COMP0050

Machine Learning with Applications in Finance

Coursework

Question 1

A Comparison of Peer-to-Peer Loan Default Prediction Models

Question 2

Regularization Techniques for the Global Minimum Variance Portfolio

---

Date:

April 2022

Question 1

A Comparison of Peer-to-Peer Loan Default Prediction Models

In this question we construct three different models for predicting defaults on a dataset of Peer-to-Peer loans: Logistic Regression, Support Vector Machines (SVM), and Decision Tree Classifiers. We perform 5-fold cross-validation for each model to ensure robustness and then compare their performance using traditional metrics.

This question is structured as follows: section one provides a brief introduction; section two discusses the data pre-processing, choices and assumptions made, and some of the details of the models themselves; section three presents the results; section four provides a discussion; and section five closes with concluding remarks and opportunities for further work.

1. Introduction

One can broadly define Peer-to-Peer (P2P) lending as the extension of credit to individuals or businesses through non-traditional means. By non-traditional we mean that, in general, lenders are not high-street financial institutions such as banks or building societies and most transactions occur online in a bid to reduce operating expenses.

One aspect remains central to the P2P industry though: Credit Risk. More specifically, companies are concerned about default risk. Reliable predictors for the likelihood of default are critical for businesses to survive. In this question we explored and evaluated several machine learning techniques for the prediction of defaults based on information typically provided during the application process.

2. Methodology

2.1 Data & Pre-processing

We were provided with a dataset containing loan information for the company Lending Club [1]. This held approximately 76,000 records with 21 features. To correctly apply our proposed models, we needed to pre-process the data.

Firstly, we considered which features to keep. Whilst all features could potentially have some bearing on whether an applicant is likely to default on their loan, we chose to take a marginally smaller subset of features to simplify our problem. The features we removed were: (*i*) Verification Status (*verification_status*), this generalised our results since the level and approach to verification may vary across the industry; (*ii*) Issue Date (*issue_d*), to simplify our problem and remove the time-series component. It would be interesting to explore whether certain time periods (e.g. pre-/post-financial crisis) give rise to structural changes in the likelihood of default, however, we do not consider that in this question. (*iii*) Earliest Credit Line on record (*earliest_credit_line*), as this information was grouped by year and was considered too general to be useful for our analysis; and (*iv*) number of Mortgage Credit Lines on record (*mort_acc*), as we felt that this information provided little additional value above that captured by the feature, Total Number of Credit Lines on record (*total_acc*). This left a total of 17 features.

We examined the number of missing values and found they existed in three features: Employment Length (*emp_length*); Revolving Utility of Credit Lines (*revol_util*); and Public Record Bankruptcies (*pub_rec_bankruptcies*). To rectify this, we used mean imputation for the first two features, however, we simply removed the missing values for the third feature as this was a highly unbalanced variable and mean imputation was inappropriate. In addition, it only accounted for a small fraction of the data (< 0.2%).

To prepare the inputs to the models, we standardised our numerical features. For the categorical features we used the One Hot Encoding [2] technique to transform the data. This involved expanding each categorical feature into its own binary indicator variable. An example of which is provided in Figure 1.

Application Type	**Application Type: *Individual***	**Application Type: *Joint***
Individual	1	0
Joint	0	1

Fig 1: An example of the transformation of categorical features using the One Hot Encoding technique. We can see we have moved from categorical to numerical data, at the expense of introducing new pseudo-features.

We deemed this approach superior to the simpler ordinal encoding technique as it does not introduce spurious ordering to our features which could disrupt the results of our models.

Finally, as we were working with predictive models, we needed to take care with the unbalanced nature of the dataset. In general, most people will not default on their loans, hence, we needed to account for this fact during the training of our models. We chose to undersample the majority class of the data to create a balanced dataset. This meant our final dataset, after pre-processing, contained ~27,500 records. Our feature of interest (*charged_off*) is binary: those that defaulted (indicated by 1) and those that did not (indicated by 0).

2.2 Logistic Regression [2]

We believed Logistic Regression was an appropriate method for consideration given the binary nature of our intended predictions: default versus no default. Logistic Regression relies on the Logistic Function,

$$f(\boldsymbol{\theta}) = \frac{1}{1+exp(-\boldsymbol{\beta}\cdot\boldsymbol{\theta})}.$$

This function (known as the Sigmoid function) produces a classification result in terms of probabilities, based on the n inputs from the feature space. Hence, the output is a value in the interval [0,1]. This means that we can assign an appropriate decision threshold, say 0.5, and classify all points lying above or below that threshold into their respective binary categories.

For our implementation we have used the Scikit-Learn package available in Python. As Question 2 focusses on regularization techniques, whereas the emphasis of this question is on comparison between models, we have removed the *LogisticRegression*(.) function's automatic regularization.

2.3 Support Vector Machines [3]

Support Vector Machines are a similar classification method and as such presented another appropriate choice for comparison. The SVM method seeks to find the optimal hyperplane that separates our binary output variable.

We can consider SVMs as an optimisation problem that seeks to maximise the distance (referred to as the margin) between the hyperplane and those output points that lie closest to the hyperplane (referred to as support vectors). We can extend SVMs to better account for non-separable data by introducing a soft-margin, that is accepting some points will inevitably be mis-classified. However, the true power of SVMs lies in its flexibility regarding the shape of the hyperplane. We can introduce a non-linear map (referred to as the kernel) which takes elements from our feature space, transforms them into a higher dimensional space, finds the optimal hyperplane there, then transforms back to our original space to improve the classification.

We have used the Scikit-Learn package available in Python for our implementation, during which we tested each of the four standard kernels provided. We found that the Gaussian Radial Basis Function performed best and, as this question is a comparison between different models, we only present these scores in our results section.

2.4 Decision Tree Classifier [2]

We also chose to consider Decision Tree Classifiers because they are easily interpretable and can be explained to non-technical audiences. They create a tree like structure starting from an initial or 'root' node before progressing through several layers. At each layer there are additional nodes which dictate a decision and split the data further eventually leading to classification.

Despite their simplicity Decision Tree Classifiers are susceptible to overfitting, that is, we can continuously split our data until we have an almost perfectly accurate tree. To address this problem, we performed a grid search on the maximum depth of the tree using the average AUC score of the test set as our indicator. We found that a maximum depth of 7 produced the best results and display the associated metrics in our results. Figure 2 shows the outcome of our grid search.

Once again, we have used the Scikit-Learn package available in Python, and we have specified the use of entropy (the information gain) as the parameter with which to assess our trees.

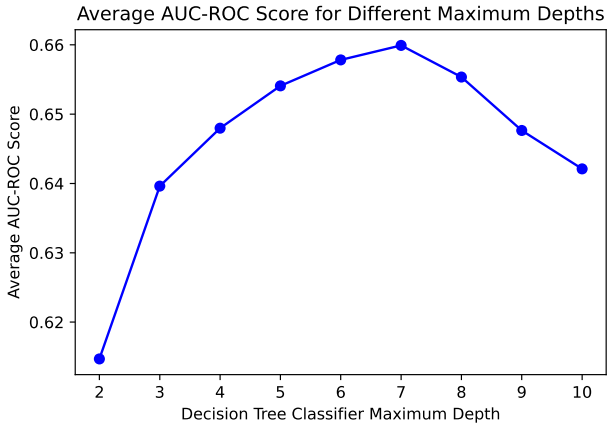

Fig 2: Graph showing the average AUC-ROC score for the grid search maximum depth of the cross-validated Decision Tree Classifier.

2.5 Cross-validation

To ensure robustness in our results we have used 5-fold cross-validation for all our models. This involves splitting the training and testing sets five different times, with each iteration using a different subset of the data (20% in the case of 5-fold cross-validation) as the test set.

3. Results

3.1 Interpretation of Results

The results of our models are displayed in Table 1. We present four commonly used metrics to evaluate our results: AUC-ROC Score; Accuracy; Precision; and Recall.

To interpret these metrics, we must first understand the confusion matrix, which shows a comparison of the actual outcome values to those predicted by our models. Figure 3 displays the definition of a confusion matrix. Whilst Figures 4–6 display the actual results for each model.

	Predicted: No Default	***Predicted***: Default
Actual: No Default	η_{00}	η_{01}
Actual: Default	η_{10}	η_{11}

Fig 3: The definition of a confusion matrix. We computed the confusion matrix for each model and calculated the metrics: Accuracy, Precision, and Recall. The results are displayed in Table 1.

The ROC curve plots a line with the False Positive Rate (FPR) on the x-axis against the True Positive Rate (TPR) on the y-axis. Where we define FPR and TPR as,

$$FPR = \frac{\eta_{01}}{\eta_{00} + \eta_{01}}$$

$$TPR = \frac{\eta_{11}}{\eta_{10} + \eta_{11}}.$$

If our model perfectly predicted all outcomes, we would see a vertical line traverse the y-axis, and a horizontal line extend from the point $y = 1$. In a purely random model, we expect the ROC curve to look like the line $y = x$. Using this ROC curve, we can compute the AUC Score which calculates the area under the ROC curve. Again, a perfect model would have an AUC score of 1, whilst a random model would have a score of 0.5. We used the random AUC score of 0.5 as a benchmark to evaluate the predictive power of our models.

We define the remaining three metrics as,

$$Accuracy = \frac{\eta_{00} + \eta_{11}}{\eta_{00} + \eta_{01} + \eta_{10} + \eta_{11}}$$

$$Precision = \frac{\eta_{11}}{\eta_{01} + \eta_{11}}$$

$$Recall = \frac{\eta_{11}}{\eta_{10} + \eta_{11}}.$$

We can interpret the accuracy as the number of datapoints our models correctly classified, that is, did our model predict defaults where a default truly occurred and likewise for no defaults. Precision can be considered as the proportion of those that truly defaulted versus the number our models predicted would default. Finally, we can interpret recall as those that we correctly predicted would default out of the total number that did indeed default. In all cases a score closer to 1 indicates theoretically better performance.

3.2 Presentation of Results

The first thing we notice is that our SVM performance was superior across all metrics. This was likely due to the non-linear (Radial Basis Function) kernel used in the SVM model, whereas Logistic Regression relied on a linear separation.

The AUC scores for all models were in the mid-to-high sixties, clearly outperforming our random model benchmark and indicating that each of them does indeed have some degree of predictive power. The scores for Logistic Regression and SVM were similar at approximately 69%, a 3% improvement over the simpler Decision Tree Classifier approach.

Similarly, the accuracy scores were well above our random model benchmark, with the SVM model producing an average score of 65.2%. This means that the model correctly classified the data into the two binary cases: default and no default an average of 65.2% of the time across the five test sets.

It is worth noting that we can be confident that this is a true reflection of the accuracy of the models because of the initial data pre-processing. If we had simply performed the analysis on the original unbalanced dataset our results could have been distorted by the significant presence of the majority class: no default.

The precision scores show a much narrower interval of values – all of which fall within a 1% range. This is an interesting outcome as it shows that a consistent number of cases are being predicted as default regardless of the model used. This can be seen by the similar numbers in the right-hand column of each of our confusion matrices (Figures 4–6).

Finally, our SVM model produced a recall score of 65.1%, meaning that of those cases that truly defaulted in the test set, our model correctly predicted 65.1% of them.

4. Discussion

Given the business context of the question, we propose the recall score is a more appropriate metric for evaluation than precision. In the P2P loan industry, default risk poses a significant concern. This means a model predicting no default when there was indeed a default is more problematic than predicting default in those that did not. The former could cause significant losses on issued loans, whereas the latter would simply reduce the number of loans issued in the first place.

This point, however, proves inconsequential in our analysis as the SVM comparatively outperforms across all metrics, thus we can infer that it is the best model of those considered.

Unfortunately, though, this superior relative performance does not imply good absolute performance. The results for all three models are disappointing. The best recall score of 65.1% still means that our model incorrectly classified over a third of cases, that is we predicted more than a third of cases would not default when they in fact did. One could certainly conclude that a business with such a loose approach to lending would not be solvent for long.

Model	AUC Score	Accuracy	Precision	Recall
Logistic Regression	68.9%	63.9%	64.2%	63.7%
Support Vector Machine	69.2%	65.2%	64.7%	65.1%
Decision Tree Classifier	66.0%	63.3%	63.7%	62.7%

Table 1: The performance metrics associated with each of our three proposed models: Logistic Regression, SVM, and Decision Tree Classifier. The results displayed are the averages obtained from 5-fold cross-validation.

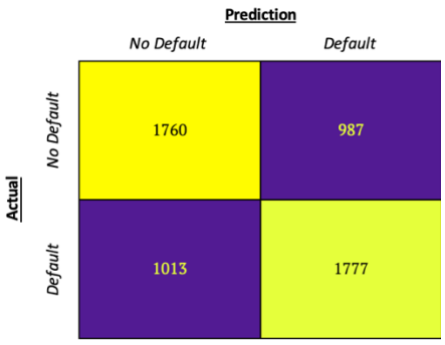

Fig 4: Logistic Regression Average Confusion Matrix

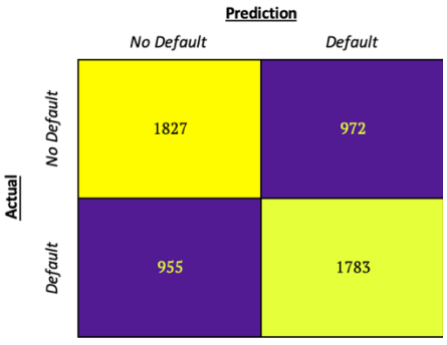

Fig 5: SVM Average Confusion Matrix

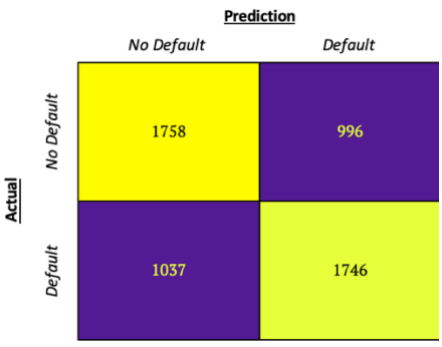

Fig 6: Decision Tree Average Confusion Matrix

There could be several reasons why these results are underwhelming. Firstly, as we might expect, loan data is inherently messy. Whilst the information collected during the application process and already in existence on an applicant's credit file will go some way towards indicating the likelihood of default, there will always be a stochastic element. This reflects the unique circumstances that every applicant will face and that cannot be captured during an application. For instance, a company may collapse during the term of the loan, leaving an employee that was previously unlikely to default unable to service their repayments. Of course, macro or economic factors are likely to be forecast elsewhere in the company and influence things such as their risk appetite.

Secondly, during our pre-processing we removed the time-series element of the data. We noted in Section 2 that this simplified our problem, but there are underlying consequences of this decision. By removing the time-series aspect we remove the ability to examine whether there is a credit environment regime change, by which we mean there could be two (or more) discrete periods within which predicting default could be substantially different. The dataset provides information for the range 2007–2017, hence, we could perhaps see different performances during the financial crisis between 2007–2010, and a relatively calmer period between 2011–2014.

Finally, the processed dataset used in our analysis is rather small at only 27,500 records. This allowed us to address the unbalanced nature and avoid misleading results but had the disadvantage of significantly reducing the volume available to train and test our models.

5. Conclusion

In this question we have compared the performance of three different methods for the prediction of P2P loan defaults. We can conclude that our models do indicate predictive power – with our SVM model being deemed superior – but sole reliance on such models would be inappropriate. In a business context though, as is so often the case with Machine Learning, the use of such models in conjunction with human oversight could bring additional efficiency, robustness, and a data-driven approach to loan applications.

We have also discussed some of the drawbacks that were encountered during our analysis, none of which are unsolvable. A natural extension of this work would be to develop some of these ideas further. For instance, one could use alternative techniques for the data pre-processing such as weighting the majority and minority classes to address the unbalanced nature of the data and use this to train and test the models. One could perform a deeper analysis of the Decision Tree Classifier model and consider whether the use of Forests improves performance. In addition, it would be interesting to perform detailed feature importance analysis to establish if there is a subset of features that drives predictions – this could then be utilised to improve the application process and remove unnecessary information capture. Alternatively, one could consider more advanced machine learning techniques such as Neural-Networks and perform further comparisons with our work.

Question 2

Regularization Techniques for the Global Minimum Variance Portfolio

In this question we examine the global minimum variance portfolio using two different regularization methods: Ridge (ℓ_2) and LASSO[1] (ℓ_1). We use cross-validation methods to ensure robustness of the results and to identify the optimal parameters associated with each penalty term. We then compare the resulting variances and portfolio composition for each of the regularization techniques. In addition, we consider the variance of an evenly weighted portfolio for each of the test sets. Finally, we discuss what this means in a financial context.

This report is structured as follows: section one provides a brief introduction; section two covers the methodology and approach; section three presents the results; section four provides a discussion; and section five closes with concluding remarks and opportunities for further work.

1. Introduction

In modern day finance risk plays a critical role. A clear understanding of the exposure relating to an individual asset or portfolio allows organisations to make informed decisions. There are a numerous metrics that can be calculated to provide insight, each with their own strengths and weaknesses. One of the most fundamental metrics is standard deviation, used synonymously with volatility. As a result, its square, or variance, is also a measure of risk and is the focus of this question.

The Markowitz portfolio optimisation problem was first introduced in the early-1950's [4]. It concerns finding the optimal weights of a portfolio of assets that minimises its variance. We focused on the simplified case, only considering the budget constraint,

$$\min_{\boldsymbol{w}}\{\boldsymbol{w}^T\,\boldsymbol{C}\,\boldsymbol{w}\} \quad s.t. \quad \sum_{i=1}^{p} w_i = 1.$$

Where $\boldsymbol{w}$ is a vector of portfolio weights, $\boldsymbol{C}$ the covariance matrix, and p the number of assets in the portfolio. In general, the true covariance matrix is not known, hence, we were required to use the empirical covariance matrix computed from the dataset.

2. Methodology

2.1 Data

The dataset contained returns for fifty (unknown) stocks from the FTSE100 index. There were 500 rows of data corresponding to a time series of daily returns. It is well known that a trading year comprises of 252 observations, hence, for the purposes of this report we treated the dataset as essentially two years' worth of returns for the specified assets. This allowed us to split the data into natural financial groups such as months (consisting of ~21 days) and quarters (consisting of ~63 days).

To improve the robustness of the outputs, traditional machine learning techniques utilise cross-validation. As opposed to simply splitting the data into training and test sets once, cross-validation performs this separation multiple times and generates multiple models using each of these pairs. One popular approach is k-fold cross-validation which performs the train/test split a fixed number, k, times with each iteration using a different subset to train and test the model.

Unfortunately, however, this technique does not work with time-series data. Randomly selecting different subsets of the data has little meaning if we used next year's data to estimate last year's portfolio variance. For our financial time-series, we were forced to take an alternative approach to validation. In this report we used a sequential approach, that is iteratively moving the testing set forward and increasing the training set as the amount of 'historical' data increases.

As mentioned, our dataset contains approximately two years of data. Considering the number of assets, p, is relatively large, we did not want our initial training set to be too small. Similarly, we did not want each test set to be too small. As a result, we chose to split the dataset into blocks that represented quarters, with the first training window composed of the first two quarters, making the first half-year. This is summarised in Table 1.

Training Set		Test Set	
1	Q1 – Q2	1	Q3
2	Q1 – Q3	2	Q4
3	Q1 – Q4	3	Q5
4	Q1 – Q5	4	Q6
5	Q1 – Q6	5	Q7
6	Q1 – Q7	6	Q8

Table 1: The sequential train/test split used. It is described by the individual quarters over the two-year time horizon.

[1] Least Absolute Shrinkage and Selection Operator

This approach had several benefits. Firstly, it increased the robustness of the outputs by performing an appropriate amount of validation without becoming burdensome or time-consuming, whilst each set contained enough datapoints not to introduce a conflict with the number of assets being considered. In addition, it made intuitive sense and could easily be interpreted. For each regularization technique, we can imagine a scenario in which our portfolio required quarterly rebalancing to the global minimum variance portfolio. Naturally, one would want to use the historical data available to make that decision, and that is precisely what we achieved.

2.2 Regularization

As with most models, finding the right balance between over and underfitting was difficult. Overfitting may lead to excellent performance in the training dataset, we may even be able to obtain a riskless or substantially low level of portfolio variance, but this is likely to be followed by high out-of-sample variance when the model is applied to the test set. Whereas underfitting may lead to overly simple models that do not capture some of the context specific, and important, information.

Regularization is a technique that aims to strike a balance between these trade-offs by introducing a penalty term. There are plenty of proposed penalty terms discussed in the literature [5] with varying levels of complexity and convexity – with the latter impacting the efficiency and our ability to solve. We explored two of the simpler cases: Ridge and LASSO regularization.

2.2.1 Ridge Regularization [5]

The concept of Ridge regularization is to use the ℓ_2 norm as a penalty for the objective function. We can modify the minimisation problem from Section 1 to introduce the penalty term.

$$\min_{\boldsymbol{w}} \left\{ \boldsymbol{w}^T \boldsymbol{C}\, \boldsymbol{w} + \lambda \sum_i w_i^2 \right\} \quad s.t. \quad \sum_{i=1}^{p} w_i = 1.$$

Where we can rewrite the objective function as,

$$\{\boldsymbol{w}^T (\boldsymbol{C} + \lambda \boldsymbol{I})\, \boldsymbol{w}\}.$$

This form allowed us to find an analytical solution, just as with the traditional minimisation problem.

In this setting λ is our regularization parameter which determines the strength of the penalty. This led to a natural question: What is the most appropriate value for lambda? Whilst, this cannot be observed directly, it can be found using validation techniques. To compute the optimal value of lambda we used a grid search method over the range $[0,10]$ with step sizes $\alpha = 0.01$. We then solved the previous equation using the different values of lambda, computed the minimum variance and subsequently identified the best value of lambda.

2.2.2 LASSO Regularization [5]

The concept of LASSO regularization is analogous with that of Ridge regularization but using the ℓ_1 norm as the penalty term,

$$\min_{\boldsymbol{w}} \left\{ \boldsymbol{w}^T \boldsymbol{C}\, \boldsymbol{w} + \lambda \sum_i |w_i| \right\} \quad s.t. \quad \sum_{i=1}^{p} w_i = 1.$$

Unfortunately, with the introduction of the absolute value we no longer had a closed form solution of the minimum. This meant we were required to use optimisation software. To minimise our objective function, we used the *fmincon(.)* function provided by MATLAB. This is a constrained optimisation solver that allows for the input of inequality, equality, and boundary constraints. In our setting we only required one equality constraint, outlined in the introduction, which corresponds to the budget constraint.

Once again, the question arose regarding the best value for lambda. We used the same grid search method as our Ridge implementation, with the domain $[0,10]$, but with larger step sizes $\alpha = 0.1$, to reduce the computational effort required.

As we will see, the portfolio suggested by LASSO is sparse. To account for the tolerances within the optimisation software, we took the decision to set any positions strictly less than 1% to zero. This marginally impacted the total sum of the weights, which must sum to one. For all test sets the implementation of our condition produced weights equal to 1 ± 0.02, which was sufficiently close for our purposes.

2.3 Evenly weighted portfolio

To benchmark our regularization approaches we also considered the evenly weighted portfolio. That is, a portfolio equally weighted in all constituent assets. In our setting the weights were each set at *2%*. We computed the variance of this portfolio for each test set and examined it alongside our other results.

3. Results

The main results of our analysis are presented in Tables 2–3. We also include Figures 1–2 and 4–5, which show the in-sample and out-of-sample risk for varying lambdas for *one* train and test set for both Ridge and LASSO regularization as an illustration. Each training and test set has its own associated graphs (available upon request). We also include Figures 3 and 6, which give an example of the distribution of portfolio weights for both Ridge and LASSO regularization.

Test Set	λ_{opt}	Minimum Variance σ^2	Evenly weighted Variance
1	2.95	0.3187	0.4413
2	0.44	0.7636	1.2502
3	0.38	0.3322	0.7714
4	2.64	0.1811	0.4025
5	0.00	0.4231	0.7343
6	6.31	0.5468	0.6786
Avg. λ_{opt}	2.12		
Avg. Minimum Variance		0.4276	0.7131

Table 2: Empirical results for **Ridge** regularization. λ_{opt} was computed using grid search. Minimum and evenly weighted variances were computed using empirical covariance and minimised portfolio weights.

Test Set	λ_{opt}	Minimum Variance σ^2	Evenly weighted Variance
1	2.30	0.2884	0.4413
2	0.50	0.7551	1.2502
3	0.20	0.3532	0.7714
4	7.70	0.1872	0.4025
5	0.00	0.4231	0.7343
6	9.80	0.6375	0.6786
Avg. λ_{opt}	3.42		
Avg. Minimum Variance		0.4407	0.7131

Table 3: Empirical results for **LASSO** regularization. Computational methods as described in Table 2.

Firstly, we can see that whilst the exact values of the lambdas are slightly different for the Ridge and LASSO regularization – indeed, for test sets 4 and 6 the values are noticeably higher for LASSO – the profile of the lambdas across the test sets 1–6 is almost identical.

We cannot infer much information from this in the context of the minimum variance portfolio, as each test set clearly produces different variances depending on factors present in that specific time window. However, it does provide us with some confidence that our two methods behave in similar ways and reduces the chance we incorrectly implemented either method.

Furthermore, we notice from Tables 2 and 3, that our average minimum variance across all the test sets is similar at approximately 0.43 and 0.44 respectively. Again, this provides us with confidence our results are, at least to some degree, valid.

One clear outcome from our results is that both the Ridge and LASSO regularized portfolios are far superior to the evenly weighted portfolio. This is obvious as the variance returned by the equally weighted portfolio is greater in every test set and the overall average.

Using Figures 1–2 and 4–5 we can explore the relationship between lambdas and in- and out-of-sample risk. The x-axis, lambda, indicates the level to which our portfolio construction has been constrained, with smaller values for lambda indicating less constrained. As we would expect the optimum performance in-sample is with no regularization. This follows as we can always expect to find a model that fits the training data well (if not perfectly). Consequently, in Figures 1 and 4, we see a monotonically increasing line for the in-sample risk. This, however, is not useful in practice and so we must examine Figures 2 and 5 showing the out-of-sample risk. Here we see steep descent followed by a slow ascent for increasing lambda, indicating there is a minimum at the specific lambda listed in our results.

It is worth noting that our closed form solution for Ridge regularization is what produces the smooth curve for both in- and out-of-sample risks. Whereas the jagged lines for LASSO follow because we have used optimisation software.

4. Discussion

Let us first discuss the superior results of our regularization methods over and above the evenly weighted portfolio. This may appear a straightforward benefit but does, in fact, have significant meaning.

An equally weighted portfolio (without good cause) is referred to as 'naïve diversification' [6] and is a problem that is seen quite frequently in the financial world. It is often seen in Defined Contribution Pension Plans, in which unsophisticated investors decide to place an equal amount of their capital in all funds without understanding the constituent components of each fund and how they interact – thereby diversifying, or more often, magnifying their risk exposure. Our results show that there is a significant benefit to introducing regularization into the asset allocation process.

Another intriguing outcome of our analysis is the high variance in test set 2 regardless of the method. The minimum variance achieved is, in some cases, two or three times that of other test sets. We can speculate that two interlinked elements were likely the cause of this: (1) there was a period of high volatility in Q4 of our dataset; and (2) the portfolio construction, trained using the previous three quarters, was ill-equipped to handle such volatility.

We now turn our attention to the optimum portfolio weights indicated by the optimal value of our regularization parameter, lambda. A new set of weights will be proposed for each test set, so we display the weights for test set 4 for illustrative purposes. There are two important differences between the weights proposed by Ridge regularization and those of LASSO.

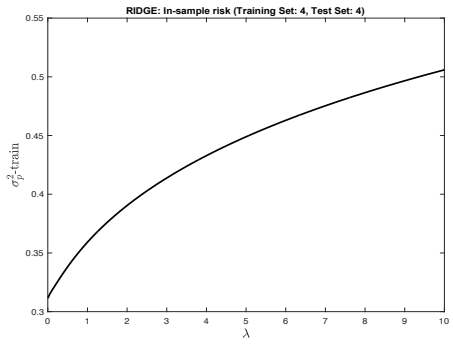

Fig 1: **Ridge** – In-sample risk for test set 4. The x-axis shows the regularization parameter lambda. Higher values of lambda indicate greater constraint.

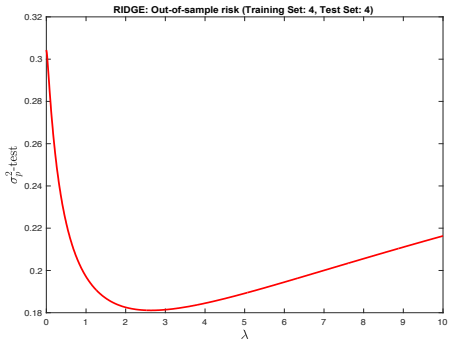

Fig 2: **Ridge** – Out-of-sample risk for test set 4. Each test set produces a similar graph. This test set is shown for illustrative purposes.

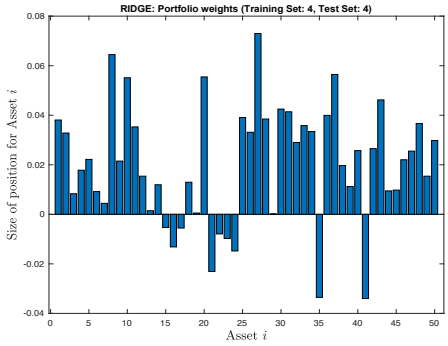

Fig 3: **Ridge** – Illustration of the optimal portfolio weights for test set 4.

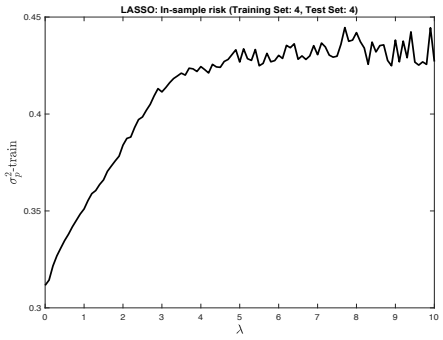

Fig 4: **LASSO** – In-sample risk for test set 4.

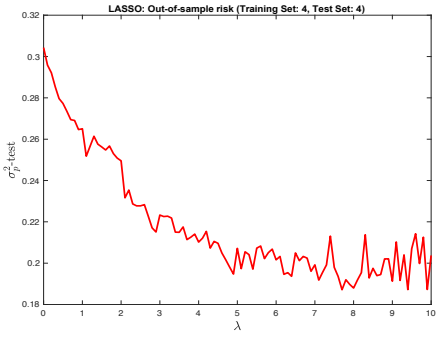

Fig 5: **LASSO** – Out-of-sample risk for test set 4.

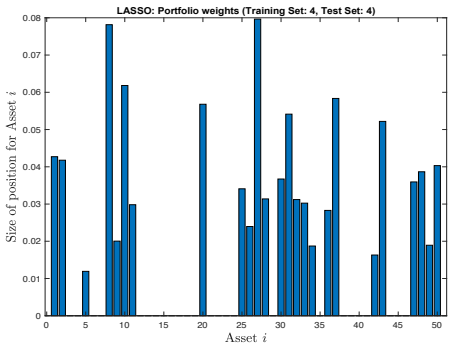

Fig 6: **LASSO** – Illustration of the optimal portfolio weights for test set 4.

Firstly, we notice the sparsity of the LASSO portfolio. One possible interpretation is the LASSO technique tends to focus on those positions which add significant levels of risk reduction to the portfolio [7]. Those positions that only produce a marginal reduction in the level of risk are discarded.

On the other hand, the Ridge portfolio take a position in almost every asset in the portfolio. This could be an explanation for the marginally higher minimum variance displayed by LASSO since we would naturally expect fewer positions to increase the level of risk, ceteris paribus. The sparsity of the portfolio could have contextual benefits though. In the real world rebalancing a portfolio is not free and an increased number of assets held could increase the amount transaction costs at each rebalancing date. Furthermore, depending on the investor, sales could trigger tax liabilities.

The second difference is the inclusion of short positions in the Ridge portfolio, whilst the LASSO portfolio in general has very few (or none for test set 4). This is simply an extension of our previous discussion as the increased number of positions suggested by Ridge must somehow be balanced to comply with the budget constraint. By taking short positions, it allows for greater investment in more assets – otherwise we would tend towards the equally weighted portfolio. Short positions also have the benefit of directly reducing portfolio variance. If two assets are perfectly correlated then taking an equal long and short position means the investor is perfectly hedged, and consequently has no risk exposure. This is clear evidence for the substantial reduction in variance over the equally weighted portfolio.

5. Conclusion

In conclusion, we have used two different regularization techniques to establish the minimum variance portfolio for our dataset. We have seen both methods produce similar values, clearly beating the equally weighted portfolio, but have starkly different portfolio constructions. In addition, the portfolio variances change considerably over time, as we include more historical data into our training set. This highlights the challenges faced in modern day finance and shows there is no perfect solution to problem.

As an extension of this work, one could generalise these results to the traditional minimum variance portfolio which includes a constraint on returns, rather than just on the budget. One could explore different penalty terms and extend the comparisons presented in this question. Alternatively, on a local level one could repeat this analysis using different optimisation software to understand whether there are improved minima.

References

[1] "Lending Club Dataset," Kaggle, 2019.

[2] A. C. Müller and S. Guido, *Introduction to machine learning with Python: a guide for data scientists.* " O'Reilly Media, Inc.", 2016.

[3] V. Vapnik, *The nature of statistical learning theory.* Springer science & business media, 1999, ch. 5.

[4] H. Markowitz, "Portfolio selection*," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1952.tb01525.x

[5] B. Fastrich, S. Paterlini, and P. Winker, "Constructing optimal sparse portfolios using regularization methods," *Computational Management Science*, vol. 12, no. 3, pp. 417–434, 2015.

[6] M. Kritzman, S. Page, and D. Turkington, "In defense of optimization: the fallacy of 1/n," *Financial Analysts Journal*, vol. 66, no. 2, pp. 31–39, 2010.

[7] J. Brodie, I. Daubechies, C. De Mol, D. Giannone, and I. Loris, "Sparse and stable markowitz portfolios," *Proceedings of the National Academy of Sciences*, vol. 106, no. 30, pp. 12 267–12 272, 2009.