



Comparison of Methods to Solve Inverse Matrix Problem in Regression

Namra Shakeel
Regn.# 321269

A thesis submitted in partial fulfillment of the requirements
for the degree of **Master of Science**
in
Mathematics

Supervised by: Dr. Tahir Mehmood

Department of Mathematics

School of Natural Sciences
National University of Sciences and Technology
H-12, Islamabad, Pakistan

Year 2021


National University of Sciences & Technology

MS THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by: NAMRA SHAKEEL, Regn No. 00000321269 Titled: "Comparison of Methods to Solve Inverse Matrix Problem in Regression" accepted in partial fulfillment of the requirements for the award of **MS** degree.

Examination Committee Members

1. Name: DR. MUJEEB UR REHMAN

Signature: 

2. Name: DR. RAZA ALI

Signature: 

External Examiner: DR. ISHFAQ AHMAD

Signature: 

Supervisor's Name: DR. TAHIR MEHMOOD

Signature: 



Head of Department

26/08/2021

Date

COUNTERSIGNED

Date: 26.8.2021



Dean/Principal

I dedicate this thesis to my beloved late parents.

Acknowledgments

Allah Almighty, the most beneficent and gracious, who created the whole universe, deserves all honor and glory. I am deeply grateful and indebted to Him for bestowing countless blessings upon me, including the courage and strength to complete my thesis effectively. Without a doubt, my sincerest appreciation goes to my supervisor, **Dr. Tahir Mehmood**, who is one of the best teachers I have ever had. I owe him a lot of gratitude for his support, advice, and especially his constant patience during this journey. May Allah bless him with an abundance of blessings. This research would not have been accomplished without his knowledge and experience. Furthermore, it is completely because of his efforts and appreciative responses to my questions that I have gained a complete grasp and respect of this field. I would like to pay my gratitude to my GEC members **Dr. Mujeeb Ur Rehman** and **Dr. Raza Ali** for their support and guidance in completing this thesis. Lastly, I want to thank the support of my siblings and friends in my studies.

Abstract

The inverse matrix problem in linear regression models is a basic issue for high dimensional data and the reason behind this issue is multicollinearity and identification problem. One of Artificial Intelligence's (AI) branches, machine learning emphasizes using data and algorithms to replicate the approach by which humans learn, to steadily increase accuracy. One of machine learning's categories is supervised learning which consists of both predictors and predicted values. The regression model is a supervised learning technique for dealing with continuous data sets. Some existing regression methods are LASSO, generalized inverse, and partial least squares (PLS) regression that is considered as a reference to evaluate the newly proposed methods. Newly proposed methods include 'Beta Cube', 'Compressed Beta Cube', 'Compressed LASSO' and 'Compressed Generalized inverse' regression. Two existing data sets 'NIR (Near-Infrared) Spectra of Biscuit Dough' and 'Raman Spectra Analysis of Contents of Polyunsaturated Fatty Acids (PUFA)' have been considered for comparing the performance of reference and proposed methods. To divide the data into training and testing sets, Monte Carlo Cross Validation has been used, and the root mean square error has been used to evaluate the performance estimation of all techniques. All models are tested through algorithms on the R language.

Contents

List of Figures	vii
1 Introduction	1
1.1 Singularity of a Matrix	2
1.1.1 Multicollinearity	2
1.1.2 Identification Problem	2
1.2 Machine learning	3
1.2.1 Definition	3
1.2.2 Theory	3
1.3 Supervised and Unsupervised Learning	4
1.3.1 Supervised Learning Models	5
1.4 Summary	5
2 Reference Regression Models	7
2.1 Introduction	7
2.1.1 Ordinary Least Square (OLS)	8
2.1.2 Assumptions of Linear Regression	10
2.1.3 Limitations of OLS	12
2.1.4 Compressed Regression	13
2.2 Least Absolute Shrinkage and Selection Operator	14
2.3 Generalized Inverse	17

2.3.1	Pseudo inverse	17
2.3.2	Left Inverse	18
2.3.3	Right Inverse	18
2.3.4	Solve-ability of Linear Equations	19
2.4	Partial Least Squares (PLS) Regression	20
3	Proposed Regression Models	25
3.1	Introduction	25
3.2	Beta Cube Regression	25
3.3	Compressed Beta Cube Regression	27
3.4	Compressed LASSO Regression	28
3.5	Compressed Generalized Inverse Regression	30
4	Model Building and Comparison	32
4.1	Cross Validation	32
4.1.1	Monte Carlo Cross Validation	33
4.1.2	Selection of Penalty Parameter	34
4.1.3	Performance Estimation	34
5	Applications and Results	37
5.1	NIR (Near-Infrared) Spectra of Biscuit Dough	39
5.2	Raman Spectra Analysis of Contents of Polyunsaturated Fatty Acids (PUFA)	46
6	Conclusions	51
	Bibliography	52
	Appendix	59

List of Figures

1.1	This figure represents complete flow of methods that has been used in this research	6
4.1	This figure represents complete flow of computational steps in OLS method	35
4.2	This figure represents complete flow of computational steps in PLS method	36
5.1	This figure represents the NIR spectra of Biscuit Dough	39
5.2	This represents RMSE of fat component of biscuit dough for each prediction method	40
5.3	This represents threshold of fat component of biscuit dough for each prediction method	41
5.4	This represents RMSE of flour component of biscuit dough for each prediction method	42
5.5	This represents threshold of flour component of biscuit dough for each prediction method	42
5.6	This represents RMSE of sucrose component of biscuit dough for each prediction method	43
5.7	This represents threshold of sucrose component of biscuit dough for each prediction method	44
5.8	This represents RMSE of water component of biscuit dough for each prediction method	45
5.9	This represents threshold of water component of biscuit dough for each prediction method	45

5.10	This figure represents the Raman spectroscopy of Fatty Acids	47
5.11	This represents RMSE of fatty acid as a percentage of total sample weight for each prediction method	48
5.12	This represents threshold of fatty acid as a percentage of total sample weight for each prediction method	48
5.13	This represents RMSE of fatty acid as a percentage of total fat content in this data set for each prediction method	49
5.14	This represents threshold of fatty acid as a percentage of total fat content in this data set for each prediction method	50

Chapter 1

Introduction

In Mathematics, linear equations are consisting of unknown variables x and y . Its solution can be found through different methods. In linear algebra, these equations can also be written in the form of a matrix. For example, let a linear equation

$$Y = X\beta \tag{1.1}$$

where Y is a column vector having dimension $n \times 1$, β vector consists dimension of $p \times 1$ while X is a matrix of dimension $n \times p$. Then solution of unknown coefficient β can be represented as

$$\beta = X^{-1}Y \text{ if } X^{-1} \neq 0 \tag{1.2}$$

where X^{-1} is an inverse matrix. But the problem comes when a square or rectangular matrix X becomes singular. So, there exists no solution for finding unknown parameter β . For example,

$$X = \begin{bmatrix} 3 & 3 & -1 \\ 3 & 3 & -1 \\ -1 & -1 & 6 \end{bmatrix} \tag{1.3}$$

In matrix (1.3), determinant of X becomes 0 and its rank is equal to 2. To overcome this problem, an American mathematician, Moore in 1935, and a British mathematician Penrose in 1955 tried to find an alternative solution of inverse matrix called the Generalized inverse or Moore-Penrose Pseudo inverse [1]. There is a detail discussion of generalized inverse given in Chapter 2.

Now, the question rises why inverse of a matrix becomes singular? What are the reasons for singularity in a matrix X ? What are the alternative solutions? How suitable solution can be determined? Firstly, we define the problem statement.

Problem Statement

How to solve the inverse of a matrix in regression analysis when its rank is less than the dimension of a matrix or matrix becomes singular?

1.1 Singularity of a Matrix

A $n \times n$ matrix is called singular matrix when its determinant becomes zero.

There are two main reasons for singularity in a matrix:

1. Multicollinearity
2. Identification Problem

1.1.1 Multicollinearity

When two or more than two independent variables are related to each other in a matrix in terms of rows or columns, then there exists multicollinearity. Multicollinearities build problems in finding out the relationship between dependent and independent variables in a linear or multivariate model. Due to multicollinearity, the coefficient value changes its sign as well as they become inaccurate and unreliable. This exaggerates variances of coefficients and gives wrong results in testing, evaluation, and prediction. Multicollinearity builds when the observer takes two or more independent variables in collecting data that may be written in a linear combination of each other [2].

1.1.2 Identification Problem

In data handling, we come across the data where sample size ' n ' is less than a number of independent variables ' p ', such a situation is known as an identification problem. With this condition, the covariance matrix of independent variables becomes singular

due to a reduction in the rank. Proof of the singularity due to rank reduction is given in the article [3].

Now, large data sets are being created and used to predict future responses from current and past observed data. To find accurate predictions, computer scientists are creating algorithms but still there exists an inverse matrix problem in data handling. So, to solve this problem, mathematicians and statisticians are working on different models based on machine learning [4].

1.2 Machine learning

1.2.1 Definition

The study of computer algorithms that improve over time as a consequence of experience and data is known as machine learning (ML). It is considered to be a part of artificial intelligence. Machine learning algorithms build a model based on training data in order to make predictions or decisions without needing to be explicitly programmed.

Machine learning is relatable with computational statistics that emphasizing estimating predictions using information technology and computers. However, not every machine learning is considered as statistical learning. The study of mathematical optimization supports the field of machine learning since it provides methodologies, theory, and applications in a variety of disciplines.

1.2.2 Theory

The concept "artificial intelligence" was created in the 1950s as a basic idea of human intellect being shown by machines. Jerrold S. Maxmen stated in 1976 that artificial intelligence (AI) will usher in the twenty-first century. AI has progressed beyond simple theory to actual application on an enormous scale during today's age of technological development and the availability of huge data sets ('big data'). Machine learning (ML), which is considered a subcategory of AI, demonstrates the empirical "acquisition" concerned with human intellect and has the potential to learn and enhance its assessment

by using computer algorithms. The machine can take an input and estimate a result with repetitions and alterations to the algorithm. The algorithm's accuracy is therefore tested by trying to compare the outcomes with ground truth, which is repeatedly revised to perfect the ability to predict future events [5].

The challenge of developing a prediction based on data that incorporates algorithms and large data calculations is tackled by machine learning. It is helpful in predicting the future prediction based on already known data. It is useful to interpreting the data according to our preferred information. Machine learning support to interpret data in many fields such as Science, Medicine, Economy, Policy-Making, etc. Machine learning works with the help of mathematical equations, statistical analysis, and computer programming techniques. It is based on a proper programming that deals with dependent and independent variables, error terms, and some estimation parameters. For getting accurate interpretations of data, researchers are still working to introduce new methods of prediction corresponding to the demand of data. Supervised and unsupervised machine learning are the two forms of machine learning.

1.3 Supervised and Unsupervised Learning

Supervised learning is a machine learning technique that includes both input and output data. Input data corresponds to independent variables while output data corresponds to the predicted response. In supervised learning, there is a possibility to test the large data by working on small training data set. In machine learning, algorithm provides the characteristics of final data sets similar to training data sets. Supervised learning is helpful in knowing the relationship between the explanatory and response variable with accuracy. There are many practical applications of supervised learning algorithms such as text categorization, signature recognition, weather forecast, stock exchange predictions, face detection, etc. [6]. Unsupervised learning is a form of statistical learning where only computations are performed on data that has not been labeled, resulting in the formation of different structures. It consists of only independent variables. There is no response variable in unsupervised learning. The algorithm

sets the link between data sets in a random way. There is no need for a human to give any input. The formation of different structures from data makes this learning more useful [7]. Unsupervised learning algorithms are useful to handle complicated tasks. Examples of unsupervised learning techniques include clustering, anomaly detection, and in some cases neural networks.

Unsupervised learning has been introduced for benefit of the reader. Detailed discussion is beyond the scope of this project.

1.3.1 Supervised Learning Models

Regression and classification models are two forms of supervised learning models. Regression models are those that deal with continuous response/output variable like real value such as height, money, intensity, length, etc. It is helpful to estimate the link between numerical value data of an outcome variable with a series of explanatory variables. While, the classification model is a type of supervised learning in which the response/output variable is categorical such as “Yes” or “No”, “True” or “false”, “male” or “female” and binary values 0 or 1. The output will be in the form of mostly 2 classes. Real-life examples are light detection, sentiment analysis, scorecard prediction of tests, etc. For this research, we are working only on the previous and proposed models of regression.

1.4 Summary

The main objective of the thesis is to propose a method for solving inverse matrix issues in regression models. Moreover, the comparison of the proposed method with the reference methods will be carried out over real-life data sets. Ordinary Least Square (OLS) is the basic method for parameter estimation in regression. OLS estimates the regression coefficients as $\beta = (X^T X)^{-1} X^T Y$. In presence of identification and perfect multicollinearity problems, the inverse of $X^T X$ does not exist. In such a situation, several solutions exist in literature. Among the existing solutions, we have considered the potential methods including LASSO, generalized inverse, and partial least squares

(PLS) regression. These reference methods are presented in Chapter 2. We have proposed beta cube regression, and compressed form of LASSO, beta cube, and generalized inverse. These methods are presented in Chapter 3. The model building of proposed methods and their comparison with the reference method is presented in Chapter 4. Real life applications and their results are discussed in chapter 5. The conclusion is presented in Chapter 6, finally, the pseudo codes and R functions are presented in Appendix. In the last of this chapter, Figure 1.1 represents the previous and newly contributed regression methods of this thesis.

**Regression
Methods
Flowchart**

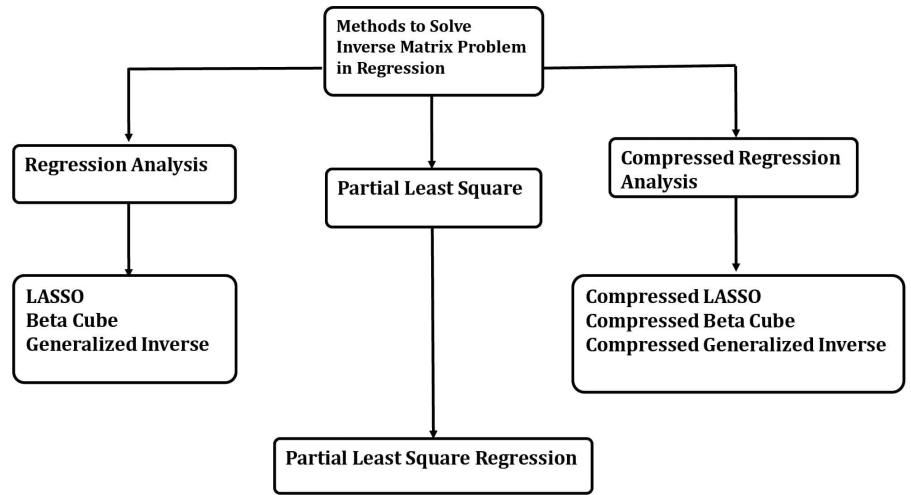


Figure 1.1: This figure represents complete flow of methods that has been used in this research

Chapter 2

Reference Regression Models

2.1 Introduction

Francis Galton was a pioneer who worked in regression analysis in social sciences in 1870s. The German mathematician Karl Gauss used the method of ordinary least squares before it in the early 1800s in astronomical data. There is intensive literature on regression analysis [8].

The method of regression is useful to determine the relationship between regressand and regressor. It represents a cause and effect relation. Regression analysis is useful in predictions and forecasting as well as to see the relationship between predictors and outcome variables [9]. There are two types of Regression Analysis.

1. Simple Linear Regression
2. Multiple Linear Regression

A method for determining a relationship between one independent variable x and one dependent variable y is known as simple linear regression. It is a linear connection that can be expressed as

$$y = \alpha + \beta x + \mu \tag{2.1}$$

In equation (2.1), α and β are two constants that represent intercept and slope and μ is a residual term in the model. These constants are called model coefficients. A model with the error terms is called the stochastic model.

$$\hat{y} = \hat{\alpha} + \hat{\beta}x \quad (2.2)$$

In equation (2.2), $\hat{\alpha}$ is the average value of \hat{y} , when there is no effect of predictor variable x and $\hat{\beta}$ represents the average change in \hat{y} with one unit change in x . Estimated sample model is called the deterministic model (model without error).

Multiple linear regression is a technique for determining the relationship between multiple independent variables $X_1, X_2, X_3, \dots, X_p$, and one dependent variable Y . It can be written as

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \dots + \beta_p X_p + \mu \quad (2.3)$$

In equation (2.3), α and $\beta_1, \beta_2, \beta_3, \dots, \beta_p$ are the model coefficients that represents the relation between response and predictors while μ is the error term in the model.

The equation for estimated multiple linear regression can be expressed as

$$\hat{Y} = \hat{\alpha} + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_3 + \dots + \hat{\beta}_p X_p \quad (2.4)$$

In equation (2.4) $\hat{\alpha}$ is the average change in \hat{Y} when all the predictors are not included. $\hat{\beta}_1$ is the unit change in \hat{Y} with one unit change in X_1 by keeping all the other predictors constant. There will be the same effect of remaining each β for their corresponding predictor [6].

2.1.1 Ordinary Least Square (OLS)

Ordinary Least Squares (OLS) is a linear least squares approach that is used to estimate the model's coefficients. In OLS, parameters are chosen by taking the set of explanatory variables in which linear function can be written as minimum of the sum of the squares of the differences between actual response variable and predicted response variable [10]. Consider the classical multiple linear regression model

$$Y = X\beta + \mu \quad (2.5)$$

Where X is an input matrix of the dimension of $n \times p$, Y is a response vector of dimension $n \times 1$, β vector has dimension $p \times 1$, and error term μ consists of $n \times 1$ dimension. Consider the assumptions that are, mean of the residual term is zero i.e $E(\mu) = 0$ and mean of variance is sigma square i.e $E(\mu^T \mu) = \sigma^2 I_n$, where I_n is an $n \times n$ identity matrix.

Residual term can be as

$$\mu = Y - X\beta. \quad (2.6)$$

To estimate the $\hat{\beta}$, the criteria is to minimize the sum of squares of residual and in matrix algebra, it can be written as sum of squares of residuals

$$\sum \mu^2 = \mu^T \mu \quad (2.7)$$

$$\sum \mu^2 = (Y - X\beta)^T (Y - X\beta)$$

$$\sum \mu^2 = (Y^T - \beta^T X^T)(Y - X\beta)$$

$$\sum \mu^2 = Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta$$

$$\sum \mu^2 = Y^T Y - (\beta^T X^T Y)^T - \beta^T X^T Y + \beta^T X^T X\beta. \quad (2.8)$$

Now, take derivative on both sides of above equation (2.8) with respect to β

$$\frac{d(\mu^T \mu)}{d\beta} = \frac{d(Y^T Y - (\beta^T X^T Y)^T - \beta^T X^T Y + \beta^T X^T X\beta)}{d\beta}$$

$$0 = 0 - X^T Y - X^T Y + 2X^T X\beta$$

$$0 = -2X^T Y + 2X^T X\beta$$

$$2X^T Y = 2X^T X \beta$$

$$X^T X \beta = X^T Y$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y. \quad (2.9)$$

The ordinary least squares (OLS) solution of above equation can be analytically calculated by considering that $X^T X$ is full rank matrix, which implies that $X^T X$ is positive definite [11].

2.1.2 Assumptions of Linear Regression

When working with data in linear regression, it's essential to recognize the model's assumptions. In regression, assumptions can be adapted according to the requirement of the model. Data can be shrunk by using regression modeling methods which required some assumptions of linear regression. These assumptions are the following:

1. Type of variable: For the linear regression model, the response variable must be continuous. Continuous variables are those in which any value can be taken between its extreme values. If the variable type is discrete such as 0 or 1, 'Yes' or 'no', then the regression model is not suitable. For discrete data, a classification model can be used.
2. Linearity: The regression model must be linear in parameters. Linear relation can be represented by a straight line.

$$Y = \alpha + \beta X \quad (2.10)$$

If explanatory variables in the model are not linear then these can be transformed into a linear model. For example, there can be a curvature in the data, which is well shown by a quadratic or cubic rather than a linear relationship.

$$Y = \alpha + \beta X^2 \quad (2.11)$$

Above equation can be modeled by

$$Y = \alpha + \beta X^* \tag{2.12}$$

where, we can assume $X^* = X^2$

3. **Distribution of Residuals:** In linear regression, residuals are normally distributed. Due to normal distribution, they lie nearly on the diagonal. The mean of residual is zero in normal distribution and variance is σ^2 .
4. **Homoscedasticity of Residuals:** Homoscedasticity means residuals or error terms have constant variance, i.e. variance is σ^2 . The standard deviation of residuals is constant and does not depend on the independent variables. Homoscedasticity occurs when the residual term is the same in all predictors, while heteroscedasticity arises when the size of the residual term varies across independent variable values.
5. **Auto Correlation:** If there is a relation between values of the same variables, then it is called autocorrelation. Autocorrelation problems mostly came in time series and cross-sectional data. It may occur when observations are dependent on each other in different factors other than time. It can also cause a problem when residuals are autocorrelated due to a wrongly identified model.
6. **Multicollinearity:** If two or more than two independent variables are related to each other then it is called multicollinearity. These correlated variables show the same aspects in a model and make it unstable. It is necessary to remove the remaining related variables from the model to make it more efficient. Multicollinearity can exist in the conditions when β coefficients are not significant, when they change completely by removing or adding a variable and when they represent a negative relationship instead of a positive.
To evaluate multicollinearity, there is a measure called variance inflation factor (VIF) which describes the increase in variance of regression estimates due to increase in multicollinearity. It exists when the VIF value is greater than 5.

Multicollinearity can be removed by eliminating variables that are causing a problem or by reducing the dimension of explanatory variables. The consequences of multicollinearity are

- If there exists an exact relationship between the predictor variables, then exact multicollinearity exists and least-squares estimators cannot be found. As $X^T X$ becomes singular, thus, coefficients and standard errors cannot be estimated.
- OLS (Ordinary Least Square) estimators become high in variance and covariance due to correlation in independent variables.
- Due to high variance and covariance, the confident interval becomes broader resulting in the acceptance of null hypothesis more freely. This is the cause of the high standard error.
- Due to small changes in the data, OLS estimators and standard errors become sensitive and give inaccurate results [12].

7. Identification problem: If sample size n is less than predictors p , then we cannot find a unique β coefficient, that is why the identification problem comes in the regression model [13].

2.1.3 Limitations of OLS

Solution of linear regression cannot be found if $X^T X$ has not a full rank matrix due to multicollinearity or identification problem. OLS performance becomes very poor due to these conditions. The main issue arises in finding the inverse matrix of $X^T X$, because it becomes singular and we cannot solve it further. This method does not remove outliers and even cannot shrink the regression coefficients. Moreover, it cannot provide an alternative way to reduce the correlated explanatory variables or variable selection method. Overall, OLS cannot perform computations in a model having big data sets. There are other methods discussed here to solve regression problems [4].

2.1.4 Compressed Regression

The development of quick and randomized approximations to significant numerical linear algebra tasks like solving least squares problems and discovering spectral decomposition has been a goal in the large-scale data processing. Compressed, modeling and preconditioning are methods for reducing the amount of data collection and creating a smaller, “compressed” solution. The variation in the optimal solution assessed at the “compressed” solution as compared to the full-data solution is further limited by probable explanations for these approximation techniques. A key objective in numerical linear algebra is to create compression methods that reduce computationally load without sacrificing too much accuracy. Comprehensive mathematical constraints with regard to the sub-sampling or arbitrary projection method are given with a high possibility due to their randomness [14].

Consider a matrix X with dimensions of $n \times p$, where n is the sample size and p is the number of predictors. Let Y is a response variable of dimension $n \times 1$ and μ of dimension $n \times 1$ satisfying the conditions, $E(\mu)=0$ and $V(\mu) = \sigma^2 I_n$, where I_n is the identity matrix. β is the estimated coefficient with dimension $p \times 1$.

$$Y = X\beta + \mu \quad (2.13)$$

Through linear regression, coefficients of $\hat{\beta}$ are estimated by using Euclidean norm as $\|X\beta - Y\|_2^2$, then a least square solution can be defined as

$$\hat{\beta}_{OLS} = \min_{\beta} \|X\beta - Y\|_2^2 \quad (2.14)$$

Equation (2.14) can be written as $\hat{\beta} = (X^T X)^{-1} X^T Y$. If $X^T X$ matrix is not a full rank matrix, then $(X^T X)^{-1}$ becomes singular and its solution can't be found with ordinary least square. This problem can be overcome by introducing a C_m that is a compression matrix having dimension $q \times n$, where q is a compression constant and $q < n$. Data can be compressed by C_m matrix as $\tilde{X} = C_m X$ and $\tilde{\mu} = C_m \mu$ [15]. The selection of C_m can be made randomly by independent Gaussian random variables. In compressed regression, the response variable is to be considered compressed as $\tilde{Y} = C_m Y$ by

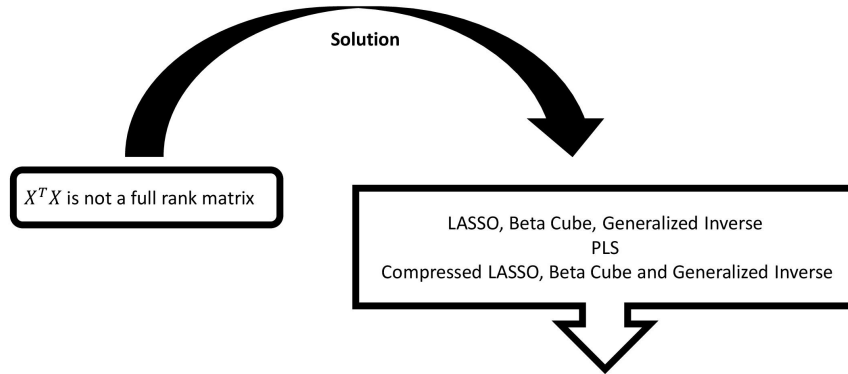
$$\tilde{Y} = C_m X \beta + C_m \mu \quad (2.15)$$

and least square compressed equation is represented as

$$\hat{\beta}_{C.OLS} = \min_{\beta} \|C_m X \beta - C_m Y\|_2^2 \quad (2.16)$$

$$\hat{\beta}_{C.OLS} = (X^T C_m^T C_m X)^{-1} X^T C_m^T C_m Y \quad (2.17)$$

Above $\hat{\beta}$ is named as compressed OLS. Same like OLS, compressed OLS does not provide results in case of $n < p$. Now, discuss the already existing methods as follows:



2.2 Least Absolute Shrinkage and Selection Operator

The LASSO operator stands for least absolute shrinkage and selection operator. It has been introduced by Fadil Santosa and William W. Symes in 1986 [16]. It came into prominence in 2006 by Robert Tibshirani [17]. This is the type of regression model that executes variable selection as well as regularization to increase the estimation of

accuracy. LASSO shrink β coefficients exactly equal to zero. LASSO is also called the L_1 norm [4].

The LASSO estimate of $\hat{\beta}$ can be written as LASSO has many practical applications that are discussed by many researchers in solving big data problems. Among the most common methods for modeling spatial auto correlation in a regression model is eigenvector-based spatial filtering. In this technique, the independent variable is a subset of eigenvectors generated from a modified spatial weight matrix. To choose the eigenvectors, the LASSO is presented in [18]. LASSO has been enhanced to address practical concerns, such as financial ones such as index tracking without short sales. Portfolio management is really a long-term focus mostly in financial sector. Modern portfolio theory (MPT) is a traditional approach to constructing optimum investment strategies [19]. One of the applications of LASSO is hard modeling multivariate curve resolution which is applied in Ion Mobility Spectra to fit broader peaks that deviate from a perfect Gaussian form [20].

The LASSO estimator can be written as

$$\hat{\beta}_{LASSO} = \arg \min_{\beta} \frac{1}{n} \sum_{i=1}^n (Y - X\beta)^2 \text{ subject to } \sum_{j=1}^p \|\beta\|_1 \leq t \quad (2.18)$$

Where t is the penalty on L_1 norm. Above equation can also be written in the form

$$\hat{\beta}_{LASSO} = \arg \min_{\beta} \frac{1}{n} \|Y - X\beta\|_2^2 + \lambda \sum_{j=1}^p \|\beta\|_1 \quad (2.19)$$

Where L_1 norm and L_2 norm are defined by $\|\beta\|_1 = \sum_{i=1}^p |\beta_i|$ and $\|\beta\|_2^2 = \sum_{i=1}^p \beta_i^2$. There is one-to-one correspondence in t and λ . This relation is because of duality and the Karush-Kuhn-Tucker (KKT) conditions. Thus, for every $t \geq 0$, there exists $\lambda \geq 0$ such that both problems play the same role [21]. The selection of λ can be done by cross validation. If $\lambda = 0$, LASSO estimator behaves similarly to the ordinary least square. If λ value rises, a number of non-zero $\hat{\beta}$ coefficients decreases and if λ value approaches to ∞ , then $\hat{\beta}$ becomes zero and LASSO provides null model [22].

Due to the non-differentiable objective function, the LASSO doesn't provide a closed-form solution to the problem. Still, there is a possibility of obtaining closed-form

by adding soft threshold operator. That's how the soft-thresholding operator is defined for LASSO regression.

$$\text{sign}_\lambda(x) = \begin{cases} x + \lambda, & \text{if } x < -\lambda \\ 0, & \text{if } |x| \leq \lambda \\ x - \lambda, & \text{if } x > \lambda \end{cases} \quad (2.20)$$

After the derivation, $\hat{\beta}$ can be written as

$$\hat{\beta} = \frac{1}{n}(X^T Y) - \frac{\lambda}{2} \text{sign}(\hat{\beta})$$

$$\hat{\beta}_{LASSO} = \begin{cases} \frac{1}{n}(X^T Y)_j + \frac{\lambda}{2}, & \text{if } \frac{1}{n}(X^T Y)_j < -\frac{\lambda}{2} \\ 0, & \text{if } \frac{1}{n}|(X^T Y)_j| \leq \frac{\lambda}{2} \\ \frac{1}{n}(X^T Y)_j - \frac{\lambda}{2}, & \text{if } \frac{1}{n}(X^T Y)_j > \frac{\lambda}{2} \end{cases} \quad (2.21)$$

The purpose of LASSO regression is to find regression parameters that correspond to a model with minimum prediction error. This is accomplished by putting a constraint on the model parameters that 'shrinks' the regression coefficients towards zero, i.e., requiring the total of the absolute values of the regression coefficients to be smaller than a predefined value λ . In a real sense, it limits the model's complexity. After shrinkage, variables having a regression coefficient of zero have been removed from the model [23].

Although LASSO regression is well-known regression but there are some pros and cons, that are as under.

Pros:

- By reducing the co-efficient towards zero, LASSO select features.
- Overfitting is avoided in it.

Cons:

The LASSO estimator has a lot of faults that make variable selection difficult in some circumstances.

- When $n < p$, the LASSO picks a maximum of n variables. If the real model has more than n variables, this might be a limitation.
- There is no grouping characteristic in LASSO, which implies it chooses only one predictor within the set of strongly correlated predictors [4].

The next method is generalized inverse regression in which we predict our data set without adding any penalty parameter.

2.3 Generalized Inverse

If A is a rectangular $n \times p$ matrix of rank $n \leq p$, then A^- of dimension $p \times n$ is called generalized inverse matrix.

$$AA^-A = A \tag{2.22}$$

Overall, the generalized inverse is not unique but it always exists. As the inverse of a matrix does not exist if its determinant becomes zero. This matrix is called the singular matrix. Moreover, the inverse could not be able to find even matrix is a non-square. The solution to finding the inverse of such matrix is presented by an American mathematician, Moore in 1935, and later in 1955, a scientist named Penrose developed a Moore inverse in a different method [24]. After this, the generalized inverse is called Moore-Penrose Inverse. In the meanwhile, an author named Rao contributed a computational method of singular matrix called pseudo inverse and used it to solve the least square theory to know estimators of linear equations [25]. But this concept has not used for high dimensional data. Several variants of generalized inverse exist, the description of these variants is as follows.

2.3.1 Pseudo inverse

Generalized inverse means that an inverse matrix A^- is associated with the matrix A in such a way that

- It happens for a class of matrices that is bigger than the non singular matrices class.
- It consists of few properties like the simple inverse.
- It shrinks to the usual matrix when the matrix is not singular [26].

Moore-Penrose inverse matrices also permit the evaluation of the system of equations with rank deficiency. There are many procedures to solve this problem but the most frequently used is the Singular Value Decomposition (SVD) method. Its function in MATLAB can be written as “pinv” and in Mathematica, it can be run by “PseudoInverse” [27]. While in R language, generalized inverse or pseudoinverse can be represented by “ginv”. This method does not much take time to execute and also provides accurate results [26].

2.3.2 Left Inverse

Generalized inverse becomes a regular inverse matrix when the number of samples n equals the number of observations p . But for $n < p$, the pseudo inverse will be the left inverse of A . This is the only inverse having rows in the row-space of A^T . The point is that $A^T A$ is invertible if A has a full column rank [28].

$$A_{left}^- = (A^T A)^{-1} A^T \quad (2.23)$$

For rectangular matrix, $A_{left}^- = (A^T A)^{-1} A^T$ is the generalized inverse of A if $(A^T A)^-$ is the ginv of $(A^T A)$.

2.3.3 Right Inverse

For $n > p$, the pseudo inverse will be the right inverse of A . This is the only inverse having a column in the column-space of A^T [28].

$$A_{right}^- = A^T (A A^T)^{-1} \quad (2.24)$$

For rectangular matrix, $A_{right}^- = A^T(AA^T)^{-1}$ is the generalized inverse of A if $(AA^T)^-$ is the ginv of (AA^T) .

2.3.4 Solve-ability of Linear Equations

One of the most applied applications of generalized inverse is to find the system of linear equations. Let

$$Y = X\beta \quad (2.25)$$

In equation (2.25), Y is a column vector with dimensions of $n \times 1$ and β is a vector with dimensions of $p \times 1$, and X is a matrix with dimensions of $n \times p$. Then the equation may be written as

$$\beta = X^{-1}Y \quad (2.26)$$

If X is non-square, singular matrix, then its generalized inverse can be represented as

$$\hat{\beta}_{ginv} = ginv(X^T X)X^T Y \quad (2.27)$$

where 'ginv' is a function in R used for generalized inverse [26].

Notable, among these variants we have used left inverse.

The properties of generalized inverse are as under

1. $\text{rank}(A) = \text{rank}(AA^-) = \text{rank}(A^-A)$.
2. $\text{rank}(A) \leq \text{rank}(A^-)$.
3. If A is non-singular and square matrix, then $A^- = A^{-1}$ and A^- is unique.
4. In addition both AA^- and A^-A is symmetric.
5. AA^- and A^-A are idempotent matrices.
6. $A^-AA^- = A^-$ and $AA^-A = A$.
7. $(AA^-)^* = AA^-$ and $(A^-A)^* = A^-A$, where A^* is the conjugate transpose of A [29] [30].

2.4 Partial Least Squares (PLS) Regression

Partial least squares (PLS) regression was firstly introduced by Herman O.A. Wold in econometrics. Then, PLS had initiated by S. Wold and H. Martens in the late seventies in branches of chemistry like in analytical, physical, and clinical chemistry. PLS is a substitute of multiple linear regression model. It is more efficient since it is robust, which implies that model parameters do not vary often when fresh samples are taken from the entire population [31].

A statistical approach for comparing response variables and numerous predictor variables is partial least squares analysis. Partial least square is the technique also known as structural equation modeling (SEM). PLS is useful for the data having less sample size, missing values, and the data of high correlation in multiple variable model. It deals in the regression as well as classification models to reduce dimension and to remove multicollinearity [32].

PLS is an all-around method used in many models of multivariate data. Other than chemometrics, it has several applications in bioinformatics, machine learning, food research, medicine, pharmacology, social sciences, and physiology. This is one of the supervised methods developed to predict accurate results in multivariate problems. PLS's primary goal is to find the associated subspace of predictor variables.

The approach of partial least squares ignores the directions in the variable space that are covered by extraneous variables. As a result, variable selection is not essential for forecasting outcomes because up and down weighting of variables is an intrinsic characteristic of the PLS estimator. However, a small sample size n and a high number of variables p might cause the regression findings to be skewed. For testing data, many irrelevant variables can be a cause of high changes in prediction. Due to these deficiencies, PLS assists to find the appropriate subspace of a p dimensional variable space when $p > n$ [33].

PLS is another possibility for solving multicollinearity and dimension reduction problems. It is an iterative procedure. In PLS, the objective is to optimize the covariance between X and Y.

$$Y = X\beta + \mu \quad (2.28)$$

Consider X to be a matrix with dimensions of $n \times p$, Y to be a column vector with dimensions of $n \times 1$, β to be a vector with dimensions of $p \times 1$, and μ is a vector with dimensions of $n \times 1$. Consider few values of A (where $A \leq p$) as the number of components to be calculated.

It works differently from regression analysis, as it calculates weights between dependent and independent variables, then finds score vector by multiplying X with weights. After this, X-loading's and Y-loading's are calculated and deflation matrix X and vector Y are estimated by equations 2.41 and 2.42. $\hat{\beta}$ can be estimated with the help of loading weights, X and Y loading.

Then the PLS regression can be written as

$$\hat{\beta}_{PLS} = W(P^T W)^{-1} Q \quad (2.29)$$

where P is the X-loading, Q is the Y-loading and W is the loading weights.

Algorithm

1. Input: X , Y , n = sample size, p = number of variables, A = maximum iteration.
2. Calculate the averages of X and Y

$$\bar{X} = \frac{\sum_1^n X}{n} \quad (2.30)$$

$$\bar{Y} = \frac{\sum_1^n Y}{n}. \quad (2.31)$$

3. Compute the standard deviation of X and Y

$$S.D(X) = \sqrt{\frac{\sum_1^n (X - \bar{X})^2}{n}} \quad (2.32)$$

$$S.D(Y) = \sqrt{\frac{\sum_1^n (Y - \bar{Y})^2}{n}}. \quad (2.33)$$

4. Centering and scaling the Data:

$$X_o = \frac{X - 1\bar{X}}{S.D(X)} \quad (2.34)$$

$$Y_o = \frac{Y - 1\bar{Y}}{S.D(Y)}. \quad (2.35)$$

5. $A \leq p$

For $a = 1$ to A

$$W_a = X_{a-1}^T Y_{a-1}. \quad (2.36)$$

- Normalize the Weights

$$W_a = \frac{W_a}{\|W_a\|_2}. \quad (2.37)$$

- Calculate Score Vector

$$t_a = X_{a-1} W_a. \quad (2.38)$$

- Calculate X-loadings

$$P_a = \frac{X_{a-1}^T t_a}{t_{a-1}^T t_a}. \quad (2.39)$$

- Calculate Y-loadings

$$Q_a = \frac{Y_{a-1}^T t_a}{t_{a-1}^T t_a}. \quad (2.40)$$

- Deflation:

$$X_a = X_{a-1} - t_a P_a^T \quad (2.41)$$

$$Y_a = Y_{a-1} - t_a Q_a. \quad (2.42)$$

where P , Q and W are matrix consisting of a vectors.

$$P = [P_1, P_2, \dots, P_a], Q = [Q_1, Q_2, \dots, Q_a], W = [W_1, W_2, \dots, W_a] \quad (2.43)$$

6. Compute the estimated beta coefficients

$$\hat{\beta} = W(P^T W)^{-1} Q. \quad (2.44)$$

As one of the major techniques to solve identification problem in regression and classification models is partial least squares. Multiple regression and PLS provide nearly the same results. But for some data sets, PLS gives more accurate performance over multiple regression. Following are the advantages of PLS.

- PLS still offers useful robust equations when the number of predictors exceeds the number of experimental sample sizes.
- It even performs when data is noisy and missing.
- PLS performance is much better than multiple regression. Comparison between performance has been explained through practical data results.
- It delivers stable results when number of predictors are correlated instead of orthogonal.
- Performance of models having more than one response variable can easily be performed through PLS.

- It can be applied to a small sample size.
- It has a strong grip over the variables like nominal, ordinal, and continuous variables [34] [32].

Along with advantages, PLS also has some deficiencies that are the following:

- There is a difficulty in constructing the loadings of predictor latent variables.
- In PLS, distributional properties of estimates are unknown.
- PLS cannot attain worthiness until it runs bootstrap.
- It has a deficiency of model test statistics [32].

PLS has many practical applications as follows:

Healthcare decision-making is difficult. Comparative studies for many fields, statistical analysis for single decision-makers, decision outcomes, and assessment criteria have all been included in coverage decision-making research. A real-world application of partial least square path modeling (PLS-PM) is being investigated to see how it may be used as a tool for empirical decision-making research in the healthcare business [35]. A multi-modal multivariate network analysis was used to define the link among the structures of data by complementary images of the brain within the same individual, as well as to demonstrate its utility by demonstrating that it can distinguish elderly people from younger adults with higher efficiency than many existing methods. By assessing each brain voxel in each person's complimentary co-registered pictures, the suggested approach constructs a composite latent variable that optimizes the covariance of all combining components using the partial least square (PLS) algorithm [36]. A chemometric approach based on partial least square methodology was applied to unfolded differential scanning calorimetry data provided by 63 samples of different vegetable oils to assess fatty acid content [37].

Chapter 3

Proposed Regression Models

3.1 Introduction

We have proposed several models for solving inverse matrix problem in regression. One of the possibilities is to introduce the cube penalty over the regression coefficient. This results in beta cube regression. Another possibility is to use the compressed regression concept with LASSO, generalized inverse, and beta cube regression. The detail of proposed methods is as under.

3.2 Beta Cube Regression

The regression coefficients are estimated using this approach by solving the following constraint, expressed as

$$\hat{\beta}_{Cube} = \arg \min_{\beta} \sum_{i=1}^n (Y - X\beta)^2 \text{ subject to } \sum_{j=1}^p \beta^3 \leq t. \quad (3.1)$$

Above equation can also be describe as

$$\hat{\beta}_{Cube} = \arg \min_{\beta} \sum_{i=1}^n (Y - X\beta)^2 + \lambda \sum_{j=1}^p \beta^3. \quad (3.2)$$

As previously mentioned in LASSO, there is a one-to-one correspondence between t and λ . As, β term comes within the $\hat{\beta}$ after the derivation of constraint. So, to

solve this issue, the normally distributed random β values are generated by "rnorm" function in the 1st iteration that will provide the value of $\hat{\beta}$ according to corresponding λ values.

Derivation of Beta Cube regression is as follow, equation (3.2) can be expressed as

$$\sum \mu^2 = (Y - X\beta)^T(Y - X\beta) + \lambda\beta^3$$

$$\sum \mu^2 = (Y^T - \beta^T X^T)(Y - X\beta) + \lambda\beta^3$$

$$\sum \mu^2 = Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta + \lambda\beta^3$$

$$\sum \mu^2 = Y^T Y - (\beta^T X^T Y)^T - \beta^T X^T Y + \beta^T X^T X\beta + \lambda\beta^3.$$

Now, take derivative on both sides of above equation with respect to β

$$\frac{d(\mu^T \mu)}{d\beta} = \frac{d(Y^T Y - (\beta^T X^T Y)^T - \beta^T X^T Y + \beta^T X^T X\beta + \lambda\beta^3)}{d\beta}.$$

$$0 = 0 - X^T Y - X^T Y + 2X^T X\beta + 3\lambda\beta^2$$

$$0 = -2X^T Y + (2X^T X + 3\lambda\beta)\beta$$

$$(2X^T X + 3\lambda\beta)\beta = 2X^T Y$$

$$\beta = (2X^T X + 3\lambda\beta)^{-1} 2X^T Y$$

$$\hat{\beta}_{Cube} = (2X^T X + 3\lambda\beta)^{-1} 2X^T Y. \quad (3.3)$$

β term in equation (3.3) can first be generated randomly by normal random distribution and against each value of λ , $\hat{\beta}$ is computed. Then from all computed $\hat{\beta}$, one optimal $\hat{\beta}$ is selected. This optimal $\hat{\beta}$ is generated again and again by applying loop in it to get the best and minimum $\hat{\beta}$. This process takes a little bit of time but gives the better performance for high dimensional data sets.

3.3 Compressed Beta Cube Regression

The compressed Beta Cube estimator is written as

$$\hat{\beta}_{C.Cube} = \arg \min_{\beta} \sum_{i=1}^n (C_m Y - C_m X \beta)^2 + \lambda \sum_{j=1}^p \beta^3 \quad (3.4)$$

$$\hat{\beta}_{C.Cube} = (2X^T C_m^T C_m X + 3\lambda\beta)^{-1} 2X^T C_m^T C_m Y. \quad (3.5)$$

Derivation of Beta Cube regression is as follow,

Residual term can be as

$$C_m \mu = C_m Y - C_m X \beta.$$

To estimate the $\hat{\beta}$, the criteria is to minimize the sum of squares of residual and in matrix algebra, sum of squares of residuals can be written as

$$\sum (C_m \mu)^2 = (C_m \mu)^T C_m \mu$$

$$\sum (C_m \mu)^2 = (C_m Y - C_m X \beta)^T (C_m Y - C_m X \beta) + \lambda \beta^3$$

$$\sum (C_m \mu)^2 = (Y^T C_m^T - \beta^T X^T C_m^T) (C_m Y - C_m X \beta) + \lambda \beta^3$$

$$\sum (C_m \mu)^2 = Y^T C_m^T C_m Y - Y^T C_m^T C_m X \beta - \beta^T X^T C_m^T C_m Y + \beta^T X^T C_m^T C_m X \beta + \lambda \beta^3$$

$$\sum (C_m \mu)^2 = Y^T C_m^T C_m Y - (\beta^T X^T C_m^T C_m Y)^T - \beta^T X^T C_m^T C_m Y + \beta^T X^T C_m^T C_m X \beta + \lambda \beta^3.$$

Now, take derivative on both sides of above equation with respect to β

$$\frac{d((C_m \mu)^T C_m \mu)}{d\beta} = \frac{d(Y^T C_m^T C_m Y - 2(\beta^T X^T C_m^T C_m Y)^T + \beta^T X^T C_m^T C_m X \beta + \lambda \beta^3)}{d\beta}.$$

$$0 = 0 - 2X^T C_m^T C_m Y + 2X^T C_m^T C_m X \beta + 3\lambda \beta^2$$

$$2X^T C_m^T C_m Y = 2X^T C_m^T C_m X \beta + 3\lambda \beta^2$$

$$2X^T C_m^T C_m Y = (2X^T C_m^T C_m X + 3\lambda \beta) \beta$$

$$\hat{\beta}_{C.Cube} = (2X^T C_m^T C_m X + 3\lambda \beta)^{-1} 2X^T C_m^T C_m Y \quad (3.6)$$

Like beta cube regression, β term in equation (3.6) can first be generated randomly by normal random distribution in compressed beta cube regression.

3.4 Compressed LASSO Regression

The compressed LASSO estimator is written as

$$\hat{\beta}_{C.LASSO} = \arg \min_{\beta} \|C_m X \beta - C_m Y\|_2^2 + \lambda \|\beta\|_1 \quad (3.7)$$

Residual term can be as

$$C_m \mu = C_m Y - C_m X \beta.$$

To estimate the $\hat{\beta}$, the criteria is to minimize the sum of squares of residual and in matrix algebra, sum of squares of residuals can be written as

$$\sum (C_m \mu)^2 = (C_m \mu)^T C_m \mu.$$

$$\sum (C_m \mu)^2 = \frac{1}{n} ((C_m Y - C_m X \beta)^T (C_m Y - C_m X \beta)) + \lambda |\beta|$$

$$\sum (C_m \mu)^2 = \frac{1}{n} ((Y^T C_m^T - \beta^T X^T C_m^T) (C_m Y - X \beta)) + \lambda |\beta|$$

$$\sum (C_m \mu)^2 = \frac{1}{n} (Y^T C_m^T C_m Y - Y^T C_m^T C_m X \beta - \beta^T X^T C_m^T C_m Y + \beta^T X^T C_m^T C_m X \beta) + \lambda |\beta|$$

$$\sum (C_m \mu)^2 = \frac{1}{n} (Y^T C_m^T C_m Y - (\beta^T X^T C_m^T C_m Y)^T + \beta^T X^T C_m^T C_m X \beta) + \lambda |\beta|$$

Now, take derivative on both sides of above equation with respect to β

$$\frac{d(\mu^T \mu)}{d\beta} = \frac{d(\frac{1}{n} (Y^T C_m^T C_m Y - (\beta^T X^T C_m^T C_m Y)^T + \beta^T X^T C_m^T C_m X \beta) + \lambda |\beta|)}{d\beta}.$$

$$0 = 0 - \frac{1}{n} (X^T C_m^T C_m Y - X^T C_m^T C_m Y + 2X^T C_m^T C_m X \beta) + \lambda \text{sign}(\beta)$$

$$0 = -\frac{2}{n} (X^T C_m^T C_m Y + X^T C_m^T C_m X \beta) + \lambda \text{sign}(\beta)$$

$$-\frac{2}{n} (X^T C_m^T C_m Y + X^T C_m^T C_m X \beta) = \lambda \text{sign}(\beta)$$

$$-\frac{1}{n} (X^T C_m^T C_m Y + X^T C_m^T C_m X \beta) = \frac{\lambda}{2} \text{sign}(\beta)$$

$$\frac{1}{n}(X^T C_m^T C_m X \beta) = \frac{1}{n}(X^T C_m^T C_m Y) - \frac{\lambda}{2} \text{sign}(\beta)$$

$$\frac{1}{n}(X^T C_m^T C_m X) \beta = \frac{1}{n}(X^T C_m^T C_m Y) - \frac{\lambda}{2} \text{sign}(\beta)$$

$\frac{1}{n}(X^T X) = I$ by considering that predictors are standardized.

$$I \beta = \frac{1}{n}(X^T C_m^T C_m Y) - \frac{\lambda}{2} \text{sign}(\beta)$$

$$\hat{\beta} = \frac{1}{n}(X^T C_m^T C_m Y) - \frac{\lambda}{2} \text{sign}(\hat{\beta})$$

$$\hat{\beta}_{C.LASSO} = \frac{1}{n}(X^T C_m^T C_m Y) - \frac{\lambda}{2} \text{sign}(\hat{\beta}) \quad (3.8)$$

$$\hat{\beta}_{C.LASSO} = \begin{cases} \frac{1}{n}(X^T C_m^T C_m Y)_j + \frac{\lambda}{2}, & \text{if } \frac{1}{n}(X^T C_m^T C_m Y)_j < -\frac{\lambda}{2} \\ 0, & \text{if } \frac{1}{n}|(X^T C_m^T C_m Y)_j| \leq \frac{\lambda}{2} \\ \frac{1}{n}(X^T C_m^T C_m Y)_j - \frac{\lambda}{2}, & \text{if } \frac{1}{n}(X^T C_m^T C_m Y)_j > \frac{\lambda}{2} \end{cases} \quad (3.9)$$

Compressed LASSO works on the same pattern of simple LASSO and also perform variable selection.

3.5 Compressed Generalized Inverse Regression

The Generalized Inverse estimator is written as

$$\hat{\beta}_{ginv} = (X_{tr}^T X_{tr})^{-1} X_{tr}^T Y_{tr}. \quad (3.10)$$

Introducing compression matrix C_m in $\hat{\beta}_{ginv}$ results in

$$\hat{\beta}_{ginv} = (X_{tr}^T C_m^T C_m X_{tr})^{-1} X_{tr}^T C_m^T C_m Y_{tr}. \quad (3.11)$$

To find the $\hat{\beta}_{ginv}$ the term $(X_{tr}^T C_m^T C_m X_{tr})^{-1}$ is calculated by "ginv" function in R such as $(X_{tr}^T C_m^T C_m X_{tr})^{-1} = \text{ginv}(X_{tr}^T C_m^T C_m X_{tr})$

Compressed regression is closely related to compressed sensing and gets an idea from it. The main goal, on the other hand, is the exact opposite of compressed sensing. Since compressed sensing of X permits the recovery of a sparse X from a small number of random samples, but in the case of compressed regression, the objective is to recover a sparse function of X . The compressed regression problem that is explained here may be a more difficult statistical inference problem, in which the objective is to choose exponentially by the large number of linear models, and one has a specific set of suitable and unknown parameters, or to estimate even the best linear model in a given class [15].

In this recent article, [38] only compressed regression was explained in detail. Now, the same approach has been applied to other linear regression models to investigate the impact of compression matrix C_m statistically.

Chapter 4

Model Building and Comparison

For model building, the model parameters are required to tune. For this we have used cross validation. Moreover cross validation is used for the reliable comparison of reference regression models and proposed regression models.

4.1 Cross Validation

Cross validation (CV) is a basic technique for determining a regression model's robustness. The main concept behind CV is to evaluate a model's prediction performance on a set of data that was not utilized to create the model. Data splitting can be done by two methods K-fold cross-validation and Monte Carlo cross-validation. In K-fold, each data point is tested once, the number of partitions is limited by k and results are unbiased but contain high variance. While, in Monte Carlo cross-validation, each data point is tested arbitrary times, partitions can be possible many times and this method result is high bias but low in variance. So, there is a trade-off between bias and variance in these cross-validation methods. As both methods provide competent results but we are considering Monte Carlo cross-validation as the splitting method. Typically, the performance of methods is judged based on new data. Novel data is not always simple to get by, and one must make do with what is available. Sample splitting is used to imitate the context of 'original' and novel data: the data set is separated into two halves (groups of samples). One is known as validation and the other is called calibration. For validation and calibration, data can be split into two sets i.e. training data set

and testing data set or hold-out set. With the training data, model parameters can be estimated. By using the estimated model parameters, if training data can be predicted and its performance can be measured then this evaluation process is called calibration. By using the trained estimators, if test data can be predicted and its performance can be monitored then this process is called validation.

Cross-validation demands it (namely the penalty parameter) to give a model with high prediction, rather than picking it to balance model fit with complexity. This method is carried out to generate a list of relevant penalty parameter options. It is preferable to choose the penalty parameter that results in the model with the best prediction performance. The obtained performance depends on the data set's original splitting. The data set is split several times into a training and test set to avoid this dependency. The model parameters for all possibilities of λ using the training data being calculated for each split and the estimated parameters are assessed on the associated test set. The penalty parameter that performs best (in certain ways) overall across the train sets is then chosen [39].

4.1.1 Monte Carlo Cross Validation

The findings of Monte Carlo Cross Validation are obtained utilizing repetitive random selection and statistical methods. This method is comparable to random experiments, in which the precise outcome is uncertain beforehand. Mathematical models are used in natural sciences, social sciences, and engineering fields to describe system dynamics using mathematical expressions. Typically, such models begin with a set of input parameters, which are then processed by the model's mathematical formulas to generate one or more outputs. Repeated random sub-sampling cross validation is another name for it. It splits the training data at random (maybe 70–30 percent, or 60-40 percent). Fit the model to the train data set with that iteration. Data is split randomly to avoid overestimate or underestimate the results. Then use the resulting model to estimate test error. Take the average of the test errors over several iterations (for example, 10, 100, or 1000) [40].

4.1.2 Selection of Penalty Parameter

The penalized term in regression is determined by a tuning parameter λ , also known as a penalty parameter. When data values are shrunk towards a central point, such as the mean, it refers to the degree of shrinking that happens. Shrinkage produces simple sparse models that are easier to understand than multi-parameter high-dimensional data models. To make a range of multiple models, a sequence of tuning parameters is being used.

4.1.3 Performance Estimation

For performance estimation of regression models, we have used root mean square error (RMSE), which is a tool for analyzing prediction quality. It predicts how far estimations fall from actual values using Euclidean distance.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{n}} \quad (4.1)$$

where n is sample size, Y_i is the i^{th} actual response and \hat{Y}_i is the i^{th} predicted response.

For computing RMSE, first, calculate the difference between actual and estimated response for each data point that is equal to error or residual term $\mu = Y - \hat{Y}$. Compute the norm of the error term and then find the mean of residuals and take its square root. Root mean square error is helpful in supervised learning to see whether predictions are correctly measured or not. In machine learning, it is tremendously useful to know the model's performance with the help of a single value in training, testing, and cross validation data. RMSE is an initiative technique to recognize and suitable with most of the common statistical assumptions [41].

Figure 4.1, represents the complete flow of data splitting and estimation of parameters in basic OLS method. Figure 4.2 is showing whole process of PLS regression model through flow chart.

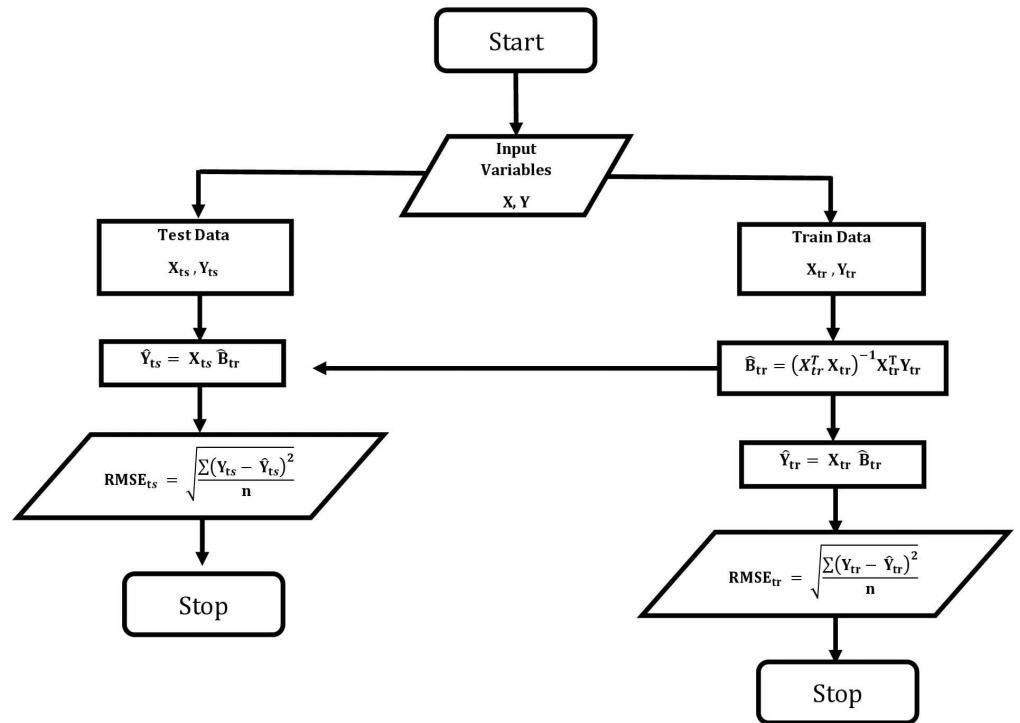


Figure 4.1: This figure represents complete flow of computational steps in OLS method

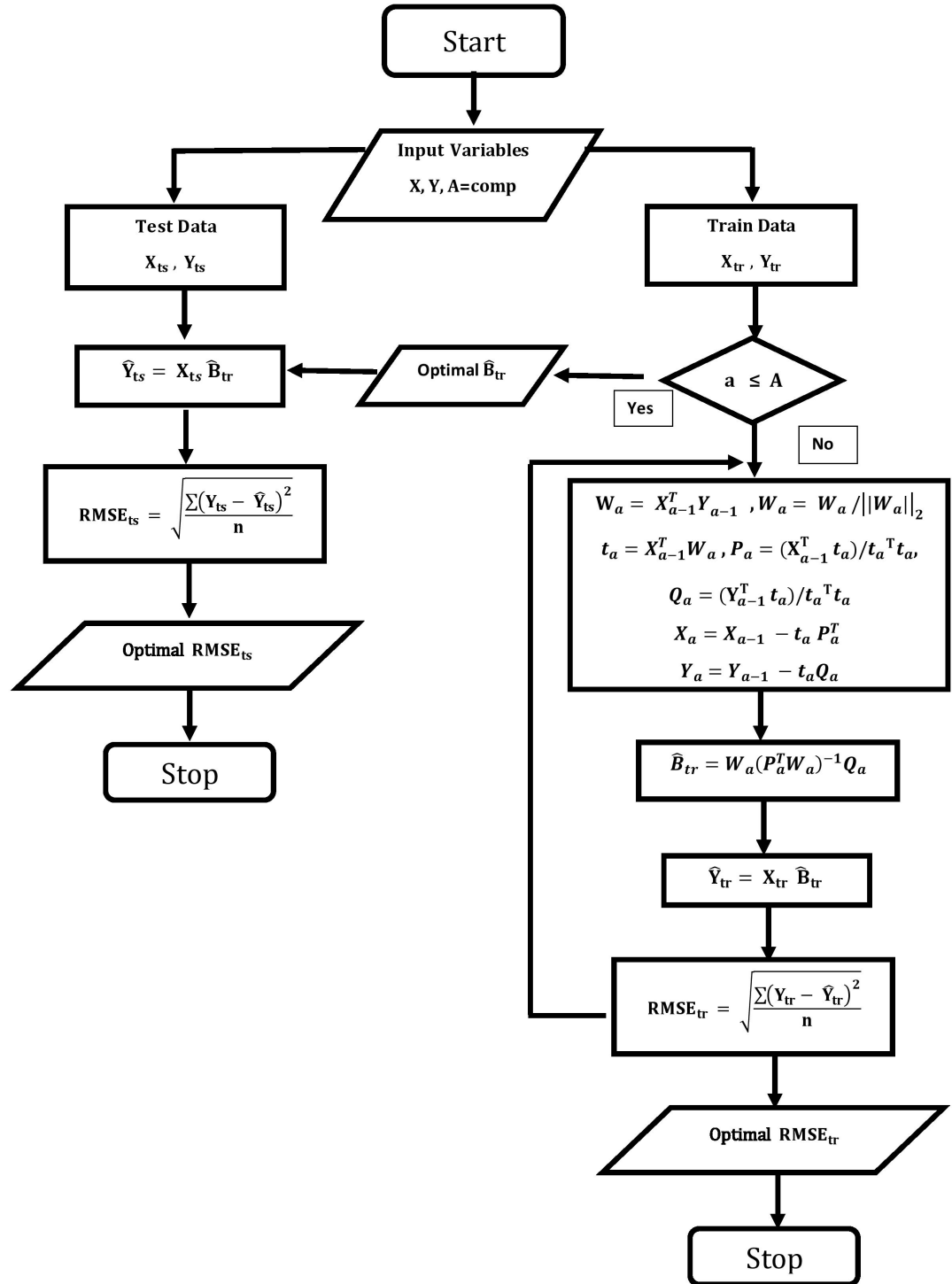


Figure 4.2: This figure represents complete flow of computational steps in PLS method

Chapter 5

Applications and Results

A statistical model is a mathematical representation of a real-world problem. The data should be summed and explained as closely as feasible by the model. It must be practical, and that it must be simple to comprehend and easy to apply. The goal is to create a model out of them without losing too much data [42].

Predictions have long been an important element of modern data science, either in the domain of statistical analysis or machine learning. Modern technology is enabling a tremendous expansion of data, yet this data frequently contains useless information, making prediction difficult. For extracting information and constructing strong prediction models, researchers are using innovative approaches and algorithms. Such models frequently include predictor variables that are either directly or indirectly related to other predictor variables. Univariate and multi-response models are being used frequently in modern inter-disciplinary research disciplines such as chemometrics, econometrics, and bioinformatics. This research compares several univariate prediction models and algorithms based on their ability to predict data using linear models for certain characteristics. A few of the characteristics include the correlation between predictor factors, the number of predictor variables, and the placement of relevant predictor components. The main purpose of this study is to compare current and newly contributed prediction approaches such as PLS, LASSO, generalized inverse, beta cube regression, and compressed versions of these techniques.

An example from near infrared (NIR) and Raman spectroscopy calibrations of chem-

ical components in food samples will be used to demonstrate the reference and proposed methods of regression. The data set is derived from a spectroscopic research whose main goal was to examine the feasibility of utilizing various spectroscopy methods to measure the fat composition in food products such as meat and fish.

The fatty acid composition and quantities of major constituents in a complex food model system were determined using Raman and NIR spectroscopy. To produce an approximate chemical replication of regular fish and meat samples, a model system consisting of 70 distinct combinations of protein, water, and oil blends was developed, exhibiting variations in both fatty acid composition and concentrations of major components. Raman and NIR techniques are being used to measure the model samples. For prediction, fatty acid characteristics were expressed in polyunsaturated fatty acid concentrations. The Raman and near-infrared spectroscopies have a lot in common. They have several characteristics that make them both appropriate for food analysis that is quick and useful. There is no need to prepare samples for either approach. It is feasible to do measurements with fiber optics. It is possible to get qualitative, quantitative, and structural information about the samples. NIR is a vibrational spectroscopy method based on overtones and combinations of basic vibrational modes, whereas Raman is a vibrational spectroscopy technique based on fundamental stretching and deformation modes. In comparison to Raman, the latter technique's spectral bands are usually wider, giving NIR a poor chemical selectivity [43].

Multivariate results are provided by spectroscopy and, more broadly, analytical chemistry methods. These experimental inputs are frequently gathered to evaluate one or more product characteristics. This assessment is often conducted on calibrating a chemometric model, such as partial least square regression [44]. Algorithms for machine learning (ML) provide mathematical tools for linking spectrum reflectance to food components. To predict numerous food components, many machine learning techniques have been explored to create calibration models on a local and worldwide scale [45]. The detail of these data sets is as under.

5.1 NIR (Near-Infrared) Spectra of Biscuit Dough

The data set contains 700 NIR spectra wavelengths (1100–2498 nm in 2 nm increments) that have been utilized as predictor variables. The yield percentages of (a) fat, (b) sucrose, (c) flour, and (d) water are being organized into four response variables. Each response's evaluation is predicted separately as uni variate response [46] [47]. There is a sample size of 72 in this model that split into train and test data by Monte Carlo cross validation. Threshold value has been taken from 0.1 to 1 with the gap of 0.1 in all linear models except PLS. The 15 number of components has been used in PLS model. Total 10 iterations have been run for all regression models. Table 5.1 is representing the mean, variance, number of testing and training data for all responses of this data set. Here all responses are predicted by 7 methods discussed in Chapter 2 and 3. Figure 5.1 represents flow for getting data from NIR spectra. [48]

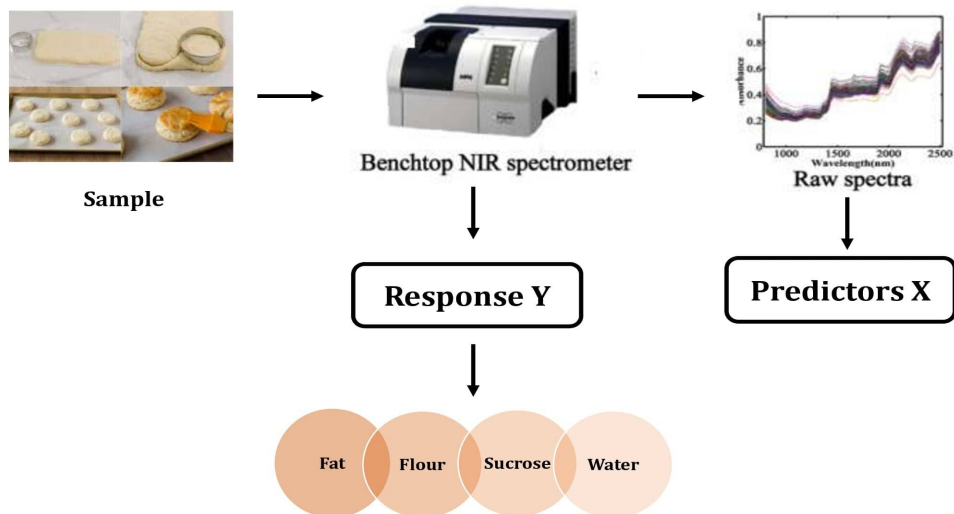


Figure 5.1: This figure represents the NIR spectra of Biscuit Dough

Responses	Fat	Flour	Sucrose	Water
Mean	18.30958	16.59375	48.98194	14.19139
Variance	3.874384	15.24049	7.464689	2.200322
Training Data	50	50	50	50
Testing Data	22	22	22	22

Table 5.1: NIR (Near-Infrared) Spectra of Biscuit Dough

PLS and LASSO methods are considered as a reference to know the prediction accuracy of other methods. In the Figure 5.2, RMSE of generalized inverse and beta cube regression is nearly equal to zero means that they are performing best for this data set. In comparison with PLS, compressed generalized inverse and compressed beta cube are giving the best performance. Compressed LASSO is performing nearly same like standard lasso for this data set.

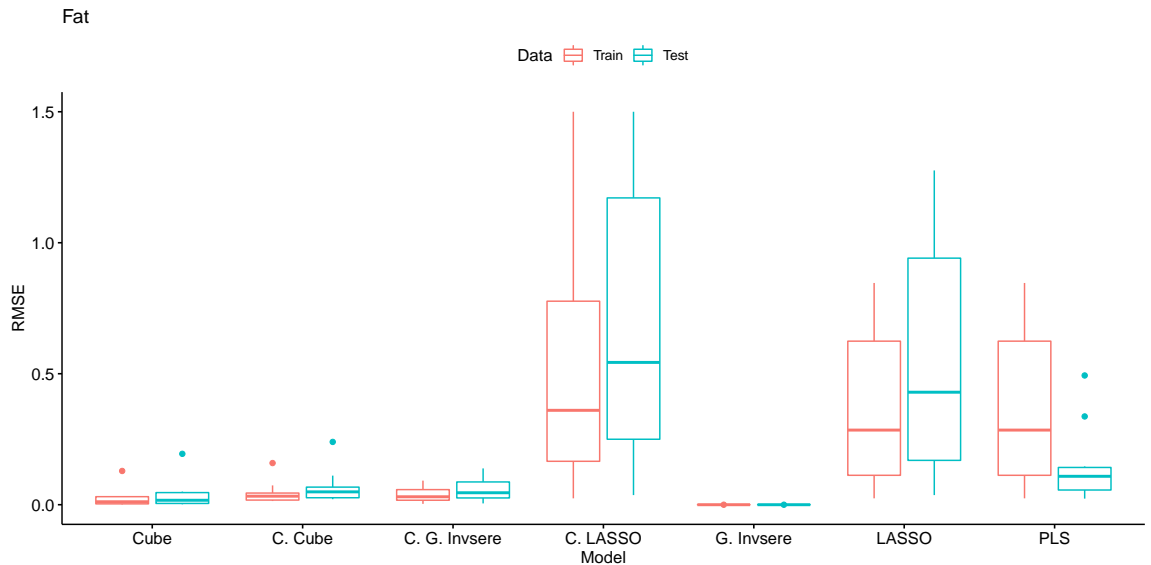


Figure 5.2: This represents RMSE of fat component of biscuit dough for each prediction method

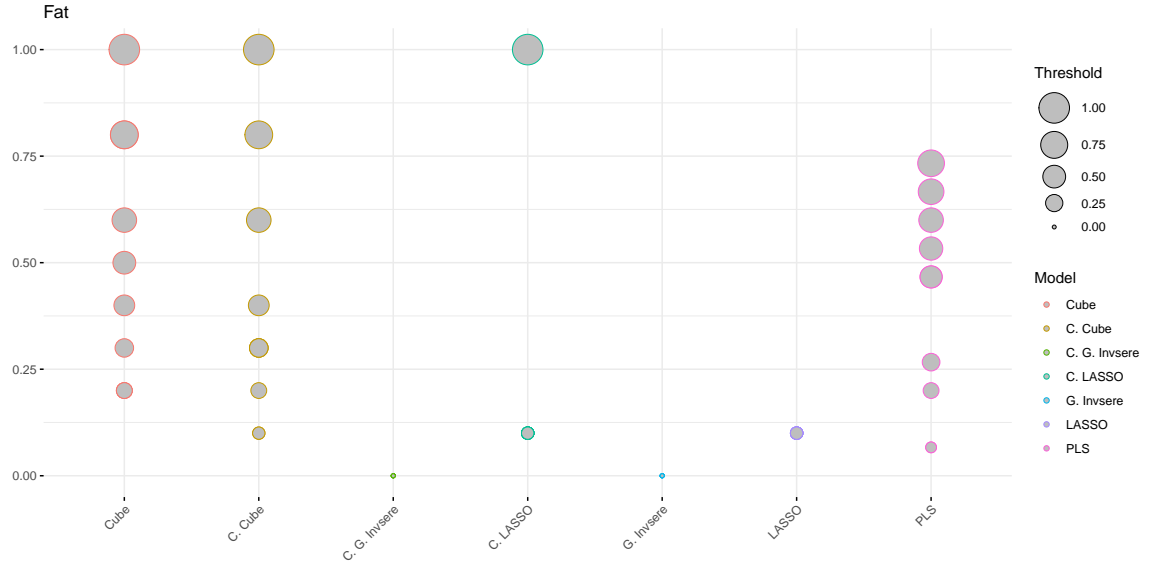


Figure 5.3: This represents threshold of fat component of biscuit dough for each prediction method

For 10 iterations, sample data is split randomly for training and testing and in each iteration, a model gets an optimal threshold for which models perform best by giving minimum root mean square error. For Fat data set, in Figure 5.3, Generalized and compressed generalized inverse are performing well for threshold = 0. This means for threshold = 0, the models are including all the $\hat{\beta}$ coefficients. Beta Cube and compressed beta cube are picking different penalty parameters between 0.1 and 1 and thus getting the mean value of λ around 0.5. This means the models are picking approximately half of the relevant $\hat{\beta}$ coefficients. In PLS, a model is considering optimal components in between the range 0.1 to 0.75. LASSO is predicting best for threshold value equal to 0.1 while compressed LASSO is choosing λ values 0.1 and 1 for getting minimum RMSE.

In Figure 5.4, the RMSE of the generalized inverse is nearly equal to zero. In comparison with PLS and LASSO; beta cube, compressed generalized inverse, and compressed beta cube are giving better performance. Compressed LASSO is performing parallel to PLS but a bit less performance than standard LASSO.

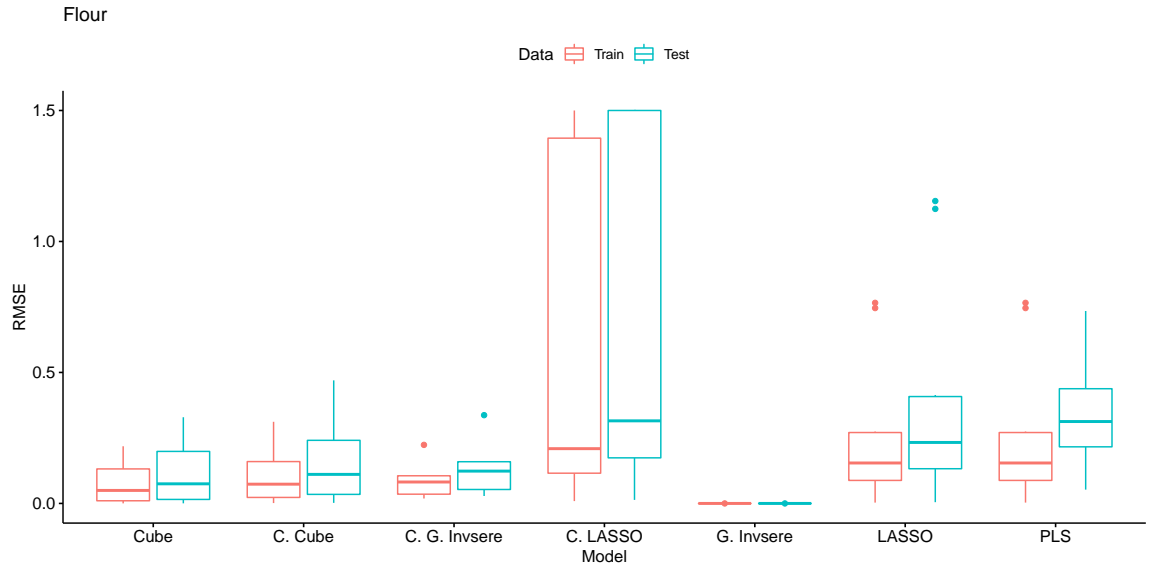


Figure 5.4: This represents RMSE of flour component of biscuit dough for each prediction method

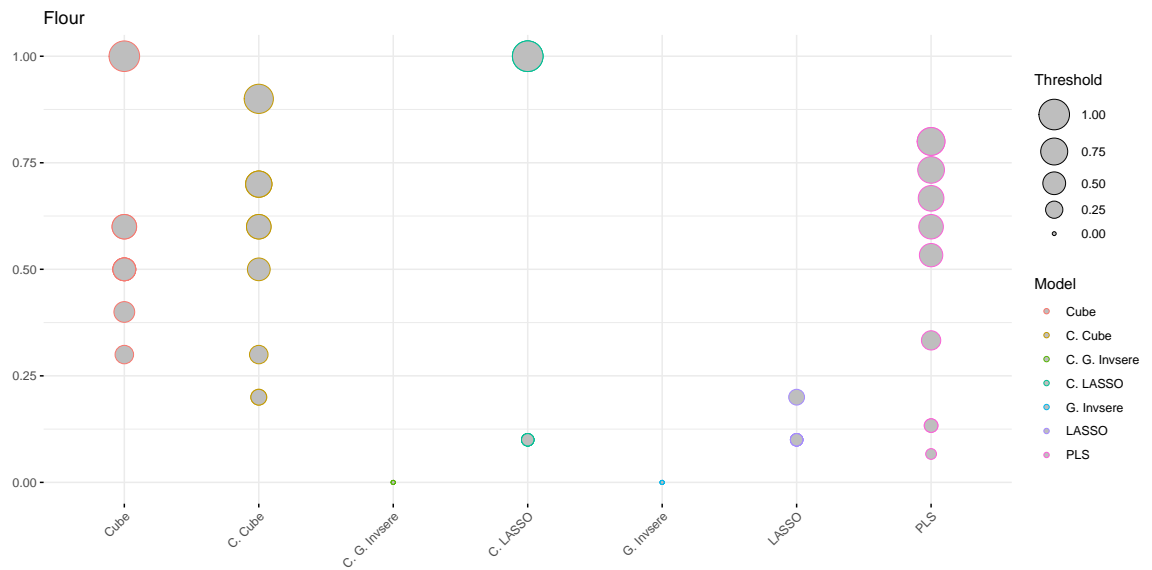


Figure 5.5: This represents threshold of flour component of biscuit dough for each prediction method

For the Flour data set, Figure 5.5, Generalized and compressed generalized inverse

are predicting best for threshold = 0. PLS is picking different penalty parameters between 0 and 0.8 and thus getting the mean value of threshold around 0.4. Beta cube is showing optimal results on the threshold from 0.25 to 0.65 and at 1 while the compressed beta cube is performing well on the threshold between 0.2 and 0.9. In LASSO, a model is considering optimal λ at 0.1 and 0.2 while in compressed LASSO, it is at 0.1 and 1. Every model is considering different threshold values according to model performance.

In the Figure 5.6, RMSE of generalized inverse and beta cube regression is again nearly equal to zero for this data set. In comparison with PLS and LASSO, compressed generalized inverse and compressed beta cube are giving better performance but lower than simple generalized inverse and beta cube. LASSO and compressed LASSO are not performing well for this data set because RMSE is much higher than other regression models.

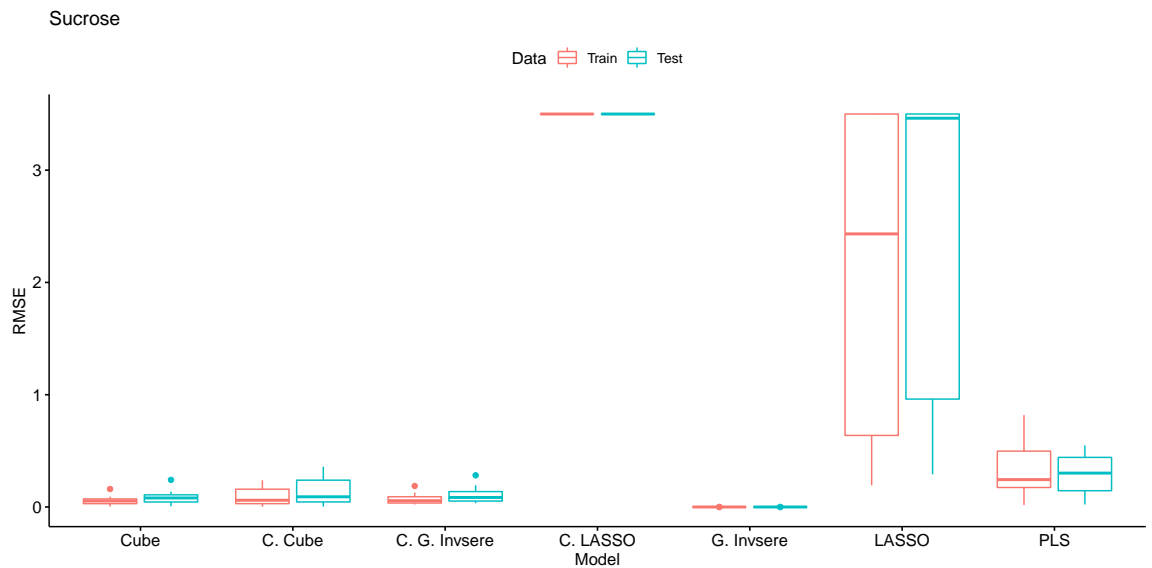


Figure 5.6: This represents RMSE of sucrose component of biscuit dough for each prediction method

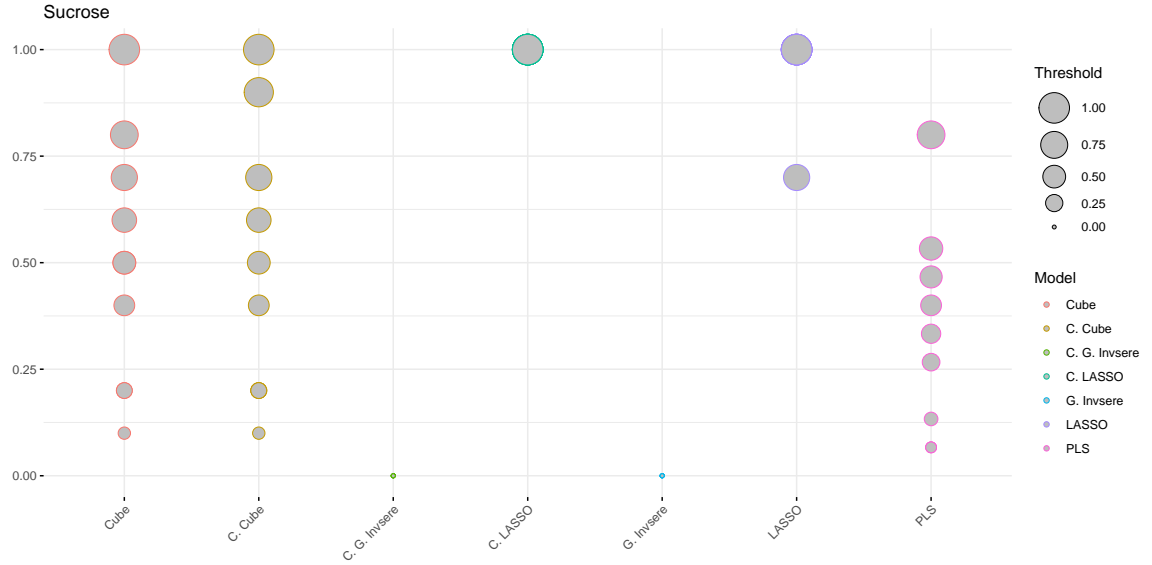


Figure 5.7: This represents threshold of sucrose component of biscuit dough for each prediction method

For Sucrose data set, figure 5.7, Generalized and compressed generalized inverse are giving best results for threshold = 0. Beta Cube and compressed beta cube are picking different penalty parameters between 0 and 1. While PLS is showing optimal results on threshold between 0 and 0.8. Standard LASSO and compressed LASSO are not performing well even at a threshold equal to 1 because of the increase in the value of λ results in the decrease of $\hat{\beta}$ coefficients. Thus, the model is not performing well with few $\hat{\beta}$ coefficients.

In the Figure 5.8, RMSE of the generalized inverse is again nearly equal to zero for the Water data set. In comparison with PLS and LASSO; beta cube, compressed generalized inverse, and compressed beta cube are giving better performance but lower than the simple generalized inverse. LASSO and compressed LASSO are not performing, this means these two are only helpful in variable selection rather than giving suitable predictions.

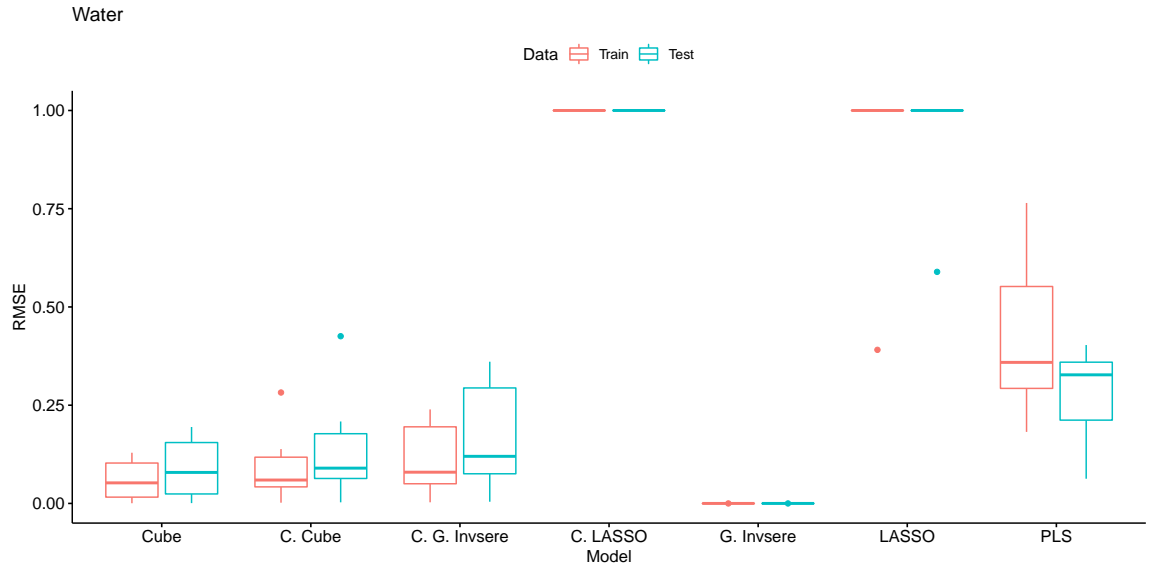


Figure 5.8: This represents RMSE of water component of biscuit dough for each prediction method

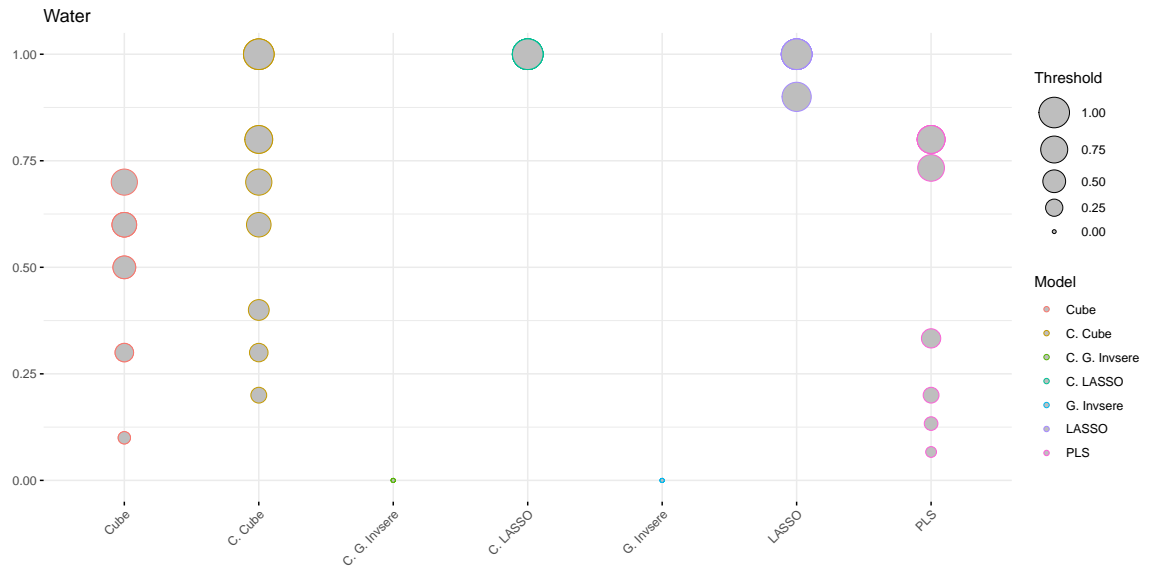


Figure 5.9: This represents threshold of water component of biscuit dough for each prediction method

For the Water data set, Figure 5.9, Generalized and compressed generalized inverse

are giving the best results for threshold = 0. Beta Cube is picking different penalty parameters between threshold 0.1 and 0.7, while the compressed beta cube is showing optimal results between threshold 0.2 and 1. PLS is performing well between thresholds 0.1 and 0.8. LASSO and Compressed LASSO are not performing well even by picking a threshold above 0.8.

5.2 Raman Spectra Analysis of Contents of Polyunsaturated Fatty Acids (PUFA)

The data comes from spectroscopic research whose main goal was to see if different spectroscopy techniques could be used to measure fat composition in food items like meat and fish. The fat content of these samples being altered by incorporating mixes of 5 different vegetable and marine oils based on a distinct mixing design. As a result, there is variance in both the major ingredients (proteins, water, and fats) and the fatty acid composition across the 69 samples in the data set. It is indeed an appropriate way to describe variables in the form of the exact amount of minor components when utilizing Raman spectroscopy to extract particular chemical information from small components in meals, such as fat content in this example [49]. Fatty acid information is presented as a) percentage of total sample weight and b) percentage of total fat content in this data set. Raman spectroscopy was used to evaluate the samples, yielding 1096 wavelength variables as predictors. Chemical information is extracted from micro food components. The purpose of this experiment is to investigate how well different prediction algorithms can estimate PUFA levels from Raman spectra. Model is split into train and test data by Monte Carlo cross validation. Table 5.2 is representing the mean, variance, number of testing and training data for all responses of this data set. Total 10 iterations have been run for all regression models. The λ values are taken from 0.1 to 1 with the gap of 0.1. Figure 5.10 represents flow for getting data from Raman spectra.

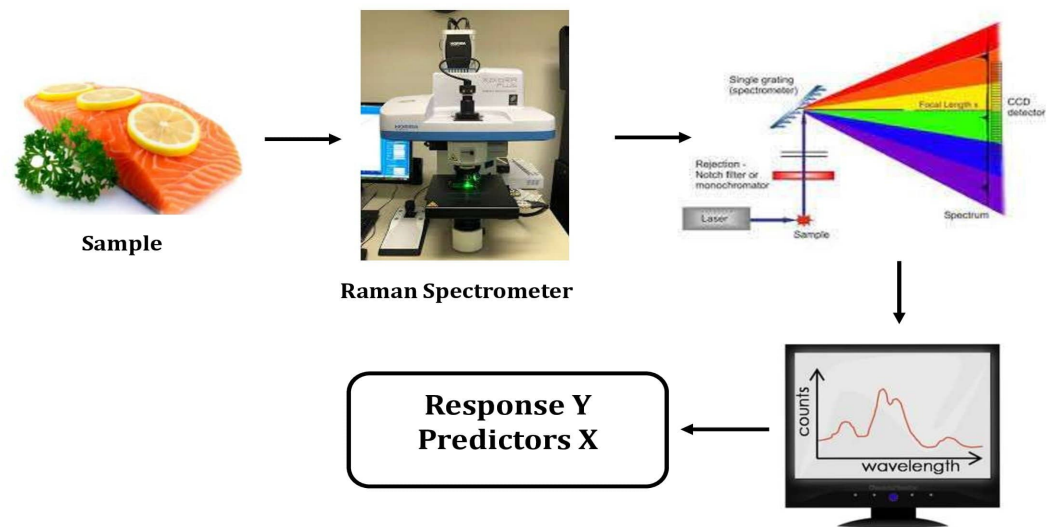


Figure 5.10: This figure represents the Raman spectroscopy of Fatty Acids

Responses	Sample weight	Fat content
Mean	4.398227	33.63645
Variance	7.899378	251.8544
Training Data	48	48
Testing Data	21	21

Table 5.2: Raman Spectra Analysis of Contents of Polyunsaturated Fatty Acids (PUFA)

PLS and LASSO methods are considered as a reference to know the prediction accuracy of other prediction methods. In Figure 5.11, RMSE of generalized inverse and beta cube regression is nearly equal to zero means that they are performing best for this data set. In comparison with PLS and LASSO, compressed forms of generalized inverse and beta cube are performing better. Along with LASSO, its compressed form is also not working well.

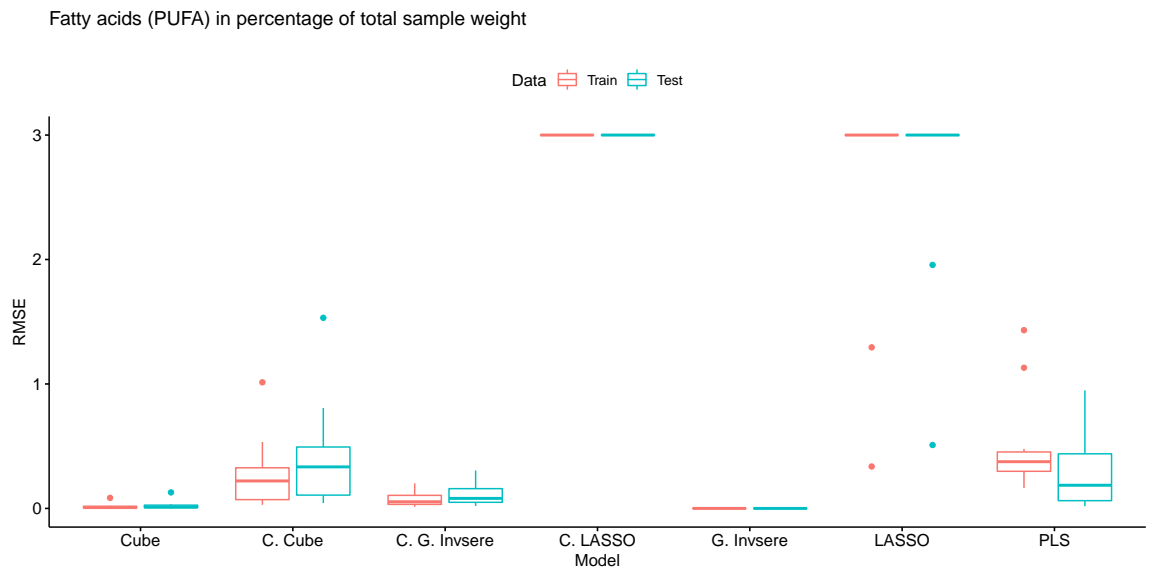


Figure 5.11: This represents RMSE of fatty acid as a percentage of total sample weight for each prediction method

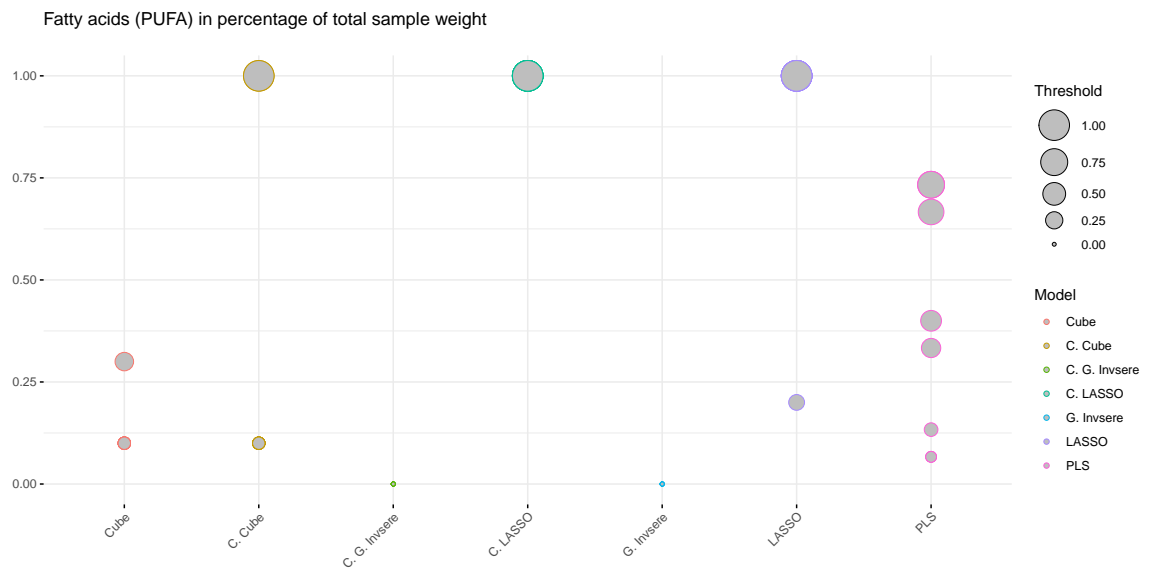


Figure 5.12: This represents threshold of fatty acid as a percentage of total sample weight for each prediction method

For 10 iterations, sample data is split randomly for training and testing and in

each iteration, a model gets an optimal threshold for which models perform best by giving minimum root mean square error. In Figure 5.12, Generalized and compressed generalized inverse are performing well for threshold = 0. Beta Cube is picking 0.1 and 0.3 as an optimal threshold while compressed beta cube is choosing the values 0.1 and 1. PLS is considering optimal components in between the range 0 to 0.75. LASSO and compressed LASSO are predicting RMSE for thresholds 0.2 and 1 but these methods are not giving suitable results.

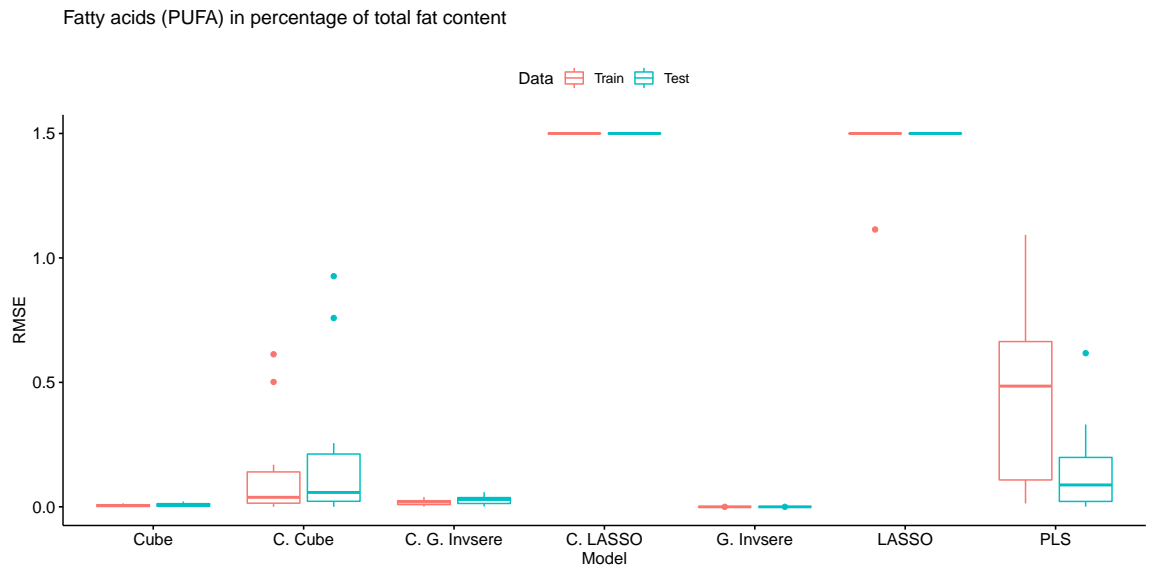


Figure 5.13: This represents RMSE of fatty acid as a percentage of total fat content in this data set for each prediction method

In Figure 5.13, RMSE of generalized inverse, and beta cube regression is nearly equal to zero. In comparison with PLS, compressed generalized inverse and compressed beta cube are performing well. But LASSO and compressed LASSO are not performing well for this data set as well.

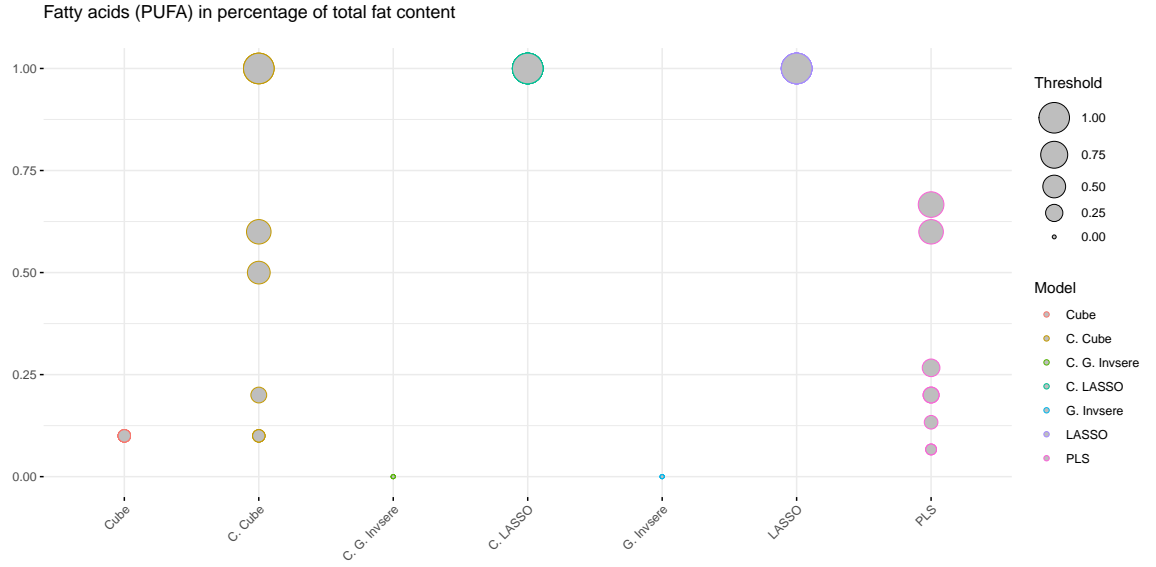


Figure 5.14: This represents threshold of fatty acid as a percentage of total fat content in this data set for each prediction method

In Figure 5.14, Generalized and compressed generalized inverse are predicting best for threshold = 0. Beta Cube is picking $\lambda = 0.1$ means it is choosing a maximum number of $\hat{\beta}$ coefficients. While compressed Beta Cube is picking penalty parameters at values 0.1 and 1. PLS is taking threshold values between 0 and 0.7. In LASSO and compressed LASSO, the model is considering an optimal value equal to 1 but not performing well at this threshold value.

Chapter 6

Conclusions

The research has concluded that by using real life practical examples of biscuit dough and polyunsaturated fatty acids, we predicted the performance of current and newly proposed regression methods. By considering using LASSO and PLS as standard methods for regression to predict response, it has been noticed that our new contributed methods are working much better than already existing models. LASSO has a benefit over other methods because it helps in variable selection but it doesn't perform well for high dimensional data and provides high value of root mean square error (RMSE). On the other side, PLS is suitable for optimizing covariance between X matrix and multiple responses. So, by working on the real data sets, it is concluded that our new contributed method 'Beta Cube Regression' is performing very well in comparison with LASSO and PLS. Moreover, our contributed compressed forms of generalized inverse and beta cube are also performing well in comparison with LASSO. Overall, all our contributed methods are performing well except compressed LASSO because it is also giving preference to provide variable selection rather than providing minimum RMSE.

These regression methods are specially designed for the data sets having a very large number of independent variables as compared to sample size. Moreover, these are helpful for the data sets which contain multicollinearity. In future, these methods can also be applied to data sets which even doesn't have identification and multicollinearity problem. Compressed LASSO regression is not well working for the above data sets, but this method can be improved further in the future by applying advanced techniques.

Moreover, PLS regression can also be transformed into compressed form by substituting C_m matrix in independent and dependent variables.

Bibliography

- [1] C. R. Rao, “A note on a generalized inverse of a matrix with applications to problems in mathematical statistics,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 24, no. 1, pp. 152–158, 1962.
- [2] A. Alin, “Multicollinearity,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 3, pp. 370–374, 2010.
- [3] G. Trenkler, “C430. on the singularity of the sample covariance matrix,” *Journal of Statistical Computation and Simulation*, vol. 52, no. 2, pp. 172–173, 1995.
- [4] F. Emmert-Streib and M. Dehmer, “High-dimensional lasso-based computational regression models: regularization, shrinkage, and selection,” *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 359–383, 2019.
- [5] J. M. Helm, A. M. Swiergosz, H. S. Haeberle, J. M. Karnuta, J. L. Schaffer, V. E. Krebs, A. I. Spitzer, and P. N. Ramkumar, “Machine learning and artificial intelligence: Definitions, applications, and future directions,” *Current reviews in musculoskeletal medicine*, vol. 13, no. 1, pp. 69–76, 2020.
- [6] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.
- [7] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, “Interpretable machine learning: definitions, methods, and applications,” *arXiv preprint arXiv:1901.04592*, 2019.

- [8] J. S. Armstrong, “Illusions in regression analysis,” *International Journal of Forecasting*, vol. 28, no. 3, p. 689–694, 2012.
- [9] R. J. Freund, W. J. Wilson, and P. Sa, *Regression analysis*. Elsevier, 2006.
- [10] G. D. Hutcheson, “Ordinary least-squares regression,” *L. Moutinho and GD Hutcheson, The SAGE dictionary of quantitative management research*, pp. 224–228, 2011.
- [11] B. Mahaboob, B. Venkateswarlu, C. Narayana, C. Ravi, and P. Balasiddamuni, “A treatise on ordinary least squares estimation of parameters of linear model,” *International Journal of Engineering & Technology*, vol. 7, no. 4.10, pp. 518–522, 2018.
- [12] R. K. Paul, “Multicollinearity: Causes, effects and remedies,” *IASRI, New Delhi*, vol. 1, no. 1, pp. 58–65, 2006.
- [13] M. Tranmer and M. Elliot, “Multiple linear regression,” *The Cathie Marsh Centre for Census and Survey Research (CCSR)*, vol. 5, no. 5, pp. 1–5, 2008.
- [14] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.
- [15] S. Zhou, J. Lafferty, and L. Wasserman, “Compressed regression,” *arXiv preprint arXiv:0706.0534*, 2007.
- [16] F. Santosa and W. W. Symes, “Linear inversion of band-limited reflection seismograms,” *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1307–1330, 1986.
- [17] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

- [18] H. Seya, D. Murakami, M. Tsutsumi, and Y. Yamagata, “Application of lasso to the eigenvector selection problem in eigenvector-based spatial filtering,” *Geographical Analysis*, vol. 47, no. 3, pp. 284–299, 2015.
- [19] L. Wu, Y. Yang, and H. Liu, “Nonnegative-lasso and application in index tracking,” *Computational Statistics & Data Analysis*, vol. 70, pp. 116–126, 2014.
- [20] V. Pomareda, D. Calvo, A. Pardo, and S. Marco, “Hard modeling multivariate curve resolution using lasso: application to ion mobility spectra,” *Chemometrics and Intelligent Laboratory Systems*, vol. 104, no. 2, pp. 318–332, 2010.
- [21] T. Hastie, R. Tibshirani, and M. Wainwright, “Statistical learning with sparsity: the lasso and generalizations,” 2015.
- [22] N. Gauraha, “Introduction to the lasso,” *Resonance*, vol. 23, no. 4, pp. 439–464, 2018.
- [23] J. Ranstam and J. Cook, “Lasso regression,” *Journal of British Surgery*, vol. 105, no. 10, pp. 1348–1348, 2018.
- [24] R. Penrose, “A generalized inverse for matrices,” in *Mathematical proceedings of the Cambridge philosophical society*, vol. 51, pp. 406–413, Cambridge University Press, 1955.
- [25] C. R. Rao, “Analysis of dispersion for multiply classified data with unequal numbers in cells,” *Sankhyā: The Indian Journal of Statistics (1933-1960)*, vol. 15, no. 3, pp. 253–280, 1955.
- [26] A. Ben-Israel and T. N. Greville, *Generalized inverses: theory and applications*, vol. 15. Springer Science & Business Media, 2003.
- [27] P. Courrieu, “Fast computation of moore-penrose inverse matrices,” *arXiv preprint arXiv:0804.4809*, 2008.

- [28] T. Greville, “The pseudoinverse of a rectangular or singular matrix and its application to the solution of systems of linear equations,” *SIAM review*, vol. 1, no. 1, pp. 38–43, 1959.
- [29] A. G. Fisher, “On construction and properties of the generalized inverse,” *SIAM Journal on Applied Mathematics*, vol. 15, no. 2, pp. 269–272, 1967.
- [30] P. McCullagh and J. A. Nelder, *Generalized linear models*. Routledge, 2019.
- [31] P. Geladi and B. R. Kowalski, “Partial least-squares regression: a tutorial,” *Analytica chimica acta*, vol. 185, pp. 1–17, 1986.
- [32] D. M. Pirouz, “An overview of partial least squares,” *Available at SSRN 1631359*, 2006.
- [33] T. Mehmood, K. H. Liland, L. Snipen, and S. Sæbø, “A review of variable selection methods in partial least squares regression,” *Chemometrics and Intelligent Laboratory Systems*, vol. 118, pp. 62–69, 2012.
- [34] R. D. Cramer, “Partial least squares (pls): its strengths and limitations,” *Perspectives in Drug Discovery and Design*, vol. 1, no. 2, pp. 269–278, 1993.
- [35] K. E. Fischer, “Decision-making in healthcare: a practical application of partial least square path modelling to coverage of newborn screening programmes,” *BMC medical informatics and decision making*, vol. 12, no. 1, pp. 1–13, 2012.
- [36] K. Chen, E. M. Reiman, Z. Huan, R. J. Caselli, D. Bandy, N. Ayutyanont, and G. E. Alexander, “Linking functional and structural brain images with multivariate network analyses: a novel application of the partial least square method,” *Neuroimage*, vol. 47, no. 2, pp. 602–610, 2009.
- [37] L. Cerretani, R. M. Maggio, C. Barnaba, T. G. Toschi, and E. Chiavaro, “Application of partial least square regression to differential scanning calorimetry data for fatty acid quantitation in olive oil,” *Food Chemistry*, vol. 127, no. 4, pp. 1899–1904, 2011.

- [38] D. Homrighausen and D. J. McDonald, “Compressed and penalized linear regression,” *Journal of Computational and Graphical Statistics*, vol. 29, no. 2, pp. 309–322, 2020.
- [39] W. N. van Wieringen, “Lecture notes on ridge regression,” *arXiv preprint arXiv:1509.09169*, 2015.
- [40] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev, “Why the monte carlo method is so important today,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, no. 6, pp. 386–392, 2014.
- [41] T. Chai and R. R. Draxler, “Root mean square error (rmse) or mean absolute error (mae)?—arguments against avoiding rmse in the literature,” *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.
- [42] V. Fonti and E. Belitser, “Feature selection using lasso,” *VU Amsterdam Research Paper in Business Analytics*, vol. 30, pp. 1–25, 2017.
- [43] N. Afseth, V. Segtnan, B. Marquardt, and J. Wold, “Raman and near-infrared spectroscopy for quantification of fat composition in a complex food model system,” *Applied spectroscopy*, vol. 59, no. 11, pp. 1324–1332, 2005.
- [44] J.-M. Roger, A. Biancolillo, and F. Marini, “Sequential preprocessing through orthogonalization (sport) and its application to near infrared spectroscopy,” *Chemometrics and Intelligent Laboratory Systems*, vol. 199, p. 103975, 2020.
- [45] R. Reda, T. Saffaj, H. Derrouz, S. E. Itqiq, I. Bouzida, O. Saidi, B. Lakssir, *et al.*, “Comparing calreg performance with other multivariate methods for estimating selected soil properties from moroccan agricultural regions using nir spectroscopy,” *Chemometrics and Intelligent Laboratory Systems*, vol. 211, p. 104277, 2021.
- [46] R. Rimal, T. Almøy, and S. Sæbø, “Comparison of multi-response prediction methods,” *Chemometrics and Intelligent Laboratory Systems*, vol. 190, pp. 10–21, 2019.

- [47] U. Indahl, “A twist to partial least squares regression,” *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 19, no. 1, pp. 32–44, 2005.
- [48] Y. Sun, Y. Wang, J. Huang, G. Ren, J. Ning, W. Deng, L. Li, and Z. Zhang, “Quality assessment of instant green tea using portable nir spectrometer,” *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, vol. 240, p. 118576, 2020.
- [49] T. Naes, O. Tomic, N. K. Afseth, V. Segtnan, and I. Måge, “Multi-block regression based on combinations of orthogonalisation, pls-regression and canonical correlation analysis,” *Chemometrics and Intelligent Laboratory Systems*, vol. 124, pp. 32–42, 2013.

Appendix

Pseudo code

The pseudo code of OLS is presented below.

1. First split data the explanatory and response into training and testing data.
2. Input: $X_{tr}, Y_{tr}, X_{ts}, Y_{ts}$, n =sample size
3. Compute the estimated beta coefficients for training Data

$$\hat{\beta}_{tr} = (X_{tr}^T X_{tr})^{-1} X_{tr}^T Y_{tr}.$$

4. Compute the estimated response \hat{Y} by multiplying the train data matrix with estimated $\hat{\beta}_{tr}$ coefficients

$$\hat{Y}_{tr} = X_{tr} \hat{\beta}_{tr}.$$

5. Compute RMSE of training data set

$$RMSE_{tr} = \sqrt{\frac{\sum_{i=1}^n (Y_{tr} - \hat{Y}_{tr})^2}{n}}. \quad (6.1)$$

6. Now compute the estimated response of testing data with the help of beta coefficients of training data

$$\hat{Y}_{ts} = X_{ts} \hat{\beta}_{tr}.$$

7. Compute the RMSE of testing data set

$$RMSE_{ts} = \sqrt{\frac{\sum_{i=1}^n (Y_{ts} - \hat{Y}_{ts})^2}{n}}. \quad (6.2)$$

8. Output: This pseudo code gives following results.

$$RMSE_{tr}, RMSE_{ts}$$

The pseudo code of LASSO regression is presented below.

1. First split data the explanatory and response into training and testing data.
2. Input: $X_{tr}, Y_{tr}, X_{ts}, Y_{ts}$, n =sample size, λ = tuning parameter
3. Scale the data X_{tr} and Y_{tr} by using scale function in R.
4. Compute the estimated $\hat{\beta}$ coefficients for training Data.
5. For $\lambda = 1$ to i

If

$$\frac{1}{n}(X_{tr}^T Y_{tr})_j < -\frac{\lambda}{2}$$

then

$$\hat{\beta}_{tr} = \frac{1}{n}(X_{tr}^T Y_{tr})_j + \frac{\lambda}{2}.$$

If

$$\frac{1}{n}|(X_{tr}^T Y_{tr})_j| \leq \frac{\lambda}{2}$$

then

$$\hat{\beta}_{tr} = 0.$$

If

$$\frac{1}{n}(X_{tr}^T Y_{tr})_j > \frac{\lambda}{2}$$

then

$$\hat{\beta}_{tr} = \frac{1}{n}(X_{tr}^T Y_{tr})_j - \frac{\lambda}{2}.$$

End

6. Compute the estimated response \hat{Y} by multiplying the train data matrix with estimated $\hat{\beta}_{tr}$ coefficients

$$\hat{Y}_{tr} = X_{tr}\hat{\beta}_{tr}.$$

7. Compute RMSE of training data set

$$RMSE_{tr} = \sqrt{\frac{\sum_{i=1}^n (Y_{tr} - \hat{Y}_{tr})^2}{n}}. \quad (6.3)$$

8. Now compute the estimated response of testing data with the help of beta coefficients of training data

$$\hat{Y}_{ts} = X_{ts}\hat{\beta}_{tr}.$$

9. Compute RMSE of testing data set

$$RMSE_{ts} = \sqrt{\frac{\sum_{i=1}^n (Y_{ts} - \hat{Y}_{ts})^2}{n}}. \quad (6.4)$$

10. Output: This pseudo code gives following results.

$$RMSE_{tr}, RMSE_{ts}$$

The pseudo code of beta cube regression is presented below.

1. First split data the explanatory and response into training and testing data.
2. Input: $X_{tr}, Y_{tr}, X_{ts}, Y_{ts}$, n =sample size, λ = tuning parameter
3. Scale the data X_{tr} and Y_{tr} by using scale function in R.
4. Compute the estimated $\hat{\beta}$ coefficients for training Data.

- Initially compute the random β vector of $p \times 1$ by using random function in R language and then compute $\hat{\beta}_{tr}$ accordingly

$$\hat{\beta}_{tr} = (2X_{tr}^T X_{tr} + 3\lambda\beta)^{-1} 2X_{tr}^T Y_{tr}.$$

- Compute the estimated response \hat{Y} by multiplying the train data matrix with estimated $\hat{\beta}_{tr}$ coefficients

$$\hat{Y}_{tr} = X_{tr}\hat{\beta}_{tr}.$$

- Compute RMSE of training data set

$$RMSE_{tr} = \sqrt{\frac{\sum_{i=1}^n (Y_{tr} - \hat{Y}_{tr})^2}{n}}. \quad (6.5)$$

- Now compute the estimated response of testing data with the help of beta coefficients of training data

$$\hat{Y}_{ts} = X_{ts}\hat{\beta}_{tr}.$$

- Compute RMSE of testing data set

$$RMSE_{ts} = \sqrt{\frac{\sum_{i=1}^n (Y_{ts} - \hat{Y}_{ts})^2}{n}}. \quad (6.6)$$

- Output: This pseudo code gives following results.

$$RMSE_{tr}, RMSE_{ts}$$

The pseudo code of generalized inverse regression is presented below.

- First split data the explanatory and response into training and testing data.
- Input: $X_{tr}, Y_{tr}, X_{ts}, Y_{ts}$, n =sample size, λ = tuning parameter
- Scale the data X_{tr} and Y_{tr} by using scale function in R.
- Compute the estimated $\hat{\beta}$ coefficients for training Data.

- Initially compute the random β vector of $p \times 1$ by using random function in R language and then compute accordingly $\hat{\beta}_{tr}$

$$\hat{\beta}_{tr} = X_{tr}^{-1}Y_{tr}$$

$$\hat{\beta}_{tr} = \text{ginv}(X_{tr})Y_{tr}$$

$$\hat{\beta}_{tr} = (X_{tr}^T X_{tr})^{-1} X_{tr}^T Y_{tr}.$$

- Compute the estimated response \hat{Y} by multiplying the train data matrix with estimated $\hat{\beta}_{tr}$ coefficients

$$\hat{Y}_{tr} = X_{tr}\beta_{tr}.$$

- Compute RMSE of training data set

$$RMSE_{tr} = \sqrt{\frac{\sum_{i=1}^n (Y_{tr} - \hat{Y}_{tr})^2}{n}}. \quad (6.7)$$

- Now compute the estimated response of testing data with the help of beta coefficients of training data

$$\hat{Y}_{ts} = X_{ts}\hat{\beta}_{tr}.$$

- Compute RMSE of testing data set

$$RMSE_{ts} = \sqrt{\frac{\sum_{i=1}^n (Y_{ts} - \hat{Y}_{ts})^2}{n}}. \quad (6.8)$$

- Output: This pseudo code gives following results.

$$RMSE_{tr}, RMSE_{ts}$$

The pseudo code of compressed LASSO regression is presented below.

- First split data the explanatory and response into training and testing data.

2. Input: $X_{tr}, Y_{tr}, X_{ts}, Y_{ts}$, n =sample size, C_m , $q < n$, λ = tuning parameter
3. Scale the data X_{tr} and Y_{tr} by using scale function in R.
4. Compute the estimated β coefficients for training Data.
5. For $\lambda = 1$ to i

If

$$\frac{1}{n}(X_{tr}^T C_m^T C_m Y_{tr})_j < -\frac{\lambda}{2} \quad (6.9)$$

then

$$\hat{\beta}_{tr} = \frac{1}{n}(X_{tr}^T C_m^T C_m Y_{tr})_j + \frac{\lambda}{2} \quad (6.10)$$

If

$$\frac{1}{n}|(X_{tr}^T C_m^T C_m Y_{tr})_j| \leq \frac{\lambda}{2} \quad (6.11)$$

then

$$\hat{\beta}_{tr} = 0 \quad (6.12)$$

If

$$\frac{1}{n}(X_{tr}^T C_m^T C_m Y_{tr})_j > \frac{\lambda}{2} \quad (6.13)$$

then

$$\hat{\beta}_{tr} = \frac{1}{n}(X_{tr}^T C_m^T C_m Y_{tr})_j - \frac{\lambda}{2} \quad (6.14)$$

End

6. Compute the estimated response \hat{Y} by multiplying the train data matrix with estimated $\hat{\beta}_{tr}$ coefficients

$$\hat{Y}_{tr} = X_{tr} \hat{\beta}_{tr}.$$

7. Compute RMSE of training data set

$$RMSE_{tr} = \sqrt{\frac{\sum_{i=1}^n (Y_{tr} - \hat{Y}_{tr})^2}{n}}. \quad (6.15)$$

8. Now compute the estimated response of testing data with the help of beta coefficients of training data

$$\hat{Y}_{ts} = X_{ts}\hat{\beta}_{tr}.$$

9. Compute RMSE of testing data set

$$RMSE_{ts} = \sqrt{\frac{\sum_{i=1}^n (Y_{ts} - \hat{Y}_{ts})^2}{n}}. \quad (6.16)$$

10. Output: This pseudo code gives following results.

$$RMSE_{tr}, RMSE_{ts}$$

Codes

Codes that are implemented for the data analysis of real data sets are as follows:

```
# Functions

rm(list=ls(all=TRUE))

# OLS Regression
ols.fit<-function(Ytr,Xtr, Yts,Xts){
Y<- as.numeric(as.matrix(Ytr))
X<- as.matrix(Xtr)
nc<- ncol(Xtr)
nr<- nrow(Xtr)
B.tr <- solve(t(Xtr)%*%Xtr) %*%t(Xtr)%*%Ytr
pr.tr <- Xtr%*%B.tr
pr.ts <- Xts%*%B.tr
RMSE.tr <- sqrt(sum(pr.tr-Ytr)^2/length(Ytr))
RMSE.ts <- sqrt(sum(pr.ts-Yts)^2/length(Yts))

res.ols<-list(B.tr=B.tr,opt.RMSE.tr=RMSE.tr, opt.RMSE.ts=RMSE.ts)
res.ols
}
```

```

#LASSO Regression
lasso.fit= function(Ytr,Xtr, Yts,Xts,lambda= seq(from=0.1, to=1, by=0.1)){

Y<- as.numeric(as.matrix(Ytr))
X<- as.matrix(Xtr)
nc<- ncol(Xtr)
nr<- nrow(Xtr)
n= length(Ytr)
B.tr<-matrix(NA,length(lambda),nc)
pr.tr<-matrix(NA,nr, length(lambda))
pr.ts<-matrix(NA,length(Yts), length(lambda))
RMSE.tr<-rep(NA,length(lambda))
RMSE.ts<-rep(NA,length(lambda))

for (i in 1:length(lambda)){
for (j in 1:nc){
if (((t(Xtr)%*%Ytr)[j]/n) < -lambda[i]/2){
B.tr[i,j] =((t(Xtr)%*%Ytr)[j]/n) + (lambda[i]/2)}
if (((t(Xtr)%*%Ytr)[j]/n) <= lambda[i]/2){
B.tr[i,j] = 0 }
if (((t(Xtr)%*%Ytr)[j]/n) > lambda[i]/2){
B.tr[i,j] =((t(Xtr)%*%Ytr)[j]/n) - (lambda[i]/2)}
}
pr.tr[,i]<- Xtr%*%B.tr[i,]
RMSE.tr[i]<- sqrt(sum(pr.tr[,i]-Ytr)^2/length(Ytr))
pr.ts[,i]<- Xts%*%B.tr[i,]
RMSE.ts[i]<- sqrt(sum(pr.ts[,i]-Yts)^2/length(Yts))

}
opt.RMSE.tr<-min(RMSE.tr)
ind.tr<- which.min(RMSE.tr)
opt.lambda<- lambda[ind.tr]
opt.RMSE.ts<-RMSE.ts[ind.tr]

res.lasso<-list( B.tr=B.tr, RMSE.tr=RMSE.tr, RMSE.ts=RMSE.ts,
opt.RMSE.tr=opt.RMSE.tr, opt.RMSE.ts=opt.RMSE.ts, opt.lambda= opt.lambda)

res.lasso
}

# Beta Cube Regression
BetaCubeReg.fit<- function(Ytr,Xtr, Yts,Xts,
lambda= seq(from=0.1, to=1, by=0.1), N=3){

Y<- as.numeric(as.matrix(Ytr))
X<- as.matrix(Xtr)
nc<- ncol(Xtr)
nr<- nrow(Xtr)

```

```

set.seed(42)
Int.B<-diag(rnorm(nc),nc,nc)
B.tr<-array(NA,c(length(lambda),nc,N))
pr.tr<-array(NA,c(length(lambda),nr, N))
RMSE.tr<-matrix(NA,N,length(lambda))
pr.ts<-array(NA,c(length(lambda),length(Yts), N))
RMSE.ts<-matrix(NA,N,length(lambda))

for (i in 1:N){
  if (i==1){
    B=Int.B} else {B<- opt.B}
  for (j in 1:length(lambda)){

    B.tr[j,,i]<-solve(2*t(Xtr)%%Xtr+3*lambda[j]*B ,
    tol =exp(-100))%%(2*t(Xtr)%%Ytr)

    pr.tr<- Xtr)%%B.tr[j,,i]
    RMSE.tr[i,j]<- sqrt(sum(pr.tr-Ytr)^2/length(Ytr))

    pr.ts<- Xts)%%B.tr[j,,i]
    RMSE.ts[i,j]<- sqrt(sum(pr.ts-Yts)^2/length(Yts))
  }
  ind<- which.min(RMSE.tr[i,])
  opt.B<- B.tr[ind,,i]
}
opt.RMSE.tr<-min(RMSE.tr)
ind.tr<- which(RMSE.tr == min(RMSE.tr), arr.ind=TRUE)
opt.lambda<- lambda[ind.tr[2]]
opt.RMSE.ts<-RMSE.ts[ind.tr]

res.BetaCube<-list( B.tr=B.tr,RMSE.tr=RMSE.tr,RMSE.ts=RMSE.ts,
opt.RMSE.tr=opt.RMSE.tr,opt.RMSE.ts=opt.RMSE.ts, opt.lambda= opt.lambda)

res.BetaCube
}

# Generalized Inverse (Moore-Penrose inverse)
library(MASS)
ginv.fit<-function(Ytr,Xtr, Yts,Xts){
Y<- as.numeric(as.matrix(Ytr))
X<- as.matrix(Xtr)
nc<- ncol(Xtr)
nr<- nrow(Xtr)
B.tr <-ginv(t(Xtr)%%Xtr)%%t(Xtr)%%Ytr
pr.tr <- Xtr)%%B.tr
pr.ts <- Xts)%%B.tr
RMSE.tr <- sqrt(sum(pr.tr-Ytr)^2/length(Ytr))
RMSE.ts <- sqrt(sum(pr.ts-Yts)^2/length(Yts))

```

```

res.ginv<-list(B.tr=B.tr,opt.RMSE.tr=RMSE.tr, opt.RMSE.ts=RMSE.ts,
opt.lambda=0)

res.ginv
}

# Partial Least Square (PLS)
PLS<- function(Ytr,Xtr, Yts,Xts,ncomp){
Y<- as.numeric(as.matrix(Ytr))
X<- as.matrix(Xtr)
nc<- ncol(Xtr)
nr<- nrow(Xtr)

W<- P<- matrix(NA, nc, ncomp)
B.tr<- matrix(NA, nc, ncomp)
S<- matrix(NA, nr, ncomp)
Q<- rep(NA, ncomp)
RMSE.tr<-rep(NA, 1,ncomp)
RMSE.ts<-rep(NA, 1,ncomp)

for( a in 1:ncomp){
ws<- t(Xtr)%*%Ytr
w<- ws/norm(ws)
s<- Xtr%*%w
p<- t(Xtr)%*%s / c(t(s)%*%s)
q<- t(Ytr)%*%s /c(t(s)%*%s)

Xtr<- Xtr - s%*% t(p)
ytr<- Ytr - s%*% t(q)

W[,a]<- w
S[,a]<- s
P[,a]<- p
Q[a]<- q
mat<- t(P[,1:a])%*% W[,1:a]

B.tr<- W[,1:a]%*% solve(mat)%*%Q[1:a]
pr.tr<- Xtr%*%B.tr
RMSE.tr[a]<- sqrt(sum(pr.tr-Ytr)^2/length(Ytr))

pr.ts<- Xts%*%B.tr
RMSE.ts[a]<- sqrt(sum(pr.ts-Yts)^2/length(Yts))
}

opt.RMSE.tr<-min(RMSE.tr)
opt.lambda<- which.min(RMSE.tr)
opt.RMSE.ts<-RMSE.ts[opt.lambda]

res.PLS<- list(B.tr=B.tr, RMSE.tr=RMSE.tr, RMSE.ts=RMSE.ts,

```

```

opt.RMSE.tr=opt.RMSE.tr, opt.RMSE.ts=opt.RMSE.ts, opt.lambda= opt.lambda)

res.PLS
}

# Compressed LASSO Regression
cs.lasso.fit<-function(Ytr,Xtr, Yts,Xts,Q = GaussianMatrix(n,q),
lambda= seq(from=0.1, to=1, by=0.1)){

Y<- as.numeric(as.matrix(Ytr))
X<- as.matrix(Xtr)
nc<- ncol(Xtr)
nr<- nrow(Xtr)
n=length(Ytr)
B.tr<-matrix(NA,length(lambda),nc)
pr.tr<-matrix(NA,nr, length(lambda))
pr.ts<-matrix(NA,length(Yts), length(lambda))
RMSE.tr<-rep(NA,length(lambda))
RMSE.ts<-rep(NA,length(lambda))

for (i in 1:length(lambda)){
for (j in 1:nc){
if (((t(Xtr)%*%t(Q)%*%Q)%*%Ytr)[j]/n) < -lambda[i]/2){
B.tr[i,j] =((t(Xtr)%*%t(Q)%*%Q)%*%Ytr)[j]/n) + (lambda[i]/2)}
if (((t(Xtr)%*%t(Q)%*%Q)%*%Ytr)[j]/n) <= lambda[i]/2){
B.tr[i,j] = 0 }
if (((t(Xtr)%*%t(Q)%*%Q)%*%Ytr)[j]/n) > lambda[i]/2){
B.tr[i,j] =((t(Xtr)%*%t(Q)%*%Q)%*%Ytr)[j]/n) - (lambda[i]/2)}
}
pr.tr[,i]<- Xtr%*%B.tr[i,]
RMSE.tr[i]<- sqrt(sum(pr.tr[,i]-Ytr)^2/length(Ytr))
pr.ts[,i]<- Xts%*%B.tr[i,]
RMSE.ts[i]<- sqrt(sum(pr.ts[,i]-Yts)^2/length(Yts))
}
opt.RMSE.tr<-min(RMSE.tr)
ind.tr<- which.min(RMSE.tr)
opt.lambda<- lambda[ind.tr]
opt.RMSE.ts<-RMSE.ts[ind.tr]

res.cs.lasso<-list( B.tr=B.tr,RMSE.tr=RMSE.tr,RMSE.ts=RMSE.ts,
opt.RMSE.tr=opt.RMSE.tr,opt.RMSE.ts=opt.RMSE.ts, opt.lambda= opt.lambda)

res.cs.lasso
}

# Compressed Beta Cube Regression
cs.BetaCubeReg.fit<-function(Ytr,Xtr, Yts,Xts,Q = GaussianMatrix(n,q),

```

```

lambda= seq(from=0.1, to=1, by=0.1), N=3){

Y<- as.numeric(as.matrix(Ytr))
X<- as.matrix(Xtr)
nc<- ncol(Xtr)
nr<- nrow(Xtr)

set.seed(42)
Int.B<-diag(rnorm(nc),nc,nc)
B.tr<-array(NA,c(length(lambda),nc,N))
pr.tr<-array(NA,c(length(lambda),nr, N))
RMSE.tr<-matrix(NA,N,length(lambda))
pr.ts<-array(NA,c(length(lambda),length(Yts), N))
RMSE.ts<-matrix(NA,N,length(lambda))

for (i in 1:N){
if (i==1){
B=Int.B} else {B<- opt.B}
for (j in 1:length(lambda)){
B.tr[j,,i]<-solve(2*t(Xtr)%*%t(Q)%*%Q%*%Xtr+3*lambda[j]*B,
tol = exp(-100))%*%(2*t(Xtr)%*%t(Q)%*%Q%*%Ytr)

pr.tr<- Xtr%*%B.tr[j,,i]
RMSE.tr[i,j]<- sqrt(sum(pr.tr-Ytr)^2/length(Ytr))

pr.ts<- Xts%*%B.tr[j,,i]
RMSE.ts[i,j]<- sqrt(sum(pr.ts-Yts)^2/length(Yts))
}
ind<- which.min(RMSE.tr[i,])
opt.B<- B.tr[ind,,i]
}
opt.RMSE.tr<-min(RMSE.tr)
ind.tr<- which(RMSE.tr == min(RMSE.tr), arr.ind=TRUE)
opt.lambda<- lambda[ind.tr[2]]
opt.RMSE.ts<-RMSE.ts[ind.tr]

res.cs.BetaCube<-list( B.tr=B.tr,RMSE.tr=RMSE.tr,RMSE.ts=RMSE.ts,
opt.RMSE.tr=opt.RMSE.tr, opt.RMSE.ts=opt.RMSE.ts, opt.lambda= opt.lambda)

res.cs.BetaCube
}

# Compressed Generalized Inverse (Moore-Penrose inverse)
library(MASS)
cs.ginv.fit<-function(Ytr,Xtr, Yts,Xts,Q = GaussianMatrix(n,q)){
Y<- as.numeric(as.matrix(Ytr))
X<- as.matrix(Xtr)
nc<- ncol(Xtr)
nr<- nrow(Xtr)

```

```

B.tr<-ginv(t(Xtr)%*%t(Q)%*%Q%*%Xtr)%*%t(Xtr)%*%t(Q)%*%Q%*%Ytr
pr.tr <- Xtr%*%B.tr
pr.ts <- Xts%*%B.tr
RMSE.tr <- sqrt(sum(pr.tr-Ytr)^2/length(Ytr))
RMSE.ts <- sqrt(sum(pr.ts-Yts)^2/length(Yts))

res.cs.ginv<-list(B.tr=B.tr,opt.RMSE.tr=RMSE.tr, opt.RMSE.ts=RMSE.ts,
opt.lambda=0)
res.cs.ginv
}

```