Assignment 6 - Model Selection

Olivia Samples

7/13/2020

Homework

For the homework, we will use a dataset from https://www.kaggle.com/c/titanic/data. Using this dataset from the Titanic shipwreck you will predict who will survive and who will not. The data set is available on canvas with the name -train.csv" and -test.csv".

Dataset description

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships. One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class. In this challenge, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply neural network to predict which passengers survived the tragedy.

VARIABLE DESCRIPTIONS: 1. Survival Survival (0 = No; 1 = Yes) 2. pclass Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd) 3. name Name 4. sex Sex 5. age Age 6. sibsp Number of Siblings/Spouses Aboard 7. parch Number of Parents/Children Aboard 8. ticket Ticket Number 9. fare Passenger Fare 10. cabin Cabin 11. embarked Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

SPECIAL NOTES:

Pclass is a proxy for socio-economic status (SES) 1st = Upper; 2nd = Middle; 3rd = Lower Age is in Years; Fractional if Age less than One (1) If the Age is Estimated, it is in the form xx.5 With respect to the family relation variables (i.e. sibsp and parch) some relations were ignored. The following are the definitions used for sibsp and parch.

Sibling: Brother, Sister, Stepbrother, or Stepsister of Passenger Aboard Titanic Spouse: Husband or Wife of Passenger Aboard Titanic (Mistresses and FiancesvIgnored) Parent: Mother or Father of Passenger Aboard Titanic Child: Son, Daughter, Stepson, or Stepdaughter of Passenger Aboard Titanic Other family relatives excluded from this study include cousins, nephews/nieces, aunts/uncles, and in-laws. Some children travelled only with a nanny, therefore parch=0 for them. As well, some travelled with very close friends or neighbors in a village, however, the de_nitions do not support such relations.

Your tasks

The training and the test data are provided in canvas, look for tit-train-f.csv and tit-test-f.csv. These are cleaned data. Use tit-train-f.csv data for model selection, applying the selection methods, please report R codes and the outcome with your interpretation. Remember that the response variable is binary outcome, consider what model should be used for model selection.

Ridge regression (10 points) Lasso regression (10 points) Elastic Net (10 points) Principal component regression (10 points)

For this question, you will focus on selected predictors after lasso model selection, re-train the model using different machine learning methods, report the R code and the model performance using tit-test-f.csv, and your interpretation

Random forest (20 points) Logistic regression (20 points) Support vector machine (20 points)

Solution

```
rm(list=ls())
library(plyr)
library(rpart)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(caTools)
library(stringr)
library(Hmisc)
```

```
## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:plyr':
##
##     is.discrete, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(ggplot2)
library(vcd)
```

```
## Loading required package: grid
```

```
library(ROCR)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(VIM)
```

```
## Loading required package: colorspace
##
## Attaching package: 'colorspace'
## The following object is masked from 'package:pROC':
##
##     coords
## VIM is ready to use.
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
##
## Attaching package: 'VIM'
## The following object is masked from 'package:datasets':
##
##     sleep
```

```
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 4.0-2
```

Read the csv files. These are previously cleaned data.

```
train <- read.csv(file='tit-train-f.csv', head=TRUE, sep=",")
test <- read.csv(file='tit-test-f.csv', head=TRUE, sep=",")
```

We will ensure that everything is a number except for our factored target variable, "Survived".

```
coltype = function(data){
    sapply(data, class)
}
coltype(train)
```

```
## PassengerId    Survived      Pclass         Sex         Age       SibSp
##   "integer"    "factor"   "integer"   "integer"   "numeric"   "integer"
##       Parch      Ticket        Fare       Cabin    Embarked        name
##   "integer"   "integer"   "numeric"   "integer"   "integer"   "integer"
##       title
##   "integer"
```

Now we have to split the training data into two sets of its own: the training data and the test data.

```
set.seed(101)
t1 <- sample(1:nrow(train), nrow(train)*0.8)
Survived= train[t1,2]
train.xy <- data.frame(train[t1,c(1, 3:13)], Survived)
test.x <- subset(train[-t1,], select = -Survived)
test.y <- train[-t1,]$Survived
```

Now we will scale the numeric features

```r
cols = c(2:6, 8:10, 11)

pre_proc_val <- preProcess(train.xy[,cols], method = c("center", "scale"))
train.xy[,cols] = predict(pre_proc_val, train.xy[,cols])
test.x[,cols] = predict(pre_proc_val, test.x[,cols])

summary(train.xy)
```

```
##   PassengerId        Pclass              Sex                Age
##  Min.   :  1.0   Min.   :-1.5067   Min.   :-1.3501   Min.   :-1.37714
##  1st Qu.:219.8   1st Qu.:-1.5067   1st Qu.:-1.3501   1st Qu.:-0.92019
##  Median :447.5   Median : 0.8536   Median : 0.7396   Median : 0.05082
##  Mean   :447.2   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.:670.2   3rd Qu.: 0.8536   3rd Qu.: 0.7396   3rd Qu.: 0.62201
##  Max.   :891.0   Max.   : 0.8536   Max.   : 0.7396   Max.   : 3.19234
##      SibSp             Parch             Ticket              Fare
##  Min.   :-0.4717   Min.   :-0.4724   Min.   :      0   Min.   :-0.63269
##  1st Qu.:-0.4717   1st Qu.:-0.4724   1st Qu.:  17421   1st Qu.:-0.48052
##  Median :-0.4717   Median :-0.4724   Median : 113052   Median :-0.35427
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 294648   Mean   : 0.00000
##  3rd Qu.: 0.4561   3rd Qu.:-0.4724   3rd Qu.: 347082   3rd Qu.:-0.03216
##  Max.   : 6.9507   Max.   : 6.9204   Max.   :3101312   Max.   : 9.20482
##      Cabin            Embarked            name              title
##  Min.   :-0.5685   Min.   :-3.5383   Min.   :-1.5216   Min.   : 1.000
##  1st Qu.:-0.5685   1st Qu.: 0.2246   1st Qu.:-0.8646   1st Qu.: 1.000
##  Median :-0.5685   Median : 0.2246   Median :-0.1149   Median : 1.000
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 1.947
##  3rd Qu.:-0.5685   3rd Qu.: 0.2246   3rd Qu.: 0.8353   3rd Qu.: 3.000
##  Max.   : 3.9794   Max.   : 2.1061   Max.   : 1.9038   Max.   :17.000
##  Survived
##  NO :431
##  YES:281
##
##
##
##
```

We will now compare performances of Ridge Regression, Lasso Regression, and Elastic Net.

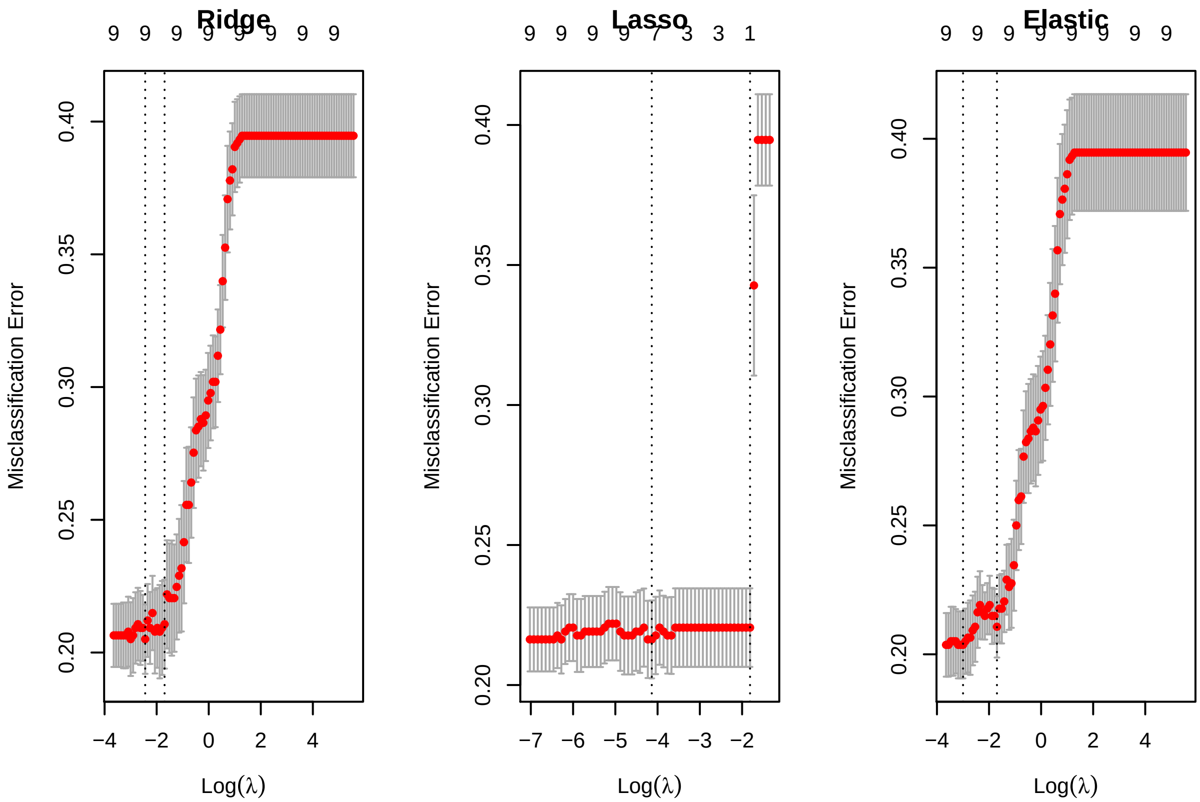
```r
x = data.matrix(train.xy[,cols])
y = Survived

set.seed(356)
# 10 fold cross validation
cvfit.ridge = cv.glmnet(x, y,
                  family = "binomial",
                  alpha = 0,
                  type.measure = "class")

cvfit.lasso = cv.glmnet(x, y,
                  family = "binomial",
                  alpha = 1,
                  type.measure = "class")
cvfit.elastic = cv.glmnet(x,y,
                      family = 'binomial',
```

```
                          alpha = seq(0, 1, length.out = 11),
                          type.measure = 'class')
```

```
## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used
```

```
## Warning in if (alpha > 1) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (alpha < 0) {: the condition has length > 1 and only the first
## element will be used
```

```r
par(mfrow=c(1,3))
plot(cvfit.ridge, main = "Ridge")
plot(cvfit.lasso, main = "Lasso")
plot(cvfit.elastic, main = "Elastic")
```



As you can see, it looks like Lasso has the smallest lambda value.

```r
cvfit.ridge$lambda.min
```

```
## [1] 0.08716399
```

```r
cvfit.lasso$lambda.min
```

```
## [1] 0.01595745
```

```r
cvfit.elastic$lambda.min
```

```
## [1] 0.04987844
```

We will compare the accuracy of the models on the training set.

```r
# Ridge Model
# Prediction on training set
PredTrain.R = predict(cvfit.ridge, newx=x, type="class")
confusionMatrix(Survived, factor(PredTrain.R))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  NO YES
##        NO  402  29
##        YES 116 165
##
##                Accuracy : 0.7963
##                  95% CI : (0.7649, 0.8254)
##     No Information Rate : 0.7275
##     P-Value [Acc > NIR] : 1.341e-05
##
##                   Kappa : 0.5495
##
##  Mcnemar's Test P-Value : 9.204e-13
##
##             Sensitivity : 0.7761
##             Specificity : 0.8505
##          Pos Pred Value : 0.9327
##          Neg Pred Value : 0.5872
##              Prevalence : 0.7275
##          Detection Rate : 0.5646
##    Detection Prevalence : 0.6053
##       Balanced Accuracy : 0.8133
##
##        'Positive' Class : NO
##
```

```
# Lasso Model
# Prediction on training set
PredTrain.L = predict(cvfit.lasso, newx=x, type="class")
confusionMatrix(Survived, factor(PredTrain.L))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  367  64
##        YES  93 188
##
##                Accuracy : 0.7795
##                  95% CI : (0.7472, 0.8094)
##     No Information Rate : 0.6461
##     P-Value [Acc > NIR] : 7.573e-15
##
##                   Kappa : 0.5301
##
##  Mcnemar's Test P-Value : 0.02544
##
##             Sensitivity : 0.7978
##             Specificity : 0.7460
##          Pos Pred Value : 0.8515
##          Neg Pred Value : 0.6690
##              Prevalence : 0.6461
##          Detection Rate : 0.5154
##    Detection Prevalence : 0.6053
##       Balanced Accuracy : 0.7719
```

```
##
##        'Positive' Class : NO
##
```

```r
# Elastic Model
# Prediction on training set
PredTrain.E = predict(cvfit.elastic, newx=x, type="class")
confusionMatrix(Survived, factor(PredTrain.E))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  402  29
##        YES 116 165
##
##                Accuracy : 0.7963
##                  95% CI : (0.7649, 0.8254)
##     No Information Rate : 0.7275
##     P-Value [Acc > NIR] : 1.341e-05
##
##                   Kappa : 0.5495
##
##  Mcnemar's Test P-Value : 9.204e-13
##
##             Sensitivity : 0.7761
##             Specificity : 0.8505
##          Pos Pred Value : 0.9327
##          Neg Pred Value : 0.5872
##              Prevalence : 0.7275
##          Detection Rate : 0.5646
##    Detection Prevalence : 0.6053
##       Balanced Accuracy : 0.8133
##
##        'Positive' Class : NO
##
```

Here, the Elastic Net performs best on our training set. Let us compare how it performs on our test set.

```r
# Ridge Model
# Prediction on validation set
PredTrain.R = predict(cvfit.ridge, newx=data.matrix(test.x[,cols]), type="class")
confusionMatrix(test.y, factor(PredTrain.R))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  107  11
##        YES  24  37
##
##                Accuracy : 0.8045
##                  95% CI : (0.7387, 0.8599)
##     No Information Rate : 0.7318
##     P-Value [Acc > NIR] : 0.01530
##
```

```
##                  Kappa : 0.5412
##
##  Mcnemar's Test P-Value : 0.04252
##
##             Sensitivity : 0.8168
##             Specificity : 0.7708
##          Pos Pred Value : 0.9068
##          Neg Pred Value : 0.6066
##              Prevalence : 0.7318
##          Detection Rate : 0.5978
##    Detection Prevalence : 0.6592
##       Balanced Accuracy : 0.7938
##
##        'Positive' Class : NO
##
```

```r
# Lasso Model
# Prediction on validation set
PredTrain.L = predict(cvfit.lasso, newx=data.matrix(test.x[,cols]), type="class")
confusionMatrix(test.y, factor(PredTrain.L))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  101  17
##        YES  16  45
##
##                Accuracy : 0.8156
##                  95% CI : (0.751, 0.8696)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 1.3e-06
##
##                   Kappa : 0.5913
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8632
##             Specificity : 0.7258
##          Pos Pred Value : 0.8559
##          Neg Pred Value : 0.7377
##              Prevalence : 0.6536
##          Detection Rate : 0.5642
##    Detection Prevalence : 0.6592
##       Balanced Accuracy : 0.7945
##
##        'Positive' Class : NO
##
```

```r
# Elastic Model
# Prediction on validation set
PredTrain.E = predict(cvfit.elastic, newx=data.matrix(test.x[,cols]), type="class")
confusionMatrix(test.y, factor(PredTrain.E))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  NO YES
##        NO  107  11
##        YES  24  37
##
##                Accuracy : 0.8045
##                  95% CI : (0.7387, 0.8599)
##     No Information Rate : 0.7318
##     P-Value [Acc > NIR] : 0.01530
##
##                   Kappa : 0.5412
##
##  Mcnemar's Test P-Value : 0.04252
##
##             Sensitivity : 0.8168
##             Specificity : 0.7708
##          Pos Pred Value : 0.9068
##          Neg Pred Value : 0.6066
##              Prevalence : 0.7318
##          Detection Rate : 0.5978
##    Detection Prevalence : 0.6592
##       Balanced Accuracy : 0.7938
##
##        'Positive' Class : NO
##
```