COMP0050: Machine Learning with Applications in Finance

Peter McMullan*
UCL, Department of Computer Science
ucabpmc@ucl.ac.uk

April 4, 2024

Task 1

Abstract

This paper seeks to build models to predict whether a bank will default. Several Machine Learning models were built based on 15 features to complete the task. Given issues with imbalanced data, as well as the niche period over which the data was taken, generalisation of the prediction was difficult. However, performance was compared across each machine learning technique and it was found that Logistic Regression performed best out-of-sample for predicting both defaulted and non-defaulted companies, when compared to Random Forests, Support Vector Machines, and Voting Classifiers.

1 Introduction

This task focuses on the prediction of defaults across 7783 banks. Default is broadly defined as the inability to fulfil obligations, particularly repaying creditors. However, for banks during this sample period (2007), default mainly came in the form of exposure to risky Collateralized Debt Obligations (CDOs) and Real Estate exposure, which led to the Great Financial Crisis of 2008-2009. Therefore, the features important to this sample may be irrelevant to accurately forecast the next crisis. For this reason, a more holistic modelling approach was deployed during this paper in an attempt to produce better generalisation for each model deployed. This meant compromising some predictive accuracy for this particular crisis.

2 Methodology[1]

2.1 Data & Pre-Processing

The data included 7783 banks for Q4 2007, including 15 features per bank. Features included a breakdown of the end-user for each banks loans, as well as various assets and liabilities of each bank. Upon investigating the data, there was no missing values and 1 duplicate row (index 6996) was removed. After this, the data was split into a train and test set, with an 80%/20% split. The parameter `stratify=y` was set to ensure balance in the shuffled data, so proportions of default/non-defaults remained equal.

Feature engineering was conducted to create more meaningful relationships between the features currently included in the data. Features such as total loans, net debt (*debt - afs_securities + cash*), and debt to liquid assets (*debt/(cash + afs_securities)*) were included. Then, to ensure each feature contributes equally to the analysis and modelling, the data was scaled using Scikit-Learn's `StandardScaler()`. Although random forest's do not require scaling, this allowed for comparison of feature importance, and therefore better interpretability. With a large feature space (23 features), variables are likely to be highly correlated. A Variance-Inflation Factor (VIF) test was ran to check for multicollinearity in the features, an essential step to ensure robust results. It was found that just 9 features gave non-infinite VIF values. This informed the decision to deploy Principal Component Analysis (PCA).

PCA was performed using 4, 6, and 8 principal components. This step was necessary to reduce the dimensionality and remove multicollinearity across the features. Analysing the outputs of this transformation, 96% of the data was explained using 8 features, which was a sufficient reduction in dimensionality while preserving almost all the variance in the data. By definition, using PCA means all 8 features were uncorrelated, since PCA leads to orthogonal vectors in the subspace variables are projected onto. Although an 8-feature PCA explains the most variation, simpler models often perform better out-of-sample, so the base model (Logistic Regression) was trained on each of the three PCA datasets (4,6, and 8 PCs) and the best feature subset performer was chosen for the remainder of the models.

After cleaning and splitting the data it was observed that there was a 96% : 4% split between defaults and non-defaults for the banks in the dependent variable (y). The three main options usually employed to address the imbalance; synthetic minority over-sampling, random under-sampling of the majority class, or using balanced weights when building

*Thanks Prof. Fabio Caccioli and Sabrina Aufiero
[1]Methods in this section adapted from [1] and [2]

the models. It was decided that oversampling would not lead to statistically significant results due to the lack of actual defaulted data (61 data points) and under-sampling of the majority class would demand significant reductions in the number of samples for non-defaulted banks, leading to a lack of data. Therefore, balanced weights were used, incorporating the `class_weights` parameter within each model. This assigns higher weights (importance) to the minority class during training, encouraging the model to pay more attention to these points, potentially improving its ability to correctly classify them.

2.2 Model Building Framework

To ensure robustness and comparability across different machine learning models, a framework was developed to build all models. The steps were as follows;

1. Define each model with parameters: `random_state`=42, `max_iter`=10000, `class_weights`='balanced', `tol`=1e-4, `penalty`=L2. Since PCA already removed a significant number of features, L1 regularisation was deemed inappropriate because dropping more features risks making the model too specific to this dataset, hence impacting its ability to generalise well.

2. Use `Pipeline` with GridSearch Cross Validation (`GridSearchCV`) and 5-folds to find the optimal hyper-parameters. Scoring metric '`balanced_accuracy`' was used throughout to ensure both positive and negative predictions were assigned equal importance for the model.

3. The best model parameters are chosen to predict the out-of-sample results. `cross_val_score` with 5-folds was used to ensure the test results were not an anomaly.

2.3 Logistic Regression

Because this is a binary classification task, and it is often good practice to start with simpler models before moving to more complex ones, a Logistic Regression Model was built as the base model. Logistic Regression uses the Sigmoid Function;

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{1}$$

to produce outputs in the range [0-1], which can be interpreted as probabilities. For binary classification, as conducted in this paper, this means the decision boundary (threshold) is at 0.5, where values above this are assigned a classification of 1 and 0 otherwise.

Deploying this model it was found that 8 principal components led to the best results, so this feature set was used for the remaining machine learning approaches. The best hyper-parameter was `C=1e-5`, as confirmed by cross-validation. A smaller value of C indicates stronger regularization, meaning the model will be penalized more for complex decision boundaries that fit the training data too closely. It is positive to see that the value of C is low, as it suggests out-of-sample results should be better [2].

2.4 Support Vector Machines

Support Vector Machines (SVMs) operate by identifying a hyperplane that maximally separates different classes in a dataset. The optimal hyperplane is one where the margin, or the distance between the nearest points of the classes, is largest. SVMs are well-suited for high-dimensional spaces and non-linear class boundaries due to their capacity to handle complex datasets. For datasets where a clear margin of separation does not exist, SVMs employ a soft-margin approach to permit some misclassifications. This introduces flexibility and enhances the generalization of the model. Another significant feature of SVMs is the kernel trick, which allows the data to be mapped into a higher-dimensional space, facilitating the identification of a hyperplane in a way that is computationally efficient. By using kernel functions, SVMs can perform classification tasks that uncover non-linear relationships without incurring the high computational cost usually associated with such operations.

The kernel trick is deployed through Radial Basis Functions (RBFs), where a parameter γ must be set. This parameter defines the influence of points surrounding each data point (like K-Nearest Neighbors), where small values make the bell-shaped curve wider meaning each data point has a larger influence and the decision boundary ends up being smoother, leading to less over-fitting. $\gamma = 0.3$ leads to the Gaussian kernel.

Sticking with the simple-to-complex modelling approach, a Linear SVM was deployed first, before moving to a Gaussian RBF in two degrees, RBF in three-degrees, and finally a Stochastic Gradient Descent (SGD) SVM. For SVMs, the hyper-parameter

[2]Three features which toggled the size of the bank measured by total loan size were also added when running the Logistic Regression model (using One-Hot-Encoding). However these features were found to have no predictive value, so were discarded. Going forward, the 8-component PCA feature set is used.

C dictates the amount of mis-classifications allowed, with a lower value of C indicating more misclassifications allowed (and therefore less chance of overfitting).

SGD SVMs are beneficial as early stopping can be used. Iterations (epochs) are ran until the validation error has risen above a minimum for n epochs (ensuring the gradient descent doesn't find a straddle point/local minima). Once it has risen n times, the best weights are restored and the model training terminates. This makes it more likely one can find the global minimum parameters in terms of the bias-variance trade off.

The optimal hyperparameters found through cross validation were:

1. Linear SVC: C = 1
2. Gaussian SVC: $\gamma = 0.3$, C = 100
3. RBF: Degree = 3, C = 10, $\gamma = 0.1$
4. SGD: $\alpha = 1e - 05$

2.5 Random Forests and Voting Classifier

Decision trees often exhibit high variance; even slight adjustments in the data or hyperparameters can result in substantially different models. They are prone to overfitting as well, since they can keep dividing the data until they achieve near-perfect accuracy (purity) on the training set. To counteract this, employing an ensemble approach where predictions are averaged across many trees can yield more consistent results for both in-sample and out-of-sample data. This can happen through an ensemble of decision trees, i.e. a random forest. For these reasons, random forests were implemented instead of decision trees.

For hyper-parameter tuning, the `max_depth` and `N_estimators` were optimised. `N_estimators` controls the number of decision trees in the 'forest' and `max_depth` is the maximum depth of each tree. Limiting both parameters reduces the chance of overfitting, for the reasons discussed above. The optimal parameters found through cross-validation were; `max_depth`: 5, `N_estimators`: 10. This makes intuitive sense, since the relatively small number of features (8) means a relatively small number of trees are effective (10).

Finally, building on this ensemble approach, a Voting Classifier was deployed. This model concatenates all models and picks a majority vote for each observation, hoping that the 'wisdom of crowds' prevails. Generally, this method can boost a number of weak learners to create one great learner.

3 Results

The results of the aforementioned models are summarised in Table 1 and Figure 1. Five commonly used metrics to evaluate the results for a classification task are displayed: Recall, Accuracy, Area-Under-Curve (AUC), Precision, and Specificity. For a holistic view of results, the confusion matrix is plotted in Figure 1 for a selection of the models deployed (top 4 performers). The matrix is split into four quadrants, True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN), therefore conveying the actual outcomes versus those predicted by each model. The aforementioned metrics are defined as;

$$\text{Recall (Sensitivity)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Recall quantifies the proportion of defaults that were predicted by the model. This is the most important metric for this analysis. Accuracy represents the proportion of total correct predictions (defaults and non-defaults) made by the model. Precision reflects the number of correct default predictions versus all predicted defaults. A value of 1 for any of these metrics suggests perfect accuracy.

Finally, the Receiver Operating Characteristic (ROC) curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR);

$$\text{True Positive Rate (TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

The Area-Under-Curve (AUC) score is derived by measuring the area beneath the ROC curve. If a model predicts every outcome correctly, the ROC curve would move vertically up the y-axis before moving horizontally at $y = 1$, leading to an AUC score of 1. In contrast, random guessing would produce an ROC curve equal to $y = x$, leading to an AUC score of 0.5. Therefore, an AUC = 0.5 serves as a baseline against which the effectiveness of classification models is assessed.

The test set (out-of-sample) results for each classifier are shown below:

Model	Recall	Accuracy	Precision	Specificity	AUC
Logistic Regression	0.75	0.72	0.10	0.71	0.78
Linear SVC	0.52	0.89	0.18	0.90	-
Gaussian SVC	0.64	0.78	0.11	0.79	0.77
RBF SVC	0.70	0.76	0.11	0.76	0.79
SGD SVC	0.66	0.80	0.12	0.80	0.81
Random Forest	0.57	0.82	0.12	0.83	-
Voting Classifier	0.69	0.77	0.11	0.77	-

Table 1: Model performance metrics out-of-sample

Results were also explored aiming to optimise for the F1-Score, but these produced worse results for the majority of the models deployed.

4 Discussion

A high Recall means the model is predicting a high percentage of the actual defaults. Therefore, this is the key metric to consider for the models developed. Because of the nature of this task, we would accept a lower precision for a higher recall, i.e. we would rather have more FPs than FNs, as FNs can be a lot more costly. In other words, we would rather incorrectly forecast a bank to default which subsequently doesn't, than forecasting a bank doesn't default when it actually does. After all, these banks may have been close to default but narrowly avoided it. For a creditor, correctly predicting such 'narrow misses' can also be very important. However, it is important to note that a strong accuracy is also important, as a high recall could mean predicting all banks will default, leading to a poor accuracy.

As can be seen in Table 1, the base model, Logistic Regression performs best in accurately predicting defaults. However, the results are somewhat disappointing, as the best model still missed 25% of defaults. It could be assumed that a business fully exposed to 25% of defaults without hedging may not survive any crisis. Assessing performance across models, although the base model performs best on the Recall metric, the RBF and SGD SVMs perform well across a number of the metrics highlighted in Table 1. This suggests they may perform better than Logistic Regression if another crisis is analysed.

There are several potential reasons for the poor classification success of these models. Firstly, as mentioned in Section 1, the models in this paper are built to generalise well, so some performance for this specific task is compromised. It is hoped that these models could be applied to a completely different crisis and still produce similar results to those published above. Additionally, because the data is very skewed towards non-defaults, the machine learning techniques applied may have struggled to identify all defaults due to a lack of data. The data is also relatively small, with only 7783 samples to build the models with.

Furthermore, the data collected, being individual bank balance sheet data, does not reflect all elements of a potential default. Macro-economic variables like the leverage in the private sector, interest rates, and other variables need to be considered. Also, exposure between banks, like correlation/contagion risk, was a major factor in this crisis, and is not incorporated into the features. Therefore there is considerable unobserved features that may not be entirely reflected in the metrics used in this data. The fact that only one quarter of data is supplied (Q4 2007) also impacts predictive ability, as the rate of change of variables over time can also be significant in predicting crises ahead of time, for example, the rise in defaults countrywide or increases in credit card loans.

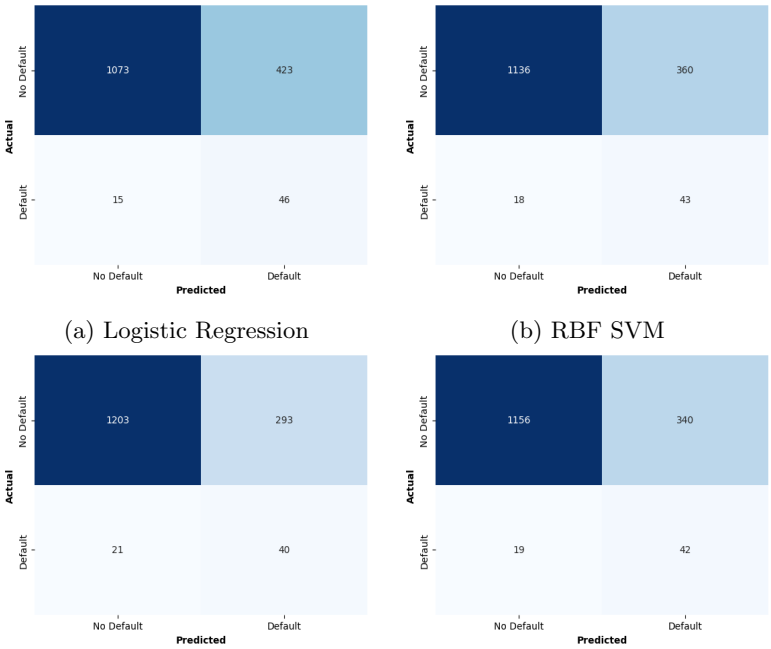
Interestingly, the random forest classifier highlighted component 2 and component 8 to be the most important features in predicting defaults/non-defaults. Component 2 is a combination of the ratio features created and component 8 is a combination of the original 15 features, so it makes sense that these are the most important and may suggest that an even lower dimension could be deployed with similar results.

5 Conclusion

The performance of several machine learning techniques was deployed in order to assess their ability to accurately predict defaults for banks in Q4 2007. It can be concluded that all models have some predictive

power, with SVM and Logistic Regression performing better than ensemble methods such as Random Forest and Voting Classifiers. This comes as a surprise, however given that this was binary classification task, simpler approaches often lead to better results, which explains this. As a final out-of-sample test, all models highlighted in this paper were used on data for Silicon Valley Bank (SVB) prior to its collapse in March 2023. Unfortunately none of the models were able to accurately predict the collapse of this bank based on the data they were trained on. This conveys that the ability to generalise well depends on the data machine learning models are trained on, as well as the difficulty in accurately forecasting defaults! From a business perspective, machine learning models can have a positive impact in helping to prevent crises in the future.

There are several extensions to this work which may help produce better in and out-of-sample results. Rather than deploying random forests, decision trees with gradient boosting could be deployed to test its success in accurately predicting defaults. Deep learning methods could also be incorporated. However, MLP/deeper neural networks are difficult in terms of explainability, which is a key part of this task (from a business perspective). This is the reason they were avoided in this paper. Other techniques could be used to select features in order to build these models, and results may improve if techniques such as backward and forward selection or L1 regularisation was used instead of PCA. Finally, upon reflection of the implementation of this task, due to the data provided (one quarter), it was unreasonable to attempt to create models which would generalise well, given the specific nature of the data the models were trained on. This could be accounted for in future studies to produce better results by using synthetic over-sampling and under-sampling.



(a) Logistic Regression

(b) RBF SVM

(c) Random Forest

(d) Voting Classifier

Figure 1: Confusion matrices for various classifiers.

References

[1] Hastie et al. Elements of statistical learning. *Springer*, 2009.

[2] Aurelien Geron. Hands-on machine learning with scikit-learn, keras, and tensorflow. *O'Reilly*, 2023.

Task 2

Abstract

This paper tested the efficacy of several portfolio optimisation techniques in order to build the minimum variance portfolio. These techniques included Ridge, Lasso, and Elastic-Net regularisation. Covariance shrinkage was included as an additional extension. It was found that as p/N increases, the estimation error increases exponentially, so regularisation helps mitigate against such errors as p/N approaches 1. This paper finds that the use of optimisation techniques outperforms equal-weight (1/N) portfolio allocations in terms of both variance and returns, suggesting that such techniques are robust in producing optimal portfolios.

1 Introduction

This task involved finding the global minimum variance (GMV) portfolio while allowing for negative weights. The optimal portfolio was calculated for different lengths of training time series and regularization schemes were introduced, the effects of which were analysed. A ban on short selling was then introduced for the unregularized problem.

Harry Markowitz [1] first introduced the problem of building the GMV portfolio in 1952. This paper focused on this optimisation problem, where the aim was to;

$$\begin{aligned} \underset{\sigma_p^2}{\text{minimize}} \quad & \sigma_p^2 = w^T \Sigma w \\ \text{subject to} \quad & \sum_{i=1}^{p} w_i = 1 \end{aligned} \tag{1}$$

where w is a vector of weights and Σ is the covariance matrix for the assets.

Through the remainder of this paper, the impact of regularisation techniques on this optimisation problem were explored. The paper concludes with an analysis of a momentum trading strategy, where performance is assessed for the different portfolio optimisation techniques discussed. This helps put each optimisation technique into the practical context of constructing an optimal portfolio for trading, where performance can easily be judged.

2 Methodology

2.1 Data & Pre-Processing

Data was collected on 48 industry portfolios from 1926-2017 via Ken French's website [2]. This data contained daily equity returns for 48 industries in the U.S. The dataset contained both equal-weight and market-capitalisation based returns for each of these industries at a daily frequency. This analysis uses the market-cap based dataset. There was a large number of missed values present in certain industries including; Guns, Gold, and Soda, which was uncovered during pre-processing. Due to the large amount of data, it was not a concern, so this period was excluded from the main analysis.

K-fold cross-validation fails in financial times series because *'observations cannot be assumed to be drawn from an IID process... and the testing set is used multiple times in the process of developing a model, leading to multiple testing and selection bias'* [3]. Therefore, the data was segmented into several non-overlapping date ranges in order to calculate the optimal hyperparameters and weights for each technique.

2.2 Optimal Portfolio

With the selected dataset, unregularised and regularised portfolios were constructed. The unregularised portfolio was built using the following steps:

1. Calculate the covariance matrix:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \text{Cov}(R_1, R_2) & \cdots & \text{Cov}(R_1, R_n) \\ \text{Cov}(R_2, R_1) & \sigma_2^2 & \cdots & \text{Cov}(R_2, R_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(R_n, R_1) & \text{Cov}(R_n, R_2) & \cdots & \sigma_n^2 \end{bmatrix} \tag{2}$$

2. Invert the covariance matrix:

$$\Sigma^{-1} = \frac{1}{\det(\Sigma)} \text{adj}(\Sigma) \tag{3}$$

3. Calculate the weights:

$$w_i = \frac{\sum_{i=1}^{p} \Sigma_i^{-1}}{\sum_{i,j=1}^{p} \Sigma_{ij}^{-1}} \tag{4}$$

   i.e. sum of the inverted covariance matrix column, divided by the sum of all inverted covariances.

4. This can also be calculated using a solver aiming to minimise;

$$\begin{aligned} & w^T \cdot \Sigma \cdot w \\ & \text{as before} \end{aligned} \tag{5}$$

Regularisation terms are added to prevent overfitting by introducing a penalty to the loss function used to train the model. For the regularised portfolios, Ridge, Lasso, and Elastic-Net regularisation were deployed. For Ridge regularisation (L2 regularisation), the inversion of the covariance matrix includes a regularisation term λI, where λ is the regularisation parameter, and I is the identity matrix. This technique leads to a circular optimisation landscape/constraint region due to the squared term. When these constraint regions intersect with the contours of the loss function, the points of intersection represent the values of the coefficients that minimize the penalized loss. For Ridge, this intersection point rarely occurs at the axes so the optimisation shrinks the size of the portfolio weights, helping control the portfolio variance. A higher value of λ leads to more regularisation (weights are shrunk more). The optimal hyperparameter for λ in Rigde and Lasso regularisation was found using the range [0, 10], with step sizes of 0.1.

In contract, Lasso (L1 regularisation) regularisation creates a diamond optimisation landscape due to the absolute coefficient terms, with corners lying on the axis. Therefore this technique can sets weights equal to zero when the contours meet at the axis. Finally, Elastic-Net (EN) regularisation is an integrated approach between Ridge and Lasso Regression, making it more flexible than each individually. Lasso helps in feature (stock) selection by allowing some coefficients (weights) to be exactly zero, whereas Ridge deals better with multicollinearity and shrinks the weights. Lasso might select one of these features, but Elastic Net tends to either select all the correlated features or none, making it more stable and interpretable. The parameter γ controls the weighting between L_1 and L_2 regularisation. The optimal hyperparameter for γ was found using the range [0, 1], with step sizes of 0.05 over each non-overlapping period. The optimal portfolio weights were found for the three techniques using the following formulas, where the penalty term for each regularisation technique is added to the covariance matrix. Both Lasso and Elastic-Net (due to Lasso) were passed through an optimiser to find the optimal weights for the portfolio using Python's `scipy.optimize.minimise` function, as they do not have closed form solutions.

$$L_1 = w^T \cdot (\Sigma + \lambda \cdot \sum_{i=1}^{p} |w_i|) \cdot w \tag{6}$$

$$L_2 = w^T \cdot (\Sigma + \lambda \cdot \sum_{i=1}^{p} w_i^2) \cdot w \tag{7}$$

$$EN = w^T \cdot (\Sigma + \gamma \sum_{i=1}^{p} |w_i| + (1-\gamma) \sum_{i=1}^{p} w_i^2) \cdot w \tag{8}$$

Within the optimiser, bounds can be set. When introducing the short-selling ban, bounds were set in the range (0, 1), therefore limiting the weights to the range 0-100%.

Finally, a momentum strategy was deployed to test the true efficacy of each portfolio construction technique. The portfolios were built based on a 12mth-1mth, long-short momentum strategy. The steps to build the strategy are as follows;

- Signal: Score each asset based on its $\frac{\text{price}_{i,t-1\text{mth}}}{\text{price}_{i,t-12\text{mth}}}$ at each time step t
- Relative Ranking: Rank the score cross-sectionally (versus all other industries)
- Selection: Long the top 20%, short the bottom 20% of industries based on relative ranking
- Portfolio Construction: Recalculate the covariance matrix every week using prior 24 weeks of returns data and deploy each optimisation technique to find optimal weights [4].

Optimisation is hindered by its assumption of constant covariance over time, hence a rolling covariance matrix was calculated. An additional optimisation technique was also included - covariance shrinkage. This has proven to be more robust out-of-sample versus traditional optimisation techniques, so was included as an additional 'hurdle' for the optimisation techniques described above [5]. An equal-weight allocation was also included as a benchmark.

3 Results

First, the impact of different sample lengths was tested. Figure 1 plots the risk (portfolio volatility) estimation error against the fraction of features (industries) versus number of samples (p/N).

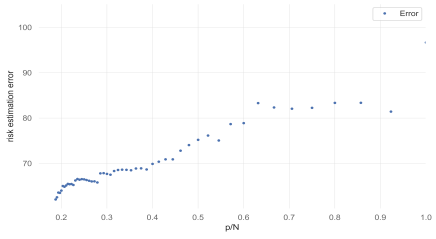


Figure 1: Volatility Estimation Error vs p/n

The benefits of regularisation for small versus large sample sizes (N) was also tested. Results are shown in Figure 2.

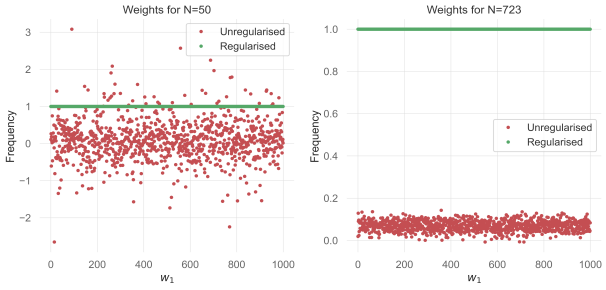


Figure 2: Regularised vs Unregularised portfolio weights for small and large N

Next, in-sample versus out-of-sample risk (volatility) was tested for all three regularisation techniques. Results for Ridge, Lasso, and Elastic-Net with different hyperparameter values are plotted in Figure 3, and the weights assigned to each asset for the GMV portfolio are plotted in Figure 4.

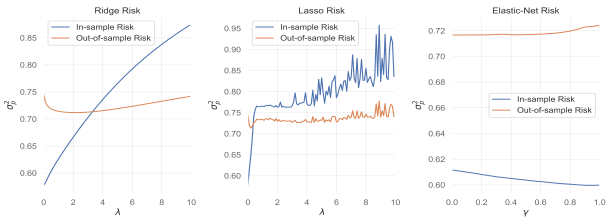


Figure 3: Ridge, Lasso and Elastic-Net regularisation risk for one sample period (2008-2009)

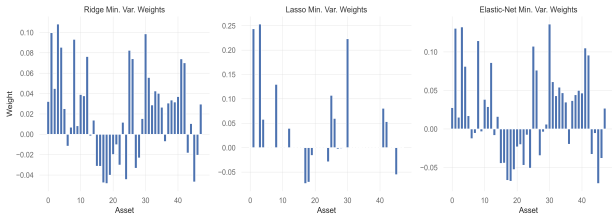


Figure 4: Ridge, Lasso and Elastic-Net regularisation weights for one sample period (2008-2009)

This aided in the selection of the optimal hyperparameters for each regularisation technique, which were used to build the momentum strategies. It should be noted that the hyperparameters deployed to build these strategies are sensitive to the period in which they are calculated. Different market regimes (bull or bear markets) will lead to different parameters and therefore results.

All portfolio optimisation techniques discussed were tested on a long-short momentum strategy, and performance was evaluated based on the Sharpe Ratio for the strategy;

$$\text{SR} = \frac{R_p - R_f}{\sigma_p}$$
$$\text{where } R_p = \text{portfolio return}, \tag{9}$$
$$R_f = \text{risk-free rate},$$
$$\sigma_p = \text{std. dev. of portfolio return}$$

Full sample performance of each strategy is plotted in Figure 5[1].

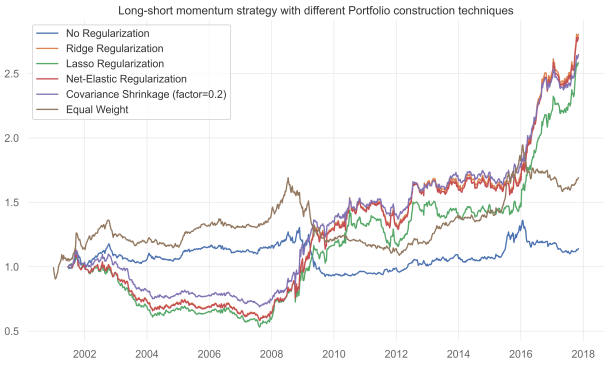


Figure 5: Full-sample performance of each optimisation technique

Covariance shrinkage performed best over the period with a Sharpe ratio of 1.4, followed by Ridge regularisation with a Sharpe of 1.19 and Elastic-Net with a Sharpe of 1.18 (due to the low value of γ=0.1). All other portfolio construction techniques significantly under-performed in terms of Sharpe ratio.

Finally, no short-selling bounds (0, 1) were set for the unregularised portfolio, as seen in Figure 6.

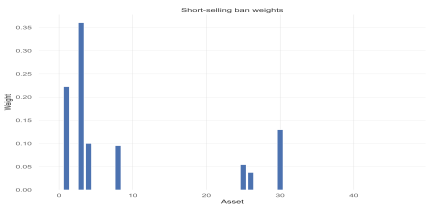


Figure 6: Un-regularisated portfolio, long-only optimal weights for GMV for one sample period (2008-2009)

[1]Risk-free rate assumed to be 0% and no transaction costs included for ease of analysis

4 Discussion

From Figure 1 and Figure 2, its clear that as the number of sample data points (N) increases, the estimation risk decreases, and there is little need for optimisation techniques as N grows very large. Without this, optimisation is helpful in creating more stable portfolios in terms of variance. In other words, as the time series length falls, overfitting occurs which leads to high variance out-of-sample. This justifies the use of subsamples of the data when applying the regularisation techniques discussed above, and suggests that the less data an investor has, the more they should rely on regularisation techniques when building GMV portfolios.

From analysing the weights allocated to the three regularisation techniques, one can recognise the sparsity of the allocations for Lasso regularisation. This occurs as this technique focuses on the few industries which reduce risk (variance) the most. This explains the underperformance of the strategy, as Lasso is over-exposed to a few assets, reducing the benefits of diversification. In contrast, Ridge is well-diversified with positions in all assets. This also explains the higher variance of Lasso versus Ridge and Net-Elastic in Figure 3. More short and long positions are included in the Ridge and EN portfolios, meaning that are closer to 'market-neutral' strategies, versus Lasso which has more directional exposure. This often leads to higher variance in trading, as positions are more correlated and variance cannot be offset/hedged. One caveat to this is that Lasso will exhibit significantly less transaction costs than other optimisation techniques due to this sparsity, which may lead to better performance for this trading strategy if such costs were factored accounted for.

Results from deploying optimisation techniques over the 18 year trading period in Figure 5 clearly show that such techniques are superior to naive 1/N allocations, supporting the research of Kritzman et al. [6]. What is surprising is that covariance shrinkage outperforms all other techniques on a Sharpe Ratio basis. Shrinkage helps to overcome 'Markowitz's curse', where diversification, which is supposed to stabilize a portfolio, leads to instability in the inverse of the covariance matrix as small estimation errors can be significantly amplified. This affects the entire portfolio's optimal weights, undermining the robustness and stability of the portfolio allocation and therefore its optimal variance.

Shrinkage improves the estimation of the covariance matrix by combining the sample covariance matrix with a structured estimator that is known to be well-behaved. This combination "shrinks" the sample covariance matrix towards the structured estimator, helping reduce the estimation error and leading to a more robust covariance matrix that is less prone to the kind of errors that can arise in high-dimensional data sets or during periods of market stress, when assets can exhibit high correlation. This means estimates of the minimum variance portfolio can be more reliable out-of-sample. This evidently leads to outperformance versus regularisation techniques, as the covariance shrinkage strategy experiences shallower drawdowns while still participating in the upside of other optimisation techniques.

Finally, it was found that most of the portfolio weights were set to zero over the sample periods when a short-selling ban was introduced for the unregularized problem. Subsequently, large amounts of capital were allocated to only a few assets, with only 7 out of 48 assets purchased in Figure 6. This is because the optimiser will only allocate to the lowest volatility and least correlated assets, since no shorting is allowed to balance/hedge such risks. It should be noted that this sparsity in allocations may have been due to the time period selected (during the Financial Crisis).

5 Conclusion

Several regularisation techniques were deployed in order to find the GMV portfolio using data based on US industries. It was encouraging to find that optimisation techniques clearly outperform equal-weight portfolio construction, confirming the benefits of using such techniques. It has been shown that Ridge regularisation provides more stability in estimating the optimal portfolio both in and out-of-sample. However, the length (and choice) of the in-sample period is a key consideration when deploying any such technique, as it can lead to different results. Extensions to this paper include using different optimisation and shrinkage methods over different time horizons and asset classes/markets to understand whether there is differences/convergence in the choice of hyperparameters and performance. Also, adding an expected return constraint could lead to superior results in terms of finding the GMV portfolio, but this adds an additional layer of complexity as one must accurately estimate returns ahead of time.

References

[1] Harry Markowitz. "Portfolio Selection". In: *Jstor* (1952).

[2] French. "48 Industry Portfolios daily". In: *Dartmouth, Ken French data library* (2017).

[3] Marcos Lopez de Prado. "Advances in Financial Machine Learning". In: *Wiley* (2018).

[4] Peter McMullan. "Recurrent Returns: How RNNs can improve momentum strategies". In: *COMP0162 Coursework* (2024).

[5] Ledoit Wolf. "Honey, I shrunk the covariance matrix". In: *SSRN* (2003).

[6] Page Kritzman and Turkington. "In Defense of Optimization: The Fallacy of 1/ N". In: *Financial Analysts Journal* (2010).