPE and RMSE statistics. We therefore denote these values as 'NA'.

We also look into the independent variables that are selected by each of the three machine learning models. As each of the three models has a different regularization penalty, it follows that the coefficients associated with the independent variables also differ from model to model. Tables 4 and 5 show the coefficients of the 94 stock-level predictive characteristics (that aren't multiplied by either one of the 8 macroeconomic predictors) for the elastic net model, ridge model and lasso model.

6 Conclusion

When looking at the comparative statistics in Table 3, we can draw a number of conclusions. The first is that we aren't able to replicate the R^2_{OOS} for the elastic net model that was presented in (Gu, Kelly, & Xiu, 2020) exactly. However, we aren't too far off. There are multiple reasons for this difference in R^2_{OOS} between our elastic net model and the elastic net model used in (Gu, Kelly, & Xiu, 2020).

The most important difference is that we choose to remove all observations that have one or more missing values from our data set. This leaves us with around 175000 usable observations instead of using 3 to 4 million observations with many missing values. As noted earlier, this also results in our data set starting in 1985 instead of 1957. The effect of this is that our training sample is much smaller, but we are able to partially combat this by increasing the training sample to 64% of our data set.

The second reason is that (Gu, Kelly, & Xiu, 2020) use the technique of recursively refitting

Table 4: Coefficients of the 94 stock-level predictive characteristics (1 of 2)

Characteristic Name	Elastic Net	Ridge	Lasso
mvel1	-3.50125E-10	-2.80944E-10	-3.49744E-10
beta	0	-0.00085	0
betasq	-0.00037	-0.00027	-0.00036
chmom	0.00119	0.00252	0.00097
dolvol	-0.00111	-0.00112	-0.00111
idiovol	0	-0.02401	0
indmom	-0.01219	-0.01708	-0.01172
mom1m	0	-0.01102	0
mom6m	-0.00101	-0.00459	-0.00072
mom12m	0	0.00357	0
mom36m	-0.00082	-0.00150	-0.00066
pricedelay	0.00007	0.00050	0.00003
turn	-0.00077	-0.00018	-0.00083
absacc	0	0.01678	0
acc	0	-0.02970	0
age	0.00004	0.00008	0.00004
agr	0.00724	0.00828	0.00696
bm	0	3.80940E-06	0
bm_ia	0.00002	0.00002	0.00002
cashdebt	-0.00045	0.00130	-0.00027
cashpr	-1.95505E-06	-1.19124E-06	-2.49607E-06
cfp	0.00095	0.00449	0
cfp_ia	0.00021	0.00036	0.00016
chatoia	0	0.00334	0
chcsho	0	0.00042	0
chempia	0	0.00207	0
chinv	0	-0.00938	0
chpmia	-0.00024	-0.00030	-0.00024
convind	0	-0.00151	0
currat	-0.00042	-0.00051	-0.00042
depr	0	0.00366	0
divi	0	-0.00508	0
divo	0	-0.00558	0
dy	0	-0.01808	0
egr	-0.00117	-0.00126	-0.00119
ep	0	-0.00132	0
gma	0	0.00360	0
grcapx	-0.00064	-0.00058	-0.00064
grltnoa	0	0.00151	0
herf	0	-0.00698	0
hire	0	0.00103	0
invest	0	0.00350	0
lev	-0.00070	-0.00075	-0.00062
lgr	0	-0.00018	0
mve_ia	2.26409E-07	2.32509E-07	2.25738E-07
operprof	0	0.00200	0
orgcap	0	-0.00638	0

Here we present the 94 stock-level coefficients of the predictive characteristics, that aren't multiplied by either one of the 8 macroeconomic predictors. As one can see, all characteristics that have a coefficient equal to 0 are not selected by that model. It is important to note that these coefficients are not standardized.

Table 5: Coefficients of the 94 stock-level predictive characteristics (2 of 2)

Characteristic Name	Elastic Net	Ridge	Lasso
pchcapx_ia	-4.65514E-06	-0.00001	-4.72954E-06
pchcurrat	0	0.00074	0
pchdepr	0	-0.00317	0
pchgm_pchsale	0	-0.00084	0
pchquick	0	0.00061	0
pchsale_pchinvt	0	-0.00289	0
pchsale_pchrect	0	-0.00020	0
pchsale_pchxsga	0	0.00264	0
pchsaleinv	0	0.00034	0
pctacc	-0.00021	-0.00003	-0.00021
ps	0	-0.00045	0
quick	0.00012	0.00024	0.00010
rd	0.00073	0.00288	0.00053
rd_mve	0	0.04045	0
rd_sale	0	-0.00242	0
realestate	0	-0.00039	0
roic	0	0.00034	0
salecash	-0.00001	-2.79924E-06	-0.00001
saleinv	0.00001	0.00002	0.00001
salerec	-0.00004	-0.00002	-0.00004
secured	0	-0.00191	0
securedind	0	-0.00081	0
sgr	0	0.00029	0
sin	0	0.00980	0
sp	0.00038	0.00039	0.00037
tang	0	-0.00255	0
tb	0.00061	0.00074	0.00058
aeavol	0.00050	0.00056	0.00048
cash	0	0.00886	0
chtx	0	0.01720	0
cinvest	0	0.00076	0
ear	0	0.02211	0
nincr	0.00025	0.00009	0.00018
roaq	0	0.00274	0
roavol	0	-0.02416	0
roeq	0	0.00183	0
rsup	0	0.00991	0
stdacc	0.00015	0.00013	0.00015
stdcf	-0.00007	-0.00006	-0.00007
ms	0.00026	-0.00044	0.00023
baspread	0	0.10764	0
ill	0	2.99998E-06	0
maxret	0	0.01225	0
retvol	0	0.00692	0
std_dolvol	-0.00280	-0.00706	-0.00253
std_turn	0.00040	0.00014	0.00040
zerotrade	-0.00104	-0.00122	-0.00104

Here we present the 94 stock-level coefficients of the predictive characteristics, that aren't multiplied by either one of the 8 macroeconomic predictors. As one can see, all characteristics that have a coefficient equal to 0 are not selected by that model. It is important to note that these coefficients are not standardized.

the model each year. This means that every time they refit, the training sample increases by one year and the size of the validation sample stays the same but moves up one year. We use a different technique without recursively refitting. The reason for this is because since we don't have that many observations, it doesn't make sense anymore to limit the training sample to save computation time. Therefore we choose to use a 64% / 16% / 20% split for the training sample, validation sample and testing sample respectively.

A third reason is that we use a regular least squares loss function, whereas (Gu, Kelly, & Xiu, 2020) use a Huber loss function. The Huber loss function might be better at fitting models on a data set that has large outliers.

The last reason is that we use coordinate descent for finding the optimal vector θ, whereas (Gu, Kelly, & Xiu, 2020) use accelerated proximal gradient descent.

When looking at our extension which covers introducing the ridge and lasso models and then comparing them to the elastic net model, we find that the ridge model performs more poorly than the elastic net model with a negative R^2_{OOS} value. The lasso model on the other hand performs a lot better, with an R^2_{OOS} that is more than three times as large as the R^2_{OOS} of our elastic net model and more than twice as large as the R^2_{OOS} of the elastic net model used in (Gu, Kelly, & Xiu, 2020). These findings are in line with the expectations that are presented in (Melkumova & Shatskikh, 2017). They note that lasso can set parameter coefficients equal to 0 whereas ridge can only set parameter coefficients close to 0, but not actually equal to 0. Therefore they expect that lasso will outperform ridge when dealing with a data set that has a large number of independent variables. In our case we have 907 independent variables, which is quite a large number. Therefore setting parameter coefficients equal to 0 greatly benefits the predictive power of the model.

If we look at the coefficients of the stock-level predictive characteristics for the three models in Tables 4 and 5, we can draw a number of conclusions. First of all, as we expected the coefficients in the ridge model are all equal to a non-zero number. This is because of the regularization penalty of the ridge model as we explained earlier. On the other hand, the elastic net model and lasso model have many coefficients equal to 0. For these two models more than half of the stock-level predictive characteristics have coefficients that are equal to 0. The reason for this is that these stock-level predictive characteristics are highly correlated, which results in many of them to not be selected or in other words having a coefficient equal to 0. Because of this, the R^2_{OOS} of the elastic net model and lasso model is a lot higher than the R^2_{OOS} of the ridge model. Furthermore, it is interesting to note that there is only 1 stock-level predictive characteristic that has a coefficient equal to 0 in the

lasso model but a coefficient not equal to 0 in the elastic net model, namely 'cfp'.

We can now conclude that the lasso regression model outperforms both the elastic net model and the ridge model when predicting excess stock return. The ridge model is by far the worst model out of the three at predicting excess stock return since it's R^2_{OOS} was negative.

7 Discussion

Looking back at this research, there are a few things that we could have done in a different manner. We have chosen to remove all of the observations that had missing values for one or more of the variables. This left us with approximately 175000 usable observations out of the total 3 to 4 million observations that were used in (Gu, Kelly, & Xiu, 2020). Therefore our model wasn't trained as well as it could have been if we decided to also use the observations that had one or more missing values.

For the code used in this research, we ran into the problem that all three of our machine learning models didn't converge even after increasing the maximum amount of iterations a number of times. Therefore we increased the tolerance until all three of our machine learning models were able to converge.

8 Recommendations

For future research, we have a number of recommendations. First of all, it would be useful to see what the predictive power of the model is when using k-fold cross validation. We have made the conscious decision to not use cross validation as (Gu, Kelly, & Xiu, 2020) don't use it and we are trying to replicate their results. But it would still be of great interest to see what the effect of that is.

Another thing is that we use an α value equal to 0.5 for the elastic net model. It would be interesting to see what different values of α could do in terms of predictive power.

9 References

Shihao Gu, Bryan Kelly, Dacheng Xiu, Empirical Asset Pricing via Machine Learning, The Review of Financial Studies, Volume 33, Issue 5, May 2020, Pages 2223–2273.

Ivo Welch, Amit Goyal, A Comprehensive Look at The Empirical Performance of Equity Premium Prediction, The Review of Financial Studies, Volume 21, Issue 4, July 2008, Pages 1455–1508.

Melkumova, L., amp; Shatskikh, S. (2017, September 27). Comparing ridge and LASSO estimators for data analysis.

10 Appendix 1: Python Code

CODE FILE 1

Here we extract the values of the comparative statistics for the range of different λ values for each of the three machine learning models.

```
import numpy as np
import pandas as pd
from sklearn.linear_model import ElasticNet

data_thesis = pd.read_csv('COMPLETE_DATA_THESIS_Igor_Lipper.csv')

#Splitting the data into Train, Validation and Test samples
#via a 64%, 16%, 20% split respectively.
X_train = data_thesis.iloc[0:110986,4:910].values
X_validation = data_thesis.iloc[110986:138732,4:910].values
X_test = data_thesis.iloc[138732:173416,4:910].values
y_train = data_thesis.iloc[0:110986,3].values
y_validation = data_thesis.iloc[110986:138732,3].values
y_test = data_thesis.iloc[138732:173416,3].values

#Choosing all the lambda (in python they are called aplhpa) values
alphas = [pow(10,-4), pow(10,-3.75), pow(10,-3.5), pow(10,-3.25), pow(10,-3),
pow(10,-2.75), pow(10,-2.5), pow(10,-2.25), pow(10,-2), pow(10,-1.75), pow(10,-1.5),
pow(10,-1.25), pow(10,-1)]


#Extracting the out of sample R-squared, MPSE and RMSE for each of the
```

```
#different lambda (here alpha) values.

#Ridge
for a in alphas:
    model = ElasticNet(alpha=a, l1_ratio=0, tol=1).fit(X_train,y_train)
    y_validation_predict = model.predict(X_validation)
    r_2_validation = 1-(sum(pow(y_validation-y_validation_predict,2))/
        sum(pow(y_validation,2)))
    mspe = (1/len(y_validation))*sum(pow(y_validation-y_validation_predict,2))
    print("Ridge Alpha:{0:.4f}, R2:{1:.5f}, MSPE:{2:.5f}, RMSE:{3:.5f}"
       .format(a, r_2_validation, mspe, np.sqrt(mspe)))

#Lasso
for a in alphas:
    model = ElasticNet(alpha=a, l1_ratio=1, tol=1).fit(X_train,y_train)
    y_validation_predict = model.predict(X_validation)
    r_2_validation = 1-(sum(pow(y_validation-y_validation_predict,2))/
        sum(pow(y_validation,2)))
    mspe = (1/len(y_validation))*sum(pow(y_validation-y_validation_predict,2))
    print("Lasso Alpha:{0:.4f}, R2:{1:.5f}, MSPE:{2:.5f}, RMSE:{3:.5f}"
       .format(a, r_2_validation, mspe, np.sqrt(mspe)))

#Elastic Net
for a in alphas:
    model = ElasticNet(alpha=a, l1_ratio=0.5, tol=1).fit(X_train,y_train)
    y_validation_predict = model.predict(X_validation)
    r_2_validation = 1-(sum(pow(y_validation-y_validation_predict,2))/
        sum(pow(y_validation,2)))
    mspe = (1/len(y_validation))*sum(pow(y_validation-y_validation_predict,2))
    print("Elastic Net Alpha:{0:.4f}, R2:{1:.5f}, MSPE:{2:.5f}, RMSE:{3:.5f}"
       .format(a, r_2_validation, mspe, np.sqrt(mspe)))
```

CODE FILE 2

Here we use the optimal λ values found in CODE FILE 1 to calculate the values of the comparative statistics for the testing sample for each of the three machine learning models.

```
import numpy as np
import pandas as pd
from sklearn.linear_model import ElasticNet


data_thesis = pd.read_csv('COMPLETE_DATA_THESIS_Igor_Lipper.csv')

#Splitting the data into Train, Validation and Test samples via
#a 64%, 16%, 20% split respectively.
X_train = data_thesis.iloc[0:110986,4:910].values
X_validation = data_thesis.iloc[110986:138732,4:910].values
X_test = data_thesis.iloc[138732:173416,4:910].values
y_train = data_thesis.iloc[0:110986,3].values
y_validation = data_thesis.iloc[110986:138732,3].values
y_test = data_thesis.iloc[138732:173416,3].values

#Exporting the names of the independent variables.
names = data_thesis.drop(['Unnamed: 0', 'date', 'permno', 'ExcessRet'],axis=1).columns
column_names = pd.DataFrame(names)
column_names.to_csv('column_names.csv',index=False)

#Ridge
model = ElasticNet(alpha=pow(10,-3.5), l1_ratio=0, tol=1,
    max_iter=10000).fit(X_train,y_train)
y_test_predict = model.predict(X_test)
r_2_test = 1-(sum(pow(y_test-y_test_predict,2))/sum(pow(y_test,2)))
mspe = (1/len(y_test))*sum(pow(y_test-y_test_predict,2))
print("Ridge R2:{0:.5f}, MSPE:{1:.5f}, RMSE:{2:.5f}".format(r_2_test,
    mspe, np.sqrt(mspe)))
```

```
#Exporting the coefficients of the Ridge regression.
ridge_coef = model.coef_
ridge_coef_values = pd.DataFrame(ridge_coef)
ridge_coef_values.to_csv('ridge_coef_values.csv',index=False)

#Lasso
model = ElasticNet(alpha=pow(10,-3.5), l1_ratio=1, tol=1,
    max_iter=10000).fit(X_train,y_train)
y_test_predict = model.predict(X_test)
r_2_test = 1-(sum(pow(y_test-y_test_predict,2))/sum(pow(y_test,2)))
mspe = (1/len(y_test))*sum(pow(y_test-y_test_predict,2))
print("Lasso R2:{0:.5f}, MSPE:{1:.5f}, RMSE:{2:.5f}".format(r_2_test,
    mspe, np.sqrt(mspe)))
#Exporting the coefficients of the Lasso regression.
lasso_coef = model.coef_
lasso_coef_values = pd.DataFrame(lasso_coef)
lasso_coef_values.to_csv('lasso_coef_values.csv',index=False)

#Elastic Net
model = ElasticNet(alpha=pow(10,-3.25), l1_ratio=0.5, tol=1,
    max_iter=10000).fit(X_train,y_train)
y_test_predict = model.predict(X_test)
r_2_test = 1-(sum(pow(y_test-y_test_predict,2))/sum(pow(y_test,2)))
mspe = (1/len(y_test))*sum(pow(y_test-y_test_predict,2))
print("Elastic Net R2:{0:.5f}, MSPE:{1:.5f}, RMSE:{2:.5f}".format(r_2_test,
    mspe, np.sqrt(mspe)))
#Exporting the coefficients of the Elastic Net regression.
elasticnet_coef = model.coef_
elasticnet_coef_values = pd.DataFrame(elasticnet_coef)
elasticnet_coef_values.to_csv('elasticnet_coef_values.csv',index=False)
```