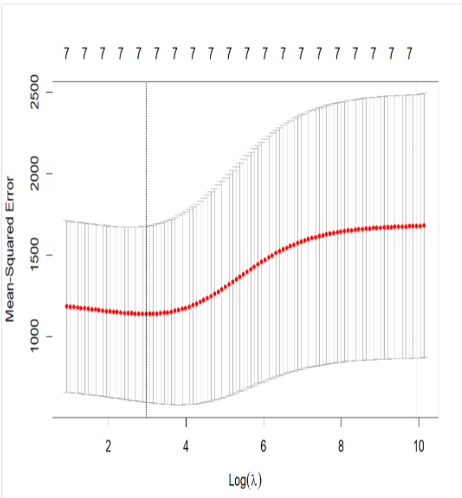
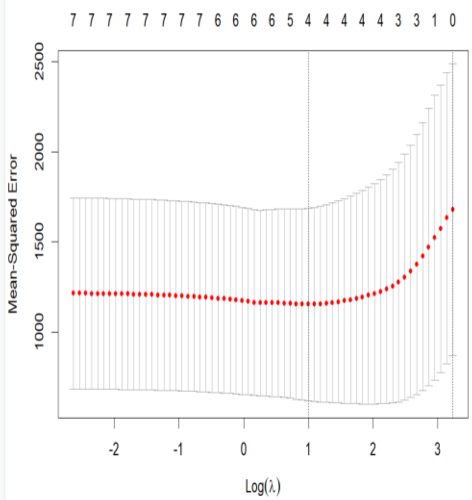
Ashish Mani Acharya STAT6307 Project 4 3/30/2022

Q1 The data has 97 rows and 8 columns. we got rid ot patient ID column as it was unnnecseay . And we treated vesinv as a factor as it was qualitative(categorical variable) We used Leave One Out Cross Validation or LOOCV method to calculate test MSE for this data.




Ridge Regression Lasso

(1a)linear regression test MSE= 1218.358

(1b) 1084.374

(1c) 1084.374

(1d) 1084.374

(1e) 1142.836

(1f) 1167.067

(1g)

| | MODEL (A) | MODEL (B) | MODEL (C) | MODEL (D) | MODEL (E) | MODEL (F) |
|---|---|---|---|---|---|---|
| (Intercept) | -15.242640 | -44.184900 | -44.184900 | -44.184900 | -24.007134 | -25.337922 |
| cancervol | 2.032250 | 2.249600 | 2.249600 | 2.249600 | 1.251679 | 1.916864 |
| weight | 0.011320 | | | | 0.013582 | 0.000000 |
| age | -0.537210 | | | | -0.233510 | 0.000000 |
| benpros | 1.298310 | | | | 0.461475 | 0.000000 |
| vesinv1 | 19.609570 | 21.880800 | 21.880800 | 21.880800 | 15.461505 | 15.237621 |
| capspen | 1.098770 | | | | 1.562614 | 0.938860 |
| gleason | 7.059220 | 6.898200 | 6.898200 | 6.898200 | 6.580142 | 4.398533 |
| test MSE | 1218.358 | 1084.374 | 1084.374 | 1084.374 | 1142.836 | 1167.067 |

As we can clearly see from the data that Model B , C and D performed better than other model. Hence I would prefer subset selection and backward and forward selection to other models that we tried like linear regression ( Model A) and Ridge Regression and LASSO which are model E and F respectively.

Ashish Mani Acharya STAT6307 Project 4 3/30/2022

Q2) The data has 1000 rows and 21 columns . for simplicity datas even some quantitative ones have been used as a factor. We already have done exploratory analysis of this data in our previous project of project 3.

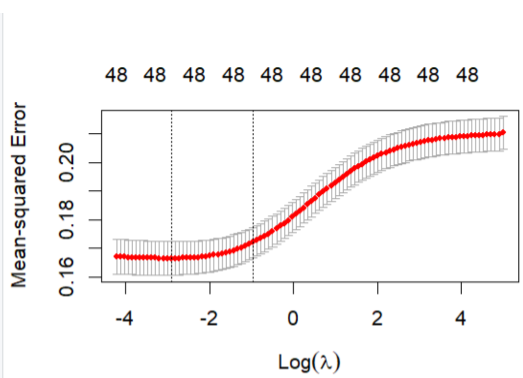
Q2a) 0.249

Q2b) 0.240000

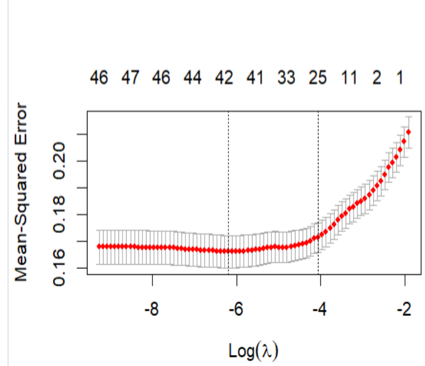
Q2c)0.240000
Q2d)0.240000

Q2e) 0.166466

Q2f) 0.166169




RidgeRegression




Lasso

Q2g)

| | MODEL (B) | MODEL (C) | MODEL (D) | | MODEL (A) | MODEL (E ) | MODEL (F) |
|---|---|---|---|---|---|---|---|
| (Intercept) | 1.750000 | 1.750000 | 1.750000 | (Intercept) | 0.400500 | 1.440543 | 1.313688 |
| checkingstatus1A12 | -0.390000 | -0.390000 | -0.390000 | checkingstatus1A12 | -0.374900 | -0.046045 | 0.000000 |
| checkingstatus1A13 | -1.024000 | -1.024000 | -1.024000 | checkingstatus1A13 | -0.965700 | -0.150145 | 0.000000 |
| checkingstatus1A14 | -1.718000 | -1.718000 | -1.718000 | checkingstatus1A14 | -1.712000 | -0.233259 | -0.182462 |
| duration | 0.025680 | 0.025680 | 0.025680 | duration | 0.027860 | 0.004513 | 0.003124 |
| historyA31 | -0.118800 | -0.118800 | -0.118800 | historyA31 | -0.143400 | -0.109649 | 0.000000 |
| historyA32 | -0.830300 | -0.830300 | -0.830300 | historyA32 | -0.586100 | -0.032774 | 0.000000 |
| historyA33 | -0.909700 | -0.909700 | -0.909700 | historyA33 | -0.853200 | -0.073370 | 0.000000 |
| historyA34 | -1.492000 | -1.492000 | -1.492000 | historyA34 | -1.436000 | -0.136059 | -0.024208 |
| purposeA41 | -1.607000 | -1.607000 | -1.607000 | purposeA41 | -1.666000 | -0.183499 | 0.000000 |
| purposeA410 | -1.435000 | -1.435000 | -1.435000 | purposeA410 | -1.489000 | -0.177925 | 0.000000 |
| purposeA42 | -0.740500 | -0.740500 | -0.740500 | purposeA42 | -0.791600 | -0.078741 | 0.000000 |
| purposeA43 | -0.919500 | -0.919500 | -0.919500 | purposeA43 | -0.891600 | -0.100315 | 0.000000 |
| purposeA44 | -0.525100 | -0.525100 | -0.525100 | purposeA44 | -0.522800 | -0.039195 | 0.000000 |
| purposeA45 | -0.142400 | -0.142400 | -0.142400 | purposeA45 | -0.216400 | -0.001133 | 0.000000 |
| purposeA46 | 0.143600 | 0.143600 | 0.143600 | purposeA46 | 0.036280 | 0.042436 | 0.000000 |
| purposeA48 | -2.164000 | -2.164000 | -2.164000 | purposeA48 | -2.059000 | -0.187927 | 0.000000 |
| purposeA49 | -0.782700 | -0.782700 | -0.782700 | purposeA49 | -0.740100 | -0.062162 | 0.000000 |
| amount | 0.000129 | 0.000129 | 0.000129 | amount | 0.000128 | 0.000016 | 0.000000 |
| savingsA62 | -0.328200 | -0.328200 | -0.328200 | savingsA62 | -0.357700 | -0.039544 | 0.000000 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| savingsA63 | -0.430400 | -0.430400 | -0.430400 | savingsA63 | -0.376100 | -0.079461 | 0.000000 |
| savingsA64 | -1.289000 | -1.289000 | -1.289000 | savingsA64 | -1.339000 | -0.139715 | 0.000000 |
| savingsA65 | -0.962800 | -0.962800 | -0.962800 | savingsA65 | -0.946700 | -0.116745 | 0.000000 |
| installment | 0.329900 | 0.329900 | 0.329900 | employA72 | -0.066910 | 0.035283 | 0.000000 |
| statusA92 | -0.287200 | -0.287200 | -0.287200 | employA73 | -0.182800 | 0.006761 | 0.000000 |
| statusA93 | -0.822800 | -0.822800 | -0.822800 | employA74 | -0.831000 | -0.077383 | 0.000000 |
| statusA94 | -0.416900 | -0.416900 | -0.416900 | employA75 | -0.276600 | -0.017210 | 0.000000 |
| othersA102 | 0.487400 | 0.487400 | 0.487400 | installment | 0.330100 | 0.038147 | 0.000000 |
| othersA103 | -1.040000 | -1.040000 | -1.040000 | statusA92 | -0.275500 | -0.001100 | 0.000000 |
| age | -0.013090 | -0.013090 | -0.013090 | statusA93 | -0.816100 | -0.070329 | 0.000000 |
| otherplansA142 | -0.078640 | -0.078640 | -0.078640 | statusA94 | -0.367100 | -0.032860 | 0.000000 |
| otherplansA143 | -0.699500 | -0.699500 | -0.699500 | othersA102 | 0.436000 | 0.071643 | 0.000000 |
| housingA152 | -0.441500 | -0.441500 | -0.441500 | othersA103 | -0.978600 | -0.145853 | 0.000000 |
| housingA153 | -0.149700 | -0.149700 | -0.149700 | residence | 0.004776 | 0.001032 | 0.000000 |
| teleA192 | -0.279400 | -0.279400 | -0.279400 | propertyA122 | 0.281400 | 0.035969 | 0.000000 |
| foreignA202 | -1.382000 | -1.382000 | -1.382000 | propertyA123 | 0.194500 | 0.026868 | 0.000000 |
| | | | | propertyA124 | 0.730400 | 0.085303 | 0.000000 |
| Misclassification Error | 0.240000 | 0.240000 | 0.240000 | age | -0.014540 | -0.001591 | 0.000000 |
| | | | | otherplansA142 | -0.123200 | 0.004791 | 0.000000 |
| | | | | otherplansA143 | -0.646300 | -0.074857 | 0.000000 |
| | | | | housingA152 | -0.443600 | -0.061099 | 0.000000 |
| | | | | housingA153 | -0.683900 | -0.062714 | 0.000000 |
| | | | | cards | 0.272100 | 0.030593 | 0.000000 |
| | | | | jobA172 | 0.536100 | 0.004268 | 0.000000 |
| | | | | jobA173 | 0.554700 | 0.017107 | 0.000000 |
| | | | | jobA174 | 0.479500 | 0.012689 | 0.000000 |
| | | | | liable | 0.264700 | 0.028450 | 0.000000 |
| | | | | teleA192 | -0.300000 | -0.041228 | 0.000000 |
| | | | | foreignA202 | -1.392000 | -0.123863 | 0.000000 |
| | | | | | | | |
| | | | | Misclassification Error | 0.249000 | 0.166466 | 0.166169 |

Since it had the least misclassification error rate I would chose model E and F i.e. Ridge regression and LOOCV respectively. In other words models using Elastic Net methods seem better than other models. Regarding the comparison with the problem I did in project 3; various other factors than simple misclassification rate came into play while making recommendations like AUC, sensitivity and specificity and of course misclassification rate. However since

Ashish Mani Acharya STAT6307 Project 4 3/30/2022

this question was not asking for anything else , here my decision is mainly based on misclassification rate unlike project 3.

```
################################section 2################################

######for Q1######

######installing required libraries for this question #
install.packages("bestglm")
library(caret)
library(ggplot2)
library(lattice)
library(boot)
library(leaps)
library(car)
library(glmnet)
library(bestglm)

pc <- read.csv("C:/Users/alexk/OneDrive/Desktop/stat 6340/mini project 4/prostate_cancer.csv",header = TRUE)
View(pc)
#using abbrevaetion for prostate cancer as pc#

###exploratory analysis of the data###

dim(pc)
head(pc)
sum(is.na(pc))
pc = pc[,-1] #removing unncessary patientID variable#
```

Ashish Mani Acharya STAT6307 Project 4 3/30/2022

```
str(pc)

pc$vesinv = as.factor(pc$vesinv) # vesinv is categorical variable#

summary(pc)


######1a#######

control = trainControl(method = "LOOCV") #using leave one out cross validation method#

a1_fit = train(psa~., data = pc, trControl = control, method = "glm", family = gaussian())

summary(a1_fit)

a1_fit


######1b######



b1_fit_output = regsubsets(psa ~ ., data = pc, nbest = 1, nvmax = NULL, method = "exhaustive")



output_summary = summary(b1_fit_output)

c1 = as.data.frame(output_summary$which)

c2 = as.data.frame(output_summary$adjr2)

c = cbind(c1, c2)

c

# the best model is the model that contains i.e. pcvol, vesinv and gleason, as per above models #



pc2 = pc[ ,c(1,2,6,8)] # as psa,pcvol,vesinv and gleason are significant variables#

control = trainControl(method = "LOOCV") #using LOOCV method#

bcd_fit = train(psa ~ ., data = pc2, trControl = control, method = "glm", family = gaussian())

summary(bcd_fit)

bcd_fit


######1c####################
```

Ashish Mani Acharya STAT6307 Project 4 3/30/2022

```
c1_fit_output = regsubsets(psa ~ ., data = pc, nbest = 1, nvmax = NULL, method = "forward")


output_summary = summary(c1_fit_output)

c1 = as.data.frame(output_summary$which)

c2 = as.data.frame(output_summary$adjr2)

c = cbind(c1, c2)




######################1d##########################

fit.1d.out = regsubsets(psa ~ ., data = pc, nbest = 1, nvmax = NULL, method = "backward")


output_summary = summary(fit.1d.out)

c1 = as.data.frame(output_summary$which)

c2 = as.data.frame(output_summary$adjr2)

c = cbind(c1, c2)

# the best model is the model that contains i.e. pcvol, vesinv and gleason, as per above calculations #


#######1e###############


y = pc$psa

#creating model matrix#

x = model.matrix(psa ~ ., pc)[, -1]


### as we know by dim(pc) command we have 97 rows #####

#RidgeRegression#

CrossValidation_RidgeRegression = cv.glmnet(x, y, alpha = 0, type.measure = "mse", nfolds = 97)




plot(CrossValidation_RidgeRegression)
```

```
CrossValidation_RidgeRegression$lambda.min

RidgeRegression_fit = glmnet(x, y, alpha = 0, lambda = CrossValidation_RidgeRegression$lambda.min, thresh = 1e-8)

coef(RidgeRegression_fit)  # getting the model coefficient for the ridgeregression#

#######1f################

#LASSO#
CrossValidation_LASSO = cv.glmnet(x, y, alpha = 1, type.measure = "mse", nfolds = 97)

plot(CrossValidation_LASSO)

CrossValidation_LASSO$lambda.min

LASSO_fit = glmnet(x, y, alpha = 1, lambda = CrossValidation_RidgeRegression$lambda.min, thresh = 1e-8)

coef(LASSO_fit)

grid.pr = 10^seq(2, 0, length = 3)
grid = 10^seq(2, 0, length = 20)
# finding s values for the grid
#97 as we got 97 rows through dim(pc)command#
RidgeRegression_MSE = matrix(NA, nrow = 20, ncol = 97)
LASSO_MSE = matrix(NA, nrow = 20, ncol = 97)

#using glmnet to make predictions#
```

```
for (j in 1:20)

{

  for (i in 1:97)

  {

    train_y = pc$psa[-i]

    train_x = model.matrix(psa ~ ., pc[-i,])[, -1]

    test_y = pc$psa[i]

    test_x = model.matrix(psa ~ ., pc[i,])[, -1]


    Ridge.Reg = glmnet(train_x, train_y, alpha = 0, lambda = grid.pr, thresh = 1e-8)

    LASSO.Reg = glmnet(train_x, train_y, alpha = 1, lambda = grid.pr, thresh = 1e-8)



    RidgeRegression_prediction = predict(Ridge.Reg, s = grid[j], newx = test_x)

    LASSO_prediction = predict(LASSO.Reg, s = grid[j], newx = test_x)


    RidgeRegression_MSE[j,i] = (RidgeRegression_prediction - test_y)^2

    LASSO_MSE[j,i] = (LASSO_prediction - test_y)^2

  }

}


for (j in 1:20)

{

  mean(RidgeRegression_MSE.s[j, ])

  mean(LASSO_MSE.s[j, ])

}


RidgeRegression_MSE.s = rowMeans(RidgeRegression_MSE, na.rm = TRUE)

LASSO_MSE.s = rowMeans(LASSO_MSE, na.rm = TRUE)


which.min(RidgeRegression_MSE.s)
```

```
min(RidgeRegression_MSE.s)

s.Ridge = grid[7]


which.min(LASSO_MSE.s)

min(LASSO_MSE.s)

s.LASSO = grid[16]


RidgeRegression_fit = glmnet(x, y, alpha = 0, lambda = s.Ridge, thresh = 1e-8)

fit.LASSO = glmnet(x, y, alpha = 1, lambda = s.LASSO, thresh = 1e-8)


#finding coefficient of Ridge Regression and LASSO #

coef(RidgeRegression_fit)

coef(fit.LASSO)


#####################################for Q2########################################


gc<- read.csv("C:/Users/alexk/OneDrive/Desktop/stat 6340/mini project 4/germancredit.csv", header=TRUE)

View(gc)

#for ease of typing german gc has been shortened to gc#


#libraries needed for the solutions #

library(caret)

library(glmnet)

library(MASS)

library(bestglm)
```

Ashish Mani Acharya STAT6307 Project 4 3/30/2022

```
#exploratory analysis of the data#

View(gc)

dim(gc)

head(gc)


# factoring Categorical Variables #

gc$Default = as.factor(gc$Default)

gc$checkingstatus1 = as.factor(gc$checkingstatus1)

gc$history = as.factor(gc$history)

gc$purpose = as.factor(gc$purpose)

gc$savings = as.factor(gc$savings)

gc$employ = as.factor(gc$employ)

gc$status = as.factor(gc$status)

gc$others = as.factor(gc$others)

gc$property = as.factor(gc$property)

gc$otherplans = as.factor(gc$otherplans)

gc$housing = as.factor(gc$housing)

gc$job = as.factor(gc$job)

gc$tele = as.factor(gc$tele)

gc$foreign = as.factor(gc$foreign)

gc$liable = as.factor(gc$liable)


#################################2a##################################

control = trainControl(method = "LOOCV") #using LOOCV method#

b1_fit = train(Default ~ ., data = gc, trControl = control, method = "glm", family = binomial())

summary(b1_fit)

b1_fit


#################################2c##################################

#forward progression #
```

```
null_fit = glm(Default ~ 1, family = binomial, data = gc)
full_fit = glm(Default ~ ., family = binomial, data = gc)

forward_model = stepAIC(null_fit, scope = list(upper = full_fit, lower = null_fit),
          direction = "forward", trace = FALSE)
summary(forward_model)
forward_model$anova

################################2d#################################
#backward progression #
backward_model = stepAIC(full_fit, scope = list(upper = full_fit, lower = null_fit), direction = "backward", trace = FALSE)
backward_model$anova
summary(backward_model)

gc2 = gc[ ,-c(8,12,13,17,18,19)]
#only significant variables are included#
control = trainControl(method = "LOOCV")
fit = train(Default ~ ., data = gc2, trControl = control, method = "glm", family = binomial())
summary(fit)
fit
################################2e#################################

y = as.numeric(gc$Default)
x = model.matrix(Default ~ ., gc)[, -1]

#using Cross Validation with Ridge Regression#
#using 1000 as we got 1000 rows from dim(gc) command #

CrossValidarion_RidgeRegression = cv.glmnet(x, y, alpha = 0, type.measure = "deviance", nfolds = 1000)
```

```
plot(CrossValidarion_RidgeRegression)

CrossValidarion_RidgeRegression$lambda.min # 0.05435123


RidgeRegression_fit = glmnet(x, y, alpha = 0, lambda = CrossValidarion_RidgeRegression$lambda.min, thresh = 1e-8)

#gettng coefficients for Ridge Regression #

coef(RidgeRegression_fit)


#################################2f##################################


CrossValidation_LASSO = cv.glmnet(x, y, alpha = 1, type.measure = "mse", nfolds = 1000)

plot(CrossValidation_LASSO)

CrossValidation_LASSO$lambda.min

LASSO_fit = glmnet(x, y, alpha = 1, lambda = CrossValidation_RidgeRegression$lambda.min, thresh = 1e-8)

#gettng coefficients for LASSO #

coef(LASSO_fit)


#using glmnet to make predictions #

RidgeRegression_MSE = numeric(1000);

LASSO_MSE = numeric(1000)

grid = 10^seq(1, -4, length = 5)

for (i in 1:1000)

{

  train_y = as.numeric(gc$Default)[-i]

  train_x = model.matrix(Default ~ ., gc[-i,])[, -1]

  test_y = as.numeric(gc$Default)[i]

  test_x = model.matrix(Default ~ ., gc[i,])[, -1]


  Ridge.Reg = glmnet(train_x, train_y, alpha = 0, lambda = grid, thresh = 1e-8)

  LASSO.Reg = glmnet(train_x, train_y, alpha = 1, lambda = grid, thresh = 1e-8)


  RidgeRegression_prediction = predict(Ridge.Reg, s = cv.RR$lambda.min, newx = test_x)
```

```
  LASSO_prediction = predict(LASSO.Reg, s = cv.LASSO$lambda.min, newx = test_x)


  RidgeRegression_MSE[i] = (RidgeRegression_prediction - test_y)^2

  LASSO_MSE[i] = (LASSO_prediction - test_y)^2

}

mean(RidgeRegression_MSE)

mean(LASSO_MSE)
```