**INTRODUCTION TO ELASTIC-NET REGRESSION AND REGULARISATION**

**by Simon Moss**

|  |
| --- |
| **Introduction** |

In many studies, researchers want to examine how various characteristics or attributes, such as IQ, EQ, and age, are associated with some outcome, such as the motivation of research candidates to study. In these circumstances, researchers tend to utilise a variant of regression—such as linear regression, logistic regression, or generalised linear models—to analyse the data. Nevertheless, in these circumstances, researchers can undertake more reliable and respected alternatives to these traditional models. Specifically

* these reliable alternatives include ridge regression, lasso regression, and elastic-net regression
* these three approaches are similar to each other
* collectively, these approaches are often assigned the label regularization

This document will introduce these methods. Specifically, the document will

* outline the benefits of these methods
* explain the rationale that underpins these methods
* demonstrate how to conduct these analyses, using software called R

**Benefits of regularization**

So, what are the benefits or objectives of ridge regression, lasso regression, and elastic-net regression? Is this document worth reading? In essence, these techniques are especially useful if

* researchers want to construct an equation or formula that predicts outcomes as precisely as possible
* the predictors are moderately or highly correlated with each other, or
* you want to impress your examiners or reviewers with your advanced knowledge—using a technique that demands only a few hours to learn

|  |
| --- |
| **Example** |

**Linear regression**

To introduce you to these methods, consider this example. Suppose you want to predict which research candidates are likely to be especially motivated. To investigate this topic, a researcher administers a survey to 500 research candidates. This survey includes questions that assess

* motivation, such as “On a scale of 1 to 10, how motivated do you feel”
* emotional intelligence or EQ—a measure of the extent to which individuals can decipher the emotions of other people and can readily control their own emotions

The following display presents a subset of the output. This output indicates that

* EQ is positively associated with motivation.
* as the unstandardised B values indicate, Motivation = .52 + .14 x EQ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predictor | Unstandardised B | SE | Standardised B or beta | t |
| Constant | .52 | .14 |  |  |
| EQ | .14 | .02 | .14 | 2.82\* |

\* p < .05, \*\* p < .01

**Rationale of linear regression**

Rather than linear regression, these data could be subjected to ridge regression, lasso regression, and elastic-net regression. To appreciate these alternatives, you need to be attuned to the notion of least squares. To illustrate, consider the following scatterplot. In this scatterplot

* each green circle corresponds to one person
* for example, the bottom left circle corresponds to a person whose EQ is about 1.7 and whose motivation is about 0.5.
* the broken line is a preliminary attempt to represent the relationship between EQ and motivation



So, how can we evaluate whether this dotted line does indeed represent the relationship between EQ and motivation effectively? How can we evaluate this model? One common approach is often called the least squares criterion. In particular

* the computer determines the vertical distance between each circle and the broken line, as represented by the arrows
* for example, the vertical distance between the bottom left circle and the broken line is 4
* the computer then squares these distance, as shown in the calculations
* finally, the computer sums these squared distances
* in this example, the sum of these squared distances is 16 + 4 + 4 + 25 + 0.05 = 49.05

To ascertain whether this number, 49.05, is high or low, we could evaluate other attempts to represent the relationship between EQ and motivation. For example, consider the next scatterplot. In this example

* the squared distances between each circle and the broken line is 0.04 + 4 + 4 + 0.04 + 25 = 33.08
* therefore, this second unbroken line is closer to the circles than is the first unbroken line



In principle, we could then continue to shift the unbroken line until the sum of these squared distances is as low as possible—often called least squares. Fortunately,

* statisticians have developed a method that can be applied to generate this unbroken line immediately; researchers do not have to experiment with thousands of lines
* indeed, linear regression utilizes this method to uncover this line
* this line can then be converted to an equation like **Motivation = .52 + .14 x EQ**—because the first number is the intercept and the second number is the slope or gradient, a principle you might remember from high school

When the research comprises more than one predictor, such as EQ and IQ, the same rationale applies. However, the method the computer uses to calculate the broken line is more complex.

**Limitations of linear regression**

This least squares criterion, utilised by linear regression, seems convincing. But actually, over time, scholars discovered a limitation of this approach. To illustrate, consider the following scatterplot. In this example, the broken line represents the equation that minimised the least squares—the equation that linear regression would uncover. As this scatterplot reveals

* the unbroken line does appear to be close to the green circles
* that is, the unbroken line represents, or least seems to represent, the association between EQ and motivation effectively.



Now, however, imagine that another research candidate arrived, after the linear regression was completed. In particular, as revealed in the following display

* the red circle represents this additional research candidate
* the EQ of this person is about 8.2
* if EQ is 8.2, the unbroken line predicts that motivation should be about 7.8, as represented by the grey arrows
* but the motivation of this research candidate is only about 3



This example illustrates a common problem with linear regression and the least squares criterion. Specifically, in this example

* the unbroken line, derived from the least squares criterion, is close to the data that were utilized to generate this equation accurately, called **low bias**
* but, the unbroken line did not predict the outcome of other cases—that is, individuals whose data were not utilized to generate the equation—accurately, sometimes called **high variance**

A flatter line might have predicted the motivation of this additional candidate better. To illustrate, in the following scatterplot

* the very light grey line represents the original equation, derived from the least squares criterion
* the black unbroken line represents a flatter line
* the flatter line diverges more from the green circles but is closer to the red circle
* in other words, the flatter line does not as accurately represent the original data—the data that was utilized to estimate the equation—but predicts the outcome of additional cases better



Indeed, scholars have shown this pattern is common. A flatter line often predicts the outcome of future cases better. The reason is that

* least squares are often too sensitive to a few outliers—participants whose scores differ appreciably from everyone else, as represented by the asterisk in the previous figure
* these outliers tend to increase the slope artificially
* the equation or line that is derived from least squares, therefore, does not represent the actual relationship between the variables in the population very closely
* therefore, this equation or line does not predict future outcomes—such as the motivation of future research candidates—especially well.

To overcome this problem, we need to generate slightly flatter lines than derived from least squares or linear regression. But, how can we achieve this goal? To what extent should these lines be flatter? Fortunately, we can utilise ridge regression, lasso regression, and elastic-net regression to answer this question.

**Illustration of ridge regression**

To reiterate, linear regression uncovers the line or equation that minimizes least squares. In contrast, ridge regression minimises another criterion. This criterion appears in the following box. This equation might look frightening but, after observing an illustration, is actually straightforward

|  |
| --- |
| Criterion = Sum of squared distances +  x slope2 |

To illustrate how to apply this criterion, consider the following scatterplot. In this scatterplot

* as shown before, sum of least squares = 0.04 + 4 + 4 + 0.04 + 25 = 33.04
* the  in the previous formula is the Greek letter lambda and will be clarified later
*  can be any positive value; at this time, merely assume that = 0.5
* the slope is represented by the red arrows and equals rise over run or 1.8 / 2.00 = 0.9
* therefore, the criterion = 33.04 + 0.5 x 0.92 = 33.445



Hence, in this example, the criterion equals 33.445. Similar to least squares, this value is meaningless, unless contrasted to another example. Therefore, consider the following scatterplot as well. In this example

* sum of least squares = 4 + 4 + 4 + 4 + 9 = 25
* again, assume = 0.5
* the slope equals 1.4 / 2.00 = 0.7
* therefore, the criterion = 25 + 0.5 x 0.72 = 25.25



This second criterion is less than is the first criterion—and is thus preferable. In principle, we could then continue to shift the line until we minimize this criterion. Consequently, the line or equation we finally choose will

* generate a low least squares value—and thus be close to the green circles in this example
* but not generate a slope that is too high; if the slope is too high, such as 5, the criterion will increase appreciably and hence this line will not be selected
* hence, this criterion is designed to penalize models or lines that are too steep.

This technique is designed to unearth lines or equations that are not too sensitive to outliers. For some reason, this goal, to diminish sensitivity to outliers, is called regularisation.

**The effect of **

In the previous example,  was assumed to be 0.5. But what is the effect of this parameter? What would happen if  was larger or smaller. To illustrate, suppose that  was tiny, like 0.00000000001. For the previous scatterplot

* sum of least squares is still 4 + 4 + 4 + 4 + 9 = 25
* the slope still equals 1.4 / 2.00 = 0.7
* therefore, the criterion—equal to the sum of least squares +  x slope2 –now equals 25 + 0.00000000001 x 0.7 = 25.000000000007 or basically 25

As this example illustrates, if  is tiny, the criterion is almost identical to the least squares criterion. Consequently, when  is miniscule

* ridge regression and linear regression apply virtually the same criterion
* so, ridge regression, like a linear regression, will generate steep lines or equations.

As this argument implies, as  increases, the line or equation that ridge regression generates will increasingly diverge from the line or equation that linear regression generates. Specifically, an increase in will typically flatten the line or equation. In addition, this increase in will increase bias but reduce variance, as defined in a previous section.

**How to choose a suitable level of **

The value of  can thus affect the results appreciably. So, what is a suitable level of ? How should you decide the level of to specify? In practice, researchers tend to apply an approach called cross-validation to choose a suitable level of . Roughly speaking

* the computer might start with a random value of and then generate a line or equation from a subset of data, such as 70% of the participants
* the computer will then assess the extent to which the line or equation predicts the outcomes of remaining participants
* the computer will then repeat this procedure and choose the value of that generates the line or equation the predicts the outcomes most accurately

|  |
| --- |
| **Lasso regression** |

The previous section demonstrated how ridge regression uncovered a line or equation that represents the data effectively but is not too steep. In particular, ridge regression distils a line or equation that minimises the sum of least squares +  x slope2. This section illustrates another technique, called lasso regression, that also uncovers a line or equation that represents the data effectively but is not too steep. Indeed, lasso regression is almost identical to ridge regression, besides two differences:

* First, the criterion is slightly different, as the following box reveals
* Specifically, the criterion depends on the absolute value of the slope instead of the square of this slope
* That is, the |…| merely eliminates the negative sign. For example, |-9| equals 9. But |9| also equals 9
* Second, lasso regression will ignore slopes that are tiny—such as less than 0.001.

|  |
| --- |
| Criterion = Sum of squared distances +  x |slope| |

This section will illustrate the application of this formula. This section will also demonstrate why these trivial changes sometimes culminate in consequential benefits.



Again, consider the previous scatterplot. To illustrate, this criterion

* as shown before, sum of least squares = 0.04 + 4 + 4 + 0.04 + 25 = 33.04
* again, assume that = 0.05
* the slope is represented by the red arrows and equals rise over run or 1.8 / 2.00 = 0.9
* the slope is not less than 0.0001 and will thus be included in this calculation
* therefore, the criterion = 33.04 + 0.5 x 0.9 = 33.49.

Similar to ridge regression, the computer would repeat this procedure many times. The computer would then choose the line or equation that minimises this criterion.

**Ridge regression versus lasso regression**

So, what are the benefits of lasso regression over ridge regression? Why did scholars develop two methods that are so exceedingly similar to one another? The following table outlines the differences.

|  |  |
| --- | --- |
| Ridge regression | Lasso regression |
| * Retains predictors that are negligibly associated with the outcome—predictors in which the slope or B value is close to 0 | * Excludes predictors that are negligibly associated with the outcome—predictors in which the slope or B value is close to 0 |
| * Thus, useful when you want to retain most of the predictors, perhaps because you believe most of the predictors are important to your study | * Thus, useful when you want to exclude predictors that are not relevant, perhaps because you want to develop a simpler equation or because you recognize that some of the predictors are not vital to your study |
| * Often retains predictors, even if these predictors are highly related to each other | * Often excludes a predictor that may be related to the outcome but is highly correlated with another predictor |
| * Demands less computer time—an issue only if the amount of data is enormous |  |

|  |
| --- |
| **Elastic net regression** |

The previous sections introduced two similar approaches: ridge regression and lasso regression. Typically

* if the predictors are highly correlated—and if researchers prefer to retain all the predictors for some reason—scholars tend to choose ridge regression
* if the predictors are not too highly correlated, or if researchers would prefer a simpler equation than a very accurate equation, scholars tend to choose lasso regression.

But, scholars cannot always know in advance whether they would like to retain all the predictors. They might not, for example, know whether most of the predictors are likely to be associated with the outcome. In these circumstances, they should instead choose elastic net regression. Elastic net regression combines ridge regression and lasso regression.

**Rationale**

You might recall that ridge regression attempts to minimize the sum of least squares +  x slope2. In contrast, lasso regression attempts to minimize the sum of least squares +  x |slope|. Elastic net regression is the same, but minimises a criterion that combines these alternatives, as delineated in the following box

|  |
| --- |
| Criterion = Sum of squared distances +  x slope2 +  x  x|slope| |

This criterion might, at first glance, look complex. But, actually, this criterion merely

* includes the criterion that corresponds to both ridge regression, slope2, and lasso regression, |slope|
* furthermore, the criterion includes another parameter: ; this parameter merely determines the extent to which the method should prioritize slope2 or |slope|
* again, researchers can utilize cross validation to determine suitable values of  and 

|  |
| --- |
| **Other regression models** |

Thus far, this document has applied ridge regression, lasso regression, and elastic-net regression to one design: a design that entails one predictor and one numeric outcome. The basic rationale, however, can be extended to many other designs.

**Multiple regression**

If the research comprises two predictors, such as IQ and EQ, rather than one predictor, the rationale hardly changes. As the following box shows, the criterion now includes two slopes: one corresponding to each predictor. If the research comprised three predictors, the criterion would include three slopes, and so forth.

|  |
| --- |
| Criterion = Sum of squared distances +  x slope for IQ2 +  x slope for EQ2 |

**Logistic regression**

Whenever the researcher wants to predict a binary outcome, such as whether or not candidates will complete their thesis, scholars often utilise a logistic regression analysis. Researchers can also regularise—that is, diminish the slope—in these circumstances too. Indeed, as the following box reveals, ridge regression needs to be adjusted slightly. That is

* rather than sums of squared differences, logistic regression tends to maximize another measure, called the likelihood
* but, otherwise, the rest of this criterion is the same

|  |
| --- |
| Criterion = Likelihood +  x slope for IQ2 +  x slope for EQ2 |

In short, ridge regression, lasso regression, and elastic net regression, although usually applied to improve linear regression, can be applied to improve other variants of regression as well. The remainder of this document illustrates how to conduct ridge regression, lasso regression, and elastic-net regression. These illustrations revolve around circumstances in which researchers want to predict numerical outcomes. If you want to predict other outcomes, such as binary, counts, or ranks, the code will need to be adjusted—and you might need to seek advice. For an example, see <https://rdrr.io/cran/ridge/man/logisticRidge.html>

|  |
| --- |
| **How to conduct these regularisation methods. Step 1: Install and use R** |

You can use a variety of statistical packages to apply these techniques. This document will show you how to conduct these techniques in software called R. If you have not used R before, you can download and install this software at no cost. To achieve this goal

* visit <https://www.cdu1prdweb1.cdu.edu.au/files/2020-08/Introduction%20to%20R.docx> to download an introduction to R
* read the section called Download R and R studio
* although not essential, you could also skim a few of the other sections of this document to familiarize yourself with R.

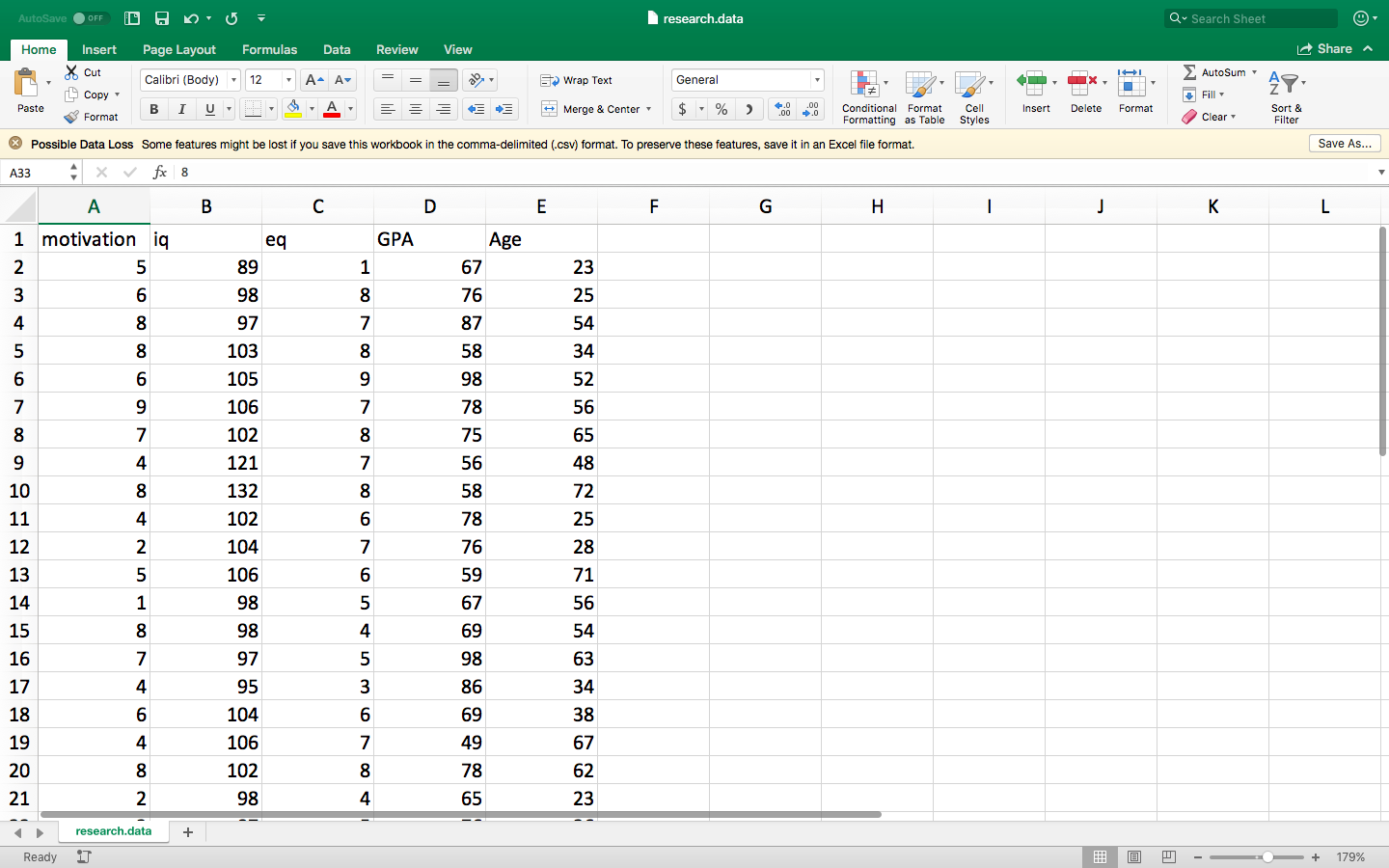
|  |
| --- |
| **How to conduct regularisation methods. Step 2: Upload the data file** |

Your next step is to upload the data into R. To achieve this goal

* open Microsoft Excel
* enter your data into Excel; you might need to copy your data from another format. Or your data might already have been entered into Excel

In particular, as the following example shows

* each column should correspond to one variable
* each row should correspond to one unit—such as one person, one animal, one specimen, and so forth
* the first row labels the variables
* to prevent complications, use labels that comprise only letters—although you could end the label with a number, such as age3.
* actually, unlike this example, you should probably just use lowercase letters to prevent confusion.



To convert this file into a csv file—such as a file called research.data—and then to upload this file into R studio

* visit <https://www.cdu1prdweb1.cdu.edu.au/files/2020-08/Introduction%20to%20R.docx> to download the introduction to R—unless you have already downloaded this document
* read the section called “Upload some data”

|  |
| --- |
| **How to conduct regularisation methods. Step 3: Enter the code and interpret the results** |

To conduct these methods, you need to enter some code. The code might resemble the following display. At first glance, this code looks absolutely terrifying. But actually this code is straightforward once explained.

|  |
| --- |
| install.packages("glmnet")  library(glmnet)  x\_var <- data.matrix(research.data[, c("iq", "eq", "GPA", "Age")])  y\_var <- research.data[, "motivation"]  train\_rows <- sample(1:33, .66\*33)  x.train <- x\_var[train\_rows, ]  x.test <- x\_var[-train\_rows, ]  y.train <- y\_var[train\_rows]  y.test <- y\_var[-train\_rows]  alpha0.fit <- cv.glmnet(x.train, y.train, type.measure="mse",  alpha=0, family="gaussian")  coefficients(alpha0.fit)  alpha0.predicted <-  predict(alpha0.fit, s=alpha0.fit$lambda.1se, newx=x.test)  mean((y.test - alpha0.predicted)^2)  alpha1.fit <- cv.glmnet(x.train, y.train, type.measure="mse", alpha=1, family="gaussian")  coefficients(alpha1.fit)  alpha1.predicted <- predict(alpha1.fit, s=alpha1.fit$lambda.1se, newx=x.test)  mean((y.test - alpha1.predicted)^2)  list.of.fits <- list()  for (i in 0:10) {  fit.name <- paste0("alpha", i/10)  list.of.fits[[fit.name]] <-  cv.glmnet(x.train, y.train, type.measure="mse", alpha=i/10,  family="gaussian")  }  results <- data.frame()  for (i in 0:10) {  fit.name <- paste0("alpha", i/10)    predicted <-  predict(list.of.fits[[fit.name]],  s=list.of.fits[[fit.name]]$lambda.1se, newx=x.test)  mse <- mean((y.test - predicted)^2)  temp <- data.frame(alpha=i/10, mse=mse, fit.name=fit.name)  results <- rbind(results, temp)  }  results |

To enter code, you could write one row at a time, called a command, in the Console. But, if you want to enter code more efficiently,

* in R studio, choose the File menu and then “New File” as well as “R script”
* in the file that opens, paste the code that appears in the left column of the following table
* to execute this code, highlight all the instructions and press the “Run” button—a button that appears at the top of this file

You should not change any of the code in the left column, except perhaps the bold characters. The right column of the following table explains this code. You do not, however, need to understand all the code.

|  |  |
| --- | --- |
| Code to enter | Explanation or clarification |
| install.packages("glmnet")  library(glmnet) | * R comprises many distinct sets of formulas or procedures, each called a package * *glmnet* is one of these packages and can be used to conduct a range of regression models * *install.packages* merely installs this package onto the computer * *library* then activates this package * the quotation marks should perhaps be written in R rather than Word; the reason is that R recognises this simple format— " —but not the more elaborate format that often appears in Word, such as “ or ”. |
| x\_var <- data.matrix(**research.data**[, c(**"iq", "eq", "GPA", "Age")])** | * This code extracts the predictors from your data file—because some of the subsequent code will be applied to your predictors or independent variables only * In this example, four predictors are extracted: iq, eq, GPA, and Age. * You would need to change these labels |
| y\_var <- **research.data[, "motivation"]** | * This code extracts the outcome variable from your data file |
| train\_rows <- sample(1:**33**, .66\***33**)  x.train <- x\_var[train\_rows, ]  x.test <- x\_var[-train\_rows, ]  y.train <- y\_var[train\_rows]  y.test <- y\_var[-train\_rows] | * This code assigns 66% of the data to a training data file—used to construct the equation or model * This code also assigns the remaining data to a testing data file—used to assess the equation or model * You do not have to change any of this code, except the number 33. This number represents the number of rows or participants in your data file |
|  | Conduct ridge regression first |
| alpha0.fit <- cv.glmnet(x.train, y.train, type.measure="mse",  alpha=0, family="gaussian") | * This code subjects the training data to ridge regression. * That is, alpha = 0 refers to ridge regression rather than lasso regression or elastic net regression * The equation is stored in a container called alpha0.fit * From now on, I will refer to this equation as the **model** |
| coefficients(alpha0.fit) | * This code generates the B coefficients in the model called alpha0.fit. * You would report these coefficients in your report.   (Intercept) 5.428571  iq 9.570194  eq 5.190628  GPA -9.285554  Age 2.067744  In this example, the equation or model is  **motivation = 5.42 + 9.5 iq + 5.2 eq + -9.3 GPA + 2.1 Age** |
| alpha0.predicted <-  predict(alpha0.fit, s=alpha0.fit$lambda.1se, newx=x.test) | * This code is designed to test this equation or model * In this example, this code applies the model, alpha0.fit, to the predictors in the testing data—generating the predicted motivation of each participant * For example, if you entered **alpha0.predicted** into the R console, you would receive a series of numbers—each representing the predicted outcome of one participant * The code **s=alpha0.fit$lambda.1se** basically sets lambda to a suitable value—a value that generates minimal cross-validation error while simplifying the model |
| mean((y.test - alpha0.predicted)^2) | * This code generates a number that gauges the accuracy of these predictions * Specifically, this code calculates the mean of the squared differences between the actual and predicted outcomes in the testing data * For example, this code might generate a number like 5.345 * You can then compare the number this model generates to the number other models generate * Ultimately, you would choose the model that generates the smallest mean of squared differences |
|  | Conduct lasso regression |
| alpha1.fit <- cv.glmnet(x.train, y.train, type.measure="mse", alpha=1, family="gaussian")  coefficients(alpha1.fit)  alpha1.predicted <- predict(alpha1.fit, s=alpha1.fit$lambda.1se, newx=x.test)  mean((y.test - alpha1.predicted)^2) | * This code is exactly the same as the code that was utilised to conduct ridge regression * The only difference is that alpha is set to 1 rather than 0 |
|  | Conduct elastic-net regression |
| alpha0.5.fit <- cv.glmnet(x.train, y.train, type.measure="mse", alpha=0.5, family="gaussian")  coefficients(alpha0.5.fit)  alpha0.5.predicted <- predict(alpha0.5.fit, s=alpha0.5.fit$lambda.1se, newx=x.test)  mean((y.test - alpha0.5.predicted)^2) | * This code is exactly the same as the code that was utilised to conduct ridge regression * The only difference is that alpha is set to 0.5 rather than 0 * When alpha is set to 0.5, the model blends ridge regression and lasso regression equally * Of course, you can choose other values of alpha. * Subsequent code will demonstrate how you can identify the optimal value of alpha |
|  | Optimise alpha |
| list.of.fits <- list()  for (i in 0:10) {    fit.name <- paste0("alpha", i/10)  list.of.fits[[fit.name]] <-  cv.glmnet(x.train, y.train, type.measure="mse", alpha=i/10,  family="gaussian")  }  results <- data.frame()  for (i in 0:10) {  fit.name <- paste0("alpha", i/10)  predicted <-  predict(list.of.fits[[fit.name]],  s=list.of.fits[[fit.name]]$lambda.1se, newx=x.test)  mse <- mean((y.test - predicted)^2)  temp <- data.frame(alpha=i/10, mse=mse, fit.name=fit.name)  results <- rbind(results, temp)  }  results | * You can execute this code without changing any of the characters * You do not need to understand this code at all * In essence, this code experiments with a range of alpha values—from 0 to 1. * The code will then generate the mean square error that corresponds to each alpha—that is, the average of the squared differences between the actual and predicted outcomes * You merely need to choose the alpha that generates the lowest mean square error, such as 07 for example * Finally, simply repeat the previous code to conduct elastic-net regression, but change alpha = 0.5 to another value, such as 0.7. |