A Robust Optimisation Approach for Stable Linear Regression

Veera Wilkki

School of Science

Bachelor's thesis
Espoo 9.10.2023

Supervisor

Prof. Fabricio Oliveira

Advisor

MSc Paula Weller

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the bachelor's thesis

Author Veera Wilkki

Title A Robust Optimisation Approach for Stable Linear Regression

Degree programme Bachelor's Programme in Science and Technology

Major Mathematics and Systems Sciences **Code of major** SCI3029

Teacher in charge Prof. Fabricio Oliveira

Advisor MSc Paula Weller

Date 9.10.2023 **Number of pages** 28+1 **Language** English

Abstract

Linear regression is a method that consists of predicting the value of a dependent variable using one or more independent variables by fitting a linear equation to the observed data. It has various applications such as making predictions, analysing trends or assessing risks.

In the context of machine learning, linear regression models are typically trained by splitting the observed data randomly into training, validation and test sets. Linear regression models created using the same observed data can differ from each other significantly because of the fact that the training/validation split is performed randomly. Therefore, this method may create models that produce inaccurate results.

In this thesis, we examine an alternative approach for training linear regression models. This approach is more resistant to errors and involves a robust optimisation problem, which is why it is referred to as the robust approach to training. Instead of choosing training and validation sets randomly, this method finds those data points that are the largest outliers, i.e., the most difficult to fit to a linear function, and uses these points to train the model.

We examine the differences between this robust approach and the classical randomised approach in terms of predictive ability (mean squared error) as well as runtime. The datasets used for the experiments describe features of two wines: a red wine and a white wine. Linear regression is used to predict the quality score of a wine, which is an integer between 0 and 10.

The results show that using the robust approach leads to models that perform better in terms of predictive ability than when using the randomised approach. However, the differences between the predictive abilities of the two approaches are quite small. In addition, the runtimes associated with the robust approach are significantly smaller than those of the randomised approach with cross validation which is often used to improve the accuracy of a model. These results demonstrate that the robust approach could potentially become a widely used method for training linear regression models due to the benefits it offers in contrast to the classical randomised approach.

Keywords linear regression, robust optimisation, mathematical optimisation, machine learning, data splitting

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Tekniikan kandidaatintyön tiivistelmä

Tekijä Veera Wilkki

Työn nimi A Robust Optimisation Approach for Stable Linear Regression

Koulutusohjelma Bachelor's Programme in Science and Technology

Pääaine Mathematics and Systems Sciences **Pääaineen koodi** SCI3029

Vastuuopettaja Prof. Fabricio Oliveira

Työn ohjaaja MSc Paula Weller

Päivämäärä 9.10.2023 **Sivumäärä** 28+1 **Kieli** Englanti

Tiivistelmä

Lineaarinen regressio on menetelmä, jossa jonkin vastemuuttujan arvoa ennustetaan yhden tai lukuisan selittävän muuttujan perusteella sovittamalla kerättyyn dataan suora. Kyseistä menetelmää voidaan soveltaa muun muassa ennusteiden tekemiseen, trendien analysoimiseen tai riskien arvioimiseen.

Koneoppimisessa lineaarinen malli koulutetaan jakamalla kerätty data satunnaisesti kolmeen eri osaan: koulutusdataan, validaatiodataan ja testausdataan. Saman datan perusteella koulutetut lineaarimallit voivat poiketa toisistaan merkittävästi kerätyn datan satunnaisen osituksen takia. Tämä voi johtaa malleihin, jotka eivät tuota täsmällisiä tuloksia.

Tässä kandidaatintyössä tutkitaan vaihtoehtoista lähestymistapaa lineaaristen mallien kouluttamiseen. Tämä tapa on vähemmän altis virheille ja hyödyntää robustia optimointia, minkä takia sitä kutsutaan robustiksi lähestymistavaksi. Sen sijaan, että koulutusdata ja validaatiodata määritettäisiin satunnaisesti, robustissa menetelmässä koulutukseen valitaan datasta eniten poikkeavat havainnot eli toisin sanoen ne datapisteet, joiden sovittaminen suoralle olisi vaikeinta.

Työssä tutkitaan eroja robustin ja klassisen lähestymistavan välillä ennusteiden täsmällisyyden (keskineliövirheen kautta) sekä koodin ajoajan avulla. Kokeissa käytetään kahta tietoaineistoa, jotka sisältävät tietoa liittyen viinien ominaisuuksiin. Ensimmäinen kuvaa punaviinejä ja toinen hieman suurempi tietoaineisto kuvaa puolestaan valkoviinejä. Lineaarista mallia käytetään ennustamaan viinin laatua kuvaavaa arviointitulosta, joka on kokonaisluku välillä 0–10.

Tulokset näyttävät, että robustia menetelmää käyttäen luodut mallit tuottavat parempia ennusteita kuin klassista menetelmää käyttäen luodut mallit. Erot ennusteiden täsmällisyydessä näiden kahden menetelmän välillä ovat kuitenkin suhteellisen pieniä. Tulokset näyttävät myös, että koodin ajoajat ovat huomattavasti lyhyempiä, kun lineaaristen mallien luomiseen käytetään robustia menetelmää klassisen ristiinvalidointia (engl. cross validation) hyödyntävän menetelmän sijaan. Näiden tulosten perusteella voidaan todeta, että robustilla menetelmällä on potentiaalia muuttua laajalti käytetyksi menetelmäksi lineaaristen mallien kouluttamiseen.

Avainsanat lineaarinen regressio, robusti optimointi, matemaattinen optimointi, koneoppiminen, datan ositus

Contents

1 Introduction

Linear regression is a method used to model the linear relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data (James et al. (2013)). Linear regression has various applications such as predictive analysis, short-term forecasting as well as trend analysis. Due to this versatility it is an essential tool used in a number of different fields such as finance, healthcare and marketing.

In machine learning, linear regression models are typically trained by initially choosing a random subset of the data as the test set and then randomly splitting the rest of the data into two more parts: a training set and a validation set (Zhou (2021)). The training set is used to create the linear regression model, i.e., solve an optimisation problem, and the validation set is used to validate results by calculating a performance metric, such as the mean squared error (MSE), for the model. Often, training needs to be performed multiple times using the same training and validation sets. For example, performing regularised regression requires the tuning of a regularisation parameter, which means that training and validation have to be performed multiple times with different values of the regularisation parameter in order to find the most accurate model (Zhou (2021)). The final accuracy of the chosen model is tested using the testing set.

Although this approach is commonly used to train linear regression models and simple to understand, it has certain drawbacks. The most significant disadvantage is caused by the fact that the subsets of data used for testing, training and validation are chosen randomly. These random splits can lead to very different models even when using the same observed data for training (Bertsimas and Paskov (2020)). This brings the reliability of linear regression models trained with this typical randomised approach into question.

Bertsimas and Paskov (2020) suggest that the splitting of the data could be optimised. Instead of performing splits randomly, the training set could be chosen in an optimal way so that the created model would produce accurate results for a wider range of input variables. In this case, the splitting of the data would be integrated into the optimisation problem directly. This would create a linear regression model that is more resistant to outliers, which is why this approach is referred to as the robust approach to linear regression.

The aim of this thesis is to assess the computational performance of the robust approach to training linear regression models by developing an implementation in Julia. The created functions are used to validate the results presented in Bertsimas and Paskov (2020) by comparing the predictive abilities of the linear regression models created using the robust approach and the randomised approach. This is realised using two datasets containing real data on red and white wine. These datasets are used to create linear regression models that estimate the quality of a wine as a score from 0 to 10 given a set of input variables. In addition to predictive ability, runtimes are also examined in order to determine the practicality of using the robust approach for training linear regression models instead of the more widely adopted randomised approach.

2 Literature review

Linear regression is a widely used data analysis method with applications in various fields. Linear regression models can be utilised for a number of different purposes such as making predictions, examining trends or assessing risks. This method has been described in much detail in a great number of publications. Information on the theory behind linear regression as well as its use in the context of machine learning can be found in Bertsimas and Tsitsiklis (1997), James et al. (2013) and Zhou (2021).

Typically, the training of linear regression models is performed using a randomised approach. However, this approach leads to models that are highly sensitive to the randomly chosen training/validation split of the data used to create the models (Bertsimas and Paskov (2020)). Therefore, the created models are not guaranteed to be robust to all subsets of the observed data. K-fold cross validation, discussed in James et al. (2013) and Zhou (2021), can alleviate this problem but only to a certain degree.

The alternative method to training linear regression models examined in this thesis involves solving a robust optimisation problem. Robust optimisation aims to immunise problems caused by the sensitivity of optimisation problem solutions to parameter perturbations (Bertsimas et al. (2011)). As previously mentioned, in the context of linear regression, the solution to the optimisation problem being solved in training is highly affected by the choice of the training/validation split. Using robust optimisation allows us to compute solutions that are not affected by this choice, therefore rendering the solutions more reliable.

Various robust linear regression methods have been developed over the years in order to create models that are not overly sensitive to outliers. The ordinary least squares (OLS) estimator is most often used to create linear regression models. To create more robust models, the OLS-estimate is replaced by other estimators such as the M-estimate, LMS estimate, LTS estimate or the S estimate (Yu and Yao (2017)). This approach differs from the robust method discussed in this thesis where the robustness of the linear regression model is ensured by selecting the training set in an optimal way instead of using a robust estimator when creating the model.

The aim of using the robust optimisation method for training linear regression models is to create models that perform well for all subsets of the observed data. Duchi and Namkoong (2018) explore this problem and introduce a distributionally robust optimisation (DRO) framework that allows one to learn models that perform well despite possible perturbations to the data-generating distribution. However, the approach introduced in Duchi and Namkoong (2018) is parametric and less computationally efficient than the method examined in this thesis.

In the linear regression method discussed in this thesis, the robustness of the created model is ensured by choosing the training and validation sets optimally instead of randomly. Using optimisation instead of randomisation to assign groups has been previously discussed by Bertsimas et al. (2015). However, this paper does not examine this in the context of linear regression but focuses on the creation of experimental groups in controlled trials.

3 Methodology

3.1 Randomised linear regression

In linear regression, the relationship between a target variable and one or more predictor variables is modeled by fitting a line to the observed data. This linear regression model in the case of multiple predictor variables can be expressed in the following way

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon, \tag{1}$$

where Y is the target variable, $X_1, ..., X_p$ are the predictor variables, β_0 is the intercept, $\beta_1, ..., \beta_p$ are the model coefficients and ϵ is a zero-mean random error term (James et al. (2013)).

The values of the intercept as well as the model coefficients can be estimated by minimising the ℓ_1 loss, i.e., the absolute difference between the true value of the target variable and the value predicted by the model (Pollard (1991)). The minimisation of ℓ_1 loss can be expressed as follows

$$\min_{\beta} \sum_{i=1}^{n} |y_i - x_i^T \beta|, \tag{2}$$

where n is the number of data points in our observed data, $\beta \in \mathbb{R}^{(p+1)\times 1}$ contains the intercept as well as the model coefficients, $y_i \in \mathbb{R}$ is the value of the target variable for the i^{th} data point of our observed data and $x_i \in \mathbb{R}^{(p+1)\times 1}$ contains the i^{th} values of the predictor variables (where the first value corresponding to the intercept is always 1, i.e., $(x_i)_1 = 1$).

Estimating the vector β using (2) can lead to over-fitting to the observed data. This would lead to models that do not perform well for unseen values of the predictor variables. A variety of regularisation methods are often used to avoid this. When applying regularisation, a penalty term is added to the optimisation problem and Problem (2) becomes

$$\min_{\beta} \sum_{i=1}^{n} |y_i - x_i^T \beta| + \lambda \sum_{i=1}^{p} \Gamma(\beta_i), \tag{3}$$

where λ is a constant regularisation parameter and $\Gamma(\cdot)$ represents the used regularisation function.

The most commonly used regularisation methods are Lasso (Tibshirani (1996)), Ridge (Hoerl and Kennard (1970)) and Elastic Net (Zou and Hastie (2005)). The following Problems (4), (5) and (6) show Problem (3) when using each of these regularisation methods.

$$\text{Lasso: } \min_{\beta} \sum_{i=1}^{n} |y_i - x_i^T \beta| + \lambda \sum_{i=1}^{p} |\beta_i|, \tag{4}$$

$$\text{Ridge: } \min_{\beta} \sum_{i=1}^{n} |y_i - x_i^T \beta| + \lambda \sum_{i=1}^{p} \beta_i^2, \tag{5}$$

$$\text{Elastic Net: } \min_{\beta} \sum_{i=1}^{n} |y_i - x_i^T \beta| + \lambda_1 \sum_{i=1}^{p} |\beta_i| + \lambda_2 \sum_{i=1}^{p} \beta_i^2. \tag{6}$$

When estimating the values of β using one of the problems (2), (4), (5) and (6), not all data is used. When training linear regression models in machine learning applications, the data is split into three parts. First, a test set is randomly chosen from the data after which the remaining data is split into a training set and a validation set (Zhou (2021)). The training set contains the data points that will be used to solve the previously described optimisation problems and the validation set contains those points that will be used to validate the created linear regression model. Due to the need to tune the regularisation parameter, the training and validation may need to be performed numerous times. The final accuracy of the model is determined using the test set.

The validation set as well as the test set can be used to validate the model created using the training set in various ways. One of the most common methods for determining how close the values predicted by the model are to the real values of the target variable is calculating the mean squared error (MSE) (James et al. (2013)). The MSE is given by

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2. \tag{7}$$

Typically, the training/validation split in linear regression is determined randomly. Due to this, models created using the same observed data can vary significantly for different training/validation splits. In order to alleviate this issue, a method called k-fold cross validation is often used (James et al. (2013)). In k-fold cross validation, the data is split into k (approximately) equal parts out of which one part is used for validation and the remainder become the training set. After this, a different part is chosen to be the validation set and the remainder are used for training. This is repeated k times so that each part is used for validation once. The model that produced the smallest MSE value calculated using the validation set is then chosen.

K-fold cross validation allows all data to be used for both training and validation so that the chances of finding the "best" model is maximised. However, this method is more complicated and time consuming than simply splitting the data into a training set and a validation set once.

3.2 Robust optimisation model

Bertsimas and Paskov (2020) propose a robust approach to training linear regression models. Instead of splitting data randomly, this step is integrated into the optimisation problem directly.

Bertsimas and Paskov (2020) suggest that training should be performed using the "hardest training set" in order to obtain accurate results even for data points that could be considered outliers. This means that the training set should include those data points that would yield the "worst" results for the optimisation problem that we wish to solve. In other words, the largest outliers of the original dataset should

be the data points used for training, as the difference between the real value and the predicted value would be the largest for these points. Therefore, a model created using such a training set is by default very resistant to outliers/errors, i.e., robust.

According to Bertsimas and Paskov (2020), these robust models perform better in terms of predictive ability and are more stable, i.e., the predictions as well as the model coefficients have smaller standard deviations, than when using the regular regression approach. In addition to this, the robust approach would allow for accurate feature selection, meaning that it would accurately predict which of the independent variables should be taken into account when predicting the value of the dependent variable. Furthermore, using the robust approach also identifies the "hardest subset" of the data, i.e., the data points that are the most difficult to fit to a linear function. Knowing the largest outliers of a dataset can be quite valuable in certain situations.

In order to perform this robust approach to training linear regression models, we need to solve the following optimisation problem presented by Bertsimas and Paskov (2020):

$$\min_{\beta} \max_{z \in Z} \sum_{i=1}^{n} z_i |y_i - x_i^T \beta| + \lambda \sum_{i=1}^{p} \Gamma(\beta_i) \quad \text{with} \quad Z = \left\{ z : \sum_{i=1}^{n} z_i = k, z_i \in \{0,1\} \right\}. \tag{8}$$

Here, z_i is an indicator variable, indicating which point (x_i, y_i) belongs to the training set and which to the validation set. When $z_i = 1$, the point (x_i, y_i) is used for training, whereas when $z_i = 0$, the point (x_i, y_i) is used for validation. The value of k represents the desired ratio between the sizes of the training and validation sets. For example, setting $k = 0.8n$ will lead to an 80/20 training/validation split.

Problem (8) is an integer optimisation problem. This makes it much more difficult to solve than a continuous linear optimisation problem. However, Problem (8) can be transformed into a linear problem by optimising over the convex hull of Z. This can be done because the inner maximisation problem is linear in z. The convex hull of Z refers to the smallest convex set that contains all points in Z (Preparata and Shamos (1985)). Figure 1 illustrates the idea of a convex hull.

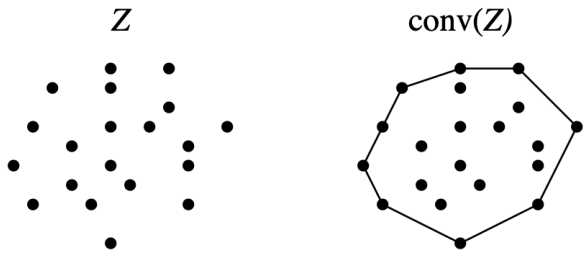

Figure 1: Illustration of the convex hull of Z

We can now write Problem (8) in the following equivalent form:

$$\min_{\beta} \max_{z \in \text{conv}(Z)} \sum_{i=1}^{n} z_i |y_i - x_i^T \beta| + \lambda \sum_{i=1}^{p} \Gamma(\beta_i) \quad \text{with conv}(Z) = \left\{ z : \sum_{i=1}^{n} z_i = k, 0 \leq z_i \leq 1 \right\}, \tag{9}$$

Having a maximisation inside of a minimisation problem makes Problem (9) difficult to solve. However, this problem can be equivalently recast as a linear optimisation dual of the inner maximisation problem. According to the principle of strong duality, a linear optimisation dual problem has the same optimal objective as the primal problem it is obtained from (Bertsimas and Tsitsiklis (1997)). On occasion, the problem reformulation using the dual is easier to solve than the original problem. This happens to be the case for the inner maximisation in Problem (9), i.e., the following problem

$$\max_{z_i} \sum_{i=1}^{n} z_i |y_i - x_i^T \beta| \quad \text{subject to} \quad \sum_{i=1}^{n} z_i = k, \quad 0 \leq z_i \leq 1. \tag{10}$$

Indeed, this approach allows us to transform Problem (9) from a 2-level "min max" problem into a 1-level "min" problem by replacing the inner maximisation with its dual.

In general, the primal optimisation problem

$$\begin{aligned} \max_{x} \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0, \end{aligned} \tag{11}$$

relates to the following dual problem (Bertsimas and Tsitsiklis (1997)):

$$\begin{aligned} \min_{y} \quad & b^T y \\ \text{s.t.} \quad & A^T y \geq c \\ & y \geq 0. \end{aligned} \tag{12}$$

In order to represent Problem (10) using the form of Problem (11), we choose

$$c = \begin{pmatrix} |y_1 - x_1^T \beta| \\ |y_2 - x_2^T \beta| \\ |y_3 - x_3^T \beta| \\ |y_4 - x_4^T \beta| \\ \vdots \\ |y_n - x_n^T \beta| \end{pmatrix} \in \mathbb{R}^{n \times 1}, A = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{n \times n}, b = \begin{pmatrix} k \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^{n \times 1}$$

Now we can formulate the dual of Problem (10) by using the formulation of Problem (12) with dual variables θ and u_i so that

$$y = \begin{pmatrix} \theta \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_n \end{pmatrix} \in \mathbb{R}^{n \times 1}$$

We obtain

$$\begin{aligned} \min_{\theta,u_i} \quad & k\theta + \sum_{i=1}^{n} u_i \\ \text{s.t.} \quad & \theta + u_i \geq y_i - x_i^T\beta, \\ & \theta + u_i \geq -(y_i - x_i^T\beta), \\ & u_i \geq 0 \end{aligned} \tag{13}$$

By replacing the inner maximisation in Problem (9) with Problem (13) we find that the final optimisation problem to solve is as follows:

$$\begin{aligned} \min_{\beta,\theta,u_i} \quad & k\theta + \sum_{i=1}^{n} u_i + \lambda \sum_{i=1}^{p} \Gamma(\beta_i) \\ \text{s.t.} \quad & \theta + u_i \geq y_i - x_i^T\beta, \\ & \theta + u_i \geq -(y_i - x_i^T\beta), \\ & u_i \geq 0 \end{aligned} \tag{14}$$

3.3 Implementation

The training and testing of linear regression models is implemented in the form of three functions that take in data along with some other necessary arguments. These arguments contain various information necessary for training, such as the desired regularisation method, training/validation split, information on which columns should be ignored (i.e., not the response variable or predictor variables), which column contains the response variable, how much of the data should be held out for testing (a percentage), the desired amount of iterations as well as the type of cross validation to be performed.

The first function (`train`) trains a linear regression model by solving the previously described robust optimisation problem (14). The second function (`random_train`) does the same but instead of solving Problem (14), it solves the optimisation problem used in the classic randomised regression approach, i.e., Problem (3), when given the training and validation sets as arguments. Both functions return the vector β as well as the mean squared error (MSE) of the created linear regression model.

The return values of the first two functions are utilised in the final function (`optimise`). This function is capable of performing linear regression using either the robust approach or the randomised approach based on the preference of the user, i.e., the value of one of its arguments. For each iteration $i = 1...I$ (the amount of iterations I is defined by the user and given to the function in the form of an argument), the function begins by randomly choosing a user-determined percentage of the data to be the held out test set. Often this percentage is relatively small (5-20%) as it is more essential to use data for training and validation.

If the function is performing robust linear regression (see Algorithm 1), it continues by tuning the regularisation parameter λ. This is done in the form of a for-loop that uses the `train`-function for values $\lambda = 10^k, k \in \{-2, -1, 0, 1, 2\}$, and chooses the combination of λ and β values that created the regression model with the smallest MSE on the validation set. Finally, a new MSE value is calculated using the held out

test set and this value is added to a variable that keeps track of the sum of MSEs (one for each iteration). This sum is used to calculate an average MSE value once all iterations have been performed.

Algorithm 1 Main for-loop in `optimise` for robust approach

for $i = 1...I$ **do**
    Shuffle data.
    Isolate test set.
    Initialise variables `best_mse`, `best_beta` and `best_lambda`.
    if regularisation is not Elastic Net **then**
        for $\lambda = [10^k$ for k in -2:2] **do**
            Train model with `train`-function.
            if new MSE is smaller than `best_mse` **then**
                Replace `best_mse`, `best_beta` and `best_lambda` with new values.
            end if
        end for
    else
        for $\lambda_1 = [10^k$ for k in -2:2] **do**
            for $\lambda_2 = [10^k$ for k in -2:2] **do**
                Train model with `train`-function.
                if new MSE is smaller than `best_mse` **then**
                    Replace `best_mse`, `best_beta` and `best_lambda` with new values.
                end if
            end for
        end for
    end if
    Calculate MSE using the held out testing set and value `best_beta`.
end for

When the function is performing the regular randomised approach to linear regression (see Algorithm 2), the split of the data into non-test data and test data is followed by the tuning of the regularisation parameter λ similarly as before. If the user does *not* want cross validation to be performed, the steps that follow are similar to the robust approach. The only difference is the use of the `random_train`-function intead of the `train`-function inside the previously described for-loop used for the tuning of the regularisation parameter λ.

Algorithm 2 Main for-loop in `optimise` for randomised approach

for $i = 1...I$ **do**
    Shuffle data.
    Isolate test set.
    Choose training and validation sets randomly.
    Initialise variables `best_mse`, `best_beta` and `best_lambda`.
    if no regularisation is used **then**
        Train model with `random_train`-function.
        Replace `best_beta` and `best_mse` values with the values returned by the `random_train`-function.
    else if regularisation is not Elastic Net **then**
        for $\lambda = [10^k$ for k in -2:2] **do**
            if no cross validation is used **then**
                Train model with `random_train`-function.
                if new MSE is smaller than `best_mse` **then**
                    Replace `best_mse`, `best_beta` and `best_lambda` with new values.
                end if
            else
                k-fold cross validation. Pseudocode in Algorithm 3.
            end if
        end for
    else
        for $\lambda_1 = [10^k$ for k in -2:2] **do**
            for $\lambda_2 = [10^k$ for k in -2:2] **do**
                if no cross validation is used **then**
                    Train model with `random_train`-function.
                    if new MSE is smaller than `best_mse` **then**
                        Replace `best_mse`, `best_beta` and `best_lambda` with new values.
                    end if
                else
                    k-fold cross validation. Pseudocode in Algorithm 3.
                end if
            end for
        end for
    end if
    Calculate MSE using the held out testing set and value `best_beta`.
end for

However, the two approaches differ more significantly when k-fold cross validation is performed. Once again, the function uses a for-loop for tuning ($\lambda = 10^k, k \in \{-2, -1, 0, 1, 2\}$) but this time the for-loop utilises an inner loop as well, which is presented in Algorithm 3. This inner loop splits the data into k parts and uses one part for validation and the rest for training. This is repeated k times so that

each part is used as the validation set once. Using these two for-loops, the function is able to find the pair (λ, β) that gives the best MSE value calculated using the validation set. Finally, this β value is used to calculate the MSE value with the held out test set similarly as in the robust version. After performing all I iterations, the `optimise`-function returns an average MSE as well as a vector containing all calculated MSE values.

Algorithm 3 K-fold cross validation inside the `optimise`-function

```
Divide non-test data into k parts (k-fold cross validation).
for 1:k do
    Redefine training and validation sets: choose one part to be the validation
    set and use the rest for training.
    Perform training using random_train-function.
    if new MSE is smaller than best_mse then
        Replace best_mse, best_beta and best_lambda with new
        values.
    end if
end for
```

3.4 Experiments

The functions are tested using two datasets containing information about red and white variants of the Portuguese "Vinho Verde" wine (Cortez et al. (2009)). Each dataset contains 12 columns representing the following numerical information: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol and quality.

The red wine dataset contains 1599 data points whereas the white wine dataset contains 4898 data points. This allows us to examine the effects of differently sized datasets on the predictive ability of a linear regression model. In addition, it provides insight into its effects on the runtime of the code used in testing.

Linear regression is used in order to predict the quality score of a wine based on the 11 input variables: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates and alcohol. This quality score is an integer between 0 and 10. However, the functions create linear regression models that lead to quality scores that are not integers. This is because the objective functions in the optimisation problems solved in training are not constrained to be integers. Nonetheless, these real-valued scores provide insight on the possible values of the true quality score for the input variables used as the obtained values can simply be rounded.

The two datasets are each tested using the `optimise`-function with six different sets of input parameters for the function. The relevant input parameters are: regression approach, regularisation method, training/validation split, percentage of data held out for testing and the number of iterations. The tested combinations are:

1. Robust, regularisation: Lasso

2. Robust, regularisation: Ridge
3. Random with 5-fold cross validation, regularisation: Lasso
4. Random with 5-fold cross validation, regularisation: Ridge
5. Random with no k-fold cross validation, regularisation: Lasso
6. Random with no k-fold cross validation, regularisation: Ridge

The following parameters remain the same in each test: training/validation split, percentage of data held out for testing and number of iterations. An 80/20 training/validation split is used in the experiments. This is a split that is often used in training linear regression models. It works well when performing comparisons between the robust approach and the regular randomised approach, specifically utilising 5-fold cross validation. Indeed, in 5-fold cross validation, the data is split into five parts out of which one at a time is used for validation, i.e., $1/5 = 20\%$ is used for validation and 80% is used for training.

The percentage of data held out for testing during each iteration is 10%. It is important that most of the available data is utilised in the training process which is why this percentage is not much higher than 10%. However, in order to obtain meaningful results for the mean squared error (MSE) it is best that this percentage is also not much lower than 10%.

The `optimise`-function performs 500 iterations for each of the used combinations. A relatively high number of iterations is necessary due to the randomness of the test sets. Indeed, as previously mentioned, the test set is chosen from the data randomly at the beginning of each iteration which means that for some iterations this test set will contain more outliers than for others. As the different approaches are not tested on the same test splits, results are not entirely comparable which creates the need for a higher number of iterations to alleviate this issue. Performing more than 500 iterations might possibly yield superior results, but a higher number of iterations naturally also leads to longer runtimes. Therefore, growing the number of iterations beyond 500 is not practical as the long runtimes outweigh the possible benefits.

The functions are implemented using Julia (version 1.8.2) and more specifically the JuMP-package (version 1.6.0) to solve the optimisation problems. The functions use the HiGHS solver (HiG) for Lasso regression and the Ipopt solver (Ipo) for Ridge regression. The code is run on a MacBook Pro with a 2 GHz Quad-Core Intel Core i5 processor.

4 Results

The results of the experiments for both the predictive ability as well as the runtimes are represented graphically. Predictive abilites are examined using histograms whereas runtimes are compared using bar charts. We begin by presenting the results for the predictive ability using the red wine dataset after which the same is done for the white wine dataset. This is followed by the results of the runtimes for both datasets separately.

4.1 Predictive ability

4.1.1 Predictive ability for red wine

Figures 2 and 3 contain histograms of the MSE values calculated for the red wine dataset over 500 test splits using the `optimise`-function. Figure 2 shows values acquired using Lasso as the regularisation method whereas Figure 3 shows those acquired using Ridge regularisation. The graphs are overlapping histograms where the blue bars represent the results of the robust approach and the red bars represent the results obtained using the random approach with 5-fold cross validation. The graphs also include blue and red vertical lines designating the average MSE values for each approach. These average values can also be found in Table 1.

The graphs include information for parameter combinations 1-4 introduced in Section 3.4. Combinations 5 and 6 representing the random approach to linear regression with no cross validation have not been included because the use of k-fold cross validation is more common when training linear regression models.

	Lasso	**Ridge**
Robust	0.429	0.427
Random	0.439	0.431

Table 1: Average MSE values for red wine

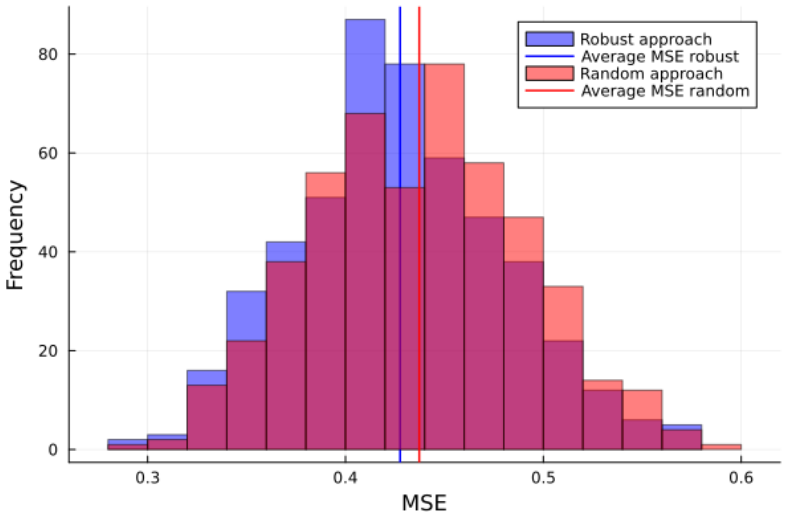

Figure 2: Histogram of MSE values for 500 test splits calculated for the red wine dataset using Lasso regularisation

In Figure 2, the MSE values calculated over the 500 test splits are approximately normally distributed for both the robust approach as well as the randomised approach. However, the distribution for the robust approach is moderately positively skewed meaning that the right tail of the distribution is more pronounced than the left tail.

The approximately normal form of the MSE distributions can be explained by the central limit theorem (CLT), which states that the sum (or average) of a large number of independent and identically distributed random variables follows a normal distribution, regardless of the distribution of the individual variables (Le Cam (1986)). In linear regression, the error term ϵ is assumed to be normally distributed. Because the MSE is a sum of these squared error terms, the MSE values must also be normally distributed. In addition to this, the MSE values are independent random variables since each value is calculated using a different randomly selected test set. Therefore, the MSE values are independent and identically distributed random variables, which means that the CLT applies to them when the amount of iterations is large enough.

In Figure 2, the tallest bin of the histogram for the robust approach contains values between 0.40 and 0.42. These values are smaller than those contained in the tallest bin of the histogram for the randomised approach, which are between 0.44 and 0.46. In addition to this, the average MSE value calculated for the robust approach is approximately 0.429 whereas the average value for the randomised approach is approximately 0.439. Therefore, the predictions of the linear regression model trained using the robust approach with Lasso regularisation seem to be more accurate than those of the random approach.

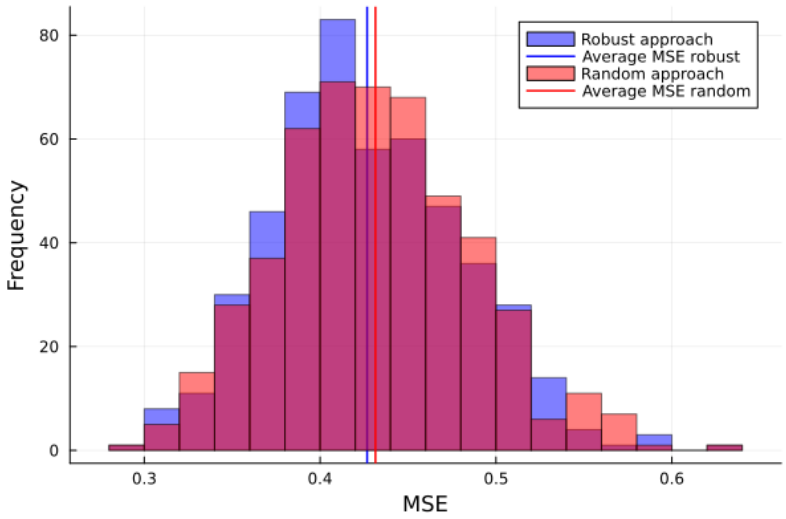

Figure 3: Histogram of MSE values for 500 test splits calculated for the red wine dataset using Ridge regularisation

Similarly to Figure 2, Figure 3 shows that the calculated MSE values seem to be roughly normally distributed as was predicted by the CLT. The robust approach shows some minor signs of positive skewness similarly to Lasso regularisation.

For the robust approach, the tallest bin of the histogram contains values between 0.40 and 0.42 similarly to the histogram for Lasso regularisation. For Ridge regularisation, the tallest bin for the randomised approach contains values between 0.40 and 0.42 as well. However, the tallest bin for the robust approach is significantly taller than the one for the randomised approach. This means that a larger amount of MSE

values were between 0.40 and 0.42 for the robust approach than for the randomised approach.

Based on the graphs, the difference between the averages of the robust approach and the randomised approach is more pronounced for Lasso regularisation than it is for Ridge regularisation. The average MSE for the robust approach is approximately 0.427 whereas the average MSE for the randomised approach is about 0.431 in the case of Ridge regularisation. The difference between these values is only 0.004 which is much smaller than the equivalent difference for Lasso regularisation which is 0.01. In addition to the difference being smaller, the average values themselves are smaller for Ridge regularisation than Lasso regularisation. These observations indicate that Ridge may be a more suitable regularisation method for this dataset.

4.1.2 Predictive ability for white wine

	Lasso	**Ridge**
Robust	0.567	0.576
Random	0.575	0.578

Table 2: Average MSE values for white wine

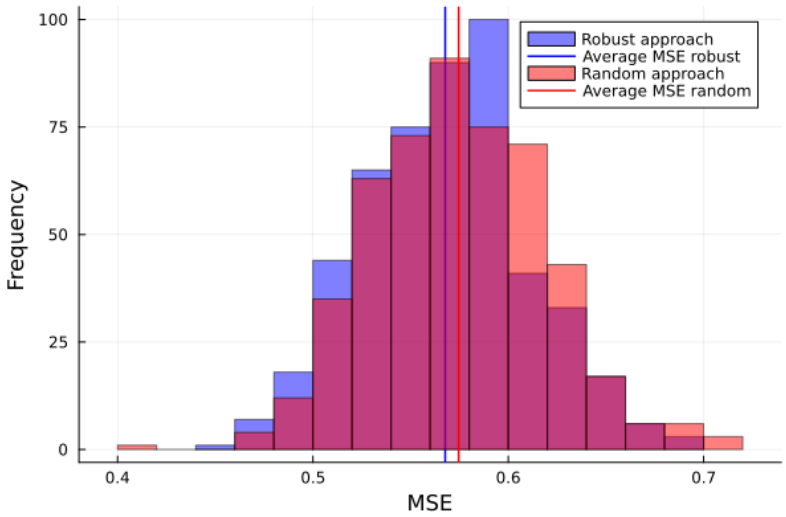

Figure 4: Histogram of MSE values for 500 test splits calculated for the white wine dataset using Lasso regularisation

Figure 4 shows the histograms for the MSE values of the white wine dataset for the robust approach as well as the randomised approach using Lasso as the regularisation method. The average MSE values can be found in Table 2. The MSE values for the randomised approach are normally distributed similarly as in the previous figures. However, the distribution for the robust approach appears to be moderately negatively skewed. This means that the tail on the left side is slightly longer and more pronounced than the tail on the right side.

In comparison to the MSE distributions of the red wine dataset, the distributions for the white wine dataset appear more compact. This indicates that the variances of the MSE values are slightly smaller for the white wine dataset than for the red wine dataset. This is most likely caused by the larger number of data points contained in the white wine dataset.

The tallest bin for the robust approach contains the values between 0.58 and 0.60. However, the average MSE value is smaller with a value of approximately 0.567. For the random approach, the tallest bin of the histogram contains the values between 0.56 and 0.58 and the average value of the MSE is approximately 0.575 which is 0.008 larger than the average MSE for the robust approach. Notably, the average MSE values for this dataset are slightly larger than the ones calculated for the red wine dataset. In addition to this, the difference between the average values for the two different regression approaches is smaller than the equivalent difference calculated for the red wine dataset when using Lasso regularisation.

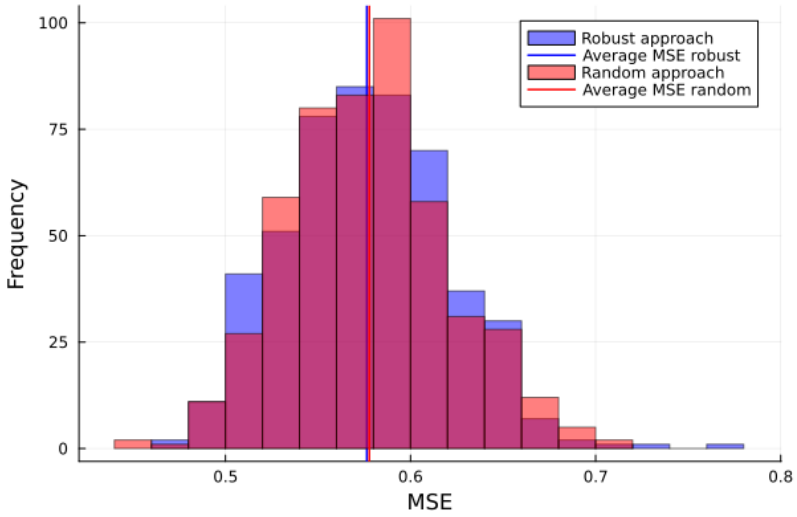

Figure 5: Histogram of MSE values for 500 test splits calculated for the white wine dataset using Ridge regularisation

Figure 5 contains the MSE histograms for the white wine dataset using Ridge regularisation. In this case, the distribution for the MSE values calculated using the random approach appears slightly negatively skewed whereas the distribution for the robust approach has a longer right tail. Once again, the distributions are more narrow for the white wine dataset than the red wine dataset.

The tallest bin for the robust approach contains the MSE values between 0.56 and 0.58 whereas the tallest bin for the random approach contains the MSE values between 0.58 and 0.60. The averages for the two regression approaches are very similar: the average MSE for the robust approach is approximately 0.576 and the average for the random approach is approximately 0.578. The difference between the two averages is half as large as the equivalent difference calculated for the red wine dataset.

4.1.3 Conclusions on predictive ability

In general, the histograms for the MSE values show that the differences in the predictive value for these datasets are not very significant, especially considering that the value being predicted represents an integer quality score. However, they still demonstrate that the robust approach consistently produces better linear regression models in terms of predictive ability than the typical random approach. Although this does not make a significant difference for the datasets in this experiment, for other datasets, especially where the predicted variable is real valued, even relatively small improvements of predictive ability could be significant.

Both datasets produced more accurate linear regression models when Ridge regularisation was used than when Lasso regularisation was used. This is probably caused by the individual properties of the datasets in combination with the characteristics of the regularisation method. For example, Lasso regularisation tends to make some of the model coefficients exactly zero, meaning that it is suitable for feature selection (James et al. (2013)). However, Ridge only allows model coefficients to become exactly zero in the very special case that the optimal solution for the original unregularised problem is exactly zero ($\beta = 0$). This means that in most cases all independent variables will have some kind of effect on the value of the dependent variable being predicted when using Ridge regularisation.

The characteristics of the datasets being used in regression have important effects on the performance of different regularisation methods. Characteristics of the data such as the correlation of the variables, the ratio between the number of features and the sample size as well as the presence of outliers all affect the applicability of different regularisation methods. Both Ridge and Lasso have their own advantages and disadvantages. For example, Ridge regularisation is more effective than Lasso when working with correlated variables or datasets with more features than observations (Zou and Hastie (2005), Friedman et al. (2010)). However, Ridge is not as robust to outliers as Lasso due to the fact that the quadratic penalty term in Ridge regularisation is more sensitive to outliers than the penalty associated with Lasso regularisation. The results indicate that Ridge is most likely the best regularisation method to use for this data when predicting the value of the quality score. This could for example indicate that some of the variables are correlated and that the datasets do not possess a significant number of outliers.

In addition to differences between Lasso and Ridge regularisation, the dataset being tested also had an effect on the amount of improvement using the robust approach brought in terms of predictive ability. Indeed, the white wine dataset produced MSE values that were very similar for both the robust approach and the randomised approach. This could indicate that larger sample sizes improve the effectiveness of the randomised approach meaning that benefits brought by the robust approach become less significant.

The significance of the obtained results is limited by several factors. The `optimise`-function chooses a different test set from the data randomly at the beginning of each iteration to calculate the MSE value. In addition to this, the function only uses one regression approach and regularisation method at a time to perform all

iterations, meaning that different approaches are never tested using the exact same test set. Due to this, the results may appear to be better for one method than the other although it might only be caused by one of them receiving a test set with more outliers than the other. However, the fact that the function performs such a large amount of iterations mitigates this problem.

During the experiments, the `optimise`-function was run numerous times and the results differed slightly each time. However, the clear majority of the results acquired using the robust approach were better than those of the randomised approach. Therefore, we can conclude that the robust approach generally produces smaller average MSE values than the random approach. However, the differences in performance between the Lasso and Ridge regularisation methods varied more during testing. Due to this, we cannot conclude with complete certainty that Lasso is clearly a less effective regularisation method to use for these datasets than Ridge. The same can be said for the differences observed between the two tested datasets.

4.2 Runtime

Figures 6 and 7 contain bar charts displaying the runtimes (in seconds) of the `optimise`-function for the red wine dataset and the white wine dataset, respectively. The six bars in each graph represent the six parameter combinations introduced in Section 3.4. Each graph contains three blue bars and three red bars. The blue bars represent the runtimes associated with Lasso regularisation while the red bars represent the runtimes for Ridge regularisation. One blue bar and one red bar is associated with each regression approach: random approach with no cross validation, random approach with 5-fold cross validation and the robust approach. The specific runtimes can be found in Tables 3 and 4.

4.2.1 Runtimes for red wine

	Lasso	**Ridge**
Random: no cross validation	351 s	756 s
Random: 5-fold cross validation	2011 s	4362 s
Robust	918 s	1495 s

Table 3: Runtimes calculated for the red wine dataset for parameter combinations 1-6

Figure 6 shows that for the red wine dataset the runtimes for the random approach with no cross validation are clearly the smallest. Running the code for this approach takes approximately 350 s (5 min 50 s) for Lasso regularisation and 760 s (12 min 40 s) for Ridge regularisation.

The runtimes associated with the robust approach are longer than for the no cross validation approach but clearly shorter than for the 5-fold cross validation approach. For the robust approach the runtime for Lasso regularisation is about 920 s (15 min 20 s) and 1490 s (24 min 50 s) for Ridge regularisation.

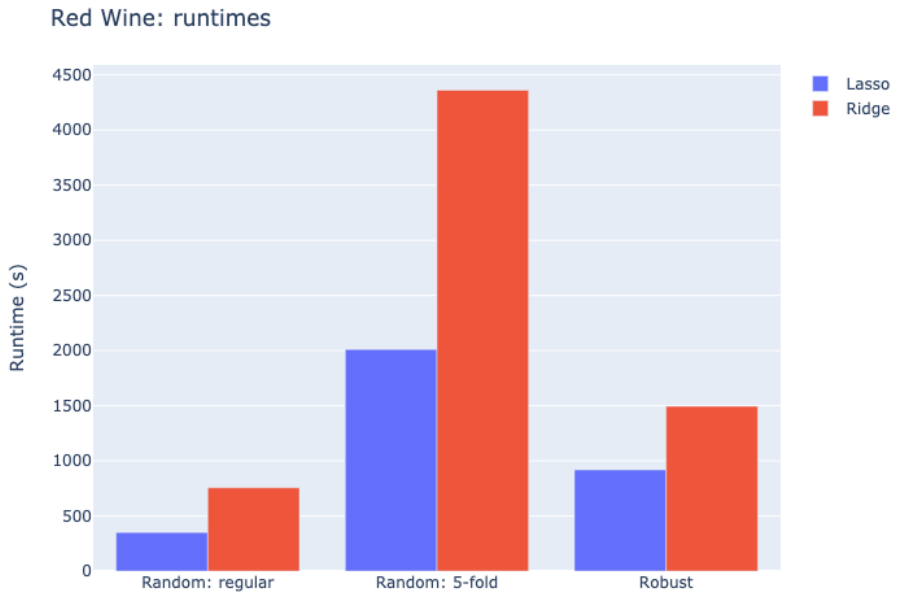

Figure 6: Runtimes calculated for the red wine dataset for parameter combinations 1-6

Runtimes become about twice as long as those for the robust approach when using the random approach with 5-fold cross validation. In this case the runtime associated with Lasso regularisation is approximately 2010 s (33 min 30 s) whereas the runtime for Ridge regularisation is approximately 4360 s (72 min 40 s = 1 h 12 min 40 s).

The results seem logical when taking into account the structure of the code used to perform the experiments. The robust approach using 5-fold cross validation takes the longest as in this case the for-loop used to tune the regularisation parameter λ also includes an inner loop for the cross validation. The random approach with no cross validation as well as the robust approach both only require one for-loop for the tuning of the regularisation parameter. However, the optimisation problem being solved when training the robust linear regression model (14) is more complex as it includes more variables than the optimisation problem for the regular random approach (3). This explains the difference in runtimes for these two approaches.

In addition to differences in runtime between the random and robust approaches, the runtimes also significantly differ depending on the used regularisation method. When using Lasso regularisation, the optimisation problems being solved for both the random approach as well as the robust approach are linear. However, in the case of Ridge regularisation, Problems (3) and (14) become convex quadratic optimisation problems.

Another essential difference between the Lasso and Ridge regularisation methods, specifically for the implementation being used in these experiments, is the used solvers. When using Lasso regularisation, the functions use the HiGHS solver, which is a high performance serial and parallel solver that can be used for solving large-scale

sparse linear programming problems (HiG). However, for Ridge regularisation, the optimisation problems are solved using the Ipopt solver, which is an interior point optimiser (Ipo). Therefore, the comparability of the runtimes is limited.

4.2.2 Runtimes for white wine

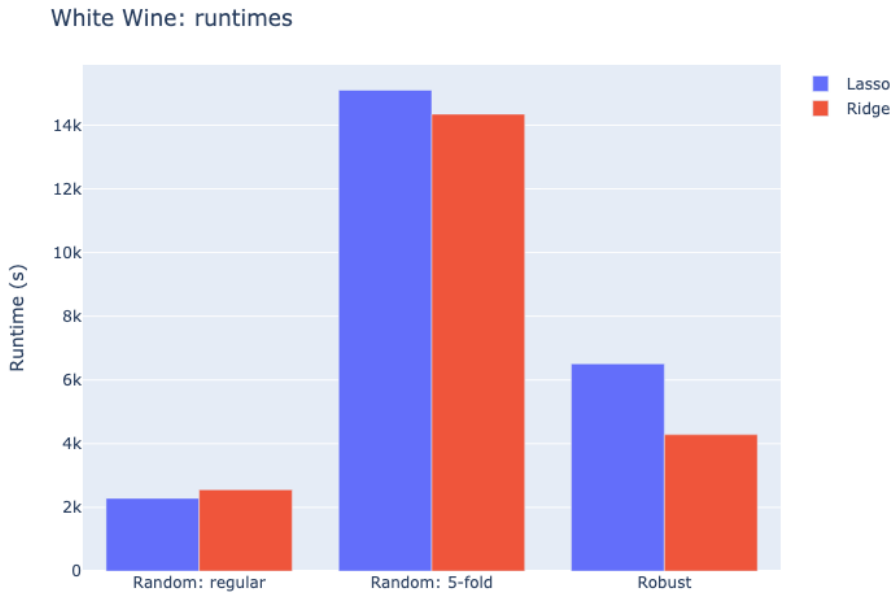

Figure 7: Runtimes calculated for the white wine dataset for parameter combinations 1-6

	Lasso	**Ridge**
Random: no cross validation	2286 s	2554 s
Random: 5-fold cross validation	15098 s	14346 s
Robust	6506 s	4284 s

Table 4: Runtimes calculated for the white wine dataset for parameter combinations 1-6

Figure 7 shows the runtimes for running the code for the white wine dataset, presenting some obvious differences compared to the runtimes for the red wine dataset. Firstly, the runtimes are much longer than for the red wine dataset. This was to be expected as having more data points will lead to longer runtimes. This is due to the fact that the number of variables (number of terms included in the (first) sum found in the optimisation problems (3) and (14)) as well as the number of constraints in (14) depend on the amount of data points. Secondly, for the red wine dataset the runtimes for Ridge regularisation are consistently much longer than for Lasso regularisation whereas here that does not hold.

Despite these differences, the white wine runtimes and red wine runtimes are consistent with respect to the relative order of runtimes for the three different regression approaches. Indeed, the randomised approach with no cross validation takes the least amount of time to run, whereas the randomised approach with 5-fold cross validation takes the longest.

The runtime for the randomised approach with no cross validation is approximately 2300 s (38 min 20 s) when using Lasso as the regularisation method and 2600 s (43 min 20 s) when using Ridge regularisation. Here, Ridge regularisation takes longer than Lasso regularisation as was the case for the red wine dataset. However, the difference in runtimes between the two regularisation methods is significantly smaller than for the red wine dataset.

For the randomised approach with 5-fold cross validation as well as the robust approach, Ridge regularisation has a smaller runtime than Lasso regularisation unlike to the red wine dataset. The runtime for the randomised approach with 5-fold cross validation using Lasso regularisation is approximately 15100 s (251 min 40 s = 4 h 11 min 40 s). When using Ridge as the regularisation method, the runtime is approximately 14300 s (238 min 20 s = 3 h 58 min 20 s). For the robust approach, the runtime is approximately 6500 s (108 min 20 s = 1 h 48 min 20 s) when using Lasso regularisation and approximately 4300 s (71 min 40 s = 1 h 11 min 40 s) when using Ridge regularisation.

4.2.3 Conclusions on runtimes

The results for the runtimes can be considered reliable. The code was run numerous times over the course of the experiment and the runtimes varied slightly during the iterations. However, the differences between the runtimes during different iterations were extremely small. In addition, the specific values of the runtimes are not the point of interest when performing comparisons between the different regression methods. It is more essential to examine the runtimes in relation to each other for the different methods.

The results for both datasets demonstrate that the robust approach to training takes longer to perform than the randomised approach with no cross validation. However, the robust approach takes significantly less time than the randomised approach using 5-fold cross validation. This is particularly relevant as using k-fold cross validation when training linear regression models is the prevailing approach due to the previously mentioned issues that using no cross validation can bring.

5 Conclusions

In this thesis, we examined a robust approach to training linear regression models and compared it to the classical randomised approach. The two methods were compared using two metrics: predictive ability and runtime. All experiments were conducted using functions created in Julia utilising the JuMP-package.

The datasets used in the experiments describe features of two different wines (red and white) and were acquired from the UCI Machine Learning Repository (Cortez

et al. (2009)). These datasets were used to predict the integer quality score of the wines using the two linear regression methods. The red wine dataset contains less data than the white wine dataset, which allowed us to make deductions on the effects of differently sized datasets to the created linear regression models.

The predictive ability of the created linear regression models was measured using the mean squared error (MSE). The MSE values calculated using the robust approach were compared to those calculated using the randomised approach with 5-fold cross validation. Both approaches were tested using two different regularisation methods: Lasso and Ridge.

Using the robust approach for training linear regression models produced smaller MSE values compared to the randomised approach for both datasets using both regularisation methods. The differences between the MSE values for the robust approach and the randomised approach were greater when using Lasso regularisation than Ridge regularisation. In addition to this, the MSE values of the robust and randomised approaches were more similar to each other when using the white wine dataset containing more data points than for the smaller red wine dataset. In general, the differences in MSE values were not very significant but still worth noting.

The runtimes were calculated for three different regression approaches: the robust approach, the randomised approach with 5-fold cross validation and the randomised approach with no cross validation. All approaches were tested using Lasso and Ridge regularisation methods. The results for the runtimes were compared using bar charts.

The runtime when performing training using the randomised approach with no cross validation was the smallest for both datasets. The randomised approach using 5-fold cross validation took significantly longer than when using the robust approach. The runtimes for the white wine dataset were longer than for the red wine dataset due to the larger number of data points. Differences in runtime could also be observed between the two regularisation methods. For the red wine dataset, Lasso regularisation produced smaller runtimes whereas for the white wine dataset Ridge was faster for both the randomised approach using 5-fold cross validation as well as the robust approach.

The results demonstrate that the robust approach to training linear regression models offers benefits compared to the classical randomised approach even when using k-fold cross validation. Although the improvements in predictive ability were not particularly significant for the datasets used in these experiments, it is still significant that the robust approach was able to consistently produce models with even slightly better predictive ability compared to the randomised approach. In addition to this, the runtimes using the robust approach proved to be significantly smaller than for the randomised approach with 5-fold cross validation, which is very promising as k-fold cross validation is a widely used method for training linear regression models. Although these experiments were rather limited and only used two datasets for testing, it is encouraging to see that the robust approach proved to be superior in every tested aspect. The results show that this robust approach has potential to become the default method for training linear regression models in the future.

This thesis focused on examining the effects on the predictive ability of the created

models as well as the runtimes when using a robust approach for training linear regression models. Although these are important aspects to take into consideration, the quality of linear models can be measured in numerous different ways such as by examining the standard deviation of the predictions as well as the model coefficients, or by calculating metrics like the R-squared value, which represents the proportion of variance in the dependent variable that is caused by the independent variables, or the Akaike Information Criterion (AIC), which also takes into account the complexity of the model (James et al. (2013)). Examining these factors might lead to different conclusions when it comes to the benefits of using the robust approach as opposed to the randomised approach.

Based on the obtained results, the robust approach shows promise when it comes to training linear regression models. However, the results obtained for linear regression cannot be generalised for all regression approaches. One may wonder whether a similar approach could be applied in order to train nonlinear regression models, which are used in situations where the dependent and independent variables do not possess a linear relationship. This is something that could be explored in future studies.

References

HiGHS - High-performance parallel linear optimization software. URL https://highs.dev/.

Ipopt: Documentation. URL https://coin-or.github.io/Ipopt/.

Dimitris Bertsimas and Ivan Paskov. Stable Regression: On the Power of Optimization over Randomization in Training Regression Problems. *Journal of Machine Learning Research*, 21:1–25, 2020.

Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*. Athena, Belmont, MA, vol. 6. edition, 1997. ISBN 1-886529-19-1.

Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011. ISSN 00361445. doi: 10.1137/080734510.

Dimitris Bertsimas, Mac Johnson, and Nathan Kallus. The power of optimization over randomization in designing experiments involving small samples. *Operations Research*, 63(4):868–876, 7 2015. doi: 10.1287/opre.2015.1361.

Paulo Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Wine Quality, 2009. URL https://archive.ics.uci.edu/dataset/186/wine+quality.

John C. Duchi and Hongseok Namkoong. Learning Models with Uniform Performance via Distributionally Robust Optimization. *Annals of Statistics*, 49(3):1378–1406, 10 2018. ISSN 21688966. doi: 10.1214/20-AOS2004.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of statistical software*, 33(1):1, 2010. ISSN 15487660. doi: 10.18637/jss.v033.i01.

Arthur E. Hoerl and Robert W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970. ISSN 15372723. doi: 10.1080/00401706.1970.10488634.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer, New York, vol. 112 edition, 2013.

Lucien Le Cam. The central limit theorem around 1935. *Statistical Science*, 1(1): 78–91, 2 1986.

David Pollard. Asymptotics for least absolute deviation regression estimators. *Econometric Theory*, 7(2):186–199, 6 1991.

Franco P. Preparata and Michael Ian Shamos. Convex hulls: Basic algorithms. In *Computational Geometry: An Introduction*, pages 95–149. 1985. ISBN 0387961313.

Robert Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1 1996. doi: 10.1111/J.2517-6161.1996.TB02080.X.

Chun Yu and Weixin Yao. Robust linear regression: A review and comparison. *Communications in Statistics: Simulation and Computation*, 46(8):6261–6282, 9 2017. ISSN 15324141. doi: 10.1080/03610918.2016.1202271.

Zhi-Hua Zhou. *Machine learning*. Springer, Gateway East, Singapore, 2021. ISBN 981-15-1967-6.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 67(2): 301–320, 2005. ISSN 13697412. doi: 10.1111/J.1467-9868.2005.00503.X.