Department of Mathematical Sciences

Project IV - (MATH4072)

Equity Fund Profitability and Sustainability Modelling

Using Multiple Response Regression

Author:
Raul Unnithan

Supervisor:
Ric Crossman

March 17, 2025

Declaration

Contents

Chapter 1 Introduction

Single-response regression models 1 response against a set of predictors. This method can be extended to multiple predictors, but what if we need to model more than 1 response? One approach is to run several independent single-response models. However, when responses are correlated, the response-covariance matrix's off-diagonal elements are non-zero, which this approach ignores, leading to suboptimal predictions. Multiple-response regression (MRR) addresses this, improving predictions.

However, selecting the most relevant predictors remains a challenge, and every MRR model does this differently. This report examines different MRR models and applies them to an equity fund dataset with 2 correlated responses: return on equity (ROE) and sustainability score. Each model's predictions are evaluated using the average normalised root mean square error, which is the root mean square error divided by the standard deviation and then averaged across each response.

Each MRR model will also be applied to a small-scale dataset with Exam Scores for clarity. These calculations will use precise values, and results will be rounded to 2 decimal places for simplicity.

Throughout this report, there will be some common terminology used. These are the correlation and covariance between the responses, which, as stated, MRR considers. This consideration is essential for every model covered to use as it is what qualifies them as multiple-response. The response intercorrelation, $\mathbf{R}$, is defined as: $\mathbf{R} = \mathbf{D}^{-1}\mathbf{\Sigma_Y}\mathbf{D}^{-1}$, where $\text{Cov}(\mathbf{Y}) = \mathbf{\Sigma_Y}$ is written as the covariance matrix of the response variables, and $\mathbf{D}$ is a diagonal matrix of the standard deviations of the response variables.[1] Each element of $\mathbf{R}$ is computed as:

$$R_{ij} = \frac{(\mathbf{\Sigma_Y})_{ij}}{\sigma_i \sigma_j},$$

where $(\mathbf{\Sigma_Y})_{ij}$ is the covariance between response variables i and j, and σ_i, σ_j are their respective standard deviations.[1] This relation ultimately means that if an MRR model considers the off-diagonal elements in the response covariance matrix, it considers the response intercorrelation and vice versa.

Let us now look at the equity fund dataset. An equity fund is a pooled investment that puts money mainly in stocks listed on major exchanges.[2] They provide investors with several benefits, such as diversification and the potential for improved returns.[2] They also come with risks associated with stock market volatility and losses.[2] They can be categorised according to factors such as their investment style and portfolio focus, which is how they are represented in this data.[2]

Modelling ROE and sustainability scores together is important as it gives a more long-term view of each equity fund's risk versus return potential. Sustainability has become more relevant as investors move to prioritise sustainable portfolios. A key driver of this trend is the growing concern over climate change and global warming, which have significant long-term social and economic implications. In a 2021 letter to other CEOs, Larry Fink, the CEO of BlackRock, wrote that 81% of a globally representative selection of BlackRock's sustainable indexes outperformed their parent benchmarks.[3] This clearly highlights the benefit of considering sustainability in equity funds.

Chapter 2 Exploratory Data Analysis

This chapter provides an outline of the process of cleaning the equity fund dataset, from its source to explaining the predictors, to examining correlation and multi-collinearity, to imputing missing values and testing individual and joint underlying distributions.

2.1 Initial Data Collection and Preprocessing

The dataset was obtained from Kaggle and simplified for this analysis. The research focuses on analysing equity funds, and the master dataset contains a mixture of equity and bond funds. The first step was to filter out the bond funds and keep the equities. This filtering was done simply using Excel's filter features. A text filter selects the relevant rows within the `category` column. Then, there was an issue with the columns; there were 130 dependent variables for this analysis. Many variables were for bonds only, so these were removed. Some variables, such as a fund's `isin` number, were irrelevant for analysis, further reducing the number of dependent variables to 29 variables. Another critical factor for the chosen independent variables was to be able to look into how they would affect the dependent variables, so the aim was to have 10-20 variables in the model because this is a manageable amount. Also, a lot of the remaining variables served the same purpose, which would lead to multicollinearity. For example, `management_fees` are a significant component of `ongoing_cost`, so the `management_fees` column was removed. The final step in preprocessing was to remove the rows corresponding to inverse equity funds as they perform the opposite of their underlying benchmark.

2.2 Variable Explanation

`category` gives the type of equity on which the data is collected. A `rating` is given by an analyst, and this is a value from 1 to 5, depending on whether the analyst wants to buy or sell. A `risk_rating` gives the market volatility of the equity.[4] `category`, `rating` and `risk_rating` are categorical variables while the rest of the variables are continuous.

`equity_style_score` is the difference between 2 investment strategies: growth and value stocks.[5] Growth stocks are expected to bring a lot of capital due to strong growth in the underlying company.[6] Value stocks are either underrated or ignored by the market, and they could gain value eventually.[6] Hence, a lower `equity_style_score` means the stock is more of a value stock, and a higher `equity_style_score` indicates more of a growth stock. `equity_size_score` measures the performance of said equity relative to the company's value.[7] `price_prospective_earnings` gives the ratio between a company's share price and earnings per share. `price_cash_flow_ratio` is a multiple that compares a company's market value to the amount of capital it generates during a select period.[7]

A dividend is the money companies may pay you if you own their shares, and the dividend yield is the dividend paid as a percentage of the share price. The `dividend_yield_factor` is the dividend yield, except it is used to compare the different equities, considering variations in dividend yield

performance. `historical_earnings_growth` measures how a stock's earnings per share has grown in the last 5 years. `sales_growth` is the increase in sales of a company's products/ services over time. `asset_cash` is the proportion of a fund's assets held in cash. `holdings_n_stock` is the number of different stock holdings each equity fund holds. A stock holding is the number of other companies in which the equity fund has invested by holding their stocks. `ongoing_cost` is the cost of the daily running of a business.[8] `fund_size` is the sum of all the assets in the fund.

2.3 Exploratory Data Analysis

Now that the dataset has been cleaned and the variables specified, the next step is to perform data analysis in R and find underlying relationships between the variables. To fully represent the dataset, it is best to use actual values rather than simulate any missing values. So, this initial analysis will be done with only rows with no missing values.

2.3.1 Underlying Distributions

QQ plots can be used to find out if a variable is normally distributed. The ROE and sustainability scores' QQ plots are shown below:

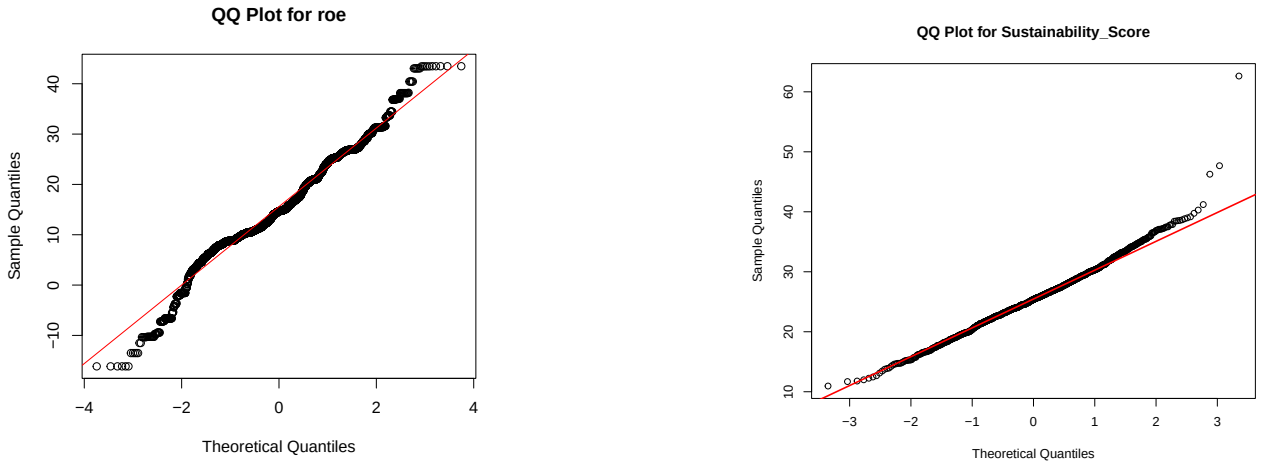

Figure 2.1: QQ Plots for ROE and Sustainability Scores

ROE's and sustainability score's underlying distribution are normally distributed because most data points can be roughly approximated as a straight line.

2.3.2 Correlation and Multi-Collinearity

It is also essential to examine the correlation within the dependent variables to determine which additional variables should be removed. High correlation implies multicollinearity, so it is necessary to check this before proceeding. After creating the correlation matrix for the dependent variables, there were two high correlation values. One was `0.8099`, which was the correlation between `price_prospective_earnings` and `price_cash_flow_ratio`. Based on the initial definitions of these variables, the latter is a more relevant factor in profitability and sustainability scores, so it is kept. The other high correlation was `-0.8029` between `dividend_yield_factor` and `equity_style_score`. Again, based on the definitions of these variables, the former is a more relevant factor in profitability and sustainability scores, so it is kept. This results in 12 independent predictor variables.

2.3.3 Imputing Missing Values

Although high correlation implies multicollinearity, the opposite is not necessarily true. Multicollinearity can be tested on variables through the virtual inflation factor (VIF). This dataset contains a mixture of categorical and continuous variables, so it is essential to consider both variable types in the analysis.

The dataset should be complete before carrying out VIF analysis. It is essential to look at how the NA values are spread and how often they occur in each row. It was found that 90% of the rows had 2 NAs or less, and the dataset is large, so only these rows will be examined upon further analysis.

Random forest imputation was used to fill in the missing categorical data for several reasons. Although it is computationally more expensive than using methods such as the mode to fill in missing data, it deals well with variables with no association. Rating and risk rating seem to have no association, as can be seen in their heat map; see Figure A.1.

Random forest imputation also works for categorical data through surrogate splits, and outliers influence them less as they use an ensemble of decision trees.[9] Random forest imputations also do not require prior assumptions of the underlying distribution, unlike methods such as linear model imputation, which requires underlying distribution normality, making them flexible.[9] The random forest model is trained on rows with no NA values, and `classification = TRUE` is set to make sure the random forest is predicting categorical values rather than numerical ones. `formula = as.formula(paste(column, "~.")` ensures the target column, the one with missing values, is not a predictor. The model learns relationships between the target variable and the other columns in the dataset.

Random forests grow many decision trees using random rows and features to improve predictions. Each tree is trained on a randomly selected subset of rows from the data. Some rows may appear multiple times, and others not at all. At each split in a tree, only a random subset of the columns is considered, further dividing the data into smaller groups. This randomness ensures the trees capture different patterns in the data. Predictions from the trees are aggregated to fill in missing values, reducing the impact of over-fitting by averaging out biases and individual tree errors.

Linear model imputation is used to fill in the missing numerical features. This method is chosen because it uses the relationship between variables, unlike more straightforward methods such as mean imputation.[10]

2.3.4 Multicollinearity

VIF analysis was done using a standard multiple response regression linear model by `lm(cbind(roe, sustainability_score)~.)`. The categorical variables are involved using frequency encoding as with linear model imputation. After this, a dummy response variable is added for VIF calculation, and the VIF values are derived from this new model. If any of the predictors have VIF values of more than 10, they would need to be removed due to multicollinearity. In this case, all the values are less than 5, see A.2, so none of these variables has multicollinearity, and the dataset is almost ready for analysis.

2.3.5 Multivariate Normality

A lot of the methods tested in this report rely on the multivariate normality of the equity fund dataset. This property was tested using Mardia's test for multivariate normality via skewness and kurtosis. Both of these tests failed to reject normality with p-values of 0.0732 and 0.0.0862, respectively. Therefore, there is sufficient evidence to suggest multivariate normality here. However, these low p-values are noteworthy as they can give an insight into why certain models perform better than others.

Chapter 3 Linear Regression

This chapter introduces the methodology of the first MRR model: multiple response linear regression by starting with single response linear regression. Next, stepwise selection methods are introduced, and how they have been extended to the multiple response case is explained. Multiple analysis of variance is then covered because it is a key part of how the predictors in multiple-response linear regression are chosen. This code is then explained, and the different models in this chapter are evaluated using the average normalised root mean square error.

3.1 Theory

3.1.1 Single Response Linear Regression

Before delving into multiple-response linear regression, let us start with single-response linear regression. Single response linear regression models the relationship between a single response (dependent) variable and a set of predictor (independent) variables. The general form of this model, with the dimensions subscripted, is:

$$\mathbf{Y}_{(n\times 1)} = \mathbf{X}_{(n\times p)}\boldsymbol{\beta}_{(p\times 1)} + \boldsymbol{\epsilon}_{(n\times 1)},$$

where n is the number of observations, p the number of predictors, $\mathbf{Y}$ the response variable, $\mathbf{X}$ the design matrix, $\boldsymbol{\beta}$ the unknown coefficients and $\boldsymbol{\epsilon}$ is the error variable.

Single response regression comes with a few underlying assumptions which come from its error terms. They are linearity, homoscedasticity, normality, and independent variables. Linearity means there is a linear relationship between the dependent variables and the independent variables: $E(Y_i|X_i) = \beta_0 + \beta_1 x_i$. Residuals should be normally distributed with a mean of 0 and variance σ. This constant variance is also known as homoscedasticity, i.e. $\text{Var}(y_i|x_i) = \sigma^2$, and it means variables are not too highly correlated.[11] Homoscedasticity, in turn, proves normality: $Y_i|X_i \sim N(\beta_0 + \beta_1 x_i, \sigma^2)$.

When modelling any data, the aim is to minimise the residuals. A residual is the difference between predicted and actual values. Reducing this ensures the model is as accurate as possible, which is done by finding the most optimal coefficient vector, $\boldsymbol{\beta}$. The optimal coefficient vector, also known as the ordinary least squares estimator (OLS), $\hat{\boldsymbol{\beta}}$, is found by minimising the sum of squared residuals. It can also be defined using $\mathbf{X}$ and $\mathbf{Y}$:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}.$$

The model's accuracy can then be evaluated using metrics such as the R-squared value, which measures the proportion of variance in the dependent variable explained by the independent variables.

Single-response linear regression can be extended to multiple-response linear regression using a few key adjustments which incorporate the covariance matrix between the responses.

3.1.2 Multiple Response Linear Regression

Multiple-response linear regression (MRLR) is the simplest model in MRR. Its general form is:

$$\mathbf{Y}_{(n\times m)} = \mathbf{X}_{(n\times p)}\,\mathbf{B}_{(p\times m)} + \mathbf{E}_{(n\times m)}, \tag{3.1}$$

with

$$E(e_i) = 0 \quad \text{and} \quad \text{Cov}(e_i, e_k) = \sigma_{ik}\mathbf{I} \quad i,k = 1,2,\ldots,m,$$

where $\mathbf{I}$ is the identity matrix, n is the number of observations, m is the number of responses, and p is the number of predictors, including the column of ones.

It is important to delve into the difference between the dimensions in single response linear regression and MRLR. The clear differences are the structures of the response, coefficients and errors. They are all matrices now, rather than vectors, due to the move to multiple-response. Here is a simple example to further this understanding: suppose $n = 4$, $p = 3$, and $m = 2$, Equation 3.1 becomes:

$$\mathbf{Y}_{(4\times 2)} = \mathbf{X}_{(4\times 3)}\,\mathbf{B}_{(3\times 2)} + \mathbf{E}_{(4\times 2)},$$

which can be written in matrix form as:

$$\begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \\ y_{31} & y_{32} \\ y_{41} & y_{42} \end{bmatrix}_{(4\times 2)} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ 1 & x_{31} & x_{32} \\ 1 & x_{41} & x_{42} \end{bmatrix}_{(4\times 3)} \cdot \begin{bmatrix} b_{01} & b_{02} \\ b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}_{(3\times 2)} + \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \\ e_{31} & e_{32} \\ e_{41} & e_{42} \end{bmatrix}_{(4\times 2)}.$$

Here, it is clear that the number of columns in $\mathbf{Y}$ is the number of responses, which in this case is 2. The number of rows in $\mathbf{Y}$ is the number of observations, which is 4 here.

Now, let us return to the general case and examine another key difference between single and MRR: the covariance matrix. The m observations on the jth trial have covariance matrix $\Sigma = \{\sigma_{ik}\}$, but observations from different trials are uncorrelated.[12] Here $\mathbf{B}$ and σ_{ik} are unknown parameters and the design matrix $\mathbf{X}$ has jth row $[x_{j0}, x_{j1}, \ldots, x_{jr}]$.[12] Each value in the covariance matrix, σ_{ik}, is the covariance between the response variables i and k. A non-zero covariance means there is intercorrelation between these responses. In MRR, the covariance matrix captures interdependencies between different response variables. It is important to capture this information as it improves response prediction accuracy.[13]

The i-th response $\mathbf{Y}_{(i)}$ follows the standard linear regression model:

$$\mathbf{Y}_{(i)} = \mathbf{X}\mathbf{B}_{(i)} + \mathbf{E}_{(i)}, \quad i = 1,2,\ldots,m,$$

with $\text{Cov}(\mathbf{E}_{(i)}) = \sigma_{ii}\mathbf{I}$ as expected, but the errors associated with different responses within the same trial may be correlated.[12] Each response variable i is modelled independently, but all responses share the same set of predictors.

The ordinary least squares (OLS) estimates, $\hat{\mathbf{B}}_{(i)}$, can be determined only from the observations $\mathbf{Y}_{(i)}$ on the i-th response.[12] This is true because the OLS method, which derives $\hat{\mathbf{B}}_{(i)}$, minimises the residual sum of squares (RSS) separately for each response in MRLR as shown here:

$$\text{RSS}_{(i)} = \|\mathbf{Y}_{(i)} - \mathbf{X}\mathbf{B}_{(i)}\|_2^2.$$

This logic for the OLS estimate is just like single-response but applied to each response variable separately:

$$\hat{\mathbf{B}}_{(i)} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{Y}_{(i)}.$$

Collecting these univariate least squares estimates gives:

$$\hat{\mathbf{B}} = \begin{bmatrix} \hat{\mathbf{B}}_{(1)} & \vdots & \hat{\mathbf{B}}_{(2)} & \vdots & \cdots & \vdots & \hat{\mathbf{B}}_{(m)} \end{bmatrix} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top, \begin{bmatrix} \mathbf{Y}_{(1)} & \vdots & \mathbf{Y}_{(2)} & \vdots & \cdots & \vdots & \mathbf{Y}_{(m)}, \end{bmatrix}$$

or equivalently:

$$\hat{\mathbf{B}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{Y}.$$ [12]

Using the least squares estimate, $\hat{\mathbf{B}}$, the matrices of the predicted values and residuals can be made. [12] The predicted values are given by:

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{B}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}.$$

And, the residuals are given by:

$$\hat{\mathbf{E}} = \mathbf{Y} - \hat{\mathbf{Y}} = [\mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}']\mathbf{Y}.$$

The predicted values and residuals can ultimately be used to evaluate MRLR's performance which is what we will do later on the equity fund dataset.

MRLR comes with a few underlying assumptions which can be extended from the single response case. Linearity comes from the formula for MRLR, which, when expanded, is:

$$\begin{aligned} Y_1 &= b_{01} + b_{11}x_1 + \cdots + b_{r1}x_r + e_1, \\ Y_2 &= b_{02} + b_{12}x_1 + \cdots + b_{r2}x_r + e_2, \\ &\vdots \\ Y_m &= b_{0m} + b_{1m}x_1 + \cdots + b_{rm}x_r + e_m. \end{aligned}$$

These rows are also independent, another MRLR property from single-response linear regression.

Normality also holds here, but it slightly differs. If we looked at using several independent single-response models, each error term is independent and identically normally distributed, with $\epsilon_i \sim \mathcal{N}(0, \mathbf{I})$. However, in MRLR, the error terms, $\mathbf{E}$, share the same distribution. They are assumed to have a multivariate normal distribution with constant variance: $\mathbf{E} \sim \mathcal{N}(0, \mathbf{\Sigma_E})$, where $\mathbf{\Sigma_E}$ is the covariance matrix of errors. The error terms associated with different responses may be correlated. [12]

This covariance matrix of the errors is important as it qualifies MRLR as an MRR model because it uses this to calculate the response covariance matrix, $\text{Cov}(\mathbf{Y}) = \mathbf{\Sigma_Y}$:

$$\begin{aligned} \mathbf{\Sigma_Y} &= \text{Cov}(\mathbf{XB} + \mathbf{E}) \\ &= \text{Cov}(\mathbf{XB}) + \text{Cov}(\mathbf{E}) \\ &= \mathbf{B}^\top \text{Cov}(\mathbf{X})\mathbf{B} + \text{Cov}(\mathbf{E}) \\ &= \mathbf{B}^\top \mathbf{\Sigma_X} \mathbf{B} + \mathbf{\Sigma_E}, \end{aligned} \tag{3.2}$$

where the linearity property of covariance does the first expansion and see A.2.1 for the simplification of $\text{Cov}(\mathbf{XB})$ to $\mathbf{B}^\top \text{Cov}(\mathbf{X})\mathbf{B}$.

If we used several independent single-response models, the off-diagonal elements of $\mathbf{\Sigma_E}$ would be

0. Still, this shared error correlation structure of MRLR ensures this is not the case and hence justifies MRLR as an MRR model.
The predictor matrix $\mathbf{X}$ must also have full column rank to ensure that the least squares estimator of $\mathbf{B}$ is well-defined.[12] This is used to determine the least squares estimates exclusively from the observations on the ith response.[12]

3.1.3 Multiple Response Selection Methods

Now that we have established MRLR and how it qualifies as an MRR model, we need to consider which predictors to use. Selecting the most significant predictors is crucial in evaluating model fit. There are always multiple ways to construct a model in a given situation, and we want to be able to judge which of those models is best. Here, we will use selection methods.

Stepwise selection includes forward, backward and bidirectional stepwise selection. These stepwise selection methods are used because they are more computationally efficient than best subset selection as they consider a smaller set of models. Forward stepwise selection starts with a model with no predictors, and it iteratively adds the most useful predictor one at a time. At each step, the variable with the greatest fit improvement is added to the model. Backward stepwise selection starts with the full model with all its predictors, and it iteratively removes the least useful predictor one at a time. Bi-directional stepwise selection combines forward selection and backward elimination. What changes from best subset selection is that this method considers the statistical impact of dropping variables that were considered previously.[14]

When using all of these selection methods, in single response regression, the "best" model is chosen for every number of predictors, and then the single best model is picked. The"best" model for each number of predictors is the one with the smallest residual sum of squares, or largest R^2. The single best model is then picked using a cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

However, this changes for MRR because the response covariance matrix needs to be considered. Therefore, we need to reevaluate our selection criteria. In this report, we will use multiple analysis of variance.

3.1.4 Analysis of Variance

ANOVA primarily analyses the differences among group means and their associated variances. It helps determine whether there is a significant difference between the means of different groups by comparing the variance within groups to the variance between groups.

It can also be used to evaluate predictor contribution to response variation. It does this by partitioning total response variance, or the total sum of squares (SST), into 2 components: the sum of squares of regression (SSR) and the sum of squares of error (SSE). The SSR is the response variation accounted for by the predictors. The SSE is the response variation not accounted for by the predictors. From their setup, these are all related via:

$$\text{SST} = \text{SSR} + \text{SSE}.$$

This decomposition is crucial in evaluating how much the predictors explain response variation. ANOVA results are displayed as a table which generates an F statistic, $F = \frac{\text{MSR}}{\text{MSE}}$, where MSR = SSR/df and MSE = SSE/df. This, in turn, generates a p-value and a low p-value (e.g. $p < 0.05$), and hence, the predictors significantly impact response variation.

Sequential ANOVA is a type of ANOVA. It decomposes the regression component of an ANOVA table by considering a sequence of nested models. This approach allows us to assess how the response variation not accounted for by predictors changes as predictors are successively added to the model. Here is how it works: first consider a sequence of m nested models $M_1 \subset M_2 \subset \cdots \subset M_m$, where each model M_j contains all predictors from M_{j-1} plus additional ones. Each model M_j is represented as:

$$M_j : \mathbb{E}[Y|X] = b_1 + b_2 X_2 + \cdots + b_{p_j} X_{p_j},$$

and the next model adds an extra predictor:

$$M_{j+1} : \mathbb{E}[Y|X] = b_1 + b_2 X_2 + \cdots + b_{p_j} X_{p_j} + b_{p_{j+1}} X_{p_{j+1}}.$$

Then, the RSS is computed for each model: $S_1 \geq S_2 \geq \cdots \geq S_m$, where the difference: $S_j - S_{j+1}$ represents the sum of squares explained by the newly added predictors. So, when constructing a model, ANOVA would add the predictor which corresponds to the largest difference: $S_j - S_{j+1}$.

3.1.5 Multiple Analysis of Variance

Multivariate Analysis of Variance (MANOVA) is an extension of ANOVA where multiple dependent variables can be analysed simultaneously.

MANOVA comes with some assumptions that should be confirmed before proceeding. The dependent variables need to have a multivariate normal distribution, which was justified in 2.3.5.[15] Also, homogeneity of variance must be the case for each dependent variable and the correlation between any two dependent variables must be the same in all groups of observations.

It tests if the means of several dependent variables differ across independent variables. MANOVA compares groups on a set of dependent variables simultaneously rather than conducting separate ANOVAs for each dependent variable.[16] This report uses MANOVA to assess the impact of the predictors across the joint variation in responses. It assesses this using variance-covariance matrices instead of scalar variance values.

The total response variation, or the total sum of squares and cross products (SSCP) matrix, $\mathbf{T}$, is partitioned into 2 matrices, like ANOVA. The first is the explained SSCP matrix, $\mathbf{H}$, representing the response variation explained by the predictor variable(s).[17] The second is the unexplained SSCP matrix, $\mathbf{W}$, representing the response variation not accounted for by the predictors.[17] By their set-up, $\mathbf{W}$ and $\mathbf{H}$ relate through the total SSCP matrix: $\mathbf{T} = \mathbf{H} + \mathbf{W}$.

MANOVA is suitable for MRR because the total SSCP matrix, $\mathbf{T}$, is an unscaled version of the response covariance matrix, i.e.:

$$\mathbf{T} = (n-1)\boldsymbol{\Sigma}_{\mathbf{Y}},$$

where n is the number of observations. This relation means MANOVA accounts for response intercorrelation by directly considering response covariances, which is ignored when using several independent univariate ANOVAs. Before preceding with these values, it is important to understand how they are computed.

From single-response ANOVA, $\mathbf{H}$ and $\mathbf{W}$ and $\mathbf{T}$ are extensions of SSE, SSR and SST, respectively. The unexplained SSCP matrix $\mathbf{W}$ is calculated as follows:

$$\mathbf{W} = (\mathbf{Y} - \hat{\mathbf{Y}})^\top (\mathbf{Y} - \hat{\mathbf{Y}}),$$

where $\mathbf{Y}$ is the $n \times m$ matrix of observed response values, and $\hat{\mathbf{Y}}$ is the $n \times m$ matrix of predicted response values from the model. $\hat{\mathbf{Y}}$ is computed using $\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{B}}$, where $\hat{\mathbf{B}}$ is the MRLR OLS estimator. The explained SSCP matrix $\mathbf{H}$ is calculated as follows:

$$\mathbf{H} = (\hat{\mathbf{Y}} - \bar{\mathbf{Y}})^\top (\hat{\mathbf{Y}} - \bar{\mathbf{Y}}),$$

where $\bar{\mathbf{Y}}$ is the overall mean response vector, $n \times 1$ given by $\bar{\mathbf{Y}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{Y}_i$.
The total SSCP matrix $\mathbf{T}$ is calculated as follows::

$$\mathbf{T} = (\mathbf{Y} - \bar{\mathbf{Y}})^\top (\mathbf{Y} - \bar{\mathbf{Y}}).$$

To test significance, MANOVA has multiple tests it can use. This report uses the Wilks' Lambda (Λ) test, which measures the proportion of variance not explained by the independent variables:

$$\Lambda = \frac{|\mathbf{W}|}{|\mathbf{T}|},$$

where $|\mathbf{W}|$ and $|\mathbf{T}|$ are the determinants of the unexplained and total SSCP matrices, respectively. A small Λ (close to 0) indicates the predictors explain a large proportion of response variation, and a large Λ (close to 1) indicates the predictors explain a small proportion of response variation.

The Wilks' Lambda value is useful because it can be transformed into an F-statistic:

$$F = \frac{(1 - \Lambda)/p}{\Lambda/(N - p - 1)}$$

where: p is the number of dependent variables and N is the total sample size. The null hypothesis in MANOVA is that all group means for the dependent variables are equal. So, a significant F-test ($p < 0.05$) suggests at least one of the dependent variables significantly differs across groups.[16]. However, this report uses the difference in Wilks' Lambda values as specified by Sequential MANOVA.

Sequential MANOVA extends sequential ANOVA by testing predictors one at a time in a nested sequence of models. It works the same as sequential ANOVA, but at each step, we compute Wilks' Lambda (Λ_j) for each model:

$$\Lambda_1 \geq \Lambda_2 \geq \cdots \geq \Lambda_m,$$

where the difference: $\Lambda_j - \Lambda_{j+1}$ is the additional variance explained by the newly added or removed predictor. A significant drop in Λ indicates that the new predictor significantly improves the model.[16]

Another advantage of using Wilks' Lambda is that it prevents the inflation of Type I errors. If multiple dependent variables are analysed separately using ANOVA, the probability of finding at least one false positive increases. Since MANOVA jointly examines multiple dependent variables, it controls for the increased risk of false positives by adjusting for intercorrelations among the responses.[16] This provides a more accurate assessment of significant predictors.

3.2 Application

3.2.1 Model Evaluation and Performance

Multiple models will be used and examined throughout this report. They will be compared using the average normalised root mean square error (ANRMSE). The ANRMSE is chosen because the root mean square error (RMSE) evaluates the model for said response variable in its units, while the

normalised root mean square error (NRMSE) returns a universal model evaluation. The average is then taken because there are 2 response variables. The ANRMSE gives the proportion of natural variability in the responses not accounted for by the model. An excellent model has a 0-0.5 ANRMSE, a good model has a 0.5-0.6 ANRMSE, a satisfactory model has a 0.6-0.7 ANRMSE, and an unsatisfactory model has a > 0.7 ANRMSE.[18] Now, a standard MRLR with the sum of predictors will be evaluated using this metric.

3.2.2 Standard Multiple Linear Regression

The first model we will evaluate on the equity fund dataset is an MRLR using all of the available predictors. The residuals from the model are used to calculate the ANRMSE, which was `0.7606`. This high value indicates that the model does not perform well on the equity fund dataset. This makes sense because the model was not built to understand the underlying complexity of the equity fund dataset, so it underfits the data. Stepwise selection will consider this more, and it has been adapted to the multiple response case using MANOVA.

3.2.3 Sequential MANOVA Stepwise Selection Code

All of the selection methods had to be manually coded because the `stepAIC` command, which does this in R, only applies to single response regression.

The selection process begins with different initial models depending on the chosen method. In forward selection, the model starts with no predictors. In backward selection, the model starts with all available predictors included. In bidirectional selection, predictors are added or removed at each step. At each iteration, the function evaluates the impact of adding (in forward selection) or removing (in backward selection) each predictor using 2 functions: `add_predictors` and `remove_predictors` respectively. This is done by computing the change in Wilks' Lambda (Λ), given by:

$$\Delta\Lambda = \Lambda_{\text{previous}} - \Lambda_{\text{new}},$$

where $\Lambda_{\text{previous}}$ represents the Wilks' Lambda value from the previous step, and Λ_{new} represents the value obtained after including or excluding a predictor.

The predictor that results in the largest reduction in Wilks' Lambda is chosen for inclusion (in forward selection) or removal (in backward selection). If the largest Wilks' Lambda difference is negative, meaning that adding or removing a predictor worsens the model, then no changes are made.

The process continues until no further improvement is observed. Specifically, if two consecutive iterations fail to improve the model, meaning that neither adding nor removing a predictor reduces Wilks' Lambda (in bidirectional selection) or all available predictors have been evaluated (in forward or backward selection), the stepwise selection terminates.

Once the stepwise selection process has been completed, all models selected during the process are evaluated using the ANRMSE via the function `calculate_anrmse`. This ensures that the final model is not only optimal in terms of Wilks' Lambda but also predicts well.

The final model is selected based on the lowest ANRMSE value. If a model with fewer predictors achieves a lower ANRMSE than a more complex model, the simpler model is chosen. This guarantees that the final model is both simple and highly effective at making accurate predictions.

For simplicity, this overall process will be called Sequential MANOVA Stepwise Selection in the rest of this paper. Now, it will be applied to a simpler dataset.

3.2.4 Small-Scale Example

Let us take a small-scale dataset evaluating correlated Mathematics and Science Scores against factors that may influence them:

X_1: Hours Studied	X_2: Time Spent on Papers	Y_1: Math Scores	Y_2: Science Scores
5	2	78	80
7	3	85	79
8	4	88	88
3	1	65	70
10	5	92	74

Table 3.1: Study Time vs. Exam Scores

Let us now walk through Sequential MANOVA Forward Stepwise Selection. The first step is evaluating each predictor one at a time to see which predictor gives the largest Wilks' lambda difference. Let us look at X_1 as a predictor to demonstrate how the Wilks' Lambda difference is computed.

The first step is computing the total, explained and hence unexplained SSCP matrices. First, compute the means to get $\bar{\mathbf{Y}} = [\bar{Y_1}, \bar{Y_2}]^\top$:

$$\bar{Y_1} = \frac{78+85+88+65+92}{5} = 81.6, \quad \bar{Y_2} = \frac{80+79+88+70+74}{5} = 78.2.$$

Now compute the deviations from the mean:

$Y_{1,i} - \bar{Y_1}$	$Y_{2,i} - \bar{Y_2}$
-3.6	1.8
3.4	0.8
6.4	9.8
-16.6	-8.2
10.4	-4.2

The total SSCP matrix, $\mathbf{T}$, is given by:

$$\mathbf{T} = \begin{bmatrix} \sum_i (Y_{1,i} - \bar{Y_1})^2 & \sum_i (Y_{1,i} - \bar{Y_1})(Y_{2,i} - \bar{Y_2}) \\ \sum_i (Y_{1,i} - \bar{Y_1})(Y_{2,i} - \bar{Y_2}) & \sum_i (Y_{2,i} - \bar{Y_2})^2 \end{bmatrix},$$

where i goes from 1 to 5, which is the number of observations. Now, compute each term:

$$\sum_i (Y_{1,i} - \bar{Y_1})^2 = 449.2, \quad \sum_i (Y_{2,i} - \bar{Y_2})^2 = 184.8, \quad \sum_i (Y_{1,i} - \bar{Y_1})(Y_{2,i} - \bar{Y_2}) = 634.$$

Thus,

$$\mathbf{T} = \begin{bmatrix} 449.2 & 634 \\ 634 & 184.8 \end{bmatrix}.$$

The next step is to compute the unexplained SSCP Matrix, $\mathbf{W}$, which uses a prediction from MRLR: $\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{B}}$. Here, we want to compute the OLS estimator for MRR, $\hat{\mathbf{B}}$. This is done like in single-response, by simply using $\hat{\mathbf{B}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$. In our case:

$$\mathbf{Y} = \begin{bmatrix} 78 & 85 & 88 & 65 & 92 \\ 80 & 79 & 88 & 70 & 74 \end{bmatrix}^\top \quad \text{and} \quad \mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 5 & 7 & 8 & 3 & 10 \end{bmatrix}^\top.$$

Remember, only X_1 is being used here because a model with 1 predictor is being evaluated first. Plugging these values into the formula for the OLS estimator, $\hat{\mathbf{B}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$, gives, to 2dp:

$$\hat{\mathbf{B}} = \begin{bmatrix} 56.47 & 72.23 \\ 3.81 & 0.90 \end{bmatrix}.$$

Therefore, the predicted values are: $\hat{Y}_{1,i} = 56.47 + 3.81X_{1,i}$ and $\hat{Y}_{2,i} = 72.23 + 0.90X_{1,i}$. Now, we can compute $\mathbf{W} = (\mathbf{Y} - \hat{\mathbf{Y}})^\top(\mathbf{Y} - \hat{\mathbf{Y}})$ to 4sf:

$$\mathbf{W} = \begin{bmatrix} 25.73 & 50.86 \\ 50.86 & 160.9 \end{bmatrix}.$$

Next, we compute the Wilks' Lambda value using the determinants of these matrices:

$$\Lambda = \frac{|\mathbf{W}|}{|\mathbf{T}|} = \frac{1553.08}{60090.2} = 0.0258,$$

This value is very low, which means the model including X_1 is explaining a large proportion of the variation in the responses, which is ideal.

Now, the difference between this and the previous Wilks' Lambda value is computed. So, the previous case is just when we have the column of ones in our design matrix, $\mathbf{X}$. This means the model only estimates the mean of each response, so no actual predictors are included to explain differences in $\mathbf{Y}$. Hence, the predicted values are the column means. As a result, the explained SSCP matrix, $\mathbf{H}$, is 0. Therefore, the unexplained SSCP matrix, $\mathbf{W} = \mathbf{T}$ and $\Lambda = 1$.

Computing the differences gives: 1-0.0258 = 0.9742. This is compared to the Wilks' Lambda Difference from using only X_2, and the predictor with the largest difference is added to the model.

3.2.5 Applying Stepwise Selection

Now, looking back to the equity fund dataset where `roe` and `sustainability_score` are the responses. Before applying Sequential MANOVA Stepwise Selection, the categorical variables were frequency encoded. Now, we are ready to analyse.

Both Forward and bi-directional Stepwise Selection lead to the same model with all of the predictors being kept. This model is the sum of all the predictors in the dataset and, hence, the same ANRMSE.

Backward elimination returned a different model to the other selection methods. The difference was that it did not include `risk_rating`. This gave a higher ANRMSE than the previous model with `0.7924`. It makes sense for the selection methods to potentially produce a different model because each selection method has different starting points and directions, which can lead to other paths through the model and, ultimately, a different set of selected predictors.

3.2.6 Extending and Evaluating Stepwise Selection

Although there are a large number of predictors in this dataset, extending the model to consider non-linear and interaction terms can seek to improve its performance. To include these, all that had to be done was making a new function which generated the new predictors. When generating the non-linear terms, the model improved as shown by the ANRMSE reducing to `0.6893`. Using the interaction terms further improved, and the ANRMSE was reduced to `0.5613`. This significance is noteworthy.

One counterargument is that these models overfit the data, but cross-validation (CV) prevents this. Another potential issue is that having too many predictors could be problematic because collecting

this information could be complex. However, all this data is already available.
This table below gives a simple outline of all the selection models evaluated in this chapter.

Model	**ANRMSE**
Model 1: MRLR with all predictors	0.7606
Model 2: Forward Selection	0.7606
Model 3: Backward Selection	0.7924
Model 4: Bidirectional Selection	0.7606
Model 5: Interaction Terms	0.5613
Model 6: Non-Linear Terms	0.6893

Table 3.2: Selection Methods and their ANRMSE Values

See A.3 for an example of Model 2: Forward Stepwise Selection in action.

Stepwise selection comes with multiple benefits. Firstly, it is easy to implement. Although manual code had to be made in this case, it proved to be easy to test on the dataset. They reduce the work of model building to the click of a button and often yield simple results.[19] Also, they can sift through lots of predictors to identify potential relationships, mainly when dealing with high-dimensional datasets.

However, they come with a few drawbacks. Firstly, they are prone to over-fitting. They can find idiosyncrasies in the population that do not exist.[19] Also, the stepwise selection method produces models that are highly sensitive to slight variations in the data, causing inconsistent results. Another issue of this method is inflated effect sizes. Even when the model identifies valid predictors, it over-estimates their effect sizes.[19] This means that the impact of significant predictors on the response variables `roe` and `sustainability_score` will be inflated.

Shrinkage Methods are another type of regression model that will be evaluated next in this report. None of the stepwise selection drawbacks apply to shrinkage methods. Shrinkage Methods address them by penalising the model coefficients, preventing over-fitting and reducing sensitivity to outliers.

Chapter 4 Shrinkage Methods

This chapter delves into Shrinkage Methods, focusing on Multivariate Response with Covariance Estimation and Reduced Rank Ridge Regression, which extend Lasso and Ridge Regression. First, it introduces the theory behind Lasso and Ridge Regression and then extends it to multiple responses. Finally, it delves into the theory of Multivariate Response with Covariance Estimation and Reduced Rank Ridge Regression and how it has been applied to the equity fund dataset.

4.1 Theory

The previous methods are known as discrete model search methods, where the best model is picked based on information criteria. The next type of model this report will look at is shrinkage methods, which are a continuous model search method. The whole model is trained but the estimated regression coefficients are minimised or zeroed, and there is a general solution of the form: model fit + penalty on coefficient size.

4.1.1 Ridge Regression

Ridge regression zeroes estimated regression coefficients, and its goal is to minimise the cost function:

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\text{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2,$$

where $\lambda \geq 0$ is a tuning parameter. The role of this parameter is crucial as it balances the trade-off between model fit and the size of the coefficients. The ridge regression cost function estimates are the solution to the least squares problem subject to the constraint $\sum_{j=1}^{p}\beta_j^2 \leq c$. A geometric interpretation of this interpretation is a L_2 ball, which is why ridge cannot penalise coefficient sizes completely to zero, even for irrelevant features. As λ increases, the penalty term gets larger, so in order to minimise the entire function (model fit + penalty), the regression coefficients will get smaller. Ridge regression minimises the cost function, which can be shown by differentiating the cost function with respect to $\boldsymbol{\beta}$, giving the closed-form coefficient estimator:

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}}=\left(\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I}_p\right)^{-1}\mathbf{X}^\top\mathbf{Y}.$$

Ridge regression comes with some benefits. It is useful when predictors have high multicollinearity and when p is close to n. It is also faster than the prior model-search methods as it only trains 1 model. Unfortunately, it also brings some drawbacks. Ridge regression does not carry out feature selection. Although it can indicate the coefficients which contribute the most, it cannot remove those coefficients entirely. Ideally, a simpler model would be created rather than including all of the predictors. Lasso Regression does exactly this.

4.1.2 Lasso Regression

Lasso regression can remove irrelevant features by shrinking their respective coefficients to zero. It penalises coefficient size through absolute values instead of squares. It also has the goal of minimising the cost function, but its formula differs:

$$\text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|,$$

Lasso minimises the cost function in a similar way to Ridge, and its cost function estimates are the solution to the least squares problem subject to the constraint $\sum_{j=1}^{p} |\beta_j| \leq c$. A geometric interpretation of this interpretation is a diamond, which is why lasso regression can penalise coefficient sizes to zero for irrelevant features.

Lasso and Ridge share some of the same benefits. They introduce bias but decrease variance, improving predictive performance. Lasso is also scalable and can work with high-dimensional data. Both methods can also deal with high multicollinearity. In this case, any multicollinearity has been sorted out in data preprocessing, but these regression models can also remove or reduce the impact of irrelevant features. The next section will discuss how they can be applied in a multivariate setting.

4.1.3 Shrinkage Methods for MRR

Previous discussions on shrinkage methods have primarily focused on the single-response case. However, they can be adapted for MRR. The extension of Ridge Regression to the multiple response setting involves the Frobenius norm of the coefficient matrix $\mathbf{B}$:

$$\min_{\mathbf{B}} \frac{1}{2} \|\mathbf{Y} - \mathbf{XB}\|_F^2 + \lambda \|\mathbf{B}\|_F^2,$$

where $\mathbf{Y}$ is the $n \times q$ matrix of response variables (for q responses), $\mathbf{B}$ is the $m \times q$ matrix of regression coefficients, and $\|\cdot\|_F^2$ is the Frobenius norm.[20] The Frobenius Norm gives the sum of squared residuals across all responses. The term $\|\mathbf{B}\|_F^2 = \sum_{j=1}^{q} \sum_{i=1}^{m} w_{ij}^2$ is the ridge penalty applied to all coefficients in $\mathbf{B}$, and λ is the regularisation parameter controlling the shrinkage. Here, the Frobenius norm replaces the standard ℓ_2-norm, instead the ℓ_2-norm penalty across the entire coefficient matrix $\mathbf{B}$.
The closed-form solution for $\mathbf{B}$ in multivariate ridge regression is:

$$\mathbf{B} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_m)^{-1} \mathbf{X}^\top \mathbf{Y}, \tag{4.1}$$

where $\mathbf{I}_m$ is the $m \times m$ identity matrix.

Similarly, Lasso regression can also be extended to the multiple response setting using the ℓ_1-norm penalty:

$$\min_{\mathbf{B}} \frac{1}{2} \|\mathbf{Y} - \mathbf{XW}\|_1^2 + \lambda \|\mathbf{B}\|_1,$$

where $\|\mathbf{B}\|_1 = \sum_{j=1}^{m} |b_j|$ represents the ℓ_1-norm penalty.[21] The ℓ_1-penalty shrinks some elements of $\mathbf{B}$ to exactly zero, enabling variable selection in the regression model like in the single response case.

Despite their flexibility to many different datasets, these methods have a major problem in the multivariate case. They do not consider the intercorrelation between responses when making predictions, which is extremely important for MRR. However, there are extensions of these methods that do. Reduced rank ridge regression and multivariate response covariance estimation are two extensions

that consider this. Over the next few sections, their methodology will be explained, including how they each consider response intercorrelation.

4.1.4 Reduced Rank Ridge Regression

The MRLR formula tells us: $\mathbf{Y}_{(n\times m)} = \mathbf{X}_{(n\times p)}\mathbf{B}_{(p\times m)} + \mathbf{E}_{(n\times m)}$, where the coefficient matrix is estimated using the OLS estimate: $\hat{\mathbf{B}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$.

Reduced rank ridge regression (RRRR) changes this coefficient matrix, $\mathbf{B}$, estimation using ridge regression and a rank constraint. It aims to solve this optimisation problem:

$$\hat{\mathbf{B}}(\lambda, r) = \arg\min_{\mathbf{B}:\text{rank}(\mathbf{B})\leq r} \|\mathbf{Y} - \mathbf{X}\mathbf{B}\|_F^2 + \lambda\|\mathbf{B}\|_F^2,$$

where $r \leq \min\{p,m\}$ forces $\mathbf{B}$ to be low-rank and λ adds a ridge penalty.

Before solving this problem, we assume that $\mathbf{X}$ and $\mathbf{Y}$ have been mean-centred, meaning that each column of $\mathbf{X}$ and $\mathbf{Y}$ has zero mean. The reason this is done will be covered more in-depth later.

The first step involves transforming this problem into a standard Reduced Rank Regression problem on an augmented dataset, incorporating the Ridge penalty.[22] This transformation is important because it transforms the RRRR problem into a standard Reduced Rank Regression problem, allowing it to be solved more efficiently using Singular Value Decomposition (SVD) rather than solving the optimisation problem directly. This transformation is done for each fixed λ because it is not known in advance what the best λ is, and this is derived using CV in this report.

The first step involves augmenting the dataset:

$$\mathbf{X}^*_{(n+p)\times p} = \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{I} \end{pmatrix}, \quad \mathbf{Y}^*_{(n+p)\times m} = \begin{pmatrix} \mathbf{Y} \\ \mathbf{0} \end{pmatrix}.$$

This augmentation is equivalent to using the closed form estimator form of a ridge, as shown in Equation 4.1, but augmenting is more efficient. The augmentation gives a new objective function:

$$\hat{\mathbf{B}}(\lambda, r) = \arg\min_{\text{rank}(\mathbf{B})\leq r} \|\mathbf{Y}^* - \mathbf{X}^*\mathbf{B}\|_F^2.$$

The solution is equivalent to an orthogonal projection onto the top r singular vectors of the ridge-adjusted response matrix as established by the Eckhart–Young theorem. As a result, the OLS estimate has an orthogonal projection property, which means the squared error loss function can be decomposed into 2 parts:

$$\|\mathbf{Y}^* - \mathbf{X}^*\mathbf{B}\|_F^2 = \|\mathbf{Y}^* - \hat{\mathbf{Y}}^*_R\|_F^2 + \|\hat{\mathbf{Y}}^*_R - \mathbf{X}^*\mathbf{B}\|_F^2,$$

where $\hat{\mathbf{Y}}^*_R = \mathbf{X}^*\hat{\mathbf{B}}^*_R$ denotes the Ridge Regression estimate.[22] The first term of this equation does not involve $\mathbf{B}$ so the objective function can be further simplified into:

$$\hat{\mathbf{B}}(\lambda, r) = \arg\min_{\text{rank}(\mathbf{B})\leq r} ||\hat{\mathbf{Y}}^*_R - \mathbf{X}^*\mathbf{B}||_F^2. \tag{4.2}$$

The next step involves breaking down the matrix $\hat{\mathbf{Y}}^*_R$ This starts by taking its SVD:

$$\hat{\mathbf{Y}}^*_R = \sum_{i=1}^{\tau} d_i\mathbf{u}_i\mathbf{v}_i^\top,$$

where the d_i's denote the singular values, $\mathbf{u}_i \in \mathbb{R}^{n\times 1}$ and $\mathbf{v}_i \in \mathbb{R}^{m\times 1}$ denote the left and right

singular vectors of $\hat{\mathbf{Y}}_R^*$, respectively.[22] Then, the Eckhart-Young Theorem says that the best rank r approximation to $\hat{\mathbf{Y}}_R^*$ in the Frobenius norm is given by:

$$\hat{\mathbf{Y}}_r^* = \sum_{i=1}^{r} d_i \mathbf{u}_i \mathbf{v}_i^\top.^{22}$$

RRRR captures response intercorrelation through $\mathbf{v}_i$. This can be proven by relating $\mathbf{Y}^\top\mathbf{Y}$ and the sample response covariance, $\mathbf{\Sigma_Y}$, which is defined as:

$$\mathbf{\Sigma_Y} = \frac{1}{n-1}(\mathbf{Y} - \bar{\mathbf{Y}})^\top(\mathbf{Y} - \bar{\mathbf{Y}}) = \frac{1}{n-1}\mathbf{Y}^\top\mathbf{Y}.$$

The last equality holds because we have mean-centred $\mathbf{Y}$. This ensures that RRRR captures response intercorrelation independently of the mean structure $\bar{\mathbf{Y}}\bar{\mathbf{Y}}^\top$. See A.2.2 for how $\mathbf{\Sigma_Y}$ and $\mathbf{Y}^\top\mathbf{Y}$ would relate without mean centring.

This is important because the SVD of $\mathbf{Y}$ in matrix form is: $\mathbf{Y} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$. Now computing $\mathbf{Y}^\top\mathbf{Y}$:

$$\begin{aligned}\mathbf{Y}^\top\mathbf{Y} &= \mathbf{V}\mathbf{D}^\top\mathbf{U}^\top\mathbf{U}\mathbf{D}\mathbf{V}^\top \\ &= \mathbf{V}\mathbf{D}^2\mathbf{V}^\top,\end{aligned}$$

where the final equality comes because $\mathbf{U}$ is orthonormal, so $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ and $\mathbf{D} = \mathbf{D}^\top$.[23] This means $\mathbf{V}$ clearly diagonalises $\mathbf{Y}^\top\mathbf{Y}$. Therefore, it directly represents the principal directions of response covariance. Thus, the columns of $\mathbf{V}$, $\mathbf{v}_i$, capture response intercorrelation.

The next step to RRRR is to define the projection matrix, $\mathbf{P}_r$, which projects the predictions onto an r-dimensional subspace: $\mathbf{P}_r = \sum_{i=1}^{r} \mathbf{v}_i\mathbf{v}_i^\top$.

Let $\hat{\mathbf{B}}(\lambda, r) = \hat{\mathbf{B}}_R^*\mathbf{P}_r$ and clearly, $\text{rank}(\hat{\mathbf{B}}(\lambda, r)) \leq r$, as $\text{rank}(\mathbf{P}_r) = r$.[22] Plugging them into 4.2:

$$\mathbf{X}^*\hat{\mathbf{B}}(\lambda, r) = \mathbf{X}^*\hat{\mathbf{B}}_R^*\mathbf{P}_r = \left(\sum_{i=1}^{\tau} d_i\mathbf{u}_i\mathbf{v}_i^\top\right)\left(\sum_{j=1}^{r}\mathbf{v}_j\mathbf{v}_j^\top\right) = \sum_{i=1}^{r} d_i\mathbf{u}_i\mathbf{v}_i^\top = \hat{\mathbf{Y}}_r^*.$$

Thus, showing that the proposed solution $\hat{\mathbf{B}}(\lambda, r) = \hat{\mathbf{B}}_R^*\mathbf{P}_r$ is the minimiser of 4.2 and hence the RRRR optimisation problem. Writing it explicitly in terms of $\mathbf{X}, \mathbf{Y}, \lambda$, and r, gives the following:

$$\hat{\mathbf{B}}(\lambda, r) = \hat{\mathbf{B}}_R^*\mathbf{P}_r = (\mathbf{X}^{*\top}\mathbf{X}^*)^{-1}\mathbf{X}^{*\top}\mathbf{Y}\mathbf{P}_r = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{Y}\mathbf{P}_r.^{22}$$

Therefore, $\hat{\mathbf{Y}}(\lambda, r) = \mathbf{X}\hat{\mathbf{B}}(\lambda, r) = \mathbf{X}\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{Y}\mathbf{P}_r = \hat{\mathbf{Y}}_\lambda\mathbf{P}_r$.

From this estimate, it is clear to see how the ridge closed-form estimator is involved. The rank r comes in because $\hat{\mathbf{Y}}_\lambda$ is normally in a m-dimensional space, but the projection matrix sends it to a lower r-dimensional space.

While RRRR models response relationships through a low-rank coefficient matrix, another approach is to explicitly model the inverse error covariance matrix, which captures conditional dependencies between responses. This forms the basis of Multivariate Regression with Covariance Estimation, where both the regression coefficients and the inverse error covariance matrix are jointly estimated.

4.1.5 Multivariate Response with Covariance Estimation

Multivariate Response with Covariance Estimation (MRCE) builds upon MRLR by jointly estimating the regression coefficient matrix $\mathbf{B}$ and the inverse error covariance matrix $\mathbf{\Omega} = \mathbf{\Sigma_E}^{-1}$.

Before solving this problem, we again assume that $\mathbf{X}$ and $\mathbf{Y}$ have been mean-centred. This is because of the effect this has when estimating $\boldsymbol{\Omega}$.
The estimated covariance matrix is:

$$\boldsymbol{\Sigma}_{\mathbf{E}} = \frac{1}{n}(\mathbf{Y} - \mathbf{X}\hat{\mathbf{B}})^\top(\mathbf{Y} - \mathbf{X}\hat{\mathbf{B}}),$$

which is then inverted to obtain the precision matrix $\boldsymbol{\Omega}$.

If $\mathbf{X}$ and $\mathbf{Y}$ are not mean-centered, the residuals $\mathbf{E}$ will contain mean effects, leading to incorrect estimation of $\boldsymbol{\Sigma}_{\mathbf{E}}$:

$$\boldsymbol{\Sigma}_{\mathbf{E}} \approx (\mathbf{Y} - \mathbf{X}\hat{\mathbf{B}})^\top(\mathbf{Y} - \mathbf{X}\hat{\mathbf{B}}) + \bar{\mathbf{Y}}\bar{\mathbf{Y}}^\top.$$

The term $\bar{\mathbf{Y}}\bar{\mathbf{Y}}^\top$, which is the mean structure in $\mathbf{Y}$, adds bias to $\boldsymbol{\Sigma}_{\mathbf{E}}$ like for RRRR.

In MRLR, the negative log-likelihood function of $(\mathbf{B}, \boldsymbol{\Omega})$ can be expressed up to a constant as:

$$g(\mathbf{B}, \boldsymbol{\Omega}) = \operatorname{tr}\left[\frac{1}{n}(\mathbf{Y} - \mathbf{X}\mathbf{B})^\top(\mathbf{Y} - \mathbf{X}\mathbf{B})\boldsymbol{\Omega}\right] - \log|\boldsymbol{\Omega}|.^{24}$$

The MRCE estimation for $\mathbf{B}$ involves adding 2 penalties to the negative log-likelihood function g to construct a sparse estimator of $\mathbf{B}$ depending on $\boldsymbol{\Omega}$:

$$(\hat{\mathbf{B}}, \hat{\boldsymbol{\Omega}}) = \arg\min_{\mathbf{B},\boldsymbol{\Omega}} \left\{ g(\mathbf{B}, \boldsymbol{\Omega}) + \lambda_1 \sum_{j' \neq j} |\omega_{j'j}| + \lambda_2 \sum_{j=1}^{p} \sum_{k=1}^{q} |b_{jk}| \right\}, \tag{4.3}$$

where $\omega_{j'j}$ are the elements of $\boldsymbol{\Omega}$, b_{jk} are the elements of $\mathbf{B}$, λ_1 controls sparsity in $\boldsymbol{\Omega}$ and λ_2 for $\mathbf{B}$.[24] A sparse estimator of $\mathbf{B}$ forces many coefficients to zero, improving feature selection and efficiency.

4.3 can be solved in 2 ways: by solving it for $\hat{\boldsymbol{\Omega}}$ with $\mathbf{B}$ fixed at a chosen $\mathbf{B}_0$ and for $\hat{\mathbf{B}}$ with $\boldsymbol{\Omega}$ fixed at a chosen $\boldsymbol{\Omega}_0$:

1. This is the formula for $\hat{\mathbf{B}}$, fixing $\hat{\boldsymbol{\Omega}}$ as $\boldsymbol{\Omega}_0$:

$$\hat{\mathbf{B}}(\boldsymbol{\Omega}_0) = \arg\min_{\mathbf{B}} \left\{ \operatorname{tr}\left[\frac{1}{n}(\mathbf{Y} - \mathbf{X}\mathbf{B})^\top(\mathbf{Y} - \mathbf{X}\mathbf{B})\boldsymbol{\Omega}_0\right] + \lambda_2 \sum_{j,k} |b_{jk}| \right\}. \tag{4.4}$$

2. This is the formula for $\boldsymbol{\Omega}$, fixing $\mathbf{B}$ as $\mathbf{B}_0$:

$$\hat{\boldsymbol{\Omega}}(\mathbf{B}_0) = \arg\min_{\boldsymbol{\Omega}} \left\{ \operatorname{tr}\left(\frac{1}{n}(\mathbf{Y} - \mathbf{X}\mathbf{B}_0)^\top(\mathbf{Y} - \mathbf{X}\mathbf{B}_0)\boldsymbol{\Omega}\right) - \log|\boldsymbol{\Omega}| + \lambda_1 \sum_{j' \neq j} |\omega_{j'j}| \right\}. \tag{4.5}$$

Before delving further into this problem, let us talk about how this model qualifies as MRR and so, how it considers response intercorrelation.

Since MRCE estimates both $\mathbf{B}$ and $\boldsymbol{\Omega}$ simultaneously, the coefficient matrix is optimised considering response covariance. This reasoning follows from the explanation for why MRLR qualifies as an MRR. See equation 3.2 for how $\boldsymbol{\Sigma}_{\mathbf{Y}}$, $\mathbf{B}$ and $\boldsymbol{\Sigma}_{\mathbf{E}}$ relate.

The error covariance matrix $\boldsymbol{\Sigma}_{\mathbf{E}}$ describes marginal dependencies between response variables, whereas the inverse error covariance matrix $\boldsymbol{\Omega} = {\boldsymbol{\Sigma}_{\mathbf{E}}}^{-1}$ describes conditional dependencies.[25] This means that if $\Omega_{ij} = 0$, then responses Y_i and Y_j are conditionally independent given all other responses and, if $\Omega_{ij} \neq 0$, then responses Y_i and Y_j have a dependency not accounted for by the

predictors.[25] Thus, MRCE does not assume independent residuals but instead models how responses are related even after accounting for predictors.

4.4 and 4.5 both have distinct ways to be solved. 4.4 is solved by the following algorithm. Let us call it Algorithm 1. This algorithm comes from Rothman et al. (2010) and needs some set-up: given $\mathbf{\Omega}$ and an initial value $\hat{\mathbf{B}}^{(0)}$, let $\mathbf{S} = \mathbf{X}^\top\mathbf{X}$ and $\mathbf{H} = \mathbf{X}^\top\mathbf{Y}\mathbf{\Omega}$.

1. For the m-th iteration, set $\hat{\mathbf{B}}^{(m)} \leftarrow \hat{\mathbf{B}}^{(m-1)}$. Visit all entries of $\hat{\mathbf{B}}^{(m)}$ in some sequence, and for entry (r, c), update $\hat{b}_{rc}^{(m)}$ with the minimiser of the objective function along its coordinate direction given by,

$$\hat{b}_{rc}^{(m)} \leftarrow \text{sign}\left(\hat{b}_{rc}^{(m)} + \frac{h_{rc} - u_{rc}}{s_{rr}\omega_{cc}}\right)\left(\left|\hat{b}_{rc}^{(m)} + \frac{h_{rc} - u_{rc}}{s_{rr}\omega_{cc}}\right| - \frac{n\lambda_2}{s_{rr}\omega_{cc}}\right)_+,$$

   where $u_{rc} = \sum_{j=1}^{p}\sum_{k=1}^{q}\hat{b}_{jk}^{(m)}s_{rj}\omega_{kc}$ and $(a)_+$ means taking $\max(a, 0)$.

2. If $\sum_{j,k}\left|\hat{b}_{jk}^{(m)} - \hat{b}_{jk}^{(m-1)}\right| < \epsilon\sum_{j,k}\left|\hat{b}_{jk}^{\text{ridge}}\right|$, then stop, otherwise go to the prior step. Here: $\mathbf{B}^{\text{ridge}}$ is the ridge closed-form estimator of the coefficient matrix. See equation 4.1 for this estimator.

4.5 is solved using the graphical lasso (glasso) algorithm because it is fast and the most commonly used method to solve it.[24] This algorithm, from Friedman et al. (2008), is as follows:

1. Initialise $\mathbf{\Omega}$ with: $\mathbf{\Omega}^{(0)} = (\mathbf{\Sigma}_{\mathbf{E}}^{(m)} + \lambda_1\mathbf{I})^{-1}$, where

$$\mathbf{\Sigma}_{\mathbf{E}}^{(m)} = \frac{1}{n}(\mathbf{Y} - \mathbf{X}\hat{\mathbf{B}}^{(m)})^\top(\mathbf{Y} - \mathbf{X}\hat{\mathbf{B}}^{(m)}),$$

   is the initial error covariance matrix.

2. For the k-th iteration and for each $j = 1, 2, \ldots, q$ in $\mathbf{\Omega}^{(k)}$, solve the graphical lasso regression problem using the previous iteration $\mathbf{\Omega}^{(k-1)}$:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}}\frac{1}{2}\|\mathbf{\Omega}_{-j,j}^{(k-1)} - \mathbf{\Omega}_{-j,-j}^{(k-1)}\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1,$$

   where $\mathbf{\Omega}_{-j,-j}^{(k-1)}$ is the $q-1$ vector containing all elements in the j-th column except for the diagonal element $\Omega_{jj}^{(k-1)}$ and $\boldsymbol{\beta}$ is an estimate of $\mathbf{\Omega}_{-j,j}^{(k)}$ by solving the lasso regression problem using $\mathbf{\Omega}^{(k-1)}$.

3. Update $\mathbf{\Omega}$ by filling in the corresponding row and column using:

$$\mathbf{\Omega}_{-j,j}^{(k)} = -\mathbf{\Omega}_{-j,-j}^{(k)}\hat{\boldsymbol{\beta}}.$$

4. Repeat steps 2 and 3 until convergence, so when $||\mathbf{\Omega}^{(k)} - \mathbf{\Omega}^{(k-1)}||_\infty < \epsilon$, where ϵ is chosen.

Now, we are ready to outline the algorithm for MRCE from Rothman et al. (2010). For fixed values of λ_1 and λ_2, initialise $\hat{\mathbf{B}}^{(0)} = \mathbf{0}$ and $\hat{\mathbf{\Omega}}^{(0)} = \hat{\mathbf{\Omega}}(\hat{\mathbf{B}}^{(0)})$.

1. Compute $\hat{\mathbf{B}}^{(m+1)} = \hat{\mathbf{B}}(\hat{\mathbf{\Omega}}^{(m)})$ by solving Equation 4.4 using Algorithm 1.

2. Compute $\hat{\mathbf{\Omega}}^{(m+1)} = \hat{\mathbf{\Omega}}(\hat{\mathbf{B}}^{(m+1)})$ by solving Equation 4.5 using the glasso algorithm.

3. If $\sum_{j,k}\left|\hat{b}_{jk}^{(m+1)}-\hat{b}_{jk}^{(m)}\right|<\epsilon\sum_{j,k}\left|\hat{b}_{jk}^{\text{RIDGE}}\right|$ then stop; otherwise, go to Step 1.

This process uses blockwise descent to compute a local solution for 4.3 with steps 1 and 2 decreasing the objective function value.[24]

MRCE performance also depends on the regularisation parameters, which are selected via CV:

$$(\lambda_1^*,\lambda_2^*)=\arg\min_{\lambda_1,\lambda_2}\sum_{k=1}^{K}\|\mathbf{Y}^{(k)}-\mathbf{X}^{(k)}\mathbf{B}_{\lambda_1,\lambda_2}^{(-k)}\|_F^2,$$

where $\mathbf{Y}^{(k)}$ is the matrix of responses with observations in the kth fold, $\mathbf{X}^{(k)}$ is the matrix of predictors of observations in the kth fold, and $\mathbf{B}_{\lambda_1,\lambda_2}^{(-k)}$ is the estimated regression coefficient matrix computed with observations outside the kth fold, with tuning parameters λ_1 and λ_2.[24]
The next section will talk about how these 2 methods were evaluated on the equity fund dataset.

4.2 Application

4.2.1 Small-Scale Example

Let us now apply both of these shrinkage methods to a small dataset before proceeding to the equity fund dataset. We will use the familiar Mathematics and Science scores data:

X_1: Hours Studied	X_2: Time Spent on Papers	Y_1: Math Scores	Y_2: Science Scores
5	2	78	80
7	3	85	79
8	4	88	88
3	1	65	70
10	5	92	74

Table 4.1: Study Time vs. Exam Scores

Reduced Rank Ridge Regression

This example will show a rank-1 reduction of RRRR using $\lambda=0.1$ as the shrinkage parameter.

Based on our prior justification of RRRR as an MRR model, mean centring is essential to prevent biasing from the mean structure. The initial predictor matrix, $\mathbf{X}$, and initial response matrix $\mathbf{Y}$ are:

$$\mathbf{X}=\begin{bmatrix}5&7&8&3&10\\2&3&4&1&5\end{bmatrix}^\top,\quad \mathbf{Y}=\begin{bmatrix}78&85&88&65&92\\80&79&88&70&74\end{bmatrix}^\top.$$

After mean-centering, i.e. $\tilde{\mathbf{X}}=\mathbf{X}-\bar{\mathbf{X}}$ and $\tilde{\mathbf{Y}}=\mathbf{Y}-\bar{\mathbf{Y}}$, the predictor and response matrices become:

$$\tilde{\mathbf{X}}=\begin{bmatrix}-1.6&0.4&1.4&-3.6&3.4\\-1&0&1&-2&2\end{bmatrix}^\top,\quad \tilde{\mathbf{Y}}=\begin{bmatrix}-3.6&3.4&6.4&-16.6&10.4\\1.8&0.8&9.8&-8.2&-4.2\end{bmatrix}^\top.$$

We apply reduced rank ridge regression with $\lambda=0.01$ by first incorporating ridge regularisation, which we define as:

$$\mathbf{X}^*=\begin{bmatrix}\tilde{\mathbf{X}}\\\sqrt{\lambda}\mathbf{I}\end{bmatrix},\quad \mathbf{Y}^*=\begin{bmatrix}\tilde{\mathbf{Y}}\\\mathbf{0}\end{bmatrix}.$$

Since $\lambda = 0.01$, we compute $\sqrt{\lambda} = \sqrt{0.01} = 0.1$, so the augmented matrices are:

$$\mathbf{X}^* = \begin{bmatrix} -1.6 & 0.4 & 1.4 & -3.6 & 3.4 & 0.1 & 0 \\ -1 & 0 & 1 & -2 & 2 & 0 & 0.1 \end{bmatrix}^\top, \quad \mathbf{Y}^* = \begin{bmatrix} -3.6 & 3.4 & 6.4 & -16.6 & 10.4 & 0 & 0 \\ 1.8 & 0.8 & 9.8 & -8.2 & -4.2 & 0 & 0 \end{bmatrix}^\top.$$

The Ridge penalty is incorporated into $\mathbf{X}^*$ through $\sqrt{\lambda}\mathbf{I}$. The augmented system behaves like a standard regression problem, meaning we solve for $\mathbf{B}$ using the standard OLS formula. Let us denote this as $\hat{\mathbf{B}}_R^*$ as a reminder that we are incorporating ridge regression.

This formula is: $\hat{\mathbf{B}}_R^* = (\mathbf{X}^{*\top}\mathbf{X}^*)^{-1}\mathbf{X}^{*\top}\mathbf{Y}^*$. After subbing in $\mathbf{X}^*$ and $\mathbf{Y}^*$, this gives us the augmented coefficient matrix, $\hat{\mathbf{B}}_R^*$ (to 2dp) as:

$$\hat{\mathbf{B}}_R^* = \begin{bmatrix} 7.40 & -2.28 \\ -6.18 & 5.47 \end{bmatrix}$$

Thus, the ridge predictions are (to 2dp):

$$\hat{\mathbf{Y}}_R^* = \mathbf{X}^*\hat{\mathbf{B}}_R^* = \begin{bmatrix} -5.67 & 2.96 & 4.19 & -14.29 & 12.81 & 0.74 & -0.62 \\ -1.82 & -0.91 & 2.28 & -2.73 & 3.19 & -0.23 & 0.55 \end{bmatrix}^\top$$

The next step is to perform singular value decomposition on this prediction, i.e. $\hat{\mathbf{Y}}_R^* = \mathbf{U}\mathbf{D}\mathbf{V}^\top$, with singular values (to 2dp) of 21.21 and 2.29, giving:

$$\mathbf{D} = \begin{bmatrix} 21.21 & 0 \\ 0 & 2.29 \end{bmatrix}.$$

For full derivation of $\mathbf{D}, \mathbf{U}$ and $\mathbf{V}$, see A.2.2.

Since $D_1 = 21.21 \gg D_2 = 2.29$, the best rank ($r$=1) approximation is taken on the first column of $\mathbf{D}$ and so, take the first columns of $\mathbf{U}$ and $\mathbf{V}$. Therefore, $\hat{\mathbf{Y}}_1^* \approx d_1\mathbf{u}_1\mathbf{v}_1^\top$, where (to 2dp):

$$d_1 = 21.21, \quad \mathbf{u}_1 = \begin{bmatrix} -0.28 & 0.13 & 0.22 & -0.69 & 0.62 & 0.03 & -0.02 \end{bmatrix}^\top, \quad \mathbf{v}_1 = \begin{bmatrix} 0.97 & 0.22 \end{bmatrix}^\top.$$

However, this is just an intermediary step as we still need to rescale the coefficient matrix. This is done through the projection matrix, $\mathbf{P}_r$ (to 2dp):

$$\mathbf{P}_r = \mathbf{v}_1\mathbf{v}_1^\top = \begin{bmatrix} 0.95 & 0.22 \\ 0.22 & 0.05 \end{bmatrix}$$

Finally, the reduced rank ridge regression estimator is:

$$\hat{\mathbf{B}}_r = \hat{\mathbf{B}}_R^*\mathbf{P}_r = \begin{bmatrix} 6.54 & 1.49 \\ -4.68 & -1.07 \end{bmatrix}$$

Now, we can do the final predicted values, which are $\hat{\mathbf{Y}} = \tilde{\mathbf{X}}^*\hat{\mathbf{B}}_r$, giving (to 2dp):

$$\hat{\mathbf{Y}} = \begin{bmatrix} -5.78 & 2.62 & 4.47 & -14.18 & 12.87 \\ -1.32 & 0.60 & 1.02 & -3.24 & 2.94 \end{bmatrix}^\top$$

Remember, these predictions are on the **mean centered Y**. We can remove this mean centring, but that does not affect the model fit evaluation using the ANRMSE.

Multivariate Response with Covariance Estimation

Now, we perform an MRCE iteration on Table 4.1 with $\lambda_1 = \lambda_2 = 0.1$. Remember we want to mean centre $\mathbf{X}$ and $\mathbf{Y}$, so let us use $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ from the RRRR calculation:

$$\tilde{\mathbf{X}} = \begin{bmatrix} -1.6 & 0.4 & 1.4 & -3.6 & 3.4 \\ -1 & 0 & 1 & -2 & 2 \end{bmatrix}^\top, \quad \tilde{\mathbf{Y}} = \begin{bmatrix} -3.6 & 3.4 & 6.4 & -16.6 & 10.4 \\ 1.8 & 0.8 & 9.8 & -8.2 & -4.2 \end{bmatrix}^\top.$$

From here on, $\mathbf{X} = \tilde{\mathbf{X}}$ and $\mathbf{Y} = \tilde{\mathbf{Y}}$. This is to make notation easier. Let us first compute $\hat{\mathbf{B}}^{(\text{ridge})}$ because that is used in determining when Algorithm 1 stops:

$$\hat{\mathbf{B}}^{(\text{ridge})} = (\mathbf{X}^\top\mathbf{X} + \lambda_2\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{Y} = \begin{bmatrix} 5.07 & -0.77 \\ -2.19 & 2.89 \end{bmatrix}$$

For Algorithm 1, initialise $\hat{\mathbf{B}}^{(0)} = \mathbf{0}$ and $\hat{\boldsymbol{\Omega}}^{(0)} = \hat{\boldsymbol{\Omega}}(\hat{\mathbf{B}}^{(0)})$. To compute $\hat{\boldsymbol{\Omega}}^{(0)}$, first do the error, $\mathbf{E}^{(0)}$:

$$\mathbf{E}^{(0)} = \mathbf{Y} - \mathbf{X}\hat{\mathbf{B}}^{(0)} = \begin{bmatrix} -3.6 & 3.4 & 6.4 & -16.6 & 10.4 \\ 1.8 & 0.8 & 9.8 & -8.2 & -4.2 \end{bmatrix}^\top.$$

Now, compute the residual covariance matrix $\boldsymbol{\Sigma}_{\mathbf{E}}^{(\mathbf{0})}$:

$$\boldsymbol{\Sigma}_{\mathbf{E}}^{(\mathbf{0})} = \frac{1}{n}\mathbf{E}^{(0)^\top}\mathbf{E}^{(0)} = \begin{bmatrix} 89.84 & 30.28 \\ 30.28 & 36.96 \end{bmatrix}$$

Next, add this to a regularisation parameter $\lambda_2 = 0.1$. The regularisation parameter avoids dividing by zero when inverting $\boldsymbol{\Sigma}_{\mathbf{E}}{}^{(0)}$ for the precision matrix (to 2sf):

$$\boldsymbol{\Omega}^{(0)} = \begin{bmatrix} 0.015 & -0.013 \\ -0.013 & 0.037 \end{bmatrix}$$

Now compute $\hat{\mathbf{B}}^{(1)} = \hat{\mathbf{B}}(\hat{\boldsymbol{\Omega}}^{(0)})$ by solving Equation 4.4 using Algorithm 1. The manual calculation of this will be shown on 1 element of $\mathbf{B}^{(1)}$: $\hat{b}_{11}^{(1)}$. First compute $\mathbf{H}$ and $\mathbf{S}$:

$$\mathbf{S} = \mathbf{X}^\top\mathbf{X} = \begin{bmatrix} 29.2 & 17 \\ 17 & 10 \end{bmatrix}, \quad \mathbf{H} = \mathbf{X}^\top\mathbf{Y}\boldsymbol{\Omega}^{(0)} = \begin{bmatrix} 1.37 & -0.41 \\ 0.78 & -0.21 \end{bmatrix}.$$

Now set $\hat{\mathbf{B}}^{(1)} = \hat{\mathbf{B}}^{(0)}$ and update $\hat{b}_{11}^{(1)}$ with the minimiser of the objective function along its coordinate direction given by:

$$\hat{b}_{11}^{(1)} \leftarrow \text{sign}\left(\hat{b}_{11}^{(1)} + \frac{h_{11} - u_{11}}{s_{11}\omega_{11}}\right)\left(\left|\hat{b}_{11}^{(1)} + \frac{h_{11} - u_{11}}{s_{11}\omega_{11}}\right| - \frac{n\lambda_2}{s_{11}\omega_{11}}\right)_+,$$

where $u_{11} = \sum_{j=1}^{2}\sum_{k=1}^{2}\hat{b}_{jk}^{(1)}s_{1j}\omega_{k1}$. After subbing in and computation, this gives: $\hat{b}_{11} = 1.95$. Doing this on every element, we get:

$$\hat{\mathbf{B}}^{(1)} = \begin{bmatrix} 1.95 & 0 \\ 0 & 0 \end{bmatrix}.$$

The next step of Algorithm 1 involves checking for convergence of $\hat{\mathbf{B}}$ relative to $\hat{\mathbf{B}}^{(\text{ridge})}$. Here, if:

$$\sum_{j,k} \left| \hat{b}_{jk}^{(1)} - \hat{b}_{jk}^{(0)} \right| < \epsilon \sum_{j,k} \left| \hat{b}_{jk}^{\text{ridge}} \right|,$$

then stop; otherwise, go to the prior step. The left-hand side simplifies too:

$$\sum_{j,k} \left| \hat{b}_{jk}^{(1)} - \hat{b}_{jk}^{(0)} \right| = \sum_{j,k} |\hat{b}_{jk}^{(1)}| = 1.95.$$

And the right-hand side, too:

$$\epsilon \sum_{j,k} \left| \hat{b}_{jk}^{\text{ridge}} \right| = 10.92\epsilon.$$

Therefore, we want $1.95 < 10.92\epsilon$. Therefore $\epsilon \gtrsim 0.18$. Let us choose $\epsilon = 0.2$ so we can move on to the next phase of MRCE, estimating $\boldsymbol{\Omega}$ using the graphical lasso algorithm. Firstly, let us set a new initial $\boldsymbol{\Omega}$, using this updated $\hat{\mathbf{B}}$:

$$\boldsymbol{\Sigma}_{\mathbf{E}}^{(1)} = \frac{1}{n}(\mathbf{Y} - \mathbf{X}\hat{\mathbf{B}}^{(1)})^{\top}(\mathbf{Y} - \mathbf{X}\hat{\mathbf{B}}^{(1)}) = \begin{bmatrix} 25.24 & 19.97 \\ 19.97 & 36.96 \end{bmatrix}.$$

When inverting, again use a regularisation parameter $\lambda_1 = 0.1$ to prevent a zero determinant, giving:

$$\boldsymbol{\Omega}^{(0)} = \begin{bmatrix} 0.069 & -0.037 \\ -0.037 & 0.047 \end{bmatrix}.$$

Like for Algorithm 1, we will only apply this manually to one element. Here, it will be $\Omega_{21}^{(1)}$. Now, let us solve the graphical lasso regression problem using $\hat{\mathbf{B}}^{(1)}$. For $j = 1$, which updates the first column and row of $\boldsymbol{\Omega}$, we define:

$$(\boldsymbol{\Sigma}_{\mathbf{E}}^{(1)})_{-1,-1} = 36.96, \quad (\boldsymbol{\Sigma}_{\mathbf{E}}^{(1)})_{-1,1} = 19.97.$$

We compute the inverse submatrix: $\Omega_{-1,-1}^{(1)} = (\boldsymbol{\Sigma}_{\mathbf{E}}^{(1)})_{-1,-1}^{-1} = 0.027$ and $\Omega_{-1,1}^{(1)} = (\boldsymbol{\Sigma}_{\mathbf{E}}^{(1)})_{-1,1}^{-1} = 0.050$ to contribute to $\hat{\boldsymbol{\beta}}$:

$$\hat{\boldsymbol{\beta}} = -\boldsymbol{\Omega}_{-1,-1}^{(1)}(\boldsymbol{\Sigma}_{\mathbf{E}}^{(1)})_{-1,1} = -0.00135.$$

Next, $\hat{\boldsymbol{\beta}}$ undergoes soft-thresholding:

$$\hat{\boldsymbol{\beta}} = \text{sign}(\hat{\boldsymbol{\beta}}) \max(|\hat{\boldsymbol{\beta}}| - \lambda_1, 0) = 0$$

The updated element is computed as:

$$\boldsymbol{\Omega}_{-1,1}^{(1)} = -\boldsymbol{\Omega}_{-1,-1}^{(1)}\hat{\boldsymbol{\beta}} = 0.$$

This occurs for each element in $\boldsymbol{\Omega}^{(1)}$ and these steps continue until convergence, so when $||\boldsymbol{\Omega}^{(k)} - \boldsymbol{\Omega}^{(k-1)}||_{\infty} < \epsilon$, where ϵ is chosen for controlled estimation403.

This completes one full MRCE iteration: initialising $\mathbf{B}^{(0)}$ and $\hat{\boldsymbol{\Omega}}^{(0)}$ and then updating to $\mathbf{B}^{(1)}$ and $\hat{\boldsymbol{\Omega}}^{(1)}$ after solving the penalised optimisation problem. This process continues iteratively until convergence. The final $\hat{\mathbf{B}}$, is then used to compute the predictions: $\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{B}}^{(n)}$. This can again have its centring removed, but it is not needed when evaluating the ANRMSE.

4.2.2 Code Explanation

The MRCE model is applied using the package `mrce`, which estimates both the regression coefficient matrix and the inverse covariance matrix of the response variables. The optimal values for these parameters are determined using a 5-fold CV procedure within the function `fit_mrce`. In this function, a range of λ_1 and λ_2 values for both penalties is iterated over, and the model is trained and validated on different subsets of the training data. The parameter combination that minimises the cross-validated ANRMSE is selected for final model training. Once the best model is identified, test-set predictions are computed by multiplying the test predictor and estimated coefficient matrices, `mrce_model$B`.

RRRR is tested using the function `fit_rrridge`, applying a rank constraint to the regression coefficient matrix. It first computes the ridge-penalised coefficient matrix to stabilise the solution, then applies SVD to retain only the top-rank singular values, forcing a lower rank. The reduced-rank coefficient matrix is then used for test set predictions, evaluated via the ANRMSE.

Next, additional feature transformations are applied to improve the performance of these methods. A polynomial expansion of degree two is performed using the function `poly` to introduce non-linear relationships between predictors and responses. Additionally, interaction terms between predictors are generated using the function `model.matrix`. Finally, all models are compared based on their test-set ANRMSE scores as specified previously.

4.2.3 Results

These regression models were applied to the dataset to evaluate how well the predictors model the response variables: `roe` and `sustainability_score`. The performance of each model was assessed using the ANRMSE, like the other models in this report, where lower values indicate better performance.

The baseline models show that RRRR achieved the lowest ANRMSE at 0.751 with MRCE having an ANRMSE of 0.761. This suggests that the reduced-rank constraint in RRRR allowed it to perform slightly better than the MRCE. MRCE performed the worst, likely due to the additional covariance estimation step, which can introduce bias when response intercorrelations are weak. Here, the response intercorrelation was -0.3171, and MRCE's strengths come with high response variable intercorrelations.

Applying polynomial features and interaction transformations improved performance across both models significantly. The ANRMSE for MRCE dropped from 0.761 to 0.678, achieving the best performance among the polynomial models. RRRR, however, only improved to 0.727, suggesting that polynomial transformations introduced additional noise not effectively handled by the reduced-rank structure. Introducing interaction terms further reduced ANRMSE, with MRCE achieving the lowest error at 0.678, with RRRR at 0.723. This means its covariance estimation benefits more from relationships between features rather than non-linear transformations of individual predictors. Conversely, RRRR only had a slight improvement, likely due to the additional complexity introduced by interaction terms, which might have increased multicollinearity.

The results suggest that performance varies depending on the terms introduced. The introduction of polynomial and interaction terms generally improved performance, but the extent of improvement varied across models. RRRR, despite its superior baseline performance, struggled with higher-order terms, whereas MRCE benefited more from interaction effects than from polynomial expansion.

Overall, RRRidge performed best here, particularly when incorporating polynomial or interaction features. Thus, it is the most effective shrinkage method for this dataset.

Model	ANRMSE
Model 1: MRCE	0.7612
Model 2: RRRR	0.7510
Model 3: MRCE including Polynomial Terms	0.6781
Model 4: RRRR including Polynomial Terms	0.7276
Model 5: MRCE including Interaction Terms	0.6777
Model 6: RRRR including Interaction Terms	0.7231

Table 4.2: Comparison of ANRMSE values across shrinkage methods

Another aspect that was looked at was how each features contribute to each response.

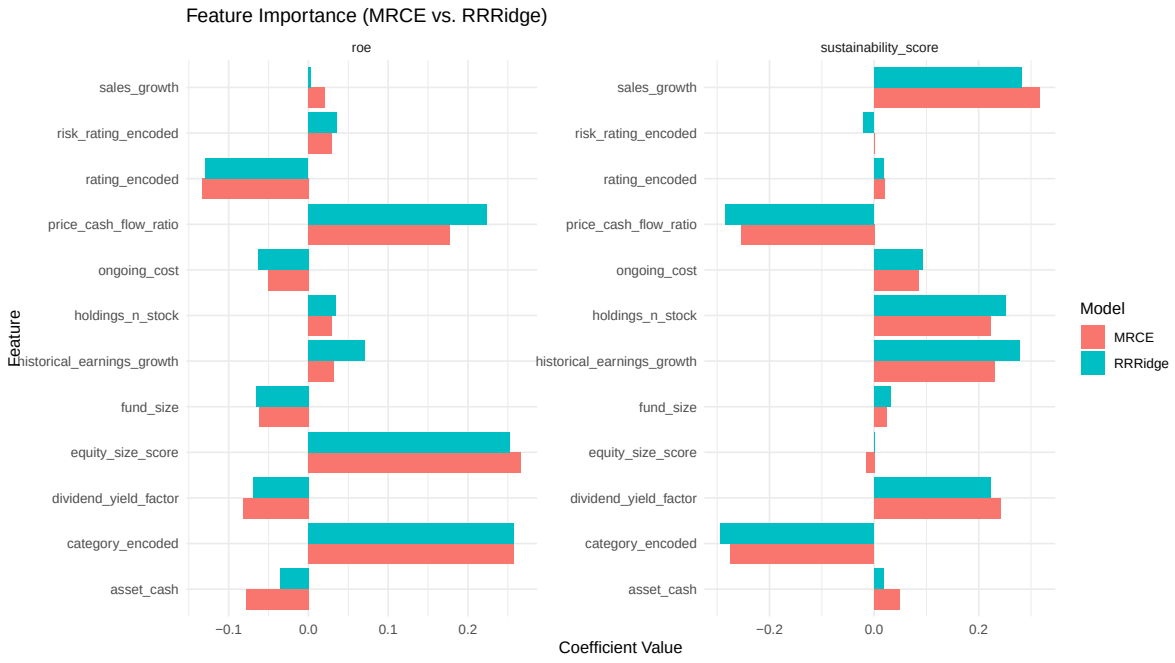

Figure 4.1: MRCE vs RRRR Feature Contributions

For `roe`, both models assign high importance to `price_cash_flow_ratio`, `dividend_yield_factor`, and `equity_size_score`. However, MRCE shrinks these coefficients more due to its estimation of regression coefficients and the precision matrix. `price_cash_flow_ratio` has a lower coefficient in MRCE, indicating that RRRR relies on it more heavily. `sales_growth` has little influence in either model, while `asset_cash` is slightly negative in MRCE but negligible in RRRR.

For `sustainability_score`, the differences between MRCE and RRRR coefficient magnitudes are clearer. `sales_growth` is the strongest positive predictor in both models, but MRCE assigns it a slightly larger coefficient. `historical_earnings_growth` has a strong positive effect on RRRR, while MRCE shrinks it. This is to mitigate multicollinearity even though it has been largely dealt with in Chapter 2. `category` has a higher coefficient in MRCE, suggesting it considers the type of equity fund more clearly. `equity_size_score` is near zero in MRCE but retains some effect in RRRR, while `price_cash_flow_ratio` has a larger absolute coefficient in RRRR.

Shrinkage methods like MRCE and RRRR have been useful, but their reliance on linear assumptions can make them ineffective with certain complex data, as shown here. Random Forests tackle these issues by using an ensemble of decision trees. They naturally handle nonlinearity and variable interactions without the need for manual feature selection. Taking an ensemble also helps reduce over-fitting, making them more effective for complex data.

Chapter 5 Random Forests

This chapter delves into Random Forests. It starts with classification and regression trees, from which we can create an ensemble of decision trees. This gives rise to random forests, which improve predictive performance. The forests are then extended to MRR, and the loss function is adjusted to form altered Covariance Regression Random Forests, which are tested on the equity fund dataset.

5.1 Theory

The prior discrete and continuous methods assume that the equity fund dataset holds certain properties, such as response multivariate normality. Tree-based models, however, are non-parametric, meaning they require no prior assumptions. These algorithms work by partitioning the feature space into smaller regions with similar response values using a set of splitting rules.

5.1.1 Classification and Regression Trees (CARTs)

Random forests are tree-based models built upon classification and regression trees (CARTs) as a building block. A CART partitions the training data into groups with similar response values and then fits a simple constant in each subgroup.

The CART starts with a root node containing all the objects and is then divided into "leaf" nodes by recursive binary splitting.[26] Each leaf node relates to a hyper-rectangle in feature (predictor) space, R_j, $j = \{1, \ldots, J\}$, i.e. the feature space defined by $X_1, X_2, \ldots, X_p$ is split into J distinct regions which do not overlap, as shown below:

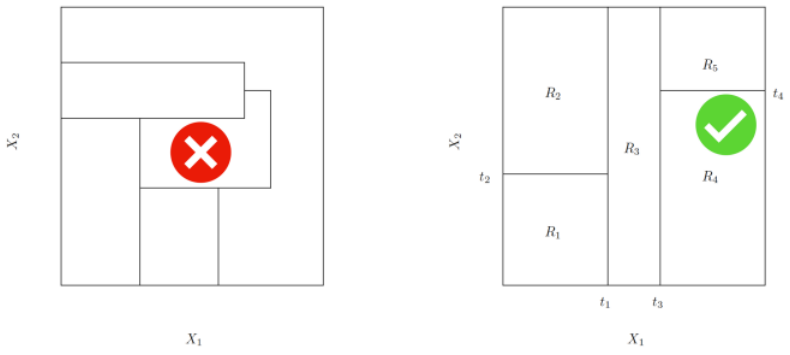

Figure 5.1: Incorrect vs. Correct Partitioning

The Greedy-Fitting Algorithm ensures each split is chosen to minimise the RSS:

$$\sum_{j=1}^{J} \sum_{i:x_i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the j-th rectangle.

A regression tree is used because the response variables here, `roe` and `sustainability_score`, are continuous. A regression tree assigns a value to each predictor space, R_j, i.e. the model predicts the output (i.e. which feature space) based on the average response values for all observations in that subgroup.

The Greedy fitting algorithm gives good predictions for the training data but could over-fit it, causing poor test data performance. The weakest link pruning algorithm prunes trees to address this bias-variance trade-off and over-fitting.

The weakest link pruning algorithm introduces a non-negative tuning parameter α, for regression trees, which minimises:

$$\sum_{j=1}^{|T|} \sum_{i:x_i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha |T|,$$

where $|T|$ is the number of terminal nodes (size) of the tree T.

5.1.2 Bagging for CARTs

Although CARTs are easy to interpret and are similar to standard decision-making processes, the trees generally have high variance, i.e. small sample changes lead to significant changes in the fit, and they tend to have poor predictive accuracy. Bootstrap aggregating, also known as bagging, improves the stability and accuracy of classification and regression algorithms by model averaging. Bagging helps reduce variance and avoid over-fitting, but the model becomes more challenging to interpret.
For regression trees, average all the predictions to obtain:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x)$$

Bagging for CART addresses the overfitting issue by growing large trees with minimal pruning and relying on the bagging procedure directly to avoid overfitting. It also addresses overfitting through pruning, which was explained previously.

Bagging is advantageous in that if there is a lot of data, which is true here, bagging is a good option; the empirical distribution will be closer to the actual underlying population's distribution. Also, it is the best option when the goal is to reduce the variance of a predictor. However, it also brings some disadvantages; interpretability is lost as the final estimate is not a tree; it is an ensemble prediction, i.e. multiple trees are combined to make the final prediction.[27]. Also, bagging trees are inherently correlated, which is a problem because the more correlated the random variables are, the less the variance reduction of their average, which can undermine one of its key advantages.

5.1.3 Random Forests

Random Forests make bagged CART models more independent, improving variance reduction in their ensemble predictions. They do this by resampling observations and restricting the model to random subspaces, $\mathcal{X}' \subset \mathcal{X}$, which ensures the tree explores different parts of the data, meaning the ensembles are more diverse and hence influential.

The Random Forests algorithm is used in the following way: a bootstrap resample $(y_i^*, x_i^*)_{i=1}^n$ is taken of the training data like for bagging. Then, the tree is built; each time a split in a tree is

considered, randomly select m predictors out of the full set of p predictors as split candidates and find the best split within those m predictors. Finally, repeat the first two steps, averaging the prediction of all the regression forest trees.

Random forests are a great option, but they are not interpretable when bagging and using random subspaces. Thus, quantifying the importance of each variable can be difficult. There are two popular approaches which quantify this importance.

One approach is to run a loop over each tree in the forest to determine the significance of each feature variable x_j, where $j = 1, \ldots, p$. Every node that splits on x_j for every tree is identified, and how much each split improves the chosen loss criterion, such as accuracy, Gini index, etc., is calculated. Each improvement is then summed for every tree in the forest, indicating the significance of x_j.

Another approach is to use out-of-bag (oob) error estimates, which can be computed via bagging, and they can also be used to determine the significance of each feature x_j, where $j = 1, \ldots, p$. Each tree's predictive accuracy is calculated after the oob samples, represented by x^{oob}, are extracted. The ties between x_j and the other variables are then broken by randomly permuting the entries of the j-th column of x^{oob}. After passing the modified $x^{\text{oob}*}$ matrix through the tree, the change in predictive accuracy is computed for that tree. The decrease in accuracy caused by the permutation is averaged over all trees, giving a measure of the importance of the feature x_j.

Random forests inherit the advantages of bagging and trees, and tuning is rarely needed. However, they are hard to implement and have extrapolation issues.

5.1.4 Multivariate Regression Trees

CARTs are limited to single-response variables, making them unsuitable for modelling relationships between multiple correlated responses. Multivariate Regression Trees (MRTs) extend CARTs by partitioning data based on multiple response variables simultaneously, ensuring that splits account for interactions among responses.[26] MRTs split a response matrix into clusters based on thresholds of explanatory variables via hierarchical clustering, ensuring that the tree structure highlights local structures and variable interactions.[28] This process is iterative and continues until each leaf node contains a single observation. This approach allows MRTs to model multiple response relationships by partitioning the data to account for interactions between predictor and response variables. Unlike single-response regression trees, MRTs highlight local data structures.[28]

MRTs are built like CARTs, but they require criteria that account for the joint distribution of multiple response variables. Unlike single-response trees that minimise variance for one variable at a time, MRTs evaluate splits via the combined variation of all responses. The impurity measure in MRTs minimises this total variance, and it is the total sum of squares of the responses around the multivariate mean at each node:

$$\text{impurity} = \sum_{i=1}^{n} \sum_{j=1}^{p} (y_{ij} - \bar{y}_j)^2 \,,$$

where y_{ij} represents the j-th response for the i-th sample, and $\bar{y_j}$ is the mean response at the node.[26] This minimising across all variations also ensures that correlated responses are clustered together in a way that preserves their joint structure, and this is how they implicitly consider response intercorrelation. However, MRTs do not explicitly model how response covariance changes with predictors.

To avoid over-fitting, tree splitting and optimal tree selection, use CV and pruning, like CARTs.

The best tree is selected by identifying the smallest tree within one standard error of the lowest cross-validated relative error tree, prioritising simpler models.[28]

MRTs produce easy-to-interpret tree structures that visualise local structures and interactions among variables. Each node represents a split based on an explanatory variable, and the leaves show the final clusters. Thus, it is easy to identify the importance of explanatory variables and interpret the results, such as in feature importance plots.

However, different visualisation tools are needed for MRTs to interpret their results. For instance, at each node of the tree, a bar plot can show the distribution of each response variable within that node, helping to visualise the multivariate outcomes and understand the impact of splits on all response variables collectively.[26]
Here is an MRT visualisation with 3 response variables and bar plots at each split:

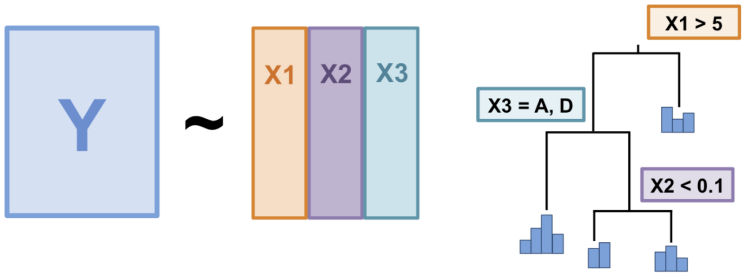

Figure 5.2: MRT Splitting.[28]

Although MRTs improve on CARTs in MRR, they come with some drawbacks, one of which is the inherent instability of tree-structured predictors caused partly by the split function's greedy optimisation.[26] Multivariate random forests can solve this.

5.1.5 Multivariate Random Forests

Multivariate random forests (MRFs) extend MRTs by generating an ensemble of MRTs via bootstrap resampling and predictors subsampling like for univariate random forests.[29] It also has a proximity matrix which captures how cases relate to each other and so provides the foundations for supervised clustering.[29] These factors mean MRFs can deal with the instability of tree-structured predictors.

MRFs are a great extension of MRTs. However, there is one glaring error that is important for MRR modelling. Although they can be adjusted to implicitly consider response correlation in their loss function, MRFs can't ensure that each split actually changes the response covariance matrix. So, when responses are correlated, MRFs fail to consider this optimally when building trees, potentially leading to suboptimal splits. This is where Covariance Regression Random Forests come in.

5.1.6 Covariance Regression Random Forests

Covariance Regression Random Forests (CRRFs) extend MRFs by incorporating a splitting criterion that maximises differences in sample covariance estimates between child nodes. Unlike MRFs, which focus solely on predicting the conditional mean, CRRFs explicitly account for response intercorrelation. They extend tree-based methods to estimate the conditional covariance matrix of a multivariate response given a set of covariates.[30] While CRRFs are designed to estimate covariance structures, they inherently compute the sample response mean at each terminal node. These mean values can be direct response estimates, allowing CRRFs to extend to MRR.

It builds decision trees using a specialised splitting rule that maximises differences in sample covariance estimates between child nodes.

Let $\Sigma_{\mathbf{y}_i}$ be the true conditional covariance matrix of $\mathbf{y}_i$, the responses, based on covariates $\mathbf{x}_i$, and $\Sigma_{\mathbf{X}}$ the collection of all conditional covariance matrices for n observations:

$$\Sigma_{\mathbf{X}} = \{\Sigma_{\mathbf{y}_i} : i = 1, \ldots, n\}.^{30}$$

Also, let $\hat{\Sigma}_{\mathbf{y}_i}$ be the estimated conditional covariance matrix of $\mathbf{y}_i$ based on covariates $\mathbf{x}_i$, and $\hat{\Sigma}_{\mathbf{X}}$ be the collection of estimated conditional covariance matrices for n observations.[30]

First, train a random forest with the set of covariates $\mathbf{X}$ to find subgroups of observations with similar covariance matrices of $\mathbf{Y}$ using decision trees to uncover the data structures. These trees are built with a splitting criterion that will be defined later.[30] The tree-growing process follows the CART algorithm.[31] The aim of this is to obtain subgroups of observations with distinct covariance matrices. Hence, a customised splitting rule at each node will be used to increase the difference in covariance matrices between two child nodes in the tree.[32,33,34,35]

Before defining this splitting let us first set up more notation. $\mathbf{\Sigma}^L$ is the sample response covariance matrix estimate, so $\hat{\Sigma}_{\mathbf{y}_i}$ of the left node, which is as follows:

$$\mathbf{\Sigma}^L = \frac{1}{n_L - 1} \sum_{i \in t_L} (\mathbf{y}_i - \bar{\mathbf{Y}}_L)(\mathbf{y}_i - \bar{\mathbf{Y}}_L)^\top,$$

where t_L is the set of indices, or the positions within the original data, of the observations in the left node, n_L is the left node size, and:

$$\bar{\mathbf{Y}}_L = \frac{1}{n_L} \sum_{i \in t_L} \mathbf{y}_i.^{30}$$

The estimate of the sample response covariance matrix for the right node, $\mathbf{\Sigma}^R$, is calculated similarly, where t_R is the set of indices of observations in the right node and n_R is its size.
The splitting criterion of CRRF is:

$$\sqrt{n_L n_R} \times d(\mathbf{\Sigma}^L, \mathbf{\Sigma}^R), \tag{5.1}$$

where $d(\mathbf{\Sigma}^L, \mathbf{\Sigma}^R)$ is the Euclidean distance between the upper triangular part of the two matrices and computed as follows:

$$d(\mathbf{D}, \mathbf{E}) = \sqrt{\sum_{i=1}^{q} \sum_{j=i}^{q} (D_{ij} - E_{ij})^2},$$

where $\mathbf{D}_{q \times q}$ and $\mathbf{E}_{q \times q}$ are symmetric matrices.[30] The best split among those possible is the one that maximises 5.1. This split is what facilitates CRRF as an MRR model as it explicitly uses a response covariance-based splitting criteria.

The final covariance matrices are estimated using random forests. For a new observation, nearest neighbour observations are used to estimate the response mean and final covariance matrix, ensuring both the predicted value and its uncertainty are captured.[30] This set of observations is called the Bag of Observations for Prediction (BOP).

For a new observation $\mathbf{x}^*$, the set of nearest neighbour observations are formed with the oob

observations.[36,37] The BOP_{oob} for a new observation is:

$$BOP_{oob}(\mathbf{x}^*) = \bigcup_{b=1}^{B} O_b(\mathbf{x}^*),$$

where B is the tree numbers and $O_b(\mathbf{x}^*)$ is the set of oob observations in the same terminal node as $\mathbf{x}^*$ in the bth tree and each tree is built with a selected random sub-sample.[30]
After training the random forest with the new split for a new observation $\mathbf{x}^*$, we form $BOP_{oob}(\mathbf{x}^*)$, which is key for response prediction.[30] The response mean and covariance matrix are then estimated using the sample mean and covariance matrix of the observations in $BOP_{oob}(\mathbf{x}^*)$, respectively:

$$\hat{\mathbf{Y}}^* = \frac{1}{|BOP_{oob}(\mathbf{x}^*)|} \sum_{i \in BOP_{oob}(\mathbf{x}^*)} y_i,$$
$$\hat{\mathbf{\Sigma}}^*_{\mathbf{Y}} = \frac{1}{|BOP_{oob}(\mathbf{x}^*)| - 1} \sum_{i \in BOP_{oob}(\mathbf{x}^*)} (y_i - \hat{\mathbf{Y}}^*)(y_i - \hat{\mathbf{Y}}^*)^\top,$$

where y_i is an observation from $BOP_{oob}(\mathbf{x}^*)$, $\hat{\mathbf{Y}}^*$ is the predicted response mean and $\hat{\mathbf{\Sigma}}^*_{\mathbf{Y}}$ represents the estimated conditional covariance matrix of the responses given $\mathbf{x}^*$.

5.2 Application

5.2.1 Small-Scale Example

We will work with this familiar dataset again:

X_1: Hours Studied	X_2: Time Spent on Papers	Y_1: Math Scores	Y_2: Science Scores
5	2	78	80
7	3	85	79
8	4	88	88
3	1	65	70
10	5	92	74

Table 5.1: Study Time vs. Exam Scores

Each tree is trained on a bootstrap sample, which is obtained by randomly drawing observations with replacements from the original dataset. For the provided dataset, a possible bootstrap sample could be, for example, rows 1,1,2,4 and 5. Here, you can see that row 1 has been repeated, and row 3 has been left out. Another possible bootstrap sample is just getting each row once, so rows 1,2,3,4 and 5. This corresponds to:

$$\mathbf{X} = \begin{bmatrix} (5,2,78,80) \\ (7,3,85,79) \\ (8,4,88,88) \\ (3,1,65,70) \\ (10,5,92,74) \end{bmatrix}.$$

After constructing a bootstrap sample, recursive partitioning is performed by selecting the split that maximises the scaled Euclidean distance between the covariance matrices of the left and right child

nodes. The Euclidean distance is computed as:

$$d(\mathbf{\Sigma}^L, \mathbf{\Sigma}^R) = \sqrt{\sum_{i=1}^{q}\sum_{j=i}^{q}(\mathbf{\Sigma}_{ij}^L - \mathbf{\Sigma}_{ij}^R)^2}, \tag{5.2}$$

where $\mathbf{\Sigma}^L$ and $\mathbf{\Sigma}^R$ are the empirical covariance matrices of the left and right nodes, respectively.

In reality, the CRRF will evaluate all possible splits and compute 5.1, but for simplicity, let us assume the split that maximises 5.1 is $X_1 < 7$. It is important to understand how 5.1 is computed for any split. First, let us show how this is done for $X_1 < 7$. This starts by splitting the data into 2 child nodes: left and right. The left child node contains the observations and responses of:

$$\mathbf{X}_L = \begin{bmatrix} 5 & 2 & 78 & 80 \\ 3 & 1 & 65 & 70 \end{bmatrix},$$

and the right child node contains:

$$\mathbf{X}_R = \begin{bmatrix} 7 & 3 & 85 & 79 \\ 8 & 4 & 88 & 88 \\ 10 & 5 & 92 & 74 \end{bmatrix}.$$

For each node, the mean response is computed as:

$$\bar{\mathbf{Y}}_L = \frac{1}{2}\sum \mathbf{Y}_L = \begin{bmatrix} 71.5 & 75 \end{bmatrix}, \quad \bar{\mathbf{Y}}_R = \frac{1}{3}\sum \mathbf{Y}_R = \begin{bmatrix} 88.33 & 80.33 \end{bmatrix}.$$

The covariance matrices are then estimated using:

$$\mathbf{\Sigma}^L = \frac{1}{n_L - 1}\sum_{i \in L}(\mathbf{Y}_i - \bar{\mathbf{Y}}_L)(\mathbf{Y}_i - \bar{\mathbf{Y}}_L)^\top, \quad \mathbf{\Sigma}^R = \frac{1}{n_R - 1}\sum_{i \in R}(\mathbf{Y}_i - \bar{\mathbf{Y}}_R)(\mathbf{Y}_i - \bar{\mathbf{Y}}_R)^\top.$$

The computed covariance matrices for the left and right nodes are:

$$\mathbf{\Sigma}^L = \begin{bmatrix} 142.25 & 105.25 \\ 105.25 & 97.25 \end{bmatrix}, \quad \mathbf{\Sigma}^R = \begin{bmatrix} 41.67 & -10.67 \\ -10.67 & 21 \end{bmatrix}.$$

Applying the Euclidean distance formula to these matrices gives (to 2dp):

$$d(\mathbf{\Sigma}^L, \mathbf{\Sigma}^R) = \sqrt{(142.25 - 41.67)^2 + 2(105.25 - (-10.67))^2 + (97.25 - 21)^2} = 206.89.$$

Finally, the scaled Euclidean distance is:

$$\sqrt{n_L n_R} \times d(\mathbf{\Sigma}^L, \mathbf{\Sigma}^R) = 206.89\sqrt{2 \times 3} = 506.77$$

Since CRRF selects the split that maximises this distance, we can see the split at $X_1 = 7$ does lead to a large Euclidean distance between the response sample covariance matrices of each child node.

When predicting, each new observation is assigned to a terminal node based on its feature values. The prediction is computed using the BOP within the same node. For a new test point $\mathbf{x}^* = (6, 3)$, which falls into the left node, the relevant BOP observations here are: $BOP_{oob}(\mathbf{x}^*) =$

$\{(78, 80), (65, 70)\}$. The predicted mean response is then given by

$$\hat{\mathbf{Y}}^* = \frac{1}{|BOP_{oob}(x^*)|} \sum_{i \in BOP_{oob}(x^*)} \mathbf{Y}_i = \begin{bmatrix} 71.5 \\ 75 \end{bmatrix}.$$

The covariance structure of the predictions is estimated using

$$\hat{\mathbf{\Sigma}}^* = \frac{1}{|BOP_{oob}(x^*)| - 1} \sum_{i \in BOP_{oob}(x^*)} (\mathbf{Y}_i - \hat{\mathbf{Y}}^*)(\mathbf{Y}_i - \hat{\mathbf{Y}}^*)^\top.$$

This results in

$$\hat{\mathbf{\Sigma}}^* = \begin{bmatrix} 142.25 & 105.25 \\ 105.25 & 97.25 \end{bmatrix}.$$

In this case, the prediction, $\hat{\mathbf{Y}}^*$, and its covariance matrix, $\hat{\mathbf{\Sigma}}^*$, correspond to $\bar{\mathbf{Y}}_L$ and $\mathbf{\Sigma}^L$ respectively because there is only 1 tree that has been made. What happens for CRRF is that lots of trees are formed from many different bootstraps because each bootstrap will lead to different splits being determined to maximise the Euclidean distance. As a result, the output $\hat{\mathbf{Y}}^*$ will be adjusted when more and more trees are added.

5.2.2 Code Explanation

Like the previous section, CRRF came with a pre-set package in R called `CovReg`. This made its implementation simpler than manual coding in R. However, there were still some considerations.

To prevent over-fitting, explicit stopping criteria are implemented in the tree growth process. A node is set as terminal if either the depth limit of the tree is reached, `depth = 0`, or the number of samples in the node falls below the threshold, $n < 5$. In such cases, the node stores the mean and covariance of the response variables rather than continuing to split further. This ensures that the tree does not grow excessively, preventing over-fitting.

Furthermore, to evaluate the performance of CRRF, we implement a 5-fold CV using the ANRMSE. The dataset is randomly partitioned into 5 equal-sized folds, where each fold acts as a validation set once while the remaining 4 folds form the training set. This ensures that every observation contributes to both training and validation. At the end of CV, the final average NRMSE of all the test folds is used to measure CRRF fit.

This code also uses parallelisation, which speeds up training by running multiple trees concurrently rather than sequentially. Parallel computing makes programs and processes run faster as more CPUs are used.[38] They are implemented using multi-threading, allowing trees to be trained simultaneously, thereby significantly reducing computation time.

5.2.3 Results

The CRRF model achieved an ANRMSE of 0.5238, the lowest value thus far. This model did not even consider interaction and polynomial terms, which were not explicitly tested as RFs already capture non-linear relationships and interactions through their splits.

The CRRF model was also analysed in how it determined predictor feature importance in modelling the 2 responses: `roe` and `sustainability_score`.

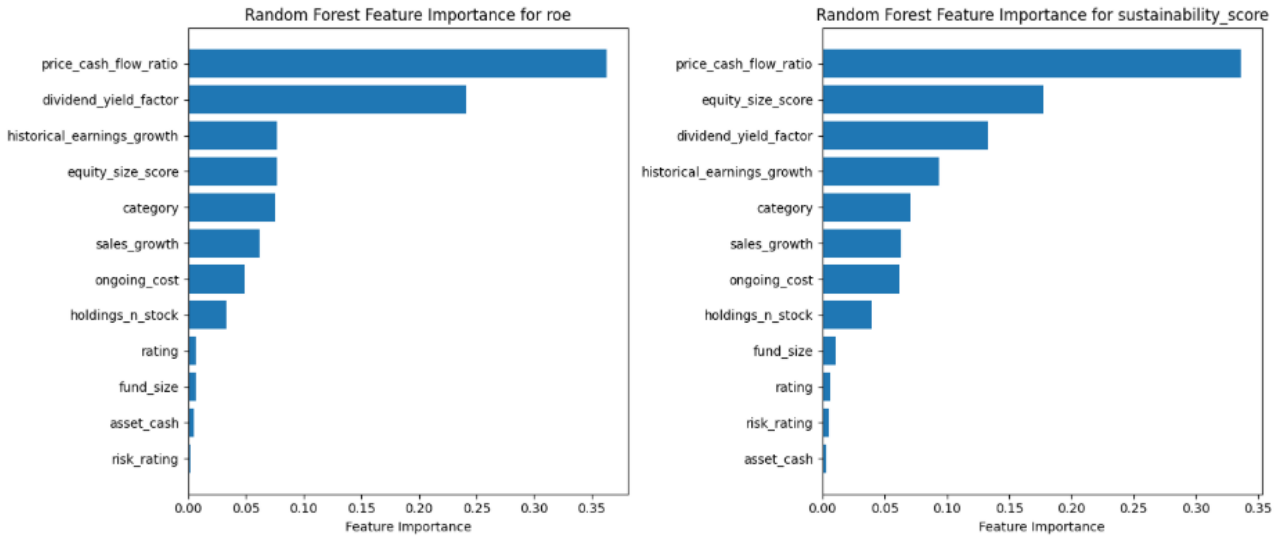

Figure 5.3: CRRF Feature Importance

For ROE, CRRF assigns the highest importance to `price_cash_flow_ratio`, `dividend_yield_factor`, and `equity_size_score`, indicating these features drive the most significant changes in the covariance structure of responses. The dominance of `price_cash_flow_ratio` suggests it effectively separates subgroups with distinct response covariances. In contrast, `sales_growth`, `risk_rating`, and `asset_cash` have minimal influence, implying they do not meaningfully alter response covariances.

For sustainability_score, `price_cash_flow_ratio` remains the most critical feature, enforcing its role in separating subgroups with distinct response covariances. `Equity_size_score` is more important here, suggesting it impacts sustainability-linked covariance shifts. `Historical_earnings_growth` and `dividend_yield_factor` remain relevant but with less impact than `price_cash_flow_ratio`. Sales growth is moderately important, while risk-related variables again show minimal effect.

Overall, the consistent importance of `price_cash_flow_ratio` highlights its strong role in dictating which equity funds should be invested in long-term based on the CRRF, whereas features with minimal covariance impact, like `risk_rating` and `asset_cash`, are not prioritised.

Although CRRFs have well-modelled this dataset, they have some drawbacks. First, they are computationally intensive because they need to calculate covariance matrices at each split. Second, although CV mitigates this somewhat, they are prone to over-fitting with irrelevant features. XGBoost can deal with all of these because it uses approximate tree learning and implements regularisation.

Chapter 6 XG Boost

This chapter covers XGBoost. It starts by investigating its theory for the single-response case using its foundational paper. Then, it extends to MRR via Multi-Output XGBoost. Finally, it combines this with Cholesky Decomposition, which removes and returns response intercorrelations. This Cholesky-Decomposed XGBoost is then applied to the equity fund dataset, and its performance is analysed.

6.1 Theory

Now, we move on to Extreme Gradient Boosting, which differs from Random Forests in that rather than generating an ensemble of trees using bagging, it builds trees sequentially. Each new tree corrects the residuals of the previous ones, forming an ensemble that refines predictions step by step.

6.1.1 Introduction

XGBoost, otherwise known as Extreme Gradient Boosting, improves upon the idea of gradient tree boosting, which involves iteratively adding trees with the goal of minimising a predefined loss function. At each iteration, the algorithm fits a new tree to the residuals of the current model, effectively correcting previous errors. Gradient tree boosting is defined as follows:

$$\hat{y}_i = \sum_{t=1}^{\top} f_t(x_i), \quad f_t \in \mathcal{F},$$

where $\mathcal{F}$ represents the space of regression trees, also known as CARTs.[39] Each tree f_t predicts an additive score for the input features x_i. This means means each new tree contributes an incremental improvement, rather than replacing the previous prediction. XGBoost improves gradient boosting by including regularisation explicitly in the loss function to address over-fitting and improve model predictions. Its objective function is:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{t=1}^{\top} \Omega(f_t),$$

where l represents the loss function, and $\Omega(f_t)$ is a regularisation term. The loss function for XGBoost regression can take many different forms, such as the log loss and the mean squared log error. However, here, the sum of squared errors (SSE) is used because it is simple and easy to interpret.

XGBoost uses regularisation to help control model complexity and mitigate over-fitting:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2,$$

where T is the number of leaves in the tree, w_j are the leaf weights, γ penalises additional leaves, and

λ controls the regularisation of leaf weights.[39]

XGBoost also uses shrinkage and column sub-sampling to prevent over-fitting further. Shrinkage scales newly added weights by a factor of η after each step of tree boosting. Shrinkage reduces the influence of each tree and leaves space for future trees to improve the model.[39] Column sub-samples also speed up computations of the parallel algorithm.[39]

The construction of XGBoost allows it to fit models well and mitigate over-fitting.[39] XGBoost optimises the objective function using a second-order Taylor approximation, and its loss function, with the constant terms removed, is approximated as:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t),$$

where $g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ and $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ are the first and second-order gradients of the loss function, respectively.

To construct decision trees, XGBoost evaluates potential splits using the Greedy Algorithm for Split Finding, which involves maximising a gain function:

$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma,$$

where G_L and H_L are the sums of first and second-order gradients for the left child (or node), L, and G_R and H_R are the corresponding sums for the right child, R, λ is a regularisation parameter, and γ is a pruning parameter.[39] A "child" is the rows selected when a decision tree splits at a node.

Each component within the Gain's bracket is called the (similarity) score. So, the Gain can be written as:

$$\text{Gain} = \frac{1}{2} \left[S_L + S_R - S_P \right] - \gamma,$$

where:

$$S_L = \frac{G_L^2}{H_L + \lambda}, \quad S_R = \frac{G_R^2}{H_R + \lambda}, \quad S_P = \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}.$$

Note: S_P is the score of the parent node of the new node being derived.

6.1.2 Iterative Steps of XGBoost

As outlined in Chen and Guestrin (2016), the exact Greedy Algorithm occurs by iterating over each feature dimension k, ranging from 1 to m, where m is the total number of predictors.

First, for each feature, potential split points are evaluated by sorting the instances, which are the rows in the training dataset, based on their corresponding feature values. During this process, cumulative sums for the left child node are tracked using G_L and H_L. These, along with the Gain, are initialised to zero at the beginning of each iteration and are accumulated:

$$G_L \leftarrow G_L + g_j, \quad H_L \leftarrow H_L + h_j.$$

Simultaneously, the right node's Gradients and Hessians are updated: $G_R = G - G_L$ and $H_R = H - H_L$.

The quality of each split is then measured by calculating the gain, which is the loss reduction:

$$\text{New Gain} = \max\left(\text{Previous Gain}, \text{New Gain} = S_L + S_R - S_P\right).$$

The scaling parameter is removed because it does not affect the ranking of split scores. Higher gain values indicate more effective splits, reducing the overall model error. The pruning parameter, γ, is applied after split selection, not during split evaluation.

Once all potential splits have been evaluated, the algorithm selects the split with the highest gain, ensuring that each node is partitioned at the most optimal point. If no split results in a positive gain, the node is left as a leaf. This approach ensures that each decision tree grows in a way that minimises residual error, leading to more accurate models.

The exact Greedy Algorithm is ideal for XGBoost, but due to its computational cost, an approximate greedy algorithm is used in practice. This method is similar to the prior one with some adjustments, which are explained below, again from Chen and Guestrin (2016). The algorithm begins by iterating over each feature dimension k, from 1 to m, where m is the total number of features. Firstly, for each feature, $k = 1, \ldots, m$, a set of candidate splits $S_k = \{s_{k1}, s_{k2}, \ldots, s_{kl}\}$ is proposed. These splits are determined by dividing the feature values into percentiles, ensuring the split proposals are distributed evenly across the feature range, capturing key points that are likely to lead to huge loss reduction. The proposals can be conducted globally or locally. Globally, split points are determined once per tree and reused at each node, whereas locally, new split points are generated for each node.

After proposing candidate splits, the algorithm evaluates their quality. For each proposed split $s_{k,v}$ within the feature k, the gradients and Hessians of the data points falling into the interval between two consecutive split points are accumulated. The gradients and Hessians are computed as:

$$G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$$
$$H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$$

where G_{kv} represents the sum of gradients and H_{kv} the sum of Hessians for the interval defined by $s_{k,v}$ and $s_{k,v-1}$. This process effectively approximates the exact split finding by limiting the evaluation to a subset of potential split points.

Once the gradients and Hessians have been computed for all proposed splits, the algorithm identifies the split that maximises the gain. This step follows the same logic as the exact greedy algorithm but is restricted to the proposed split points, reducing the number of calculations required.

Then, the same steps are taken as in the previous section to find the maximum score only among proposed splits. While this method sacrifices precision by not evaluating all possible splits, the improved efficiency allows XGBoost to scale effectively to large datasets.

Once the split that optimises the gain is found, this is then used to adjust the former prediction using the leaf weight for each iteration:

$$w_j = -\frac{G_j}{H_j + \lambda},$$

where G_j is the sum of gradients for all instances in the leaf, H_j is the sum of Hessians for all instances in the leaf, and λ is the regularisation parameter.[39]

Each instance's prediction is updated as follows:

$$\hat{y}^{(t)} = \hat{y}^{(t-1)} + \eta w_j$$

where $\hat{y}^{(t-1)}$ represents the previous prediction, w_j is the weight of the leaf the instance falls into, and η is the learning rate, which controls the step size of updates.[39]

6.1.3 Cholesky Decomposition

Although assuming a prior distribution is not necessarily required for a Cholesky Decomposition, assuming a Multivariate Normal Distribution (MVN) helps in satisfying its requirements, which are a positive definite symmetric matrix. Here is the density of an MVN Distribution:

$$f(\mathbf{Y}|\boldsymbol{\mu}_\mathbf{Y}, \Sigma_\mathbf{Y}) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_\mathbf{Y}|}} \exp\left(-\frac{1}{2}(\mathbf{Y}-\boldsymbol{\mu}_\mathbf{Y})^\top \Sigma_\mathbf{Y}^{-1}(\mathbf{Y}-\boldsymbol{\mu}_\mathbf{Y})\right),$$

where $\boldsymbol{\mu}_\mathbf{Y} \in \mathbb{R}^D$ represents a vector of conditional means, $\Sigma_\mathbf{Y}$ is the positive definite symmetric $D \times D$ response covariance matrix, where D is the number of responses, and $|\cdot|$ denotes the determinant.[40]

For the bivariate case $D = 2$, the local response covariance matrix, $\Sigma_{i\mathbf{y}}$ can be written as:

$$\Sigma_{i\mathbf{y}} = \begin{bmatrix} \sigma_{i,1}^2(\mathbf{x}) & \rho_i(\mathbf{x})\sigma_{i,1}(\mathbf{x})\sigma_{i,2}(\mathbf{x}) \\ \rho_i(\mathbf{x})\sigma_{i,1}(\mathbf{x})\sigma_{i,2}(\mathbf{x}) & \sigma_{i,2}^2(\mathbf{x}) \end{bmatrix},$$

with the variances on the diagonal and covariances on the off-diagonal, for $i = 1, \ldots, N$.[40]

To ensure positive definiteness of $\Sigma_\mathbf{x}$, the $D(D+1)/2$ entries of the covariance matrix must satisfy specific conditions.[40] In the bivariate case, applying exponential functions to the variances is a suitable transformation to restrict the coefficient of correlation $\rho \in [-1, 1]$ ensures this.[40] However, in higher dimensions, where all moments are modelled as functions of covariates, ensuring positive definiteness of the covariance matrix becomes challenging since joint restrictions for the elements of $\boldsymbol{\Sigma}$ are necessary.[41] A more computationally efficient approach to ensure positive definiteness is by using the Cholesky decomposition, which breaks down the covariance matrix as follows:

$$\boldsymbol{\Sigma}_\mathbf{Y} = \mathbf{L}\mathbf{L}^\top,$$

where $\mathbf{L} \in \mathbb{R}^{D \times D}$ is a lower triangular matrix.[40] To ensure $\boldsymbol{\Sigma}$ to be positive definite, the D diagonal elements ℓ_{ii} of $\mathbf{L}$ need to be strictly positive, whereas all $D(D-1)/2$ off-diagonal elements ℓ_{ij} can take on any value in $\mathbb{R}$. Cholesky decomposition has an algorithm to derive $\mathbf{L}$:

1. Initialise $\mathbf{L}$ as an $n \times n$ zero matrix.
2. For each row i (from 1 to n):
   - Compute the diagonal element L_{ii}:
   $$L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2}$$
   - For each column j (below diagonal, $j > i$):
   $$L_{ji} = \frac{A_{ji} - \sum_{k=1}^{i-1} L_{jk}L_{ik}}{L_{ii}}$$
3. Return $\mathbf{L}$.

The Cholesky Decomposition can be used to remove response intercorrelation, so it sets the off-diagonal elements to zero by applying $\mathbf{L}^{-1}\mathbf{Y}^\top$ to give transformed responses which are independent.[42] Doing

this means XGBoost can be applied separately for each transformed response to generate predictions. Re-applying $\mathbf{L}$ to these then returns the actual predictions.

There does exist a version of XGBoost, Multi-Output XGBoost, which natively supports MRR. However, its methodology still needs to be fleshed out.

6.2 Application

6.2.1 Small-Scale Example

We will work with the familiar dataset evaluating Mathematics and Science Scores:

X_1: Hours Studied	X_2: Time Spent on Papers	Y_1: Math Scores	Y_2: Science Scores
5	2	78	80
7	3	85	79
8	4	88	88
3	1	65	70
10	5	92	74

Table 6.1: Study Time vs. Exam Scores

We have:

$$\mathbf{X} = \begin{bmatrix} 5 & 7 & 8 & 3 & 10 \\ 2 & 3 & 4 & 1 & 5 \end{bmatrix}^\top, \quad \mathbf{Y} = \begin{bmatrix} 78 & 85 & 88 & 65 & 92 \\ 80 & 79 & 88 & 70 & 74 \end{bmatrix}^\top, \quad \bar{\mathbf{Y}} = \begin{bmatrix} 81.6 & 78.2 \end{bmatrix}^\top.$$

First, we need the response covariance matrix, $\mathbf{\Sigma_Y}$:

$$\mathbf{\Sigma_Y} = \frac{1}{n-1}(\mathbf{Y} - \bar{\mathbf{Y}})^\top(\mathbf{Y} - \bar{\mathbf{Y}}) = \begin{bmatrix} 112.30 & 37.85 \\ 37.85 & 46.20 \end{bmatrix},$$

so each response value has its column mean subtracted from it. The matrix is symmetric and positive definite because its (1,2) and (2,1) entries are the same, and it has positive eigenvalues.

Next comes the Cholesky Decomposition of the response covariance matrix. This subsection is for clarity, so for ease of decomposition, let us show Cholesky Decomposition on a "nicer" matrix:

$$\mathbf{A} = \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix}.$$

We aim to find a lower triangular matrix $\mathbf{L}$ such that:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^\top, \quad \mathbf{L} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}$$

The first step is to compute L_{11} using the formula for diagonal elements:

$$L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2} \Rightarrow L_{11} = \sqrt{A_{11}} = \sqrt{4} = 2 \Rightarrow \mathbf{L} = \begin{bmatrix} 2 & 0 \\ L_{21} & L_{22} \end{bmatrix}$$

Next, compute L_{21}. For off-diagonal elements:

$$L_{ji} = \frac{A_{ji} - \sum_{k=1}^{i-1} L_{jk}L_{ik}}{L_{ii}} \Rightarrow L_{21} = \frac{A_{21}}{L_{11}} = \frac{2}{2} = 1 \Rightarrow \mathbf{L} = \begin{bmatrix} 2 & 0 \\ 1 & L_{22} \end{bmatrix}$$

Finally, compute L_{22}, going back to our diagonal element formula:

$$L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2} \Rightarrow L_{22} = \sqrt{A_{22} - L_{21}^2} = \sqrt{3-1} = \sqrt{2} \Rightarrow \mathbf{L} = \begin{bmatrix} 2 & 0 \\ 1 & \sqrt{2} \end{bmatrix}$$

See A.2.3 for verification this decomposition is correct.

Applying this algorithm to $\mathbf{\Sigma_Y}$ gives, when passing it through R's inbuilt `chol` function:

$$\mathbf{\Sigma_Y} = \mathbf{L}\mathbf{L}^\top = \begin{bmatrix} 10.60 & 0 \\ 3.57 & 5.78 \end{bmatrix} \begin{bmatrix} 10.60 & 3.57 \\ 0 & 5.78 \end{bmatrix}$$

Now we can decorrelate the responses using $\mathbf{L}^{-1}$:

$$\tilde{\mathbf{Y}}^\top = \mathbf{L}^{-1}\mathbf{Y}^\top = \begin{bmatrix} 7.36 & 8.02 & 8.30 & 6.13 & 8.68 \\ 9.29 & 8.71 & 10.09 & 8.32 & 7.43 \end{bmatrix}$$

Here you can see that the off-diagonal elements of the covariance matrix of the transformed responses are zero, and it leaves the identity matrix, $\mathbf{I}_2$:

$$\text{Cov}(\tilde{\mathbf{Y}}) \approx \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Now that we have decorrelated the responses, we can apply an XGBoost model to each response without needing to consider the response covariance matrix.

Let us apply XGBoost, with an SSE loss function, to response $\tilde{Y}_1$, Mathematics Scores against the predictors X_1 and X_2. XGBoost starts by initialising predictions for all samples. The initial prediction is typically 0.5, but let us start here instead by taking the mean of $\tilde{Y}_1$:

$$\hat{Y}_1^{(0)} = \frac{7.36 + 8.02 + 8.30 + 6.13 + 8.68}{5} = \frac{38.49}{5} = 7.70. \tag{6.1}$$

Thus, the initial prediction for all samples is:

$$\hat{Y}_1^{(0)} = \begin{bmatrix} 7.70 & 7.70 & 7.70 & 7.70 & 7.70 \end{bmatrix}^\top.$$

Next, the residuals are calculated as the difference between the actual values $\tilde{Y}_1$ and the initial predictions (to 2dp):

$$\begin{aligned} r_1 = \tilde{Y}_1 - \hat{Y}_1^{(0)} &= \begin{bmatrix} 7.36 & 8.02 & 8.30 & 6.13 & 8.68 \end{bmatrix}^\top - \begin{bmatrix} 7.70 & 7.70 & 7.70 & 7.70 & 7.70 \end{bmatrix}^\top \\ &= \begin{bmatrix} -0.34 & 0.32 & 0.60 & -1.57 & 0.98 \end{bmatrix}^\top. \end{aligned}$$

The residuals are the errors in the initial predictions, which will be used to fit the first XGBoost tree. The residuals are taken because these are half the gradients when using the SSE loss function.

The Gradients, g_i, and Hessians, h_i, are needed because they are critical in evaluating the Score and computing the Gain. The gradients are calculated as:

$$g_i = \frac{\partial L}{\partial \hat{Y}_i} = \frac{\partial}{\partial \hat{Y}_i} \sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2 = 2(Y_i - \hat{Y}_i) = 2r_i,$$

where the summation vanishes because we are looking at each i^{th} element when differentiating. From the residuals, the gradients are (to 2dp):

$$g = \begin{bmatrix} -0.68 & 0.64 & 1.21 & -3.13 & 1.96 \end{bmatrix}^\top.$$

For squared error loss, the second-order gradient (Hessian) is:

$$h_i = \frac{\partial^2 L}{\partial \hat{Y}_i^2} = 2.$$

Since we are using the SSE, the Hessian for all points is:

$$h = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \end{bmatrix}^\top$$

Now, we train a small decision tree to predict the gradients. This starts with the parent node, P, which contains all the data. Each tree starts with a single leaf, and all residuals go to that leaf.[43] All that needs to be done here is to compute its score using the exact residuals:

$$S_P = \frac{(\sum g_P)^2}{\sum h_P + \lambda} = 0.$$

where g_p, h_p are the Gradients and Hessians of the parent node, respectively. This makes sense because, in this rare case, the gradients are proportional to the residuals, which always sum to zero. When the gradients in a node are very different, they cancel each other out and the Score is relatively small and you expect large gradients when initialising.[43] In contrast, when the gradients are similar, or there is just one of them, they do not cancel out, and the Score is relatively large.[43]

XGBoost will then evaluate splits after this by comparing the Gain for each split in X_1 and X_2 in this dataset and finding the split which maximises this. Maximising the Gain means there is a better splitting of the gradients into clusters of similar values.[43] Iterating through all potential split points can be computationally expensive, so this is where a weighted quantile sketch approximates the best-split points using weighted quantiles. See A.2.3 for more. This speeds up training by reducing the number of candidate split points.

Now, let us show how the gain is computed for an example split: $X_1 < 7$. $X_1 < 7$ splits the data into 2 sets of predictors and corresponding transformed responses:

X_1: Hours Studied	X_2: Time Spent on Papers	$\tilde{Y}_1$: Mathematics Scores	r_0: Residuals
5	2	7.36	-0.34
3	1	6.13	-1.57

Table 6.2: Study Time vs. Mathematics Scores for $X_1 < 7$

X_1: Hours Studied	X_2: Time Spent on Papers	$\tilde{Y}_1$: Mathematics Scores	r_0: Residuals
7	3	8.02	0.32
8	4	8.3	0.60
10	5	8.68	0.98

Table 6.3: Study Time vs. Mathematics Scores for $X_1 \geq 7$

Science Scores, $\hat{Y}_2$, have been removed as we are just modelling $\tilde{Y}_1$ here, and the initial residuals are placed for clarity when calculating the gradients and Hessians.

Let us call 6.2 the "left", L, node and 6.3 the "right", R, node. From each node, calculate the cumulative gradients and Hessians as:

$$G_L = \sum_{i=1}^{2} 2(r_i)_L = -3.81, \quad H_L = \sum_{i=1}^{2} 2 = 4, G_R = \sum_{i=1}^{2} 2(r_i)_R = 3.81, \quad H_R = \sum_{i=1}^{3} 2 = 6.$$

This is used to calculate the similarity scores for the left and right nodes with $\lambda = 0.1$:

$$S_L = \frac{G_L^2}{H_L + \lambda} = 3.62, \quad S_R = \frac{G_R^2}{H_R + \lambda} = 2.42.$$

Now compute the Gain for $X_1 < 7$ in comparison with the parent node:

$$\text{Gain} = S_L + S_R - S_P = 6.04.$$

For simplicity, assume the tree split of $X_1 < 7$ gives the optimal gain so we can adjust the predictions. The leaf weight differs by the child node each row falls in and is calculated as follows (to 2dp):

$$w_L = -\frac{G_L}{H_L + \lambda} = 0.95, \quad w_R = -\frac{G_R}{H_R + \lambda} = -0.63$$

The predictions are then adjusted using the leaf weight with the learning rate η, which we set as 0.3. Here, the initial prediction was the mean, from 6.1. So, the final transformed predictions are:

$$\hat{Y}_{1,L}^{(1)} = \hat{Y}_1^{(0)} + \eta \times w_L = 7.98, \quad \hat{Y}_{1,R}^{(1)} = \hat{Y}_1^{(0)} + \eta \times w_R = 7.51.$$

These need to be converted back to the original response space with the original correlation structure, using: $\mathbf{L}\hat{Y}_{1,L}^{(1)}$ or $\mathbf{L}\hat{Y}_{1,R}^{(1)}$. This gives the following original predictions (to 2dp):

$$\text{Left Node} : \hat{Y}_1^{(1)} = 84.55, \quad \text{Right Node} : \hat{Y}_1^{(1)} = 79.61.$$

6.2.2 Code Explanation

Now, we will explain how the Cholesky-Gaussian XGBoost was applied to the equity fund dataset. The only change when applying now is k-fold CV.

First, the dataset is split into five folds using `KFold()`, ensuring randomness in the CV process. For each fold, the empirical response covariance matrix of the target variables is computed using `numpy.cov()` to capture dependencies between responses. The Cholesky decomposition is applied using `numpy.linalg.cholesky()`, yielding a lower triangular matrix, $\mathbf{L}$. $\mathbf{L}^{-1}$ is then computed with `numpy.linalg.inv()` to decorrelate the response variables before training. Each transformed response

variable is then modelled separately using an XGBoost regression model via `XGBRegressor()`. The models are trained on the transformed targets using `model.fit(X_train, Y_train)`, and predictions are generated for the validation set using `model.predict(X_val)`. Since the predictions are in the transformed space, they are reverted to the original scale by multiplying them with the Cholesky factor using `numpy.dot()`. This ensures that the predicted responses keep their original correlation structure. The adjusted predictions are then used to compute the ANRMSE across all folds.

6.2.3 Results

XGBoost's ANRMSE is 0.426, which indicates an excellent fit - the best so far. This value indicates XGBoost's capabilities when modelling complex datasets.

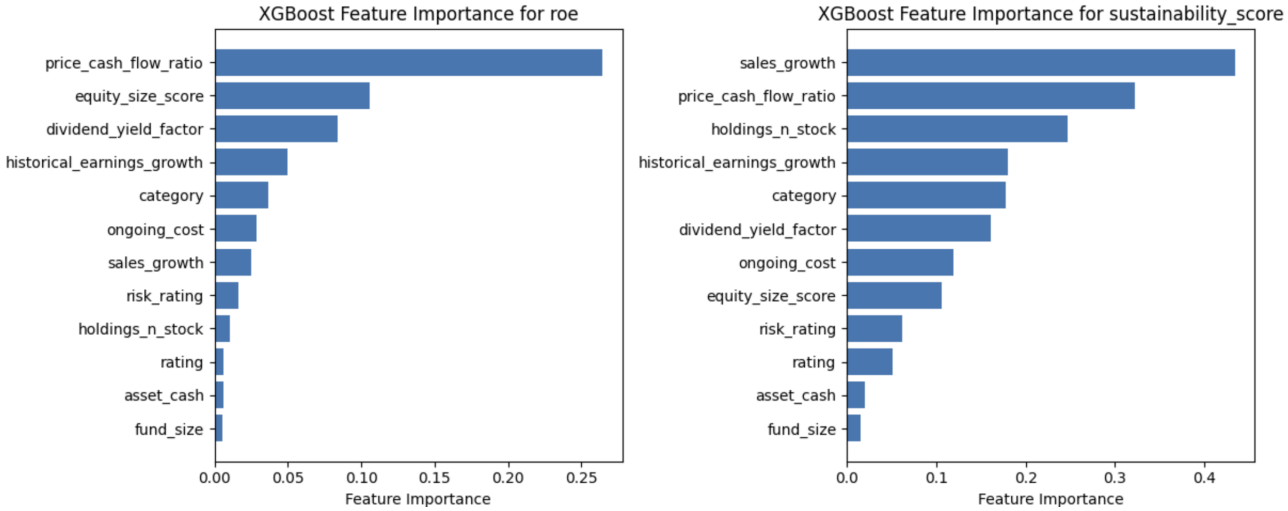

Figure 6.1: XGBoost Feature Importance

The most important feature in predicting the transformed response associated with ROE is the `price_cash_flow_ratio`. This suggests after decorrelation this ratio has the strongest impact on explaining variation in the response variable. However, due to the Cholesky transformation, its influence may be partially shared with sustainability score when reverting to the original response space.

Other significant features include the `equity_size_score` and `dividend_yield_factor`, indicating their substantial contribution to predicting the decorrelated target. The lower importance of variables such as asset cash and fund size suggests that these features provide minimal marginal information once dependencies between responses have been removed.

The most significant feature for the sustainability score is `sales_growth`. Unlike ROE, where a single dominant predictor emerges, sustainability appears to be influenced by multiple factors, as reflected in the more even distribution of feature importance. The `price_cash_flow_ratio` also ranks highly in importance, similar to its role in predicting ROE. This suggests that, before decorrelation, ROE and sustainability score shared some degree of variance, which was separated through the Cholesky decomposition. Other notable contributors include stock holdings, historical earnings growth, and ongoing costs, reinforcing the notion that sustainability is shaped by a broader set of weaker influences rather than a single dominant predictor.

XGBoost has proven to be very effective, but it has limitations in certain scenarios. One major weakness is its inability to capture highly non-linear relationships due to its reliance on piecewise constant decision trees. Neural networks, especially deep learning models, overcome this by using stacked layers with non-linear activations, allowing them to approximate complex functions.

Chapter 7 Neural Networks

The final chapter of this report will cover neural networks. It will start by explaining what a single-output neural network is and then move into a multi-output setting. Then, it will explain a Gaussian Process and a Multi-Output Gaussian Process. Ending with the methodology of a Multi-Output Gaussian Process Neural Network, a type of multi-output neural network, which will be explained and applied to the equity fund dataset.

7.1 Theory

While XGBoost refines predictions by iteratively improving decision trees, it still relies on rule-based partitioning of the feature space. Neural networks, however, take a different approach. They learn complex patterns through layers of interconnected neurons, which adapt to data.

7.1.1 Introduction

A neural network is a model which makes decisions like the human brain. It does this by a process similar to the way biological neurons work together to identify phenomena, weigh decisions and arrive at conclusions.[44]

Nodes are a building block of a neural network. Each node is a linear regression model with input data, weights, a bias (or threshold), and an output.[44] Within a neural network, there are a series of layers of nodes. The first of which is called the input layer and the last is called the output layer. The layers in between are called the hidden layers.

A Feed-Forward Neural Network (FFNN), otherwise known as a multi-layer perceptron, is a type of neural network whose vertices are numbered such that all connections go from a neuron (vertex) to one with a higher number. In other words, the information that enters all the nodes in the layers except the input is the sum of the information from those previous nodes. As information passes through the node, it is also changed by the weights of their respective prior corresponding nodes. The weights transform this information using the activation function.
Each node in the network except the input layer is defined as:

$$h_k^{(n)}(x) = \sigma_n \left(w_{k,0}^{(n)} + \sum_j w_{k,j}^{(n)} o_j^{(n-1)}(x) \right),$$

where $h_k^{(n)}$ is the linear combination of the k-th neuron in layer n, σ_n is the activation function at layer n, $w_{k,0}^{(n)}$ is the bias term for the k-th neuron in layer n, $w_{k,j}^{(n)}$ is the weight from neuron j in layer $n-1$ to neuron k in layer n, $o_j^{(n-1)}(x)$ is the output of neuron j in the previous layer $n-1$, and $o_j^{(0)}(x) = x_j$ are the input features for the input layer $n = 1$.

From what has been described, neural networks can capture linear models. They can be extended to capture potential non-linear relationships through the choice of activation function. Activation functions are non-increasing and are often sigmoids or threshold functions. Some examples of them include: the threshold sigmoid $\sigma(\alpha) = 1$ $(\alpha > 0)$, the logistic sigmoid: $\sigma(\alpha) = (1 + \exp(-\alpha))^{-1}$ and the rectified linear unit, RELU: $\text{RELU}(\alpha) = \max(\alpha, 0)$.
Here is a graphic for a FFNN:

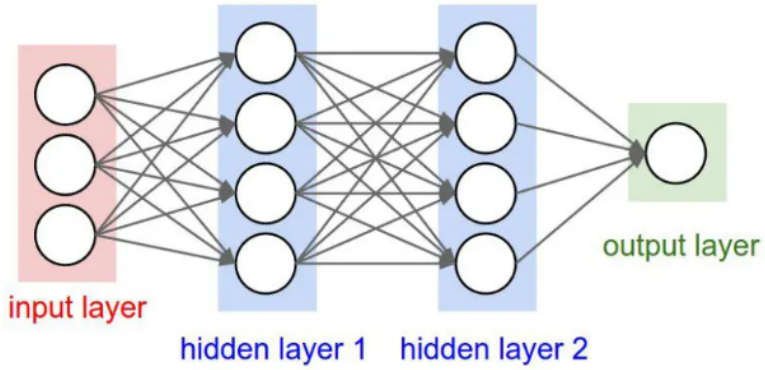

Figure 7.1: Neural Network Representation [45]

Neural networks can learn response inter-correlations between multiple outputs through their shared hidden layers. They capture correlations between these responses by using adaptive hidden units that were shared between the outputs. [46] The shared hidden layers act as a latent space for the inputs, which is then transformed by weights. [47]

Each neuron in the output layer corresponds to one response variable. The weights connecting the hidden layer to the output neurons allow the FFNN also to learn output-specific mappings. These outputs are now coupled through a weight matrix, $W(x)$.

When training neural networks for a dataset, the choice of the loss function is critical in ensuring that all outputs are effectively optimised. A common approach is to compute the loss for each output individually and then aggregate the errors. [47] This can be done using the mean squared error or a weighted loss function to emphasise specific outputs. [47] The general form of the multi-output loss is:

$$L(y, \hat{y}) = \frac{1}{p} \sum_{i=1}^{p} (y_i - \hat{y}_i)^2 ,$$

where p represents the number of outputs, y_i is the actual value for the i-th output, and $\hat{y}_i$ is the predicted value. For cases where outputs have different magnitudes or importance, a weighted loss can be applied:

$$L(y, \hat{y}) = \frac{1}{p} \sum_{i=1}^{p} w_i (y_i - \hat{y}_i)^2 ,$$

where w_i is the weight assigned to the i-th output. This approach ensures that higher-priority outputs receive greater attention during training.

7.1.2 Multi-Output Neural Networks

Multi-Output neural networks do not differ much to single-output neural networks unlike all the other models discussed in this report. However, they do bring their own benefits and drawbacks.

Training neural networks with multiple outputs introduces unique challenges, including gradient imbalance, where outputs with larger gradients dominate the optimisation process.[47] This can lead to poor performance for smaller-gradient outputs. Several techniques can mitigate this issue.
The first of which is Gradient Clipping, which limits the magnitude of gradients during back-propagation to prevent a single output from dominating the learning process. Next, output-specific learning rates can also ensure optimisation is not rushed. Finally, batch normalisation, which involves normalising the activations in each layer, ensures gradients remain stable and uniform across all outputs.

The choice of activation function in the output layer depends on the type of regression problem. For continuous outputs, linear activation functions are typically used:

$$o_j^{(n)}(x) = w_{k,j}^{(n)} o_j^{(n-1)}(x) + b_k^{(n)}$$

However, sigmoid or softmax activation functions can be applied to bounded outputs (e.g., probabilities). For unbounded outputs, the Rectified Linear Unit (ReLU) activation is used due to its simplicity and effectiveness in learning non-linear patterns.

Regularisation helps prevent over-fitting in neural networks. This is especially important in multi-output settings where the network may over-fit one output at the expense of others. An example of this is L2 Regularisation, which penalises large weights in the output layer, effectively shrinking them during training:

$$R(W) = \lambda \sum_{i,j} W_{i,j}^2$$

where λ is the regularisation strength.[47]

Dropout is another regularisation method neural networks can use. It works by literally 'dropping' neurons out from internal layers during training to introduce noise, forcing the network to learn more relevant features. It can also be applied selectively to the output layer and used in MRR. Extending dropout by deactivating the hidden layer neurons ensures that no single output becomes over-fitted.

A key change to a standard Multi-Output Neural Network that I wanted to make was to introduce task-specific layers as well as the shared hidden layers because although we want the model to learn the responses using the same inputs so we consider their dependencies, what is important to consider is how dependent they are. `roe` and `sustainability_score` have a -0.3171 correlation. So, giving them the exact same input nodes would lose some information.

Multi-output neural networks can provide insight into the relationships between outputs and features. However, they do not explicitly consider response intercorrelations. This lack of response correlation matrix consideration in the multi-output neural network makes it not usable for a lot of MRR. A Multi-Output Gaussian Process Neural Network, however, does consider this.

7.2 Multi-Output Gaussian Process Neural Network

7.2.1 Gaussian Process

First, let us define a Gaussian Process, which is a building block of a Multi-Output Gaussian Process (MOGP).

A Gaussian Process model is a probabilistic model over possible functions that fit a set of points.[48] As a result, we can derive the means and variances to represent the function's maximum likelihood estimate and indicate prediction confidence, respectively.[48]

A function, $f(x)$, is drawn from a Gaussian Process if for any finite set of input points $\mathbf{X} = \{\mathbf{x_1}, ..., \mathbf{x_n}\}$, the corresponding function values $\boldsymbol{f}(\mathbf{X}) = [f(\mathbf{x}_1), ..., f(\mathbf{x}_n)]$ follows a multivariate normal distribution:

$$P(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{f}|\mathbf{m}, \mathbf{K}),$$

where $\mathbf{m}(\mathbf{x})$ is the mean function and $\mathbf{K}(x, x')$ is the kernel function, which is the positive definite covariance between function values.[48]
From here, we predict at new points $\mathbf{X}_*$ as $\boldsymbol{f(X_*)}$ for the joint distribution of $\boldsymbol{f}$ and $\boldsymbol{f_*}$:

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m}(\mathbf{X}) \\ \mathbf{m}(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix} \right),$$

where $\mathbf{K} = K(\mathbf{X}, \mathbf{X})$, $\mathbf{K}_* = K(\mathbf{X}, \mathbf{X}_*)$ and $\mathbf{K}_{**} = K(\mathbf{X}_*, \mathbf{X}_*)$ and with no observation, the mean, $(m(\mathbf{X}, m(\mathbf{X}_*)) = \mathbf{0}$.[48]

This equation is the joint probability distribution $P(\boldsymbol{f}, \boldsymbol{f_*}|\mathbf{X}, \mathbf{X}_*)$ over $\boldsymbol{f}$ and $\boldsymbol{f_*}$ but this model is regression meaning only the conditional distribution $P(\boldsymbol{f_*}|\boldsymbol{f}, \mathbf{X}, \mathbf{X}_*)$ is needed.[48] The conditional distribution can be derived using the joint distribution, and it uses the marginal and conditional distributions of the multivariate normal theorem, which is explored further in Bishop et al. (2006). This proves the conditional distribution:

$$\boldsymbol{f_*}|\boldsymbol{f}, \mathbf{X}, \mathbf{X}_* \sim \mathcal{N}(\mathbf{K}_*^\top \mathbf{K}^{-1} \boldsymbol{f}, \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{K}_*).$$

The conditional distribution gives Gaussian Process regression's predictive equations:

$$\bar{\boldsymbol{f}}_*|\mathbf{X}, \mathbf{y}, \mathbf{X}_* \sim \mathcal{N}(\bar{\boldsymbol{f}}_*, \mathrm{Cov}(\boldsymbol{f}_*)),$$

where $\bar{\boldsymbol{f}}_* = \mathbf{K}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$ and $\mathrm{Cov}(\boldsymbol{f}_*) = \mathbf{K}_{**} - \mathbf{K}_*^{\mathrm{T}} [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{K}_*$.[48]
The adjustment to $\mathbf{K}$ happens because of the noise we have to add to true function values in reality: $y = f(x) + \epsilon$, where ϵ is a standard independent and identically distributed Gaussian noise with variance σ_n^2.[48] This adjusts the prior covariance to: $\boldsymbol{\Sigma}_\mathbf{Y} = \mathbf{K} + \sigma_n^2 \mathbf{I}$, which gives the previously defined predictive distribution.

7.2.2 Multi-Output Gaussian Process Neural Networks

Now Gaussian Processes have been set up, let us define a basic Multi-Output Gaussian Process (MOGP). This involves L independent Gaussian processes $\{\boldsymbol{f}_\ell\}$ with $\ell = 1, \ldots, L$, each with kernel K_ℓ and leads to the definition of the marginal probability of MOGP:[49]

$$p(\mathbf{Y}|\mathbf{X}) = \int d\mathbf{M} p(\mathbf{M}) \prod_{\ell=1}^{L} d\boldsymbol{f}_\ell p(\boldsymbol{f}_\ell|\mathbf{X}) \times \prod_{i=1}^{N} \mathcal{N}\big(y_i \mid \mathbf{MF}(\mathbf{x}_i), \beta\big), \tag{7.1}$$

where $\mathbf{M}$ is a $D_Y \times L$ mixing matrix, $\mathbf{MF}$, the covariance structure, is being matrix multiplied, and $p(\mathbf{M})$ is a unit Normal prior on $\mathbf{M}$.[50] Other covariance structures are possible for the Gaussian process, because owe are trying to uncover changes to the likelihoods, the model employs this basic pattern. For this reason, we use radial basis function kernels throughout.[50]

Equation 7.1 can be extended by considering a model specified by its marginal likelihood as:

$$\int d\mathbf{M} p(\mathbf{M}) \prod_{\ell=1}^{L} d\boldsymbol{f}_\ell p(\boldsymbol{f}_\ell|\mathbf{X}) \prod_{i=1}^{N} \mathcal{N}\big(y_i \mid \mathbf{M}\sigma(\widetilde{\mathbf{M}}\mathbf{F}(\mathbf{x}_i)), \beta\big), \tag{7.2}$$

where $\mathbf{F}$ is passed through a neural network before it is used to compute the mean function for the likelihood.[50] $\widetilde{\mathbf{M}}$ is a $D_H \times D_L$ matrix and $\mathbf{M}$ is a $D_Y \times D_H$ matrix, where D_H is a new hyper-parameter which controls the number of 'hidden units'.[50] $\sigma(\cdot)$ is the activation function and $\mathbf{M}$ has a unit Normal prior. This is the mathematical framework of a MOGP Neural Network (NN). Now, let us go into how this actually generates and refines predictions.

Practically, what will happen is, is that the neural network will come up with an initial prediction using its standard mechanisms. So, it uses forward and backward propagation to determine the optimal weights and biases for the most accurate initial prediction, $\hat{\mathbf{Y}}$. This prediction is then refined by the MOGP in the following way.

When a new data point, $\mathbf{x}_*$, is provided for task l, it is given by

$$\bar{f}_l(\mathbf{x}_*) = \left(\mathbf{k}_l^f \otimes \mathbf{k}_*^x\right)^T \boldsymbol{\Sigma}^{-1}\hat{\mathbf{Y}}, \quad \Sigma = \mathbf{K}^f \otimes \mathbf{K}^x + \mathbf{D} \otimes \mathbf{I},$$

where $\otimes$ denotes the Kronecker product, $\mathbf{k}_l^f$ selects the l^{th} column of $\mathbf{K}^f$, $\mathbf{k}_*^x$ is the vector of covariances between the test point $\mathbf{x}_*$ and the training points, $\mathbf{K}^x$ is the matrix of covariances between all pairs of training points, $\mathbf{D}$ is an $M \times M$ diagonal matrix in which the $(l,l)^{\text{th}}$ element is σ_l^2, and $\boldsymbol{\Sigma}$ is an $MN \times MN$ matrix with the covariances of the responses and predictors all in one.[51] $\mathbf{K}^f = \boldsymbol{\Sigma}_{\mathbf{Y}}$ so it is essential because it is what makes MOGP NN an MRR model.

The predictor covariance matrix $\mathbf{K}^x$ is computed using the radial basis function kernel:

$$K^x(i,j) = \exp\left(-\frac{||X_i - X_j||^2}{2\ell^2}\right),$$

where ℓ is calculated by taking the median of all the pairwise Euclidean distances in the dataset and normalising by $\sqrt{2}$.[51]

The task covariance, or coregionalisation matrix, $\mathbf{K}_f$, is estimated using the sample response covariance, $\boldsymbol{\Sigma}_{\mathbf{Y}}$:

$$K_f(l,k) = \frac{1}{N}\sum_{i=1}^{N}(Y_{i,l} - \mu_l)(Y_{i,k} - \mu_k),$$

where μ_l is the mean response for each task. The noise variance matrix $\mathbf{D}$ is then computed as the diagonals of the $\mathbf{K}_f$ matrix.[51]

Finally, there is a potential issue of dimensionality but this is sorted by a shape restructuring of the neural network prediction from $N \times M$ to $MN \times 1$.

7.3 Application

7.3.1 Small-Scale Example

We will apply the MOGP NN to this familiar dataset.

X_1: Hours Studied	X_2: Time Spent on Papers	Y_1: Math Scores	Y_2: Science Scores
5	2	78	80
7	3	85	79
8	4	88	88
3	1	65	70
10	5	92	74

Table 7.1: Study Time vs. Exam Scores

The first step is normalising all of the data. We then define a two-layer neural network with a shared hidden layer and a task-specific layer to keep simplicity. See A.2.4 for the image. Although the neural network does learn response intercorrelation in the hidden layer, this is not explicit. As a result, let us assume that after forward and backward propagation, we have the following weight matrix and bias vector (to 2dp):

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & w_{h1} \\ w_{21} & w_{22} & w_{h2} \end{bmatrix} = \begin{bmatrix} 1.40 & -0.72 & 0.70 \\ -0.47 & 0.41 & 0.72 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_{h1} \\ b_{h2} \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0 \\ 0.53 \\ 0.14 \end{bmatrix}.$$

This gives the normalised inputs and neural network predictions as (to 2dp):

X_1: Hours Studied	X_2: Time Spent on Papers	Y_1: Maths Score	Y_2: Science Score
0.29	0.25	0.27	0.28
0.57	0.50	0.70	-0.16
0.71	0.75	0.84	-0.54
0.00	0.00	0.19	0.36
1.00	1.00	1.03	-1.09

Table 7.2: Normalised Maths and Science Scores

These values correspond to $\hat{\mathbf{Y}}$ in the MOGP notation. Now, we need to adjust these predictions.

The first step involves calcaulting the input and response covariance matrices, $\mathbf{K}^x$ and $\mathbf{K}_f$. The input covariance matrix $\mathbf{K}^x$ is computed using the radial basis function kernel:

$$K^x(i,j) = \exp\left(-\frac{||X_i - X_j||^2}{2\ell^2}\right),$$

where ℓ is calculated by taking the median of all the pairwise Euclidean distances in the dataset and normalising by $\sqrt{2}$. For simplicity, let us assume $\ell = 2$. Here is how you compute $K^x(i,j)$ for 1 element: $(i,j) = (1,2)$. This manual computation gives:

$$||X_1 - X_2||^2 = (0.29 - 0.25)^2 + (0.57 - 0.5)^2 = 0.0065.$$

Therefore, $K^x(1,2) = \exp\left(-\frac{0.0065}{2(2.5)^2}\right) \approx 1.00$. The task covariance matrix, $\mathbf{K}_f$, is estimated using the sample response covariance:

$$K^f(l,k) = \frac{1}{N}\sum_{i=1}^{N}(Y_{i,l} - \mu_l)(Y_{i,k} - \mu_k),$$

where μ_l is the mean response for each task:

$$\mu_1 = \frac{0.27 + 0.70 + 0.84 + 0.19 + 1.03}{5} = 0.606, \quad \mu_2 = \frac{0.28 - 0.16 - 0.54 + 0.36 - 1.09}{5} = -0.23$$

Plugging in for $\mathbf{K}_f$ gives:

$$\mathbf{K}^f = \begin{bmatrix} 0.28 & -0.28 \\ -0.28 & 0.35 \end{bmatrix}$$

The noise variance matrix $\mathbf{D}$ is then computed as the diagonals of the full $\mathbf{K}_f$ matrix. Therefore, $\mathbf{D}$ is:

$$\mathbf{D} = \begin{bmatrix} 0.28 & 0 \\ 0 & 0.35 \end{bmatrix},$$

The full covariance matrix, $\mathbf{\Sigma}$, for the responses and the predictors, is then given by:

$$\mathbf{\Sigma} = \mathbf{K}_f \otimes \mathbf{K}^x + \mathbf{D} \otimes \mathbf{I},$$

which is a 10×10 matrix, so for simplicity, let us avoid using it and talk more abstractly. This matrix helps us to calculate the updated normalised prediction using:

$$\bar{f}(\mathbf{x}_*) = (\mathbf{k}_l^f \otimes \mathbf{k}_*^x)^\top \mathbf{\Sigma}^{-1}\hat{\mathbf{Y}}.$$

Here, we remember the potential issue of dimensionality is sorted by a shape restructuring of the neural network prediction from 5×2 to 10×1. Finally, this prediction will be in its normalised form, so the MOGP NN reverts the normalised predictions back to normal and then assess the ANRMSE.

7.3.2 Code Explanation

The code integrates a Neural Network (`Feature Extractor`) with a Multi-Output Gaussian Process (`MultiTaskGPModel`) to handle multi-task regression. The neural network extracts shared patterns from input features, while the Gaussian Process explicitly models dependencies between outputs using a coregionalisation matrix. A custom loss function ensures the predicted covariance structure aligns with actual relationships between outputs.

The neural network processes inputs through shared layers in `FeatureExtractor.forward()`, transforming them into a latent feature representation. Task-specific layers (`TaskHead.forward()`) then refine these representations, allowing the model to capture common structures while preserving flexibility for each task.

The Gaussian Process (GP) component models the outputs as a multivariate Gaussian distribution, capturing both uncertainty and inter-task correlations. It is implemented in `MultiTaskGPModel` using a constant mean function (`ConstantMean()`), an RBF kernel (`RBFKernel()`) for smooth variations, and a multitask covariance kernel (`MultitaskKernel()`) to structure dependencies between outputs.

A standard MSE loss function is used to monitor how accurate predictions are.

The model is trained using k-fold CV in `train_model_kfold()`, which splits the dataset into training and validation folds. Then, it trains the neural network using `FeatureExtractor.train()`, optimising parameters via stochastic gradient descent. Next, it passes transformed features into

the Gaussian Process, optimising its hyperparameters through marginal log-likelihood maximisation (`gpytorch.mlls.ExactMarginalLogLikelihood()`). Finally, it jointly updates both components to align the neural network representations with the Gaussian Process structure.

7.3.3 Results

The ANRMSE for the model is 0.411, indicating a reasonable fit, which is the best model. This value indicates the best fit, which makes sense because of how complex a MOGP NN is.

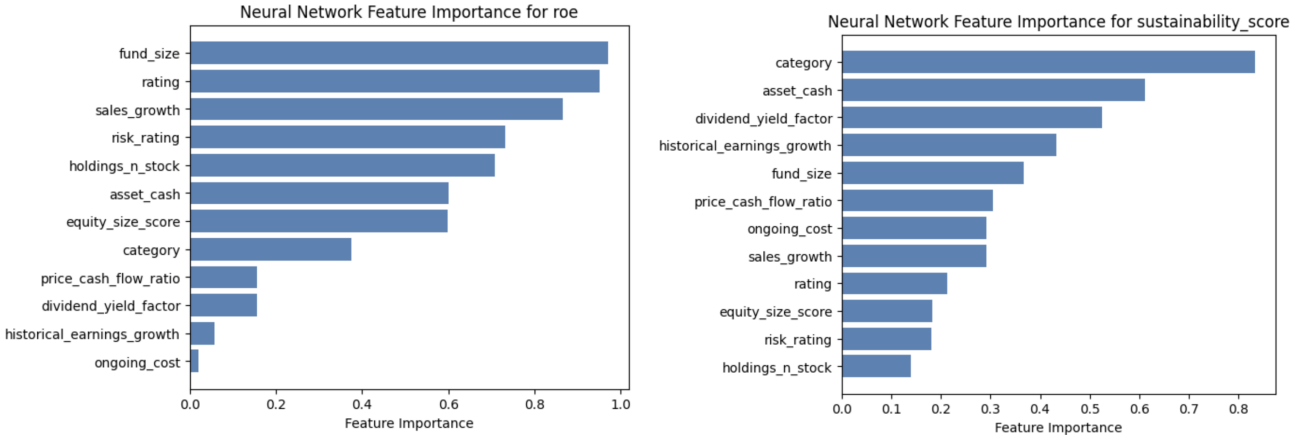

Figure 7.2: MOGP NN Feature Importance

For return on equity, fund size, rating, sales growth, and risk rating are the most influential features. These features contribute heavily to the hidden layer activations in the neural network, shaping the learned representations. The GP component then refines these representations by modelling the covariance structure between outputs, leveraging correlations to improve predictions. Higher-importance features significantly reduce posterior variance, making predictions more stable and reliable.

For sustainability score, category, asset cash, and dividend yield factor play a dominant role. Since the Gaussian process integrates these features into a shared latent function space, their high importance indicates that they strongly influence the joint distribution of outputs. The neural network learns a nonlinear transformation of these features, which the Gaussian process then smooths through the RBF. This allows the model to fit the test data well while maintaining response intercorrelation.

In essence, highly ranked features act as strong priors in the network, guiding weight updates in the neural layers and refining the covariance estimation in the Gaussian process. This allows the model to balance assessing feature importance and accurate predictions.

Although Neural Networks have proven to be very effective when combined with Multi-Task Gaussian Processes, they come with some setbacks. One major drawback is their dependency on large amounts of data. Neural networks require extensive labelled datasets to learn meaningful patterns, and without sufficient data, they are prone to overfitting or failing to fit test data effectively. Another significant limitation is their computational complexity. Training deep neural networks requires substantial computational power, often necessitating the use of specialised hardware such as GPUs or TPUs. The high memory requirements for storing weights and activations add to their resource intensity, making them expensive to train and deploy.

Chapter 8 Conclusion

8.1 Summary of Findings and Model Comparisons

8.1.1 Summary of Findings

Here is a table summarising the key figures generated from this report from the model methods:

Table 8.1: Model Fit Comparison on Equity Fund Dataset

Model	ANRMSE	Model Fit
Chapter 3: Linear Regression		
Model 1: MRLR with all predictors	0.7606	Unsatisfactory
Model 2: Forward Selection	0.7606	Unsatisfactory
Model 3: Backward Selection	0.7924	Unsatisfactory
Model 4: Bidirectional Selection	0.7606	Unsatisfactory
Model 5: Interaction Terms	0.5613	Good
Model 6: Non-Linear Terms	0.6893	Satisfactory
Chapter 4: Shrinkage Methods		
Model 7: MRCE	0.7612	Unsatisfactory
Model 8: RRRR	0.7510	Unsatisfactory
Model 9: MRCE including Polynomial Terms	0.6781	Satisfactory
Model 10: RRRR including Polynomial Terms	0.7276	Unsatisfactory
Model 11: MRCE including Interaction Terms	0.6777	Satisfactory
Model 12: RRRR including Interaction Terms	0.7231	Unsatisfactory
Chapter 5: Random Forests		
Model 13: Covariance Regression with Random Forests	0.5238	Good
Chapter 6: XGBoost		
Model 14: Cholesky-Decomposition XGBoost	0.4267	Excellent
Chapter 7: Neural Networks		
Model 15: Multi-Output Gaussian Process Neural Network	0.4112	Excellent

The model fit was set out based on criteria defined in the exploratory data analysis.

8.1.2 Model Comparisons

The linear regression models tested different variable selection and interaction strategies. The MRLR model with all predictors performed poorly (ANRMSE = 0.7606), and forward, backwards, and bidirectional selection failed to improve performance significantly. Backward selection performed the worst (ANRMSE = 0.7924), suggesting that eliminating variables in this context led to worse predictive performance. However, interaction terms (ANRMSE = 0.5613) substantially improved accuracy, achieving a "Good" model fit classification. The non-linear terms model (ANRMSE = 0.6893) also improved upon basic regression, receiving a "Satisfactory" classification.

Shrinkage methods aimed to improve regression stability by using penalisation techniques. The MRCE (ANRMSE = 0.7612) and RRRR (ANRMSE = 0.7510) models performed similarly to standard regression, with "Unsatisfactory" results. Introducing polynomial terms in MRCE (ANRMSE = 0.6781, "Satisfactory") improved performance, but the same approach for RRRR (ANRMSE = 0.7276, "Unsatisfactory") was less effective. Adding interaction terms improved MRCE (ANRMSE = 0.6777, "Satisfactory") but did not sufficiently improve RRRR (ANRMSE = 0.7231, "Unsatisfactory").

Tree-based models significantly improved over linear and shrinkage-based methods. The Random Forest model (ANRMSE = 0.5238) significantly outperformed previous methods and was classified as "Good" in terms of model fit. Further improvement was observed with XGBoost (ANRMSE = 0.4267), which was classified as "Excellent." This indicates that gradient-boosted decision trees effectively captured complex relationships among predictors and responses.

The MOGP NN (ANRMSE = 0.4112) achieved the best performance, receiving an "Excellent" classification. This suggests deep learning approaches, particularly those incorporating Gaussian processes, are highly effective in capturing the underlying dependencies in MRR.

From this comparison, it is evident that traditional regression models struggle with multiple-response prediction unless interaction or polynomial terms are introduced. Shrinkage methods improve stability but do not significantly improve performance over non-regularised models. Tree-based models (Random Forest, XGBoost) and Neural Networks achieve the best performance, with XGBoost and Gaussian Process Neural Networks leading the results.

8.2 Challenges and Limitations

There were some challenges in this report and limitations in how this could be set out. The first of which was determining the method to impute the equity fund dataset because there were a lot of valid approaches and trying to find one I thought was most appropriate was difficult.

The next challenge was determining what was and was not an MRR model. Initially, I coded up models that were not strictly multiple-response. So, they would implicitly consider the response-covariance matrix, for example, in their loss function, but they would not be rigorous in how they used it. An example of this was multivariate ridge because its penalty applies jointly across all responses, meaning it effectively regularises the entire coefficient matrix, making it a natural choice for MRR. However, it does not directly use the response-covariance matrix, so it is not valid as an MRR model. Covariance is unscaled correlation, and considering correlation is important because it reduces standard errors compared to the estimates based on independent correlation.[52] Standard errors determine confidence in the results.

One of the most significant limitations of MRR outputs is that they are not always able to interpret.[53] This was especially true for the latter methods. These types of models can be known as a

"black-box" because although they are very accurate, you do not understand how to interpret them. This sounds like a big problem, however, having this is not necessarily an issue because you can still use that to your favour.

8.3 Conclusion

This report first examined MRLR, which succinctly follows single-response linear regression. It was combined with stepwise selection and sequential MANOVA to determine the appropriate predictors. Next, this report delved into multiple-response shrinkage methods, specifically MRCE and RRRR.

Then, random forests were examined through CRRFs, which aimed to maximise the Euclidean distance between nodes when splitting. After this, XGBoost was covered and extended to multiple responses using the Cholesky-Gaussian Decomposition, which removes any correlation between the responses so 2 independent models can be applied. The final model that was looked at was a Neural Network, which was combined with a MOGP to consider response covariance directly.

8.4 Future Work

This dissertation has explored MRR models using various methodologies and assessed their effectiveness on a given dataset. However, several avenues for further research remain.

One extension is to evaluate these MRR models on a diverse set of datasets. This would enable a more hierarchical understanding of their performance and help identify patterns in model effectiveness across different levels of multicollinearity, and response variable dependencies. Such an analysis could be useful in selecting the most suitable MRR model for a given context.

Alternatively, further work could look at more MRR models on this dataset to identify an optimal model. This would involve testing additional regression techniques, including hybrid models that integrate shrinkage, dimensionality reduction, and non-linear approaches. A search for the "perfect" MRR model for this dataset could also involve methods such as hyper-parameter tuning or ensembles.

Bibliography

1. Kanti V Mardia, John T Kent, and Charles C Taylor. *Multivariate analysis*. John Wiley & Sons, 2024.

2. James Chen. Investing in equity funds: A beginner's guide. *Investopedia*, April 2024. Reviewed by Gordon Scott, Fact checked by Ariel Courage.

3. Larry Fink. Larry fink's 2021 letter to ceos. `https://www.blackrock.com/corporate/about-us/sustainability-resilience-research`, January 2021. Chairman and CEO of BlackRock.

4. Stockopedia. Risk rating, 2024. Accessed: 27 November 2024.

5. AMG Funds LLC. Equity style analysis: Growth vs. value, 2023. Accessed: 27 November 2024.

6. Adam Hayes. Growth stock: What it is, examples, vs. value stock, December 2023. Updated 11 December 2023. Reviewed by Chip Stapleton. Fact checked by Kirsten Rohrs Schmitt. Accessed: 27 November 2024.

7. Hitul Adatiya. Eda on european mf dataset, 2022. Version 2 of 4. Accessed: 27 November 2024. Licensed under Apache 2.0 open source.

8. Chris B. Murphy. Operating costs definition: Formula, types, and real-world examples, 2024. Updated 28 June 2024. Reviewed by David Kindness. Fact checked by Amanda Jackson. Accessed: 27 November 2024. Part of the series: The Evolution of Accounting and its Terminology.

9. Shangzhi Hong and Henry S Lynn. Accuracy of random-forest-based imputation of missing data in the presence of non-normality, non-linearity, and interaction. *BMC medical research methodology*, 20:1–12, 2020.

10. Cátia M. Salgado, Carlos Azevedo, Hugo Proença, and Susana M. Vieira. Missing data. In Ross C. Mitchell, editor, *Secondary Analysis of Electronic Health Records*, pages 143–162. Springer, Cham, 2016. Open Access chapter distributed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License.

11. CFA Institute. Basics of multiple regression and underlying assumptions, 2024. Refresher Reading for CFA Program Level II, Quantitative Methods.

12. Richard Johnson and Dean Wichern. Multivariate linear regression models: Section 7.7. In *Applied Multivariate Statistical Analysis: Pearson New International Edition*, pages 360–429. Pearson Education, Limited, 6th edition, 2013. Accessed: 27 November 2024.

13. Wonyul Lee and Yufeng Liu. Simultaneous multiple response regression and inverse covariance matrix estimation via penalized gaussian maximum likelihood. *Journal of multivariate analysis*, 111:241–255, 2012.

14. Smith Gary. Step from stepwise. *Journal of Big Data*, 5(1):1–12, 2018.

15. Gudmund R. Iversen. Multivariate analysis of variance and covariance (manova and mancova). In Michael S. Lewis-Beck, Alan Bryman, and Tim Futing Liao, editors, *The SAGE Encyclopedia of Social Science Research Methods*, volume 2, pages 702–703. SAGE Publications, Inc., Thousand Oaks, CA, 2004.

16. Newsom. Multivariate analysis of variance. *Psy 522/622 Multiple Regression and Multivariate Quantitative Methods*, 2024. Winter 2024 Lecture Notes.

17. STAT 505 Pennsylvania State University. Lesson 8: Multivariate analysis of variance (manova), 2024.

18. Daniel N Moriasi, Jeffrey G Arnold, Michael W Van Liew, Ronald L Bingner, R Daren Harmel, and Tamie L Veith. Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. *Transactions of the ASABE*, 50(3):885–900, 2007.

19. Kristin L. Sainani. Multivariate regression: The pitfalls of automated variable selection. *PM&R*, 5(9):791–794, 2013.

20. Nikolaos Ignatiadis and Panagiotis Lolas. Group regularised ridge regression via empirical bayes noise level cross-validation. *arXiv preprint arXiv:2010.15817*, 2020.

21. Aaron J Molstad. New insights for the multivariate square-root lasso. *arXiv preprint arXiv:1909.05041*, 2019.

22. Ashin Mukherjee and Ji Zhu. Reduced rank ridge regression and its kernel extensions. *Statistical analysis and data mining: the ASA data science journal*, 4(6):612–622, 2011.

23. Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

24. Adam J Rothman, Elizaveta Levina, and Ji Zhu. Sparse multivariate regression with covariance estimation. *Journal of Computational and Graphical Statistics*, 19(4):947–962, 2010.

25. Jian Guo, Elizaveta Levina, George Michailidis, and Ji Zhu. Joint estimation of multiple graphical models. *Biometrika*, 98(1):1–15, 02 2011.

26. Frederik Questier, Raf Put, Danny Coomans, Beata Walczak, and Yvan Vander Heyden. The use of cart and multivariate regression trees for supervised and unsupervised feature selection. *Chemometrics and Intelligent Laboratory Systems*, 76(1):45–54, 2005.

27. Elsevier Inc. *Data Science Process*. Elsevier, Amsterdam, Netherlands, 2019. DOI: https://doi.org/10.1016/B978-0-12-814761-0.00002-2.

28. Quebec Centre for Biodiversity Science. Qcbs r workshop series: Workshop 10: Advanced multivariate analyses in r, 2023. Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Licenses all content.

29. Mark Segal and Yuanyuan Xiao. Multivariate random forests. *WIREs Data Mining and Knowledge Discovery*, 1(1):80–87, 2011.

30. Cansu Alakus, Denis Larocque, and Aurélie Labbe. Covariance regression with random forests. *BMC bioinformatics*, 24(1):258, 2023.

31. L Breiman, JH Friedman, RA Olshen, and CJ Stone. Classification and regression trees. boca raton, florida: Chapman hall/crc, 1984.

32. Hoora Moradian, Denis Larocque, and François Bellavance. L_1 l 1 splitting rules in survival forests. *Lifetime data analysis*, 23:671–691, 2017.

33. Sami Tabib and Denis Larocque. Non-parametric individual treatment effect estimation for survival data with random forests. *Bioinformatics*, 36(2):629–636, 2020.

34. Cansu Alakuş, Denis Larocque, Sébastien Jacquemont, Fanny Barlaam, Charles-Olivier Martin, Kristian Agbogba, Sarah Lippé, and Aurélie Labbe. Conditional canonical correlation estimation based on covariates with random forests. *Bioinformatics*, 37(17):2714–2721, 2021.

35. Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests. *Project Euclid*, 2019.

36. Benjamin Lu and Johanna Hardin. A unified framework for random forest prediction error estimation. *Journal of Machine Learning Research*, 22(8):1–41, 2021.

37. Cansu Alakus, Denis Larocque, and Aurelie Labbe. Rfpredinterval: An r package for prediction intervals with random forests and boosted forests. *arXiv preprint arXiv:2106.08217*, 2021.

38. N Azizah, LS Riza, and Y Wihardi. Implementation of random forest algorithm with parallel computing in r. In *Journal of Physics: Conference Series*, volume 1280, page 022028. IOP Publishing, 2019.

39. Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

40. Alexander März. Multi-target xgboostlss regression. *arXiv preprint arXiv:2210.06831*, 2022.

41. T. Muschinski, G. J. Mayr, T. Simon, N. Umlauf, and A. Zeileis. Cholesky-based multivariate gaussian regression. *Econometrics and Statistics*, 2022.

42. DSG Pollock and Emi Mise. The cholesky decomposition of a toeplitz matrix and a wiener-kolmogorov filter for seasonal adjustment. Technical report, 2020.

43. StatQuest with Josh Starmer. XGBoost Part 1 (of 4): Regression, 2019. [Online; accessed 6-March-2025].

44. IBM Team. What is a neural network? `https://www.ibm.com/topics/neural-networks`, 2024. Accessed December 18, 2024.

45. Adrian Rosebrock. A simple neural network with python and keras. `https://pyimagesearch.com/2016/09/26/a-simple-neural-network-with-python-and-keras/`, September 2016. Accessed December 18, 2024.

46. Andrew Gordon Wilson, David A Knowles, and Zoubin Ghahramani. Gaussian process regression networks. *arXiv preprint arXiv:1110.4411*, 2011.

47. Nicolas Menet, Michael Hersche, Geethan Karunaratne, Luca Benini, Abu Sebastian, and Abbas Rahimi. Mimonets: Multiple-input-multiple-output neural networks exploiting computation in superposition. *Advances in Neural Information Processing Systems*, 36:39553–39565, 2023.

48. Jie Wang. An intuitive tutorial to gaussian process regression. *Computing in Science & Engineering*, 25(4):4–11, 2023.

49. Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

50. Martin Jankowiak and Jacob Gardner. Neural likelihoods for multi-output gaussian processes. *arXiv preprint arXiv:1905.13697*, 2019.

51. Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction. *Advances in neural information processing systems*, 20, 2007.

52. Asokan Mulayath Variyath and Anita Brobbey. Variable selection in multivariate multiple regression. *Plos one*, 15(7):e0236067, 2020.

53. Jo Jackson. Multivariate techniques: Advantages and disadvantages. `https://classroom.synonym.com/multivariate-techniques-advantages-and-disadvantages-123456.html`, 2018. Updated September 05, 2018. Accessed December 06, 2024.

Chapter A Appendices

A.1 Additional Data Visualisations

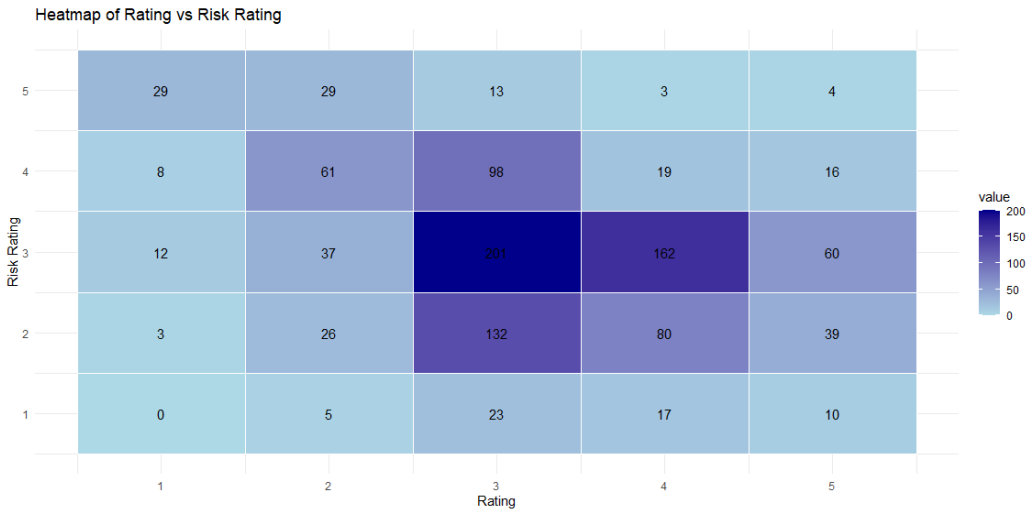

Figure A.1: Rating vs Risk Rating

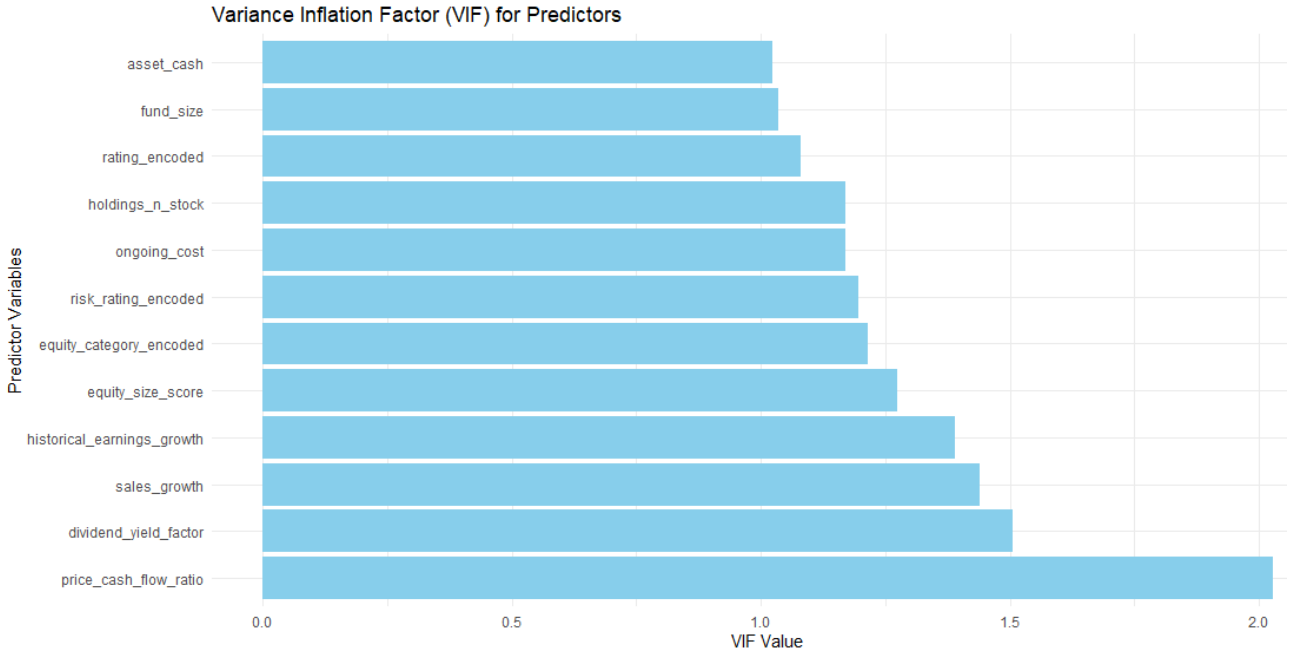

Figure A.2: VIF Values post Dataset Cleaning

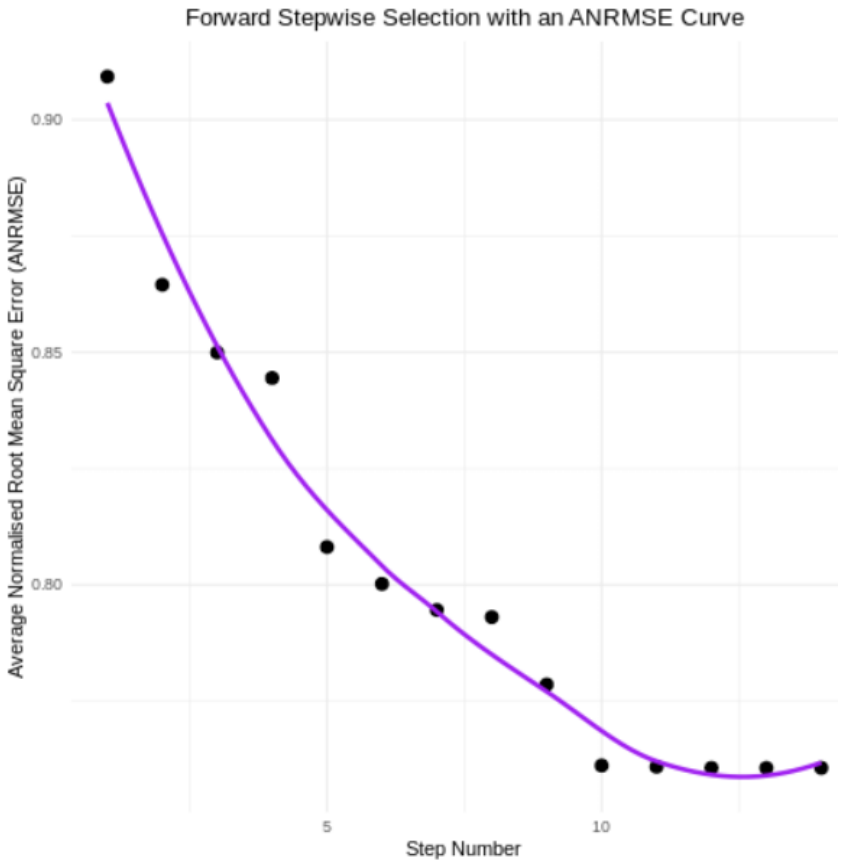

Figure A.3: Forward Stepwise Selection with an ANRMSE Curve

A.2 Additional Calculations

A.2.1 Multiple Response Linear Regression

Covariance Expansion

$$\text{Cov}(\mathbf{XB}) = \text{E}[(\mathbf{XB} - \text{E}[\mathbf{XB}])(\mathbf{XB} - \text{E}[\mathbf{XB}])^\top]$$

Since expectation is linear: $\text{E}[\mathbf{XB}] = \text{E}[\mathbf{X}]\mathbf{B}$, the centered term becomes:

$$(\mathbf{XB} - \text{E}[\mathbf{X}]\mathbf{B}) = (\mathbf{X} - \text{E}[\mathbf{X}])\mathbf{B}$$

Substitute this into the covariance definition:

$$\text{Cov}(\mathbf{XB}) = \text{E}\left[(\mathbf{X} - \text{E}[\mathbf{X}])\mathbf{B} \cdot (\mathbf{X} - \text{E}[\mathbf{X}])^\top \mathbf{B}^\top\right]$$

Factor out $\mathbf{B}$ because it is constant:

$$\text{Cov}(\mathbf{XB}) = \mathbf{B}^\top \text{E}\left[(\mathbf{X} - \text{E}[\mathbf{X}])(\mathbf{X} - \text{E}[\mathbf{X}])^\top\right]\mathbf{B}$$

The middle term is just $\text{Cov}(\mathbf{X})$, therefore: $\text{Cov}(\mathbf{XB}) = \mathbf{B}^\top \text{Cov}(\mathbf{X})\mathbf{B}$.

A.2.2 Reduced Rank Ridge Regression

Mean Centring and Response-Covariance Consideration

The response matrix $\mathbf{Y}$ can be written as the sum of its mean-centred component and the mean matrix: $\mathbf{Y} = (\mathbf{Y} - \bar{\mathbf{Y}}) + \bar{\mathbf{Y}}$. In this decomposition, $\mathbf{Y} - \bar{\mathbf{Y}}$ is the mean-centred version of $\mathbf{Y}$, while $\bar{\mathbf{Y}}$ is a matrix where each row is the mean response vector $\bar{\mathbf{y}}$.

Substituting the decomposition $\mathbf{Y} = (\mathbf{Y} - \bar{\mathbf{Y}}) + \bar{\mathbf{Y}}$ into $\mathbf{Y}^\top\mathbf{Y}$, gives $[(\mathbf{Y} - \bar{\mathbf{Y}}) + \bar{\mathbf{Y}}]^\top[(\mathbf{Y} - \bar{\mathbf{Y}}) + \bar{\mathbf{Y}}]$. Expanding this product results in

$$\mathbf{Y}^\top\mathbf{Y} = (\mathbf{Y} - \bar{\mathbf{Y}})^\top(\mathbf{Y} - \bar{\mathbf{Y}}) + \bar{\mathbf{Y}}^\top(\mathbf{Y} - \bar{\mathbf{Y}}) + (\mathbf{Y} - \bar{\mathbf{Y}})^\top\bar{\mathbf{Y}} + \bar{\mathbf{Y}}^\top\bar{\mathbf{Y}}.$$

The two cross terms $\bar{\mathbf{Y}}^\top(\mathbf{Y} - \bar{\mathbf{Y}})$ and $(\mathbf{Y} - \bar{\mathbf{Y}})^\top\bar{\mathbf{Y}}$ are equal to zero. This vanishing comes from the fact that the mean response matrix $\bar{\mathbf{Y}}$ is constant across all rows, meaning that the sum of all deviations from the mean is equal to zero. Since:

$$\sum_{i=1}^{n}(\mathbf{y}_i - \bar{\mathbf{y}}) = 0,$$

it follows that when multiplied by $\bar{\mathbf{Y}}$, the result also vanishes:

$$\sum_{i=1}^{n}\bar{\mathbf{y}}^\top(\mathbf{y}_i - \bar{\mathbf{y}}) = \bar{\mathbf{y}}^\top\sum_{i=1}^{n}(\mathbf{y}_i - \bar{\mathbf{y}}) = 0.^{23}$$

Since this holds for all response variables, it means that $\bar{\mathbf{Y}}^\top(\mathbf{Y} - \bar{\mathbf{Y}}) = 0 \quad \text{and} \quad (\mathbf{Y} - \bar{\mathbf{Y}})^\top\bar{\mathbf{Y}} = 0$. Substituting these results back into the expansion simplifies the equation to:

$$\mathbf{Y}^\top\mathbf{Y} = (\mathbf{Y} - \bar{\mathbf{Y}})^\top(\mathbf{Y} - \bar{\mathbf{Y}}) + \bar{\mathbf{Y}}^\top\bar{\mathbf{Y}}.$$

The term $\bar{\mathbf{Y}}^\top\bar{\mathbf{Y}}$ can be further simplified. Since each row of $\bar{\mathbf{Y}}$ is equal to the mean response vector $\bar{\mathbf{y}}$, and there are n such rows, the sum simplifies as follows:

$$\bar{\mathbf{Y}}^\top\bar{\mathbf{Y}} = n\bar{\mathbf{y}}^\top\bar{\mathbf{y}}.$$

This scaling by n occurs because the same mean vector is repeated for all observations. Substituting this result into the equation gives:

$$\mathbf{Y}^\top\mathbf{Y} = (\mathbf{Y} - \bar{\mathbf{Y}})^\top(\mathbf{Y} - \bar{\mathbf{Y}}) + n\bar{\mathbf{Y}}^\top\bar{\mathbf{Y}}.^{23}$$

This equation shows that $\mathbf{Y}^\top\mathbf{Y}$ consists of the variance of the mean-centred data plus the contribution of the mean structure. In its final form, this is:

$$\mathbf{Y}^\top\mathbf{Y} = (n-1)\boldsymbol{\Sigma}_{\mathbf{Y}} + n\bar{\mathbf{Y}}^\top\bar{\mathbf{Y}},$$

where $\bar{\mathbf{Y}}^\top\bar{\mathbf{Y}}$ captures the mean effects of $\mathbf{Y}$.

Singular Value Decomposition

Given the matrix $\hat{\mathbf{Y}}^*_{\mathbf{R}}$:

$$\hat{\mathbf{Y}}_{\mathbf{R}}^{*} = \begin{bmatrix} -5.67 & 2.96 & 4.19 & -14.29 & 12.81 & 0.74 & -0.62 \\ -1.82 & -0.91 & 2.28 & -2.73 & 3.19 & -0.23 & 0.55 \end{bmatrix}^{\top}.$$

We aim to compute its Singular Value Decomposition: $\hat{\mathbf{Y}}_{\mathbf{R}}^{*} = \mathbf{UDV}^{\top}$. First, compute the Gram matrix:

$$\hat{\mathbf{Y}}_{R}^{*\top}\hat{\mathbf{Y}}_{R}^{*} = \begin{bmatrix} 427.7629 & 96.57640 \\ 96.57640 & 27.33693 \end{bmatrix}.$$

This symmetric matrix is used to compute the eigenvalues and eigenvectors, which in turn provide the singular values. These singular values are the square roots of the eigenvalues of $\hat{\mathbf{Y}}_{R}^{*\top}\hat{\mathbf{Y}}_{R}^{*}$. The eigenvalues of $\hat{\mathbf{Y}}_{R}^{*\top}\hat{\mathbf{Y}}_{R}^{*}$ are: $\lambda_1 = 449.838519$ and $\lambda_2 = 5.261272$.

Hence:

$$\mathbf{D} = \begin{bmatrix} \sqrt{449.838519} & 0 \\ 0 & \sqrt{5.261272} \end{bmatrix} = \begin{bmatrix} 21.209397 & 0 \\ 0 & 2.293746 \end{bmatrix}.$$

The eigenvectors of $\hat{\mathbf{Y}}_{R}^{*\top}\hat{\mathbf{Y}}_{R}^{*}$ are the matrix $\mathbf{V}$:

$$\mathbf{V} = \begin{bmatrix} 0.9748562 & -0.2228349 \\ 0.2228349 & 0.9748562 \end{bmatrix}.$$

Now, we compute $\mathbf{U}$ using the formula: $\mathbf{U} = \hat{\mathbf{Y}}_{\mathbf{R}}^{*}\mathbf{VS}^{-\mathbf{1}}$. Since $\mathbf{V}$ is an orthonormal matrix, we have $\mathbf{V}^{\top} = \mathbf{V}^{-1}$, leading to (2dp):

$$\mathbf{U} = \begin{bmatrix} -0.28 & 0.13 & 0.22 & -0.69 & 0.62 & 0.03 & -0.02 \\ -0.22 & -0.68 & 0.56 & 0.23 & 0.11 & -0.17 & 0.29 \end{bmatrix}^{\top}$$

A.2.3 Cholesky-Gaussian XGBoost

Cholesky-Decomposition Verification

We verify by computing:

$$\mathbf{LL}^{\top} = \begin{bmatrix} 2 & 0 \\ 1 & \sqrt{2} \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 0 & \sqrt{2} \end{bmatrix}$$

$$= \begin{bmatrix} (2\times 2)+(0\times 0) & (2\times 1)+(0\times\sqrt{2}) \\ (1\times 2)+(\sqrt{2}\times 0) & (1\times 1)+(\sqrt{2}\times\sqrt{2}) \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix} = \mathbf{A}$$

As desired.

Weighted Quantile Sketches

In the approximate split-finding algorithm, as already stated, candidate split points are chosen based on percentiles of the feature values. The rank function $r_k(z)$ calculates the proportion of instances where feature k is less than z, weighted by second-order gradient statistics h.[39] The algorithm aims

to find split points $s_{k1}, s_{k2}, \ldots, s_{kl}$ such that the difference between the ranks of consecutive points is small, controlled by ϵ, which is an approximation factor.[39] This process ensures that the split points are well-distributed across the range of the feature values.

The Hessian, h_i, is used as a weight for each instance or row in the dataset, meaning that data points with larger Hessians carry more influence in determining the split. This weighting reflects the importance of the instance in minimising loss. Higher-weighted instances show where the model struggles, guiding the algorithm to focus on these critical areas during split finding.

Traditional quantile sketch algorithms work for datasets where all instances have equal weights.[39] However, no existing algorithm efficiently handles weighted data. This creates challenges in approximating split points for datasets with imbalanced or sparse data distributions. Randomised sorting or heuristic approaches have been used to address this issue, but these methods lack theoretical guarantees.[39] As a result, they may fail to identify the optimal split points, leading to suboptimal model performance. To overcome this limitation, XGBoost introduces a novel weighted quantile sketch algorithm that can handle weighted data with provable theoretical guarantees.[39] This algorithm ensures accurate split findings by accounting for the distribution of weights across the dataset.

A.2.4 Multi-Output Gaussian Process Neural Network

Example Multi-Output Neural Network Visualisation

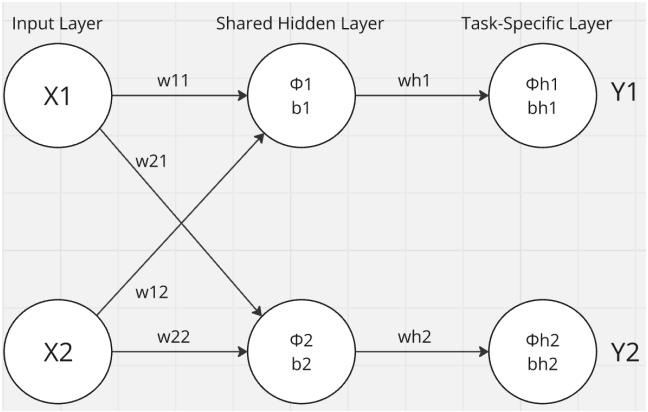

Figure A.4: Neural Network Example Diagram