Machine Learning with Applications in Finance Assignment

Adam Keys

April 2024

1 Bank Default Classification

On March 10th, 2023, after careful consideration of the bank's financial health, the Federal Deposit Insurance Corporation (FDIC) announced the seizure of Silicon Valley Bank, ultimately marking its demise. Five days later, another four small-medium sized American banks had followed suit, sparking mass hysteria and panic in capital markets and the broader financial system. Paradoxically, the catalyst attributed to these collapses was a Federal Deposit Rate which was too *high*. Sharp interest rises had eroded the value of balance sheets overly exposed to older government debt yielding the historical, and far lower, rate (Ozili 2024). The consequences of bank failure both to the depositor and wider economy are catastrophic, with depositors suffering large losses not covered by the FDIC, and confidence in financial institutions falling precipitously. Effective predictive models which can shed light on which banks are most at risk can help mitigate the risk of surprise collapses. The following will explore which classification models, paired with different resampling schemes, are best suited for these purposes.

1.1 Methodology

In broad terms, chosen given the results in Mahboob Alam et al. (2020), the efficacy of the neural network, XGBoost algorithm, and logistic model will be tested on a cross-sectional dataset pertaining to the balance sheets of 7783 commercial US banks in 2007. In order to garner a holistic view of the different classification models, each model will be tested under different resampling techniques on a randomised 80%/20% train and test split of the data. Moreover, to get a more robust estimation of efficacy, each model will be applied 25 times under each scheme in order to generate a mean ROC curve and thereby an area-under-the-curve (AUC) statistic[1]. These measures constitute the primary method evaluation metrics.

For the sake of parameter robustness, all the classification models' hyperparameters will be optimised using 5-fold cross validation applied to the training set. In particular, to reflect the critical importance of correct classifications in the context of bank failure and systemic risk, the recall measure will be used in conjunction with overall classification accuracy in a custom measure as follows.

$$\text{Combined Performance}_j = \sum_{i=1}^{n} w_i(\text{Metric}_i^{\text{mean}} - \alpha\text{Metric}_i^{\text{std}}),$$

[1]The 'no resampling' case involves bootstrapping the train dataset at each iteration to evoke some stochasticity.

where j refers to the hyperparameter combination under consideration, w_i is equal to 0.3 for accuracy and 0.7 for recall, and $\alpha = 0.1$ reflects the relative importance of statistic spread.

1.1.1 Resampling Methods

A dataset comprising entities primarily belonging to a single class is considered *imbalanced*. Examples include the detection of mobile malware through network traffic (Chen et al. 2018), detecting oil spills through satellite imagery (Keramitsoglou et al. 2006), or predicting credit card defaults (Yang & Zhang 2018). Classification models, seeking to maximise accuracy, are fundamentally impaired by this imbalance; a trivial classifier can perform arbitrarily well by classifying all predictions as the majority class in a highly imbalanced setting. In essence, through their maximisation of classification accuracy, most algorithms assume a balanced distribution of classes (Provost 2000). This importance compounds in cases where prediction concerns that of a rare yet highly impactful case. Naturally, the classification of bank failures falls securely within this category given the dramatic impact on those customers not covered by deposit insurance and, more importantly, the systemic health of the wider financial apparatus as a whole.

Methods which overcome imbalance on the data level typically fall into one of three categories: undersampling, oversampling, or feature selection (Kotsiantis et al. 2006). Random undersampling targets the majority class wherein random entities are eliminated until the desired (balanced) class distribution is reached. The obvious drawback here is the potential to lose rare and important training data which aids an algorithm in classification. Moreover, the aim of ML models is to effectively approximate the target population distribution. Given the use of systematic undersampling, the sample distribution is no longer purely random, and therefore the approximation of the target distribution is inherently biased. Random oversampling takes the opposite approach, duplicating minority class instances to balance the relative class distributions. According to both Chawla et al. (2002) and Kubat (2000), this form of resampling induces greater overfitting due to the nature of making *exact* copies of minority entities (Weiss et al. 2007). Finally, feature selection can be utilised to combine distinctly selected positive and negative features in order to circumvent the issue of imbalance. Zheng et al. (2004) use this methodology in the context of text categorisation and find only limited efficacy.

Mahboob Alam et al. (2020) investigate the relative efficacy of the available resampling methods in the classification of credit card default risk. A naturally imbalanced problem similar to the one studied here, the authors find resampling methods to enhance the effectiveness of a number of ML algorithms. Based upon their results, the over- and under-sampling techniques used for the purposes of this work are the K-Means SMOTE and Cluster Centroids algorithms respectively.

K-Means SMOTE

Douzas et al. (2018) introduce K-Means SMOTE as an extension of the seminal contribution of Chawla et al. (2002). Synthetic minority over-sampling (SMOTE) aims to overcome the issue of duplicating-induced overfitting through interpolating between existing data to create artificial minority samples. An issue arising with the SMOTE methodology is the incidental amplification of noise. Namely, a noisy minority sample situated amongst majority samples linearly interpolated to its nearest minority neighbours will produce additional noisy synthetic samples within the majority class boundaries (see Bunkhumpornpat et al. 2009). K-Means SMOTE circumvents this problem by only oversampling in so-called *safe areas*.

The algorithm works in three steps: clustering, filtering, and oversampling.

1. The *clustering* phase involves using the k-means algorithm to partition the input data into k clusters through iteration. The algorithm assigns each entity to the closest of k cluster centroids before updating the centroids' positions such that they sit at the center of the observations belonging to them. This process is repeated until no more observations are reassigned, at which point convergence has been achieved. Convergence is guaranteed within a finite number of iterations, although usually to a local optimum (MacQueen 1967).

2. Once the data is partitioned, the *filtering* step selects clusters for oversampling and determines how many samples are to be drawn from each. The guiding tenet here is to prioritise clusters which are dominated by the minority class; applying SMOTE to these clusters is less likely to propagate noise. The default criterion is to select clusters with more than 50% of entities belonging to the minority class (this is a hyperparameter). Filtered clusters are thereafter assigned sampling weights between zero and one. Higher sampling weights indicate a low density of minority entities and thus correspond to more synthetic samples. Further details are deferred to Appendix A.1.

3. Finally, the *oversampling* phase involves oversampling each filtered cluster using SMOTE. Namely, each cluster is passed to the oversampling procedure which samples $||\text{samplingWeight}_i \times n||$ points, where n is the total number of samples needed. To generate an artificial sample, SMOTE randomly selects a minority $\vec{a}$ entity within a cluster along with one of its neighbours $\vec{b}$, then randomly interpolates between the two to generate the new sample $\vec{x}$. This process is continued until the desired number of samples has been generated.

Cluster Centroids

Lin et al. (2017) address the class imbalance problem from the undersampling perspective. They utilise clustering to generalise a series of majority data into a single point, thus reducing the number of entities whilst preserving some of the information. Specifically the number of clusters k, normally equal to the number of minority class data points, is first defined. Thereafter, the centers (centroids) of each cluster are determined using the k-means algorithm applied to the majority class. The result is a new series of k data points which accurately reflect and ultimately replace the majority class. Combined with the k entities from the minority class, the data is considered balanced and can be more effectively utilised by a classification algorithm.

1.1.2 Classification Models

The following classification models, like the resampling techniques, were chosen based upon the results presented in Mahboob Alam et al. (2020). It seemed prudent to use their wide-ranging research – which should generalise to this problem – to filter out a number of models *ex-ante*[2].

Logistic Regression

An old yet effective statistical model, logistic regression attempts to fit the following to data:

$$p(y|\mathbf{x}, \mathbf{w}) = \text{Ber}(y|\text{sigm}(\mathbf{w}^T\mathbf{x})),$$

[2] It was initially the case that the Gaussian discriminant model, a Naïve Bayes model without the assumption of independent class features, was also tested. The results however were so poor that the model was excluded.

where $y \in \{0, 1\}$ is the output variable, Ber denotes the Bernoulli distribution and sigm the sigmoid (or logistic) function. Notably, the feature vector $\mathbf{x}$ depends only linearly on a series of weights $\mathbf{w}$, before being passed through the (nonlinear) sigmoid function to normalise the output to within the range of a probability: zero and one. The negative log-likelihood of the model is:

$$NLL(\mathbf{w}) = -\sum_{i=1}^{N} [y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)],$$

where μ_i corresponds to the probability an entity's output variable y_i is equal to 1. Unfortunately the logistic model has no analytic expression for the maximum likelihood estimator. That said, the Hessian matrix is positive definite and therefore the negative log-likelihood is a convex function. Hence the model's parameters can be derived numerically through a gradient-based method. In the implementation here, the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (Liu & Nocedal 1989), a quasi-newton method, is utilised.

Neural Network

The neural network (NN) essentially operates as a large parallel computational model which broadly mirrors the functioning of the human brain (Dongare et al. 2012). Comprised of a series of nodes, or "neurons", organised into layers, the NN is characterised by the complex interconnections between these nodes. Each individual node produces an output which is solely a function of the local information available to it, arriving from nodes in the previous layer; in effect, each node is a non-linear, parameterised, and bounded function (Dreyfus & Dreyfus 2005). Mathematically, each neuron uses parameters $\{w_i\}$ to take a weighted sum v of its inputs $\{x_i\}$, together with a bias $\{w_0\}$, before passing this through a nonlinear *activation* function f.

$$v = w_0 + \sum_{i=1}^{n-1} w_i x_i$$

Common choices for the activation function are the hyperbolic tangent, rectified linear unit, and the sigmoid functions. Each designed around the neuron, NNs are most commonly classified as feedforward or recurrent networks. Our focus will be on the former, represented graphically below. This structure offers a vast range of different network topologies, given the condition of connection acyclicality. A multilayered feedforward network can be cleanly represented using recursion and matrix notation.

$$\mathbf{x}^i = f^i(\mathbf{w}^i \mathbf{x}^{i-1})$$

In this depiction, the superscript i denotes the layer which is being traversed. Hence $\mathbf{x}^i$ is either the initial feature vector at $\mathbf{x}^0$ or the output of a layer's activation function f^{i-1}. In the 1-dimensional case it is of size $(N_i + 1) \times 1$, where N_i is the number of features or neurons in the layer $i - 1$, along with an additional unit for incorporating the bias. The weights matrix is of size $(N_i + 1) \times (N_{i-1} + 1)$, with the additional values again accounting for the bias. This computation is continued recursively until the final output vector of dimension $(N_o \times 1)$. Understandably, NNs do not have an analytic solution for their MLE, instead relying on a method of backpropogation (see Rojas & Rojas 1996) and a gradient descent algorithm to fit parameters.

XGBoost

Chen & Guestrin (2016) introduce a scalable tree-boosting model named XGBoost (eXtreme Gradient Boosting). The authors built XGBoost with computational efficiency as a central importance,

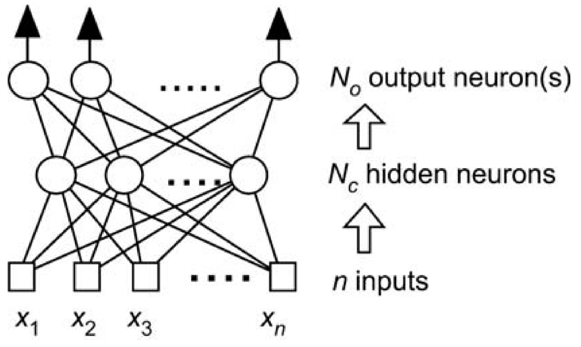

Figure 1: Graphical representation of the NN architecture (Dreyfus & Dreyfus 2005).

including innovations such as a new tree learning algorithm optimised for sparse data, distributed and parallel computing to enhance speed which, in turn, allows for greater model exploration, and a cache-aware block structure for out-of-core learning.

XGBoost is built upon the decision tree, an algorithmic structure which enables classification based upon features. At each of its nodes, a decision tree conditionally partitions the feature space such that different entities are thereafter better separated into their respective classes. The preeminent algorithm used in training decision trees is C4.5 from Salzberg (1994). At each node, C4.5 selects the attribute which most effectively splits the samples into subsets primarily consisting of one class or the other. The splitting criterion utilised is the normalised information gain, with the attribute maximising this value chosen for splitting at each step. The algorithm then recurses on the divided subsets to create further splits until the samples are sufficiently divided into their labelled classes.

Pairing the decision tree architecture with gradient boosting is one of the innovations brought forth by Chen & Guestrin (2016). This technique involves adjusting learning weights applied to various weaker learners[3] based upon errors made by prior learners, meaning that a model is able to compensate for the weaknesses of predecessors. Thousands of these optimised weak learners are then aggregated to create a forest which is used for prediction. For a more technical breakdown of how XGBoost incorporates boosting into the decision tree, along with its computational enhancements, readers are referred to the original paper.

1.2 Results

The experimental results, detailed in Table 1 and Figure 2, are relatively informative. The plots in Figure 2 exhibit the receiver operating characteristic curve (ROC), a visual representation of classifier performance. With the true positive rate plotted against the false positive rate for varying thresholds, the curve being farther to the upper left indicates a more effective classifier. This region represents a balanced performance whereby a model can achieve both high sensitivity and high specificity simultaneously. As such, the area under the curve (AUC) measure acts as a scalar indication of efficacy.

A number of conclusions can be drawn from the results presented in Figure 2. Firstly, across all plots, the 95% confidence intervals in ROC are relatively closely clustered about the mean function,

[3]These are typically decision trees with a very limited depth or small number of aggregate splits.

AUC	Logistic	N.N.	XGBoost	Method Avg.
No resampling	0.7	0.84	0.75	0.76
K-Means SMOTE	0.76	0.86	0.95	0.86
Central Centroids	0.79	0.57	0.68	0.68
Model Avg.	0.75	0.76	0.79	

Table 1: Summary of classifier performance on the testing data.

indicating relatively minimal variation resulting from resampling using the same technique. The degree of stochasticity in choosing the nearest neighbour, and the interpolation scalar, in the K-Means SMOTE algorithm for instance is evidently relatively minor. The NN had the lowest variation in mean results across the different resampling methods, whilst XGBoost had the largest. Notably, the logistic model has a quicker growing false positive rate relative to the true positive rate across all techniques, indicating that the model is not very robust at higher thresholds. An anomalous result, which was retested numerous times, concerns the NN under the central centroids scheme. The model is markedly poorer across all thresholds when paired with this scheme, with an AUC of only 0.57. Perhaps the expressiveness lost when clustering data into centroids restricts the network's ability to fit as closely to the true signals within the data. Finally, the performance of XGBoost is remarkably high under the K-Means SMOTE scheme, with a near-perfect 0.95 AUC. This is in contrast to its more modest performance under the different schemes; perhaps it is the case that XGBoost has considerable synergies with K-Means SMOTE, with the artificial data generated easily categorised using boosted decision trees.

Noting the results in Table 1, it is evident that K-Means SMOTE is the most effective resampling methodology, with an average AUC of 0.86. Although this value is skewed by the very impressive performance of the XGBoost algorithm, it still have scores strictly higher results with this model discarded. Central centroids was the least effective overall, with the two lowest AUCs out of the sample. That said, this method achieved the highest measure for the logisitic model, indicating a potentially advantageous coupling. Concerning each model's performance, the logistic model was the most reliable with the smallest standard deviation in performance. The neural network followed, with very high scores for both the no resampling and K-Means SMOTE algorithms. Importantly, given an inability to utilise a resampling method, the neural network is by far the most effective model with an AUC of 0.84. XGBoost overall has the best average performance with 0.79 AUC, although this is relatively marginal and undoubtedly skewed by the efficacy under K-Means SMOTE.

1.3 Discussion

Overall, the experimental results are beneficial in shedding light on which methodologies one should use when performing binary classification on imbalanced data. Namely, the logisic model is the most reliable overall, although it can be inefficient at higher thresholds and cannot match the upper bounds of the other models. In the context of bank default prediction, wherein the cost of misclassification can be very large, the most optimal model would undoubtedly capitalise on the K-Means SMOTE algorithm, the best performing resampling technique. In conjunction, one should ultimately choose wisely between the higher theoretical upper bound of XGBoost, or the more predictable neural network.

A facet of this analysis which could have been improved involves not only widening the range of models, but also resampling across the train and test data. This would thus involve double resampling wherein a model is tested repeatedly under a scheme, and then randomising the underlying test/train data too. Another potential limitation was the lack of data scaling, which could have enhanced model performance.

There are a number of interesting additional avenues for further analysis generated by this work, the most interesting of which may be the interaction between NNs and the cluster centroids scheme. The performance here was highly unusual, and further work may explore the mechanism driving this potentially ineffective combination. Moreover, it would be interesting to further understand the potential overarching underperformance of the undersampling techniques. Is it the case that these methods are substantively inferior to their oversampling counterparts? Or was it just a consequence of the circumstances of this work?

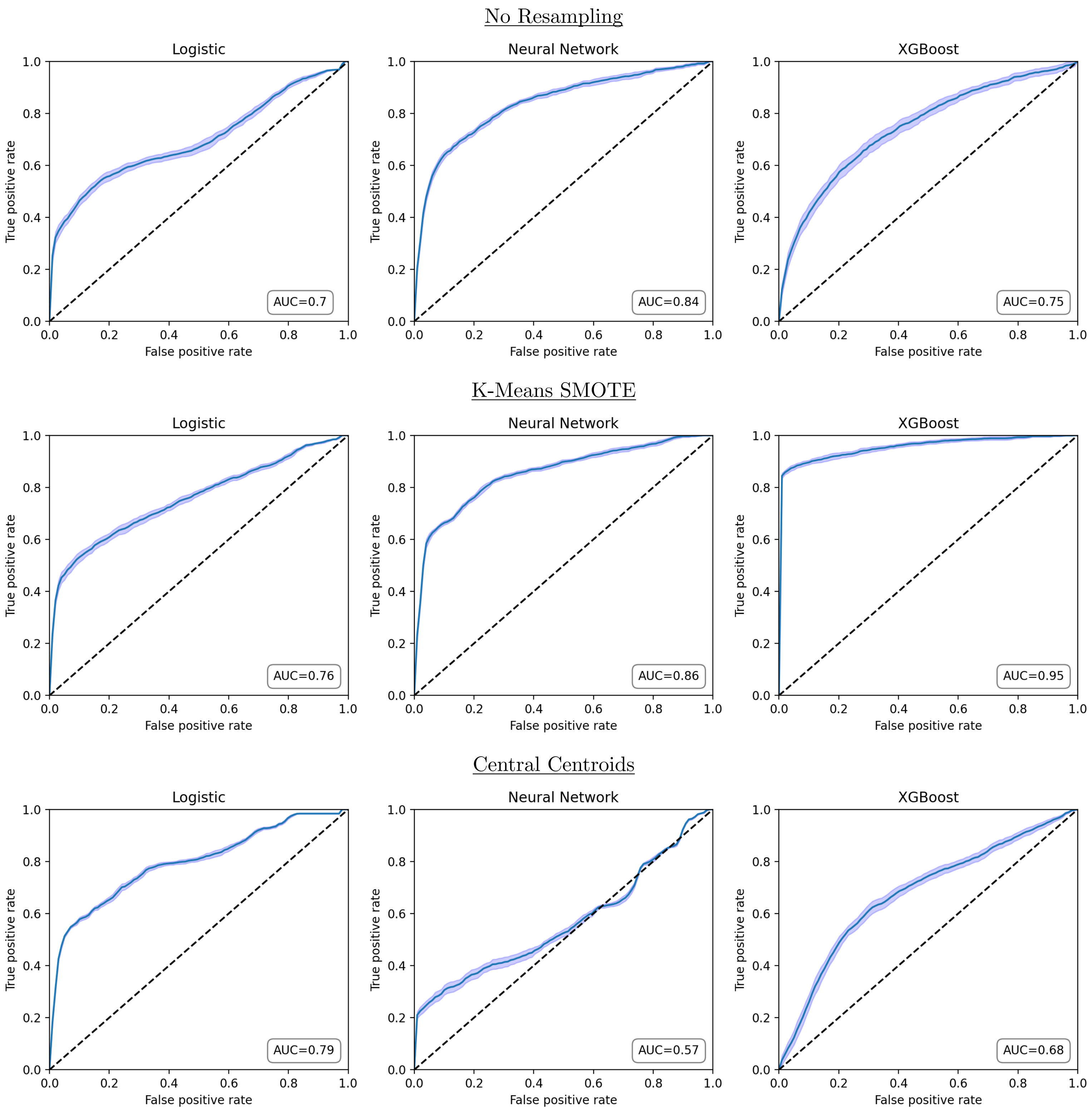

Figure 2: Performance of each model under the test set. The dashed 45° line represents random classification, the blue function represents mean ROC whilst its surrounding bands are 95th percentile confidence intervals.

2 Portfolio Variance Minimisation

Inherent to human nature appears to be the trait of risk-aversion. Kahneman & Tversky (1979) illustrate this notion elegantly, finding that individuals prefer strictly worse monetary payoffs that are certain, rather than an uncertain payoff of greater expected value. This notion translates seamlessly to the representative investor who seeks the minimal amount of risk for a given return. The following work seeks to understand the mechanisms underpinning risk minimisation in the Markovitz framework, namely the critical role of the covariance matrix and its inverse in finding global optima, and the manner through which this inverse can be more effectively estimated.

2.1 Methods

In order to assess such themes, daily equity return data for 48 different American industries are to be used, sourced from the Ken French datasets. The chosen returns are calculated on an equally weighted basis and range from 1969 until 2018. The data will be divided in an 80%/20% train and test split, translating to the test set comprising 2,440 days from 2008 onwards. A number of different portfolio optimisation techniques will be applied to the training data, before evaluation on the test set through means of numerous criteria.

2.1.1 Variance Minimisation

Modern portfolio theory emerged from seminal work by Markowitz (1952) who hypothesised that a so-called "optimal" portfolio could be derived through minimising a portfolio's variance, constrained by a given expected return. The primary focus here is on the returns-unconstrained problem, wherein a portfolio's variance is minimised through an analytic solution or otherwise. The financial purpose of this mathematical formalism is to construct a portfolio with limited risk, measured by standard deviation in returns, meaning that there is a low probability an investor faces a significant deviation from the expected return.

Mathematically, the problem an investor faces is thus:

$$\min_{\mathbf{w}} \mathbf{w}^T \hat{\boldsymbol{\Sigma}} \mathbf{w}$$
$$\text{s.t. } \mathbf{1}^T \mathbf{w} = 1$$

In this formulation, $\hat{\boldsymbol{\Sigma}}$ is the estimated covariance matrix of sector returns, $\mathbf{w}$ is an $n \times 1$ vector holding sector weights restricted by the simplex $\{\mathbf{w} : w_i \in \mathbb{R}, \sum_{i=1}^{n} w_i = 1\}$, and $\mathbf{1}$ is a vector of ones. An implicit assumption here is that sector returns are stationary in their multivariate distribution, given the covariance matrix is estimated using historical data. Were the interdependencies to suddenly change *ex-post*, then the optimal portfolio would no longer produce the minimum available variance.

The solution to this problem is well-known and can be found using the method of Lagrange multipliers (see Appendix A.2).

$$\mathbf{w}^* = \frac{\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{1}}{\mathbf{1}^T\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{1}}$$

Note that $\mathbf{1}$ denotes a vector of ones whilst $\hat{\boldsymbol{\Sigma}}^{-1}$ is the inverse covariance matrix sitting at the epicentre of this problem.

Adding the additional constraint that weights must be greater than 0 restricts the problem into disallowing short-selling, whereby an investor borrows a security, immediately sells it on the market, before repurchasing and returning the same security at a later date. Any differential in the initial sale and repurchase prices represents a realised gain. Many financial institutions such as pension companies are indeed restricted from short exposure to the market. This is because the short-sale is an inherently more risky trade: the most a long position can lose is 100% of the equity, whilst an asset can continue rising in price *ad infinitum*, thus leading to potential losses in excess of 100%. Imposing this second constraint means the problem can no longer be solved analytically[4] and must thus be solved using a numerical scheme. The optimisation is characterised as a convex programme given the objective function is quadratic in $\mathbf{w}$ and the covariance matrix is positive semi-definite. Convexity follows from the fact the Hessian matrix of the objective function is also positive semi-definite. The problem is thus solved using the quadratic cone programming functions in the CVXOPT package, which is designed for convex optimisation (see CVXOPT 2023).

2.1.2 Regularising the Problem

To solve the variance minimisation problem an estimate for the asset return covariance matrix is required. Most commonly, as is the case here, this estimate comes in the form of the sample covariance matrix. This choice however can be very poor given it is frequently nearly singular and sometimes not invertible. Such issues predominantly arise for two reasons: the number of assets p may be very large relative to the sample ($p/n \approx 1$), leading to a nearly singular sample covariance matrix even if the population equivalent is not, or due to highly correlated asset returns which implies that the population covariance itself is nearly singular. Regularisation helps to abate the issue of an ill-conditioned covariance matrix and stabilise the inverse, thereby reducing the propagation of estimation errors from the sample covariance into the optimisation problem. In order to gauge the impact of matrix singularity, the condition number will be collected under the regularised and unregularised schemes, alongside measures of in- and out-of-sample portfolio variance.

Ridge Regression

Regularising our optimisation problem using the Ridge scheme (Hoerl & Kennard 2000) involves adding the ℓ_2 norm of the parameter vector to the objective function. Through doing so, large deviations in the value of the portfolio weights will increase the objective function's value and thus be discouraged in the minimisation. The optimisation problem and its closed form solution are as follows.

$$\min_{\mathbf{w}} \mathbf{w}^T \hat{\boldsymbol{\Sigma}} \mathbf{w} + \lambda ||\mathbf{w}||_2^2, \quad \text{s.t. } \mathbf{1}^T \mathbf{w} = 1$$

$$w_i^* = \frac{\sum_j \left((\hat{\boldsymbol{\Sigma}} + \lambda \mathbf{1})^{-1} \right)_{ij}}{\sum_k \sum_j \left((\hat{\boldsymbol{\Sigma}} + \lambda \mathbf{1})^{-1} \right)_{kj}}$$

In this formulation, $||\cdot||_2$ represents the ℓ_2 norm, w_i^* is the optimal weight for asset i, and λ is a hyperparameter. Given the freedom to vary λ, the extent to which deviations in portfolio weights are penalised, this parameter will be optimised using a form of 5-fold cross validation tailored for time series. Namely, the test data, accounting for the first 80% of observations, will be split equally

[4]There exists some research in-fact offering analytical solutions for this problem, including Kondor et al. (2017)

into 5 folds. Thereafter, the first fold will act as the training set, with the optimal weights applied to the second fold to deduce the average variance. Both the first and second fold will then be used in training, and the third will become the test. This process continues until the fifth fold, where the variance for each $\lambda \in \{0, 0.02, .., 2\}$ is aggregated across each fold for a holistic view of performance.

Elastic Net

This section will extend the analysis and explore the use of the elastic net regularisation technique in mean-variance portfolio optimisation, inspired by the works of Ho et al. (2015) and Bohne & Scully (2022). This will first involve regularising the historical asset returns with the James-Stein scheme (Jorion 1986), which will then suffice as a proxy for their future values. Stein's work illustrated the benefits of using a Bayes estimator in place of the traditional sample mean. The estimated average returns $\hat{\boldsymbol{\mu}}$ will therefore be the in-sample mean returns $\boldsymbol{\mu}_s$, regularised in the direction of the portfolio's overall mean return.

$$\hat{\boldsymbol{\mu}} = (1-\rho)\boldsymbol{\mu}_s + \rho\eta\mathbf{1},$$

$$\eta = \min\left(\frac{1}{p}\sum_{i=1} \mu_{s,i}, 0.0004\right), \quad \rho = \min\left(1, \frac{p-2}{n_{\text{train}}(\boldsymbol{\mu}_s - \eta\mathbf{1})^T \hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\mu}_s - \eta\mathbf{1})}\right)$$

The elastic net regularisation adds a combination of LASSO and Ridge penalties, modifying the optimisation problem as follows.

$$\min_{\mathbf{w}} \mathbf{w}^T\hat{\boldsymbol{\Sigma}}\mathbf{w} - \mathbf{w}^T\hat{\boldsymbol{\mu}} + \lambda_1||\mathbf{w}||_1 + \lambda_2||\mathbf{w}||_2^2,$$

where $\hat{\boldsymbol{\Sigma}}$ is the estimated covariance matrix for asset returns, $\hat{\boldsymbol{\mu}}$ is a vector of expected future returns, λ_i represents a weighting parameter, and $||\cdot||_p$ is the norm of order p. Using ℓ_1 regularisation restricts the techniques which can solve this problem to the numerical domain. Typically, solving a robust optimisation problem necessitates the use of semi-definite programming techniques which become intractable as the number of assets grow. Reformulating the weighted elastic-net problem into a quadratic programme and using general purpose solvers adds N primal variables and $2N$ dual variables. A variant of the Split-Bregman algorithm (Goldstein & Osher 2009), which is far more extendable, will be used here to find the optimal weights (Algorithm 1).

Algorithm 1 Split-Bregman Algorithm

```
Require: $(\hat{\Sigma}, \hat{\mu}, \lambda), k = 1, b^k = 0, w^k = 0, d^k = 0$
  while $||w^k - w^{k-1}||_2 > tol$ do
    $w^{k+1} = \arg\min_w(w^T\hat{\Sigma}w - w^T\hat{\mu} + \frac{\lambda}{2}||d^k - \beta w^k - b^k||_2^2)$
    $d^{k+1} = \arg\min_d(\frac{\lambda}{2}||d^k - \beta w^k - b^k||_2^2 + ||d||_1$
    $b^{k+1} = b^k + \beta w^{k+1} - d^{k+1}$
    $k = k + 1$
  end while
```

The parameter λ will be optimised using the same 5-fold cross validation technique outlined earlier. Given this problem now concerns both variance and returns, the derived optimal weights will be applied to the test set to determine performance relative to a benchmark equally weighted strategy. In particular, the Sharpe ratio, a measure of risk-adjusted returns, will be calculated in order to evaluate the efficacy of the optimisation.

2.2 Results

Non-Regularised Optimisation

The data presented in Figure 3 (overleaf) summarise the unregularised results. An interesting phenomenon is observed in the upper left figure, wherein the train and test return variance is plotted in the unrestricted case as the ratio p/n increases from approximately 0 towards 1. These results were achieved through calculating the optimal weights given a decreasingly large training set, with the test set kept constant. As the covariance matrix becomes increasingly singular, as p/n increases, one can observe that the test variance begins to rapidly increase, whilst the train variance decreases. This worsening out-of-sample performance can be explained by a decreasing ability of the covariance matrix to accurately capture the true population interdependencies as the train set decreases in size (Maillet et al. 2015). Mathematically, the eigenvalues of the estimated covariance matrix will become increasingly dispersed relative to the true and unobservable eigenvalues (Marčenko & Pastur 1967), resulting in inconsistent eigenvectors (Johnstone & Lu 2009).

Such problems arising as a covariance matrix becomes increasingly unstable can be encapsulated by the condition number of the matrix, defined as its ratio of maximal and minimal eigenvalues. In a general sense, the condition number is a measure of error propagation, representing how much, in the worst case, a function is affected by the change in the input argument (Aste 2024). In this case, uncertainty in parameter values increases alongside p/n, resulting in a divergent condition number as can be seen in the lower left figure. This detail indicates that parameter measurement errors are propagated to an ever-increasing extent, causing the observed out-of-sample performance to falter.

Interestingly, when the portfolio weights are limited to restrict short sales, the results change drastically. There are a few details worth investigation. Firstly, there are sharp decreases for both in- and out-of-sample variance as the size of the training data decreases. This is quite unusual behaviour which could be explained purely by the particulars of the data. A further observation is that there is no relationship whereby the train variance tends to zero as the test equivalent rapidly increases as p/n tends to 1. It is possible that this behaviour is due to the use of a numerical scheme in calculating the optimal weights, rather than inverting an increasingly ill-conditioned covariance matrix.

Finally, the portfolio's out-of-sample returns, although not the focus of the optimisation, are reported in the lower right figure for the sake of completeness. Interestingly, the overall return generated by each scheme is approximately equivalent, although the long-only portfolio diverged slightly lower in 2013 and 2016. That said, when one considers the rolling variance, exhibited at the bottom, it is clear that the minimum variance portfolio suffers from far less risk. In quantitative terms, assuming a generous 2% risk-free rate across period, the pure variance-minimising portfolio had a Sharpe ratio of 2.0 and a variance of 1.19, followed by the long-only portfolio with values 1.65 and 1.32, and finally an equally weighted portfolio with Sharpe 1.23 and a much higher variance of 1.92. Such results demonstrate the efficacy of the Markowitz framework and why it is so widely adopted.

Regularised Optimisation

Noting the results from the regularised case in Figure 4 (overleaf), the relationships are very different. Firstly, under a Ridge-regularised scheme, the in- and out-of-sample variances do not scale as in the non-regularised case. Considering the upper left figure, the variances plotted signify the ability of the Ridge scheme to stabilise the inverse covariance matrix through its dampening towards the scaled identity matrix. This notion is summarised in the plot below, wherein the condition number can be plotted on a linear plot and does not asymptotically tend to infinity. This indicates that the covariance matrix is better able to capture the codependencies in the data, even as p/n increases; a notion which is validated by the data displayed in the upper left plot.

The elastic net scheme is shown to reduce the variance to an ever-increasing extent as the λ parameter increases in the Split-Bregman algorithm (upper right). The relation between λ and the return in this scheme is different, with the return following a more parabolic functional shape (not shown). The out-of-sample returns (lower right) do illustrate its efficacy, with the portfolio weights under this regularisation achieving the highest returns. This is largely to be expected, given the technique simultaneously minimises variance *and* maximises returns, in contrast to the Ridge scheme wherein only the former is considered. Interestingly however, when adjusted for risk, the elastic net portfolio yields a Sharpe ratio of 1.92, only marginally higher than the Ridge portfolio with 1.91, but considerably above the null method with a value of 1.23.

2.3 Discussion

Overall, the results presented are both informative and broadly align with what the theory predicts. An interesting contradiction does arise however with the highest Sharpe ratio belonging to the unregularised minimum variance portfolio. Although the differential is relatively marginal, one would imagine that a portfolio optimising both return and risk should capture the highest risk-adjusted returns. That said, both risk and return, and thereby the Sharpe ratio, are naturally random variables; one could apply the exact same analysis to a slightly different dataset or time horizon and achieve completely different results. Therefore it would be interesting to explore as to whether this was purely an anomalous result, and what the relative risk and return tradeoffs are when discriminating between schemes.

Another interesting avenue of exploration is as to the rapidly declining test- and train-set variances shown in the data as n initially decreases; surely this is due to statistical circumstance rather than being driven by an underlying theoretical factor. Further exploration may also consider the ability of numerical schemes to outperform their analytical counterparts as p/n tends to 1. It could be the case that through avoiding the inversion of the covariance matrix, numerical schemes are inherently more stable and thus better for use in these circumstances. Finally, a thorough analysis of weight dispersion under different schemes would also be informative, given they ultimately determine resulting portfolio performance.

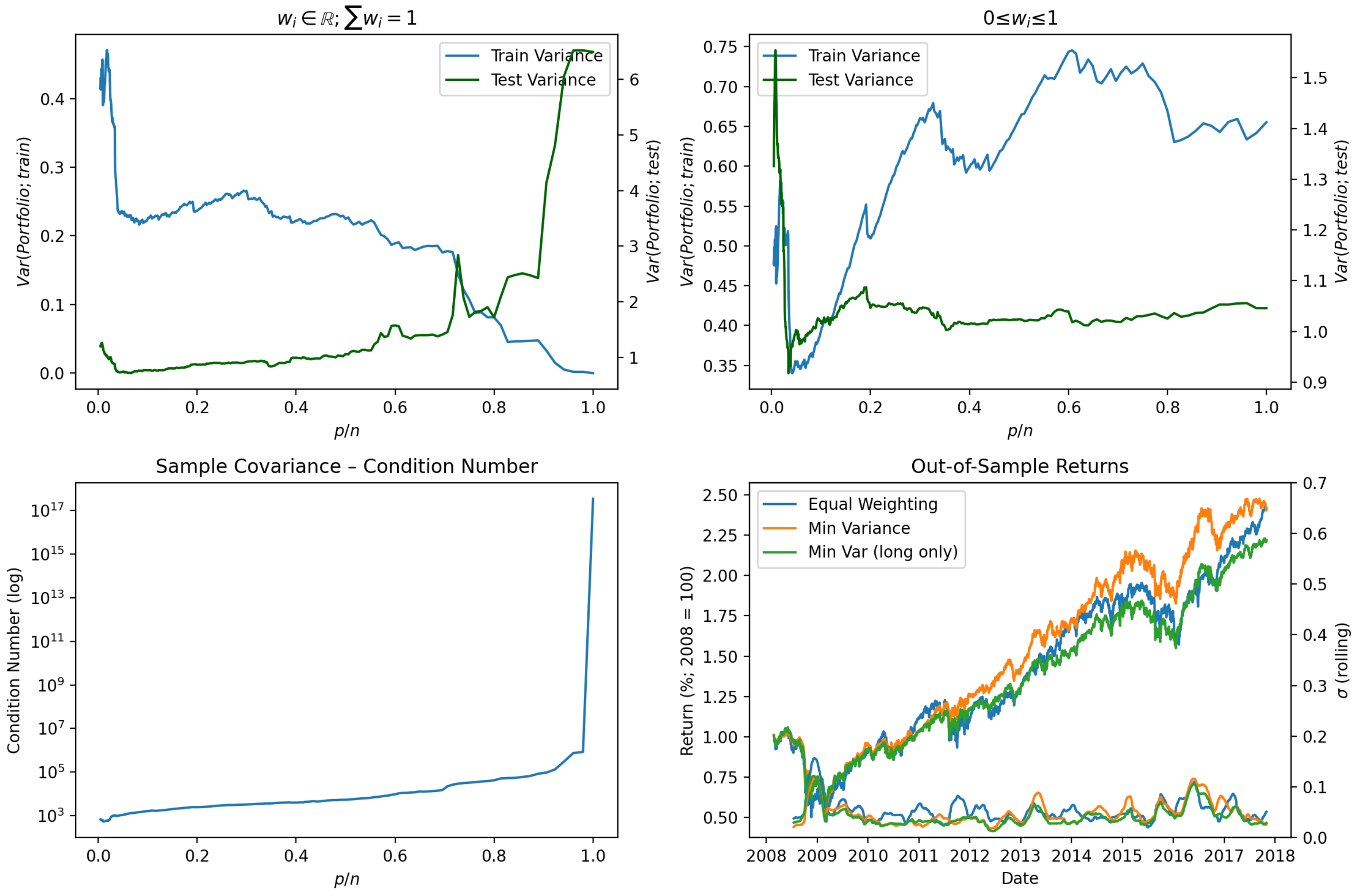

Figure 3: Illustration of results in the non-regularised case.

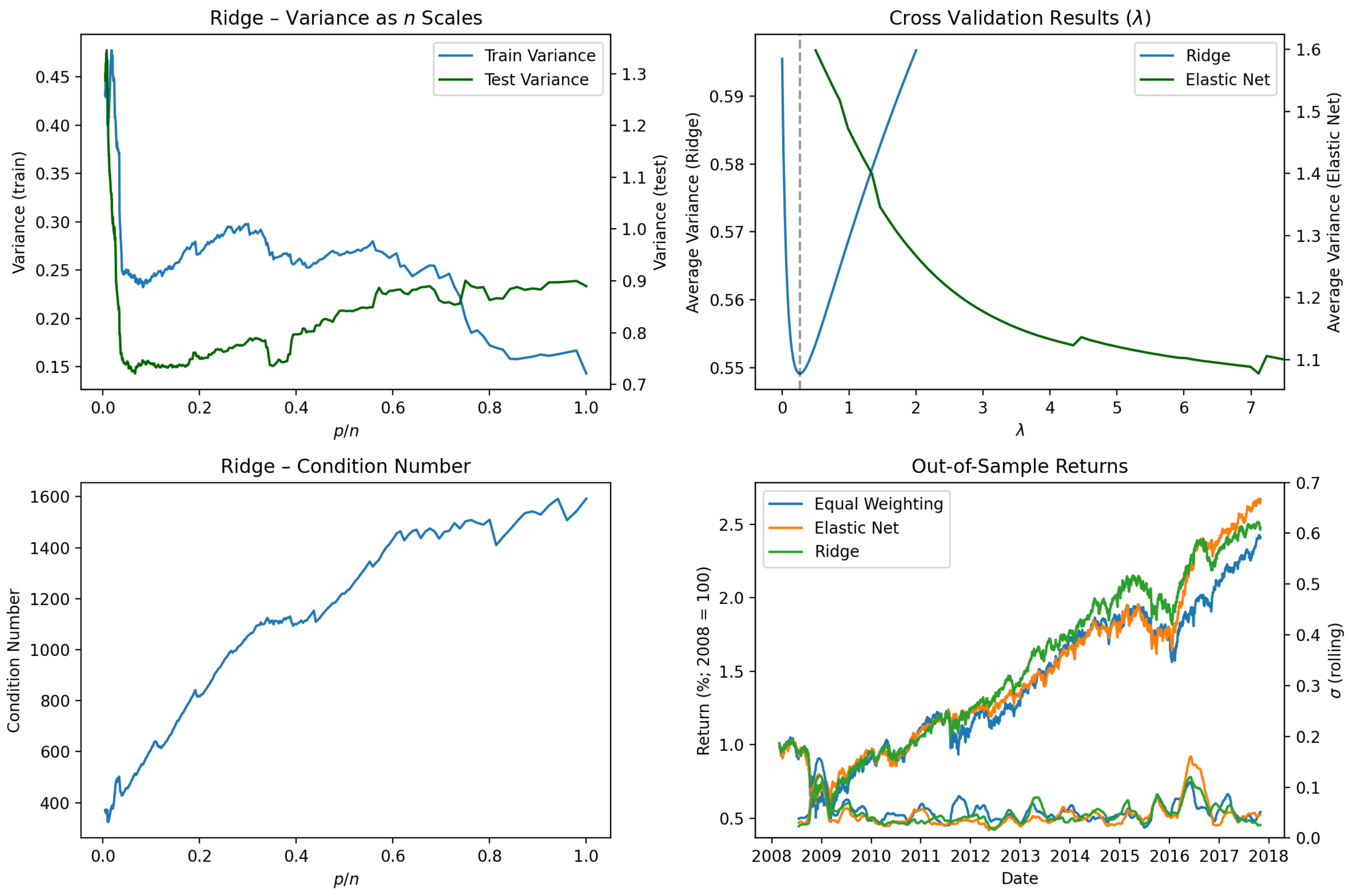

Figure 4: Summary of results in the regularised case.

A Appendix

A.1 K-Means SMOTE Clustering

To calculate this weight, the density of a cluster is considered in the context of the average cluster density. Specifically, the Euclidean distance matrix is first computed for each filtered cluster. The mean distance is found through summing the diagonals and dividing by the number of non-diagonal elements. Thereafter the cluster density is calculated as the number of minority instances divided by its average minority distance to the power of the number of features. This measure is then inverted to get a measure of sparsity. Finally, each cluster's sampling weights are defined as their sparsity divided by the sum of all cluster sparsity factors.

A.2 Global Minimum Variance Portfolio Derivation

This is a constrained minimisation exercise as follows:

$$\min_{\mathbf{w}} \mathbf{w}^T \hat{\mathbf{\Sigma}} \mathbf{w}$$
$$\text{s.t. } \mathbf{1}^T \mathbf{w} = 1$$

The Lagrangian equation can be formed:

$$\mathcal{L}(\mathbf{w}, \lambda) = \mathbf{w}^T \hat{\mathbf{\Sigma}} \mathbf{w} + \lambda(\mathbf{1}^T \mathbf{w} - 1)$$

The resulting first-order-conditions are found using calculus:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathbf{w}^T \hat{\mathbf{\Sigma}} \mathbf{w}}{\partial \mathbf{w}} + \frac{\partial(\mathbf{1}^T \mathbf{w} - 1)}{\partial \mathbf{w}} = 2\hat{\mathbf{\Sigma}}\mathbf{w} + \lambda \mathbf{1} = \mathbf{0},$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = \frac{\partial \mathbf{w}^T \hat{\mathbf{\Sigma}} \mathbf{w}}{\partial \lambda} + \frac{\partial \lambda(\mathbf{1}^T \mathbf{w} - 1)}{\partial \lambda} = \mathbf{w}^T \mathbf{1} - 1$$
$$\mathbf{w} = -\frac{1}{2}\lambda \hat{\mathbf{\Sigma}}^{-1} \mathbf{1}$$

Multiply both sides and solve for λ:

$$\mathbf{1}^T \mathbf{w} = -0.5\lambda \mathbf{1}^T \hat{\mathbf{\Sigma}}^{-1} \mathbf{1}$$
$$\lambda = -2 \frac{1}{\mathbf{1}^T \hat{\mathbf{\Sigma}}^{-1} \mathbf{1}}$$

And finally, substituting λ back into the original equation for w:

$$\mathbf{w}^* = -0.5(-2) \frac{1}{\mathbf{1}^T \hat{\mathbf{\Sigma}}^{-1} \mathbf{1}} \hat{\mathbf{\Sigma}}^{-1} \mathbf{1} = \frac{\hat{\mathbf{\Sigma}}^{-1} \mathbf{1}}{\mathbf{1}^T \hat{\mathbf{\Sigma}}^{-1} \mathbf{1}}$$

References

Aste, T. (2024), *Data Science Textbook (unfinished).*

Bohne, J. & Scully, J. (2022), 'Mean-variance portfolio optimization using elastic net penalty'.

Bunkhumpornpat, C., Sinapiromsaran, K. & Lursinsap, C. (2009), 'Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem'.
URL: *https://api.semanticscholar.org/CorpusID:9477920*

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), 'Smote: Synthetic minority over-sampling technique', *Journal of Artificial Intelligence Research* **16**, 321–357.
URL: *http://dx.doi.org/10.1613/jair.953*

Chen, T. & Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '16, ACM.
URL: *http://dx.doi.org/10.1145/2939672.2939785*

Chen, Z., Yan, Q., Han, H., Wang, S., Peng, L., Wang, L. & Yang, B. (2018), 'Machine learning based mobile malware detection using highly imbalanced network traffic', *Information Sciences* **433**, 346–364.

CVXOPT (2023), 'Cvxopt website', `https://cvxopt.org/userguide/coneprog.html?highlight=cvxopt%20solvers%20qp`.

Dongare, A., Kharde, R., Kachare, A. D. et al. (2012), 'Introduction to artificial neural network', *International Journal of Engineering and Innovative Technology (IJEIT)* **2**(1), 189–194.

Douzas, G., Bacao, F. & Last, F. (2018), 'Improving imbalanced learning through a heuristic oversampling method based on k-means and smote', *Information Sciences* **465**, 1–20.
URL: *http://dx.doi.org/10.1016/j.ins.2018.06.056*

Dreyfus, G. & Dreyfus, G. (2005), 'Neural networks: an overview', *Neural Networks: Methodology and Applications* pp. 1–83.

Goldstein, T. & Osher, S. (2009), 'The split bregman method for l1-regularized problems', *SIAM J. Imaging Sciences* **2**, 323–343.

Ho, M., Sun, Z. & Xin, J. (2015), 'Weighted elastic net penalized mean-variance portfolio design and computation'.

Hoerl, A. E. & Kennard, R. W. (2000), 'Ridge regression: Biased estimation for nonorthogonal problems', *Technometrics* **42**, 80 – 86.
URL: *https://api.semanticscholar.org/CorpusID:28142999*

Johnstone, I. M. & Lu, A. Y. (2009), 'On consistency and sparsity for principal components analysis in high dimensions', *Journal of the American Statistical Association* **104**(486), 682–693.

Jorion, P. (1986), 'Bayes-stein estimation for portfolio analysis', *Journal of Financial and Quantitative Analysis* **21**, 279–292.

Kahneman, D. & Tversky, A. (1979), 'Prospect theory: An analysis of decision under risk', *Econometrica* **47**(2), 263–291.
URL: *http://www.jstor.org/stable/1914185*

Keramitsoglou, I., Cartalis, C. & Kiranoudis, C. T. (2006), 'Automatic identification of oil spills on satellite images', *Environmental modelling & software* **21**(5), 640–652.

Kondor, I., Papp, G. & Caccioli, F. (2017), 'Analytic solution to variance optimization with no short positions', *Journal of Statistical Mechanics: Theory and Experiment* **2017**(12), 123402.
URL: *http://dx.doi.org/10.1088/1742-5468/aa9684*

Kotsiantis, S., Kanellopoulos, D., Pintelas, P. et al. (2006), 'Handling imbalanced datasets: A review', *GESTS international transactions on computer science and engineering* **30**(1), 25–36.

Kubat, M. (2000), 'Addressing the curse of imbalanced training sets: One-sided selection', *Fourteenth International Conference on Machine Learning* .

Lin, W.-C., Tsai, C.-F., Hu, Y.-H. & Jhang, J.-S. (2017), 'Clustering-based undersampling in class-imbalanced data', *Information Sciences* **409-410**, 17–26.
URL: *https://www.sciencedirect.com/science/article/pii/S0020025517307235*

Liu, D. C. & Nocedal, J. (1989), 'On the limited memory bfgs method for large scale optimization', *Mathematical programming* **45**(1), 503–528.

MacQueen, J. (1967), Some methods for classification and analysis of multivariate observations.
URL: *https://api.semanticscholar.org/CorpusID:6278891*

Mahboob Alam, T., Shaukat Dar, K., Hameed, I., Luo, S., Sarwar, M., Shabbir, S., Li, J. & Khushi, M. (2020), 'An investigation of credit card default prediction in the imbalanced datasets', *IEEE Access* **8**.

Maillet, B., Tokpavi, S. & Vaucher, B. (2015), 'Global minimum variance portfolio optimisation under some model risk: A robust regression-based approach', *European Journal of Operational Research* **244**(1), 289–299.

Markowitz, H. (1952), 'Portfolio selection', *The Journal of Finance* **7**(1), 77–91.

Marčenko, V. & Pastur, L. (1967), 'Distribution of eigenvalues for some sets of random matrices', *Math USSR Sb* **1**, 457–483.

Ozili, P. K. (2024), Causes and consequences of the 2023 banking crisis, *in* 'Governance and Policy Transformations in Central Banking', IGI Global, pp. 84–98.

Provost, F. (2000), 'Machine learning from imbalanced data sets 101', **68**(2000), 1–3.

Rojas, R. & Rojas, R. (1996), 'The backpropagation algorithm', *Neural networks: a systematic introduction* pp. 149–182.

Salzberg, S. L. (1994), 'C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993'.

Weiss, G., McCarthy, K. & Zabar, B. (2007), 'Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs?', pp. 35–41.

Yang, S. & Zhang, H. (2018), 'Comparison of several data mining methods in credit card default prediction', *Intelligent Information Management* **10**(5), 115–122.

Zheng, Z., Wu, X. & Srihari, R. K. (2004), 'Feature selection for text categorization on imbalanced data', *SIGKDD Explor.* **6**, 80–89.
URL: *https://api.semanticscholar.org/CorpusID:7956405*