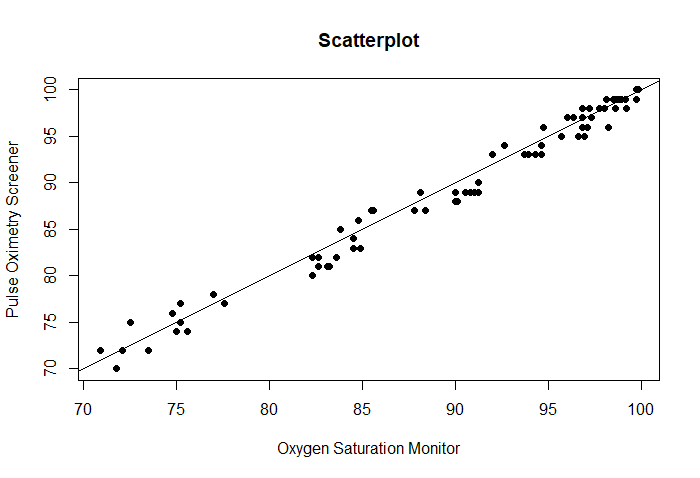
STAT 6340 Statistical and Machin Learning

Project 4

Buddhika Jayawardana

**Section 1**

**Problem 1**

1. Two populations are in perfect agreement, when they are identical. Then and and vice versa. From the Pearson's correlation coefficient, we have

where when and are identical. Thus .

1. Since and

Thus

Therefore

We know that the correlation between two population satisfy

Hence we have

1. Assume . Then

Since all these parts are positive, each of them should equal to zero separately.

This implies and

Now assume and . Then by substituting, we get

From part b) this imply perfect agreement.

Where are population means, are population standard deviations and is covariance of the two populations.

With this function CCC = 0.9892748

1. With my bootstrap function, following results were obtained.

CCC\_original CCC\_my bias std\_error

0.9892748 0.9889337 -0.0003411283 0.0002304046

Confident Interval computed using Standard Error.

lower\_bound\_SE upper\_bound\_SE

0.9887033 0.9891641

Confident Interval computed using percentile method.

lower\_bound\_Perc

47.5% 0.9889895

upper\_bound\_Perc

52.5% 0.9892421

1. With the boot package, results are

Bootstrap Statistics :

original bias std. error

t1\* 0.9892748 -0.0003717729 0.002032808

and the Confident Interval using percentile method,

Intervals :

Level Percentile

95% ( 0.9841, 0.9923 )

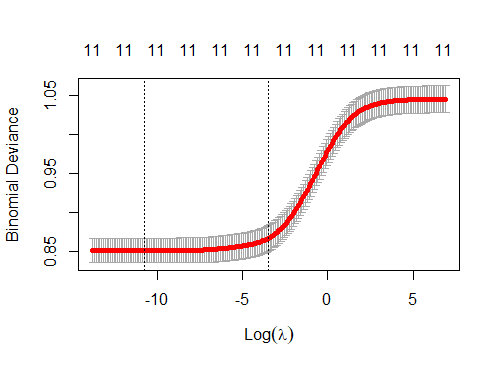
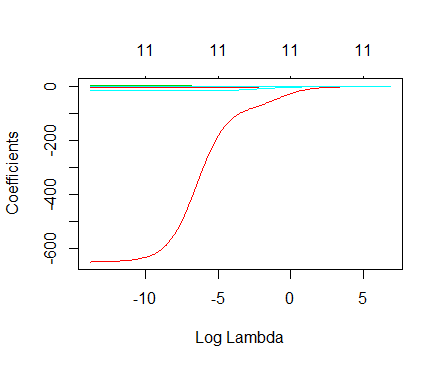
Calculations and Intervals on Original Scale

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bias | Standard Error | Lower bound for CI |
| My code | -0.0003411283 | 0.0002304046 | 0.9889895 |
| bootstrap | -0.0003717729 | 0.002032808 | 0.9841 |

1. Results shows the two methods gives nearly the same Bias and CI except for the standard Error. Thus both these methods can be used interchangeably in practice but with the low standard error, my code gives slightly better results.

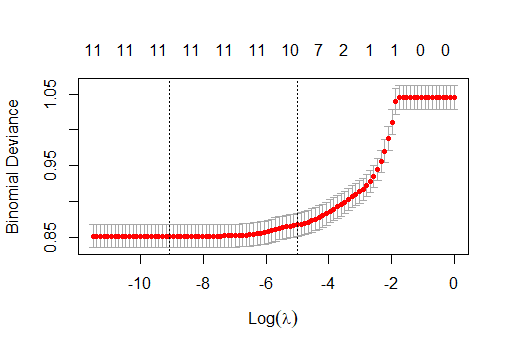
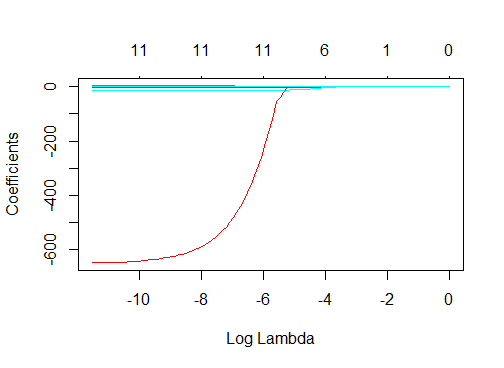
**Problem 2**

1. Test error rate = 0.2000789
2. Test error rate = 0.1990572
3. Test error rate = 0.1990572
4. Test error rate = 0.1994666



Best 2.049075e-05

Test error rate = 0.1976317



Best 0.0001149757

Test error rate = 0.1978359

Full Best Subset Backward Forward Ridge Lasso

X.Intercept. 6.361576e+02 7.443217e+02 7.443217e+02 641.2029336 6.195350e+02 6.066693e+02

fixed.acidity 5.521067e-01 6.301870e-01 6.301870e-01 0.5546123 5.391177e-01 5.262977e-01

volatile.acidity -3.784880e+00 -3.686958e+00 -3.686958e+00 -3.8031154 -3.781939e+00 -3.767113e+00

citric.acid -7.377814e-01 -6.927095e-01 -6.927095e-01 -0.7431531 -7.391644e-01 -7.252818e-01

residual.sugar 2.951950e-01 3.324403e-01 3.324403e-01 0.2967673 2.889508e-01 2.837514e-01

chlorides -1.263994e+01 -1.238154e+01 -1.238154e+01 -12.6752942 -1.276662e+01 -1.273346e+01

free.sulfur.dioxide 8.644758e-03 8.176799e-03 8.176799e-03 0.0083011 8.748570e-03 8.690595e-03

total.sulfur.dioxide -2.696022e-04 NA NA NA -3.514132e-04 -3.507052e-04

density -6.590763e+02 -7.683960e+02 -7.683960e+02 -664.1842221 -6.422076e+02 -6.290951e+02

pH 3.342979e+00 3.682339e+00 3.682339e+00 3.3514899 3.286873e+00 3.231570e+00

sulphates 2.167765e+00 2.306562e+00 2.306562e+00 2.1707966 2.144495e+00 2.118287e+00

alcohol 1.423452e-01 NA NA 0.1384844 1.602747e-01 1.732598e-01

AIC 4.167223e+03 4.164778e+03 4.164778e+03 4165.2553390 NA NA

Test.Error 2.000789e-01 1.990572e-01 1.990572e-01 0.1994666 1.976317e-01 1.978359e-01

From reduced models, Best subset selection method and Backward method gives the same model and their Test error rates are low compared to the full model. Out of all the models compared, Ridge regression method gives the lowest Test error rate. Thus the preferred model is with the Ridge regression.

All the models build in the previous project for wine data set have larger test error rates. Thus the best model is still with the Ridge regression.

**Section 1**

# problem 1

library(boot) # for bootstrap

library(gdata) # for resample

data\_set <- read.delim("oxygen\_saturation.txt")

attach(data\_set)

n <- dim(data\_set)[1]

# problem 1 a)

plot(osm, pos, main="Scatterplot",

xlab="Oxygen Saturation Monitor",

ylab="Pulse Oximetry Screener", pch=19)

abline(0, 1)

# problem 1 d)

# function to compute concordance correlation coefficient (CCC)

CCC.fn <- function(data, indices) {

x <- data[indices,1]

y <- data[indices,2]

mu1 <- mean(x)

mu2 <- mean(y)

sigma1 <- var(x)

sigma2 <- var(y)

sigma12 <- cov(x, y)

result <- 2\*sigma12/((mu1-mu2)^2+sigma1+sigma2)

return(result)

}

CCC.fn(data\_set, 1:n) # original CCC

# [1] 0.9892748

# my own code for the bootstrap method

my\_boot.fn <- function(data, n, B) {

para <- c()

i = 1

while(i<=B) {

sample <- data[resample(1:n, n, replace = TRUE, prob = NULL),]

para[i] <- CCC.fn(sample, 1:n) # CCC of the sample

i <- i+1

}

CCC\_original <- CCC.fn(data\_set, 1:n) # original CCC

CCC\_my <- mean(para) # mean of the CCC from samples

bias <- CCC\_my - CCC\_original

#computation of the CI from standard error

std\_error<-sd(para)/sqrt(n) # standard error

lower\_bound\_SE <- mean(para)-std\_error

upper\_bound\_SE <- mean(para)+std\_error

#computation of the 95% CI of the mean

perc <- quantile(para, probs = c(0.475, 0.525))

lower\_bound\_Perc <- perc[1]

upper\_bound\_Perc <- perc[2]

return(list(data.frame(CCC\_original, CCC\_my, bias, std\_error),

"Standerd Error CI" = data.frame(lower\_bound\_SE, upper\_bound\_SE),

"Percentile method CI" = data.frame(lower\_bound\_Perc),

"Percentile method CI" = data.frame(upper\_bound\_Perc)))

}

set.seed(1)

my\_boot.fn(data\_set, n, 1000)

# CCC\_original CCC\_my bias std\_error

# 1 0.9892748 0.9889337 -0.0003411283 0.0002304046

#

# $`Standerd Error CI`

# lower\_bound\_SE upper\_bound\_SE

# 1 0.9887033 0.9891641

#

# $`Percentile method CI`

# lower\_bound\_Perc

# 47.5% 0.9889895

#

# $`Percentile method CI`

# upper\_bound\_Perc

# 52.5% 0.9892421

# problem 1 f)

set.seed(1)

CCC.boot <- boot(data\_set, CCC.fn, R = 1000) # bootstrap estimates of bias and standard error for CCC

# ORDINARY NONPARAMETRIC BOOTSTRAP

#

#

# Call:

# boot(data = data\_set, statistic = CCC.fn, R = 1000)

#

#

# Bootstrap Statistics :

# original bias std. error

# t1\* 0.9892748 -0.0003717729 0.002032808

boot.ci(CCC.boot, conf = 0.95, type = "perc") # 95% lower confidence bound for CCC

# BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

# Based on 1000 bootstrap replicates

#

# CALL :

# boot.ci(boot.out = CCC.boot, type = "perc")

#

# Intervals :

# Level Percentile

# 95% ( 0.9841, 0.9923 )

# Calculations and Intervals on Original Scale

# problem 2

library(caret) # for cross-validation

library(leaps) # for best-subset selection

# library(bestglm) # for

# library(MASS)

library(glmnet) # for Ridge and lasso Regression

wine <- read.csv("winequality-white.csv", header = T, sep=';')

wine$quality <- ifelse(wine$quality >= 7, 1, 0)

wine$quality <- as.factor(wine$quality)

attach(wine)

str(wine)

# problem 2 a)

# Using caret package to fit glm and calculate test error rate with 10-fold cross-validation

set.seed(1234)

fit.full.GLM.CARET <- train(quality ~ . ,

data = wine,

method ="glm",

trControl = trainControl(method = "cv", number = 10))

summary(fit.full.GLM.CARET)

# Call:

# NULL

#

# Deviance Residuals:

# Min 1Q Median 3Q Max

# -2.1436 -0.6725 -0.4114 -0.1798 2.8331

#

# Coefficients:

# Estimate Std. Error z value Pr(>|z|)

# (Intercept) 6.362e+02 9.412e+01 6.759 1.39e-11 \*\*\*

# fixed.acidity 5.521e-01 9.053e-02 6.099 1.07e-09 \*\*\*

# volatile.acidity -3.785e+00 4.885e-01 -7.749 9.28e-15 \*\*\*

# citric.acid -7.378e-01 4.010e-01 -1.840 0.065776 .

# residual.sugar 2.952e-01 3.564e-02 8.283 < 2e-16 \*\*\*

# chlorides -1.264e+01 3.816e+00 -3.312 0.000926 \*\*\*

# free.sulfur.dioxide 8.645e-03 3.130e-03 2.762 0.005749 \*\*

# total.sulfur.dioxide -2.696e-04 1.506e-03 -0.179 0.857936

# density -6.591e+02 9.540e+01 -6.909 4.89e-12 \*\*\*

# pH 3.343e+00 4.268e-01 7.832 4.81e-15 \*\*\*

# sulphates 2.168e+00 3.475e-01 6.238 4.42e-10 \*\*\*

# alcohol 1.423e-01 1.139e-01 1.250 0.211334

# ---

# Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

#

# (Dispersion parameter for binomial family taken to be 1)

#

# Null deviance: 5116.8 on 4897 degrees of freedom

# Residual deviance: 4143.2 on 4886 degrees of freedom

# AIC: 4167.2

#

# Number of Fisher Scoring iterations: 6

summary(fit.full.GLM.CARET)$coefficients[,1]

# (Intercept) fixed.acidity volatile.acidity citric.acid

# 6.361576e+02 5.521067e-01 -3.784880e+00 -7.377814e-01

# residual.sugar chlorides free.sulfur.dioxide total.sulfur.dioxide

# 2.951950e-01 -1.263994e+01 8.644758e-03 -2.696022e-04

# density pH sulphates alcohol

# -6.590763e+02 3.342979e+00 2.167765e+00 1.423452e-01

Test.full <- 1 - fit.full.GLM.CARET$results$Accuracy # Test error rate

# [1] 0.2000789

pred <- predict(fit.full.GLM.CARET, wine, type = 'raw')

mean(quality != pred) # training error rate

# [1] 0.1976317

# problem 2 b)

X <- wine[,1:11]

y <- wine[,12]

Xy <- cbind(as.data.frame(X), y)

res.best.logistic <- bestglm(Xy, family = binomial, # binomial family for logistic

IC = "AIC", # Information criteria

method = "exhaustive")

## Show top 5 models

res.best.logistic$BestModels

# fixed.acidity volatile.acidity citric.acid residual.sugar chlorides free.sulfur.dioxide

# 1 TRUE TRUE TRUE TRUE TRUE TRUE

# 2 TRUE TRUE TRUE TRUE TRUE TRUE

# 3 TRUE TRUE FALSE TRUE TRUE TRUE

# 4 TRUE TRUE TRUE TRUE TRUE TRUE

# 5 TRUE TRUE FALSE TRUE TRUE TRUE

# total.sulfur.dioxide density pH sulphates alcohol Criterion

# 1 FALSE TRUE TRUE TRUE FALSE 4162.778

# 2 FALSE TRUE TRUE TRUE TRUE 4163.255

# 3 FALSE TRUE TRUE TRUE FALSE 4163.877

# 4 TRUE TRUE TRUE TRUE FALSE 4164.774

# 5 FALSE TRUE TRUE TRUE TRUE 4164.778

## Show result for the best model: Same model was chosen

summary(res.best.logistic$BestModel)

# Call:

# glm(formula = y ~ ., family = family, data = Xi, weights = weights)

#

# Deviance Residuals:

# Min 1Q Median 3Q Max

# -2.3552 -0.6766 -0.4103 -0.1794 2.8148

#

# Coefficients:

# Estimate Std. Error z value Pr(>|z|)

# (Intercept) 7.443e+02 3.464e+01 21.490 < 2e-16 \*\*\*

# fixed.acidity 6.302e-01 6.587e-02 9.567 < 2e-16 \*\*\*

# volatile.acidity -3.687e+00 4.671e-01 -7.893 2.95e-15 \*\*\*

# citric.acid -6.927e-01 3.972e-01 -1.744 0.081151 .

# residual.sugar 3.324e-01 1.932e-02 17.205 < 2e-16 \*\*\*

# chlorides -1.238e+01 3.800e+00 -3.259 0.001119 \*\*

# free.sulfur.dioxide 8.177e-03 2.470e-03 3.310 0.000932 \*\*\*

# density -7.684e+02 3.574e+01 -21.501 < 2e-16 \*\*\*

# pH 3.682e+00 3.313e-01 11.115 < 2e-16 \*\*\*

# sulphates 2.307e+00 3.296e-01 6.999 2.58e-12 \*\*\*

# ---

# Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

#

# (Dispersion parameter for binomial family taken to be 1)

#

# Null deviance: 5116.8 on 4897 degrees of freedom

# Residual deviance: 4144.8 on 4888 degrees of freedom

# AIC: 4164.8

#

# Number of Fisher Scoring iterations: 6

res.best.logistic$BestModel$coefficients

# (Intercept) fixed.acidity volatile.acidity citric.acid

# 7.443217e+02 6.301870e-01 -3.686958e+00 -6.927095e-01

# residual.sugar chlorides free.sulfur.dioxide density

# 3.324403e-01 -1.238154e+01 8.176799e-03 -7.683960e+02

# pH sulphates

# 3.682339e+00 2.306562e+00

res.best.logistic$BestModel$aic

# [1] 4164.778

set.seed(1234)

fit.bestset.GLM.CARET <- train(quality ~ . - total.sulfur.dioxide - alcohol,

data = wine,

method ="glm",

trControl = trainControl(method = "cv", number = 10))

summary(fit.bestset.GLM.CARET)

Test.best <- 1 - fit.bestset.GLM.CARET$results$Accuracy # Test error rate

# [1] 0.1990572

pred <- round(predict(res.best.logistic$BestModel, newdata=X, type='response'))

mean(quality != pred) # Traning error rate

# [1] 0.1984483

# problem 2 c)

full <- glm(quality ~ ., family = binomial, data = wine)

backwards <- step(full)

# Step: AIC=4164.78

# b<-stepAIC(full)

# formula(b)

# # Step: AIC=4164.78

backwards$coefficients

# (Intercept) fixed.acidity volatile.acidity citric.acid

# 7.443217e+02 6.301870e-01 -3.686958e+00 -6.927095e-01

# residual.sugar chlorides free.sulfur.dioxide density

# 3.324403e-01 -1.238154e+01 8.176799e-03 -7.683960e+02

# pH sulphates

# 3.682339e+00 2.306562e+00

set.seed(1234)

# 10 fold crossvalidation

fit.backward.GLM.CARET <- train(formula(backwards),

data = wine,

method ="glm",

trControl = trainControl(method = "cv", number = 10))

Test.backward <- 1 - fit.backward.GLM.CARET$results$Accuracy # Test error rate

# [1] 0.1990572

pred <- predict(fit.backward.GLM.CARET, wine, type='raw')

mean(quality != pred) # training error rate

# [1] 0.1984483

# problem 2 d)

null.model <- glm(quality ~ 1, family = binomial, data = wine)

forwards <- step(null.model,scope=list(lower=formula(null.model),upper=formula(full)),

direction="forward")

# Step: AIC=4165.26

f <- stepAIC(null.model,scope=list(lower=formula(null.model),upper=formula(full)),

direction="forward")

formula(f)

# Step: AIC=4165.26

forwards$coefficients

# (Intercept) alcohol volatile.acidity chlorides

# 641.2029336 0.1384844 -3.8031154 -12.6752942

# residual.sugar pH density sulphates

# 0.2967673 3.3514899 -664.1842221 2.1707966

# fixed.acidity free.sulfur.dioxide citric.acid

# 0.5546123 0.0083011 -0.7431531

set.seed(1234)

# 10 fold crossvalidation

fit.forward.GLM.CARET <- train(formula(forwards),

data = wine,

method ="glm",

trControl = trainControl(method = "cv", number = 10))

Test.forward <- 1 - fit.forward.GLM.CARET$results$Accuracy # Test error rate

# [1] 0.1994666

pred <- predict(fit.forward.GLM.CARET, wine, type='raw')

mean(quality != pred) # Training error rate

# [1] 0.1982442

# problem 2 e)

# Create response vector and the design matrix (without the first column of 1s)

y <- as.numeric(levels(quality))[quality]

x <- model.matrix(quality ~ ., wine)[, -1]

grid <- 10^seq(3, -6, length = 200)

# Fit ridge regression for each lambda on the grid -->

out <- glmnet(x, y, alpha = 0, lambda = grid, family = "binomial")

plot(out, xvar = "lambda")

# 10 fold cross-validation

set.seed(1)

cv.out <- cv.glmnet(x, y, alpha = 0, lambda = grid, family = "binomial")

plot(cv.out)

# Find the best value of lambda

bestlam <- cv.out$lambda.min

# [1] 2.049075e-05

# log(bestlam) = -10.79554

coef.ridge <- predict(out, type = "coefficients", s = bestlam)[1:12, ]

# (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar

# 6.195350e+02 5.391177e-01 -3.781939e+00 -7.391644e-01 2.889508e-01

# chlorides free.sulfur.dioxide total.sulfur.dioxide density pH

# -1.276662e+01 8.748570e-03 -3.514132e-04 -6.422076e+02 3.286873e+00

# sulphates alcohol

# 2.144495e+00 1.602747e-01

ridge.pred <- round(predict(out, s = bestlam, newx = x, type='response'))

Test.ridge <- mean((ridge.pred - y)^2)

# [1] 0.1976317

# problem 2 f)

grid <- 10^seq(0, -5, length = 100)

# Fit lasso regression for each lambda on the grid -->

out <- glmnet(x, y, alpha = 1, lambda = grid, family = "binomial")

plot(out, xvar = "lambda")

# 10 fold cross-validation

set.seed(1)

cv.out <- cv.glmnet(x, y, alpha = 1, lambda = grid, family = "binomial")

plot(cv.out)

# Find the best value of lambda

bestlam <- cv.out$lambda.min

# [1] 0.0001149757

# log(bestlam) = -9.07079

coef.lasso <- predict(out, type = "coefficients", s = bestlam)[1:12, ]

# (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar

# 6.066693e+02 5.262977e-01 -3.767113e+00 -7.252818e-01 2.837514e-01

# chlorides free.sulfur.dioxide total.sulfur.dioxide density pH

# -1.273346e+01 8.690595e-03 -3.507052e-04 -6.290951e+02 3.231570e+00

# sulphates alcohol

# 2.118287e+00 1.732598e-01

lasso.pred <- round(predict(out, s = bestlam, newx = x, type='response'))

Test.lasso <- mean((lasso.pred - y)^2)

# [1] 0.1978359

# problem 2 g)

table <- data.frame(t(rbind.fill(data.frame(t(summary(fit.full.GLM.CARET)$coefficients[,1]),

"AIC"=summary(fit.full.GLM.CARET)$aic, "Test Error"=Test.full),

data.frame(t(res.best.logistic$BestModel$coefficients),

"AIC"=res.best.logistic$BestModel$aic, "Test Error"=Test.best),

data.frame(t(backwards$coefficients),

"AIC"=backwards$aic, "Test Error"=Test.backward),

data.frame(t(forwards$coefficients),

"AIC"=forwards$aic, "Test Error"=Test.forward),

data.frame(t(coef.ridge), "Test Error"=Test.ridge),

data.frame(t(coef.lasso), "Test Error"=Test.lasso))))

colnames(table) <- c("Full", "Best Subset", "Backward", "Forward", "Ridge", "Lasso")

table