Assignment6\_Okan

Aysenur Okan

2024-03-05

knitr::opts\_chunk$set(echo = TRUE)  
options(repos = c(CRAN = "https://cloud.r-project.org"))  
install.packages("leaps")

##   
## The downloaded binary packages are in  
## /var/folders/\_v/3cy8vcps2738xrj0l2k96lkh0000gq/T//Rtmp0KpaJ7/downloaded\_packages

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.0 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(dplyr)  
install.packages('leaps')

##   
## The downloaded binary packages are in  
## /var/folders/\_v/3cy8vcps2738xrj0l2k96lkh0000gq/T//Rtmp0KpaJ7/downloaded\_packages

library(leaps)  
library(glmnet) #for regularization

## Loading required package: Matrix  
##   
## Attaching package: 'Matrix'  
##   
## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack  
##   
## Loaded glmnet 4.1-8

library(caret) #tuning the elastic net

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(Matrix)  
  
  
setwd("/Users/aysenur/Documents/GitHub/Machine\_Learning")  
  
tedf <- read\_csv("tedata.csv")

## Rows: 100 Columns: 7  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## dbl (7): Balance, Income, Limit, Rating, Cards, Age, Education  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

trdf <- read\_csv("trdata.csv")

## Rows: 300 Columns: 7  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## dbl (7): Balance, Income, Limit, Rating, Cards, Age, Education  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

# Regression

reg1 <- lm(Rating ~ ., trdf)  
summary(reg1)

##   
## Call:  
## lm(formula = Rating ~ ., data = trdf)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -23.6103 -6.8054 -0.3134 6.6066 24.6752   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 30.093008 4.009464 7.505 7.39e-13 \*\*\*  
## Balance 0.008787 0.003573 2.459 0.0145 \*   
## Income 0.083445 0.039809 2.096 0.0369 \*   
## Limit 0.064218 0.001041 61.673 < 2e-16 \*\*\*  
## Cards 5.137488 0.460654 11.153 < 2e-16 \*\*\*  
## Age 0.021938 0.034675 0.633 0.5274   
## Education -0.285735 0.192145 -1.487 0.1381   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 10.22 on 293 degrees of freedom  
## Multiple R-squared: 0.9958, Adjusted R-squared: 0.9957   
## F-statistic: 1.164e+04 on 6 and 293 DF, p-value: < 2.2e-16

pr1 <- predict(reg1,newdata=tedf)  
mse1 <- mean(tedf$Rating-pr1)^2 #Mean square error  
r2 <- cor(tedf$Rating, pr1)^2 #Variance explained  
mse1

## [1] 0.5190247

r2

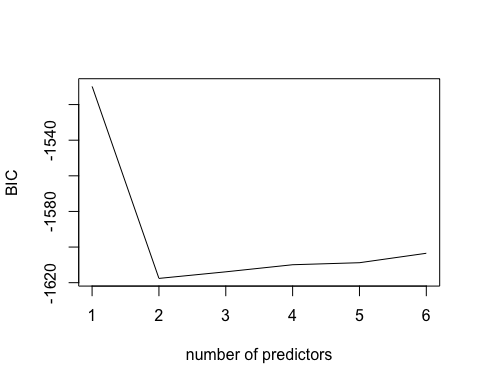
## [1] 0.9953172

# Forward Selection in Regression

forwardreg <- regsubsets(Rating~.,data=trdf,nvmax=6,method='forward')  
(forwardreg\_summary <- summary(forwardreg))

## Subset selection object  
## Call: regsubsets.formula(Rating ~ ., data = trdf, nvmax = 6, method = "forward")  
## 6 Variables (and intercept)  
## Forced in Forced out  
## Balance FALSE FALSE  
## Income FALSE FALSE  
## Limit FALSE FALSE  
## Cards FALSE FALSE  
## Age FALSE FALSE  
## Education FALSE FALSE  
## 1 subsets of each size up to 6  
## Selection Algorithm: forward  
## Balance Income Limit Cards Age Education  
## 1 ( 1 ) " " " " "\*" " " " " " "   
## 2 ( 1 ) " " " " "\*" "\*" " " " "   
## 3 ( 1 ) " " " " "\*" "\*" " " "\*"   
## 4 ( 1 ) "\*" " " "\*" "\*" " " "\*"   
## 5 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 6 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"

plot(forwardreg\_summary$bic,type='l', xlab='number of predictors',ylab='BIC') #plot relationship between BIC and # of predictors



which(forwardreg\_summary$bic == min(forwardreg\_summary$bic))

## [1] 2

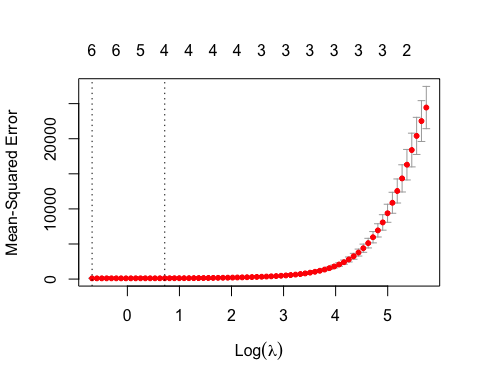
coef(forwardreg,2) #final coefficients

## (Intercept) Limit Cards   
## 23.25672267 0.06675044 5.32684583

# Regularized Regression

Lambda is the regularization parameter that controls the amount of shrinkage applied to the coefficients. In glmnet, as lambda increases, more coefficients are pushed towards zero, leading to simpler models. A very high value of lambda can lead to underfitting, while a very low value can lead to overfitting. Alpha parameter balances the type of regularization applied between L1 (Lasso, alpha = 1) and L2 (Ridge, alpha = 0). An elastic net regularization is a combination of both, and by adjusting alpha, you can control the mix of L1 and L2 regularization. An alpha close to 1 puts more emphasis on Lasso, promoting sparsity (more coefficients set to exactly zero), while an alpha closer to 0 favors Ridge, which tends to distribute the penalty among more coefficients. I use an elastic net, which combines both approaches.

set.seed(202)  
  
# Prepare the training data  
x2 <- model.matrix(~., data=trdf[, -which(names(trdf) == "Rating")]) # Remove Rating column for predictors  
y <- trdf$Rating # Response vector  
  
# Standardize the predictors (note: glmnet will do this by default, but doing manually for clarity)  
x2 <- scale(x2)  
x2 <- x2[,-1] #delete the intercept  
# For test data (only if needed for prediction or validation later)  
# I want to standardize using training data means and sd, which is why I'm not doing it here directly.  
# x4 and y2 are prepared but not used in cv.glmnet below. They could be used for prediction after model selection.  
x4 <- model.matrix(~., data=tedf[, -which(names(tedf) == "Rating")])  
y2 <- tedf$Rating  
x4 <- x4[,-1]  
  
  
# 10-fold cross-validation for tuning parameter lambda with Elastic Net  
i33 <- cv.glmnet(x = x2, y = y, alpha = 0.5) # alpha = 0.5 for elastic net  
  
# Plot the result  
plot(i33)



# Extract the lambda value that gives the minimum mean cross-validated error  
lambda\_min <- i33$lambda.min  
print(lambda\_min)

## [1] 0.5078007

# Extract the lambda value at which the cross-validated error is within 1 standard error of the minimum  
lambda\_1se <- i33$lambda.1se  
print(lambda\_1se)

## [1] 2.05

# Coefficients at lambda.1se  
coef\_1se <- coef(i33, s = "lambda.1se")  
print(coef\_1se)

## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 350.940000  
## Balance 11.558102  
## Income 8.546338  
## Limit 137.140683  
## Cards 5.247286  
## Age .   
## Education .

# Predict on test data using the lambda.1se from the cross-validation  
pelnet = predict(i33, newx = x4, s = "lambda.1se")  
# Calculate the Mean Squared Error (MSE) between the predictions and actual values  
mseelnet = mean((pelnet - y2) ^ 2)  
print(mseelnet)

## [1] 557290207263

# Calculate the R-squared value  
r2elnet = cor(pelnet, y2) ^ 2  
print(r2elnet)

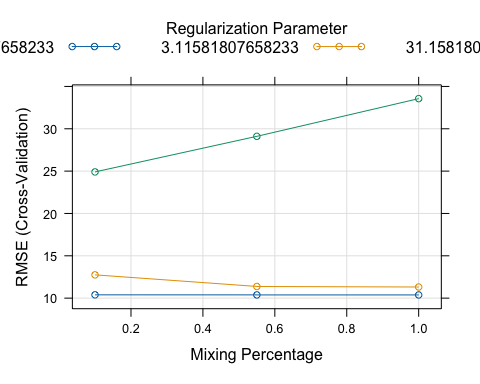
## [,1]  
## lambda.1se 0.9938716

# Tuning the Elastic Net

Performance metric (e.g., RMSE for regression tasks) on the y-axis and the log of lambda on the x-axis. Each line represents a different alpha value if the training explored multiple alphas. For each curve, the point with the lowest metric (highest accuracy or lowest RMSE/error) indicates the best-performing lambda value for that alpha. The optimal alpha is the one that results in the best (lowest) cross-validated performance metric.

By observing how the performance metric changes with lambda, I can gauge the sensitivity of your model to regularization. A steep curve suggests that regularization has a significant impact, potentially indicating a model that’s very sensitive to overfitting without regularization.

# Set up control for cross-validation  
ctrl1 <- trainControl(method = 'cv', number = 5)  
# Train the glmnet model  
trglmnet <- train(x = x2, y = y, method = "glmnet", trControl = ctrl1)  
  
# Plot the training result  
plot(trglmnet)



# Retrain the model with the best-tuned parameters on the full training set  
gl1 <- glmnet(x = x2, y = y, alpha = trglmnet$bestTune$alpha, lambda = trglmnet$bestTune$lambda)  
  
# Display coefficients of the tuned model  
coef(gl1)

## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) 350.94000000  
## Balance 3.97615207  
## Income 2.77499309  
## Limit 149.90317796  
## Cards 6.42556846  
## Age 0.09550243  
## Education -0.56277756

# Predict on test data using the tuned model parameters  
elnet\_tuned <- predict(gl1, newx = x4, s = trglmnet$bestTune$lambda)  
  
# Calculate the Mean Squared Error (MSE) for the tuned model  
(msetunedelnet <- mean((elnet\_tuned - y2) ^ 2))

## [1] 656265767639

# Calculate the R-squared value for the tuned model  
(r2tunedelnet <- cor(elnet\_tuned, y2) ^ 2)

## [,1]  
## s1 0.9938938