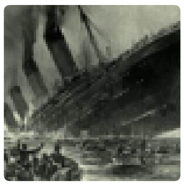


Random Forest Benchmark (R)
└ forked from *Random Forest Benchmark (R)* by *Ben Hamner* (+602/-25/~15)

2 voters

by AakashAgrawal · last run a month ago · R notebook · 248 views
using data from Titanic: Machine Learning from Disaster

Report

Titanic: Lasso/Ridge Implementation

Bisaria

Friday, May 20, 2016

Introduction

This is an attempt at predicting survivors in the Titanic dataset, using lasso and ridge regression methods, specifically glmnet package in R. Since an early exploration of data divulges huge disparity in survival ratio between men and women, separate predictive models were trained for both. As many observations had their Age variable missing, the Multiple Imputation by Chained Equations (MICE) package has been used for imputing missing age.

Models trained using ridge regression on male and lasso on female datasets yielded an accuracy of 0.81818 on public leaderboard. Interestingly, although lasso model on female dataset had marginally better misclassification error, performance on leaderboard improved on using ridge model. Furthermore, classifying the ladies above 14.5 years as ‘Ms’ after imputing the

Age led to improved performance of 0.82297 on the public leaderboard.

Since male model's prediction of survivors on the training and held-out test set was not good, it needs to be further explored using other algorithms.

Load data

```
titanic.train <- read.csv("../input/train.csv", stringsAsFactor=FALSE)
titanic.test <- read.csv("../input/test.csv", stringsAsFactor=FALSE)
```

Load R library

```
library(plyr)
library(rpart)
library(caret)
library(caTools)
library(mice)
library(stringr)
library(Hmisc)
library(ggplot2)
library(vcd)
library(ROCR)
library(pROC)
library(VIM)
library(glmnet)
```

Exploratory Data Analysis

First a quick peak into the dataset.

```
str(titanic.train)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)" "Heikkinen, Mi
ss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

```
summary(titanic.train)
```

```
##   PassengerId       Survived          Pclass          Name          
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000   Length:891        
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000   Class :character  
##  Median :446.0   Median :0.0000   Median :3.000   Mode  :character  
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309                     
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000                     
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000                     
##                                                                     
##      Sex                 Age            SibSp           Parch       
##  Length:891         Min.   : 0.42   Min.   :0.000   Min.   :0.0000  
##  Class :character   1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000  
##  Mode  :character   Median :28.00   Median :0.000   Median :0.0000  
##                     Mean   :29.70   Mean   :0.523   Mean   :0.3816  
##                     3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.0000  
##                     Max.   :80.00   Max.   :8.000   Max.   :6.0000  
##                     NA's   :177                                     
##     Ticket               Fare           Cabin             Embarked        
##  Length:891         Min.   :  0.00   Length:891         Length:891        
##  Class :character   1st Qu.:  7.91   Class :character   Class :character  
##  Mode  :character   Median : 14.45   Mode  :character   Mode  :character  
##                     Mean   : 32.20
##                      3rd Qu.: 31.00
##                      Max.   :512.33
##
```

```
table(titanic.train$Survived, titanic.train$Sex)
```

```
##
##     female male
##   0     81  468
##   1    233  109
```

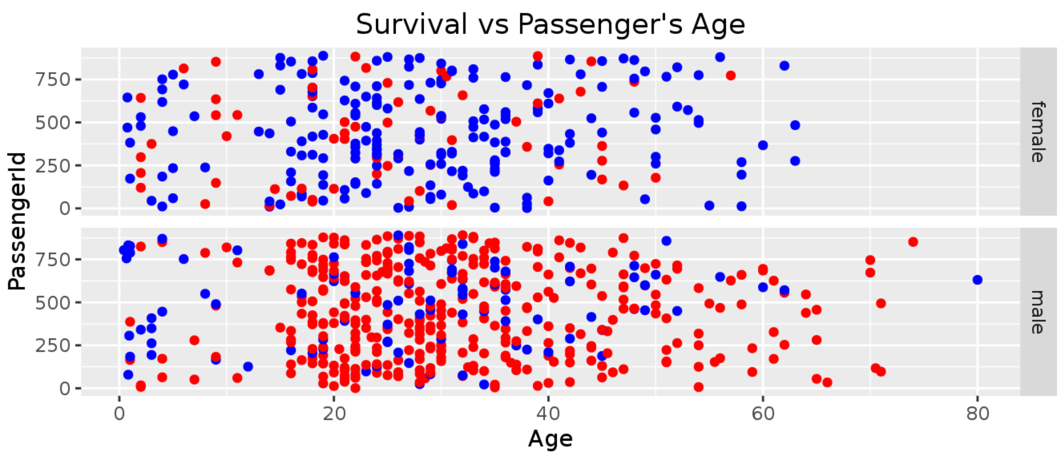
There are 891 passangers in the training data, comprising of 314 female and 577 male passengers, out of which 233 females and 109 males survived the disaster. The dataset provides information about each passenger’s name, age, gender, their port of embarkation, which cabin they booked, their ticket number and fare they paid along with number of family members they were travelling with.

First step would be to check if these variables have any correlation with the response variable Survived. In the dataset provided, some of the data like Age, Cabin etc. are missing. They will be ignored for preliminary exploration of the data.

Did age influence survival?

A scatter plot of age of each passenger, incorporating the information if the passenger perished or survived the disaster, show that while most of the casualty were male, those below the age of 15 years seem to have more or less same survival rate as females within that age group. Also, females above 50 seem to have better survival rate, while males above 50 are worse off than their middle-aged counterparts within their own gender groups.

```
ggplot(titanic.train, aes(x=Age, y=PassengerId, color = as.factor(Survived))) +
    geom_point() +
    facet_grid(Sex ~.) +
    ggtitle("Survival vs Passenger's Age")+
    xlab("Age") +
    theme(legend.position = "none")+
    scale_colour_manual(values = c("#FF0000","#0000FF"))
```

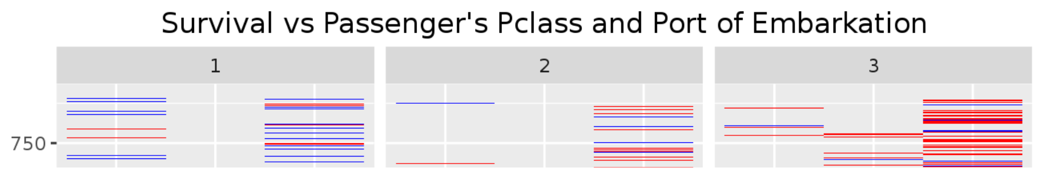


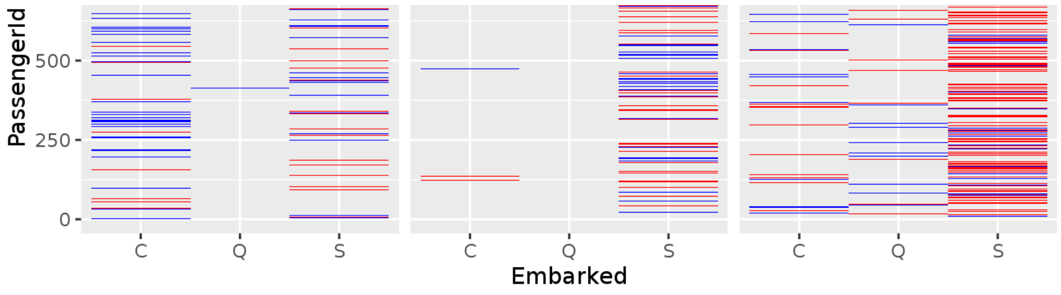
This information can be used later while creating new features based on age.

Is survival of a passenger related to his/her Pclass and port of embarkation?

Since passengers travelled in different classes, it would be worthwhile to check if there exists any correlation between his/her class of travel, from whence he/she boarded and whether they finally perished or survived.

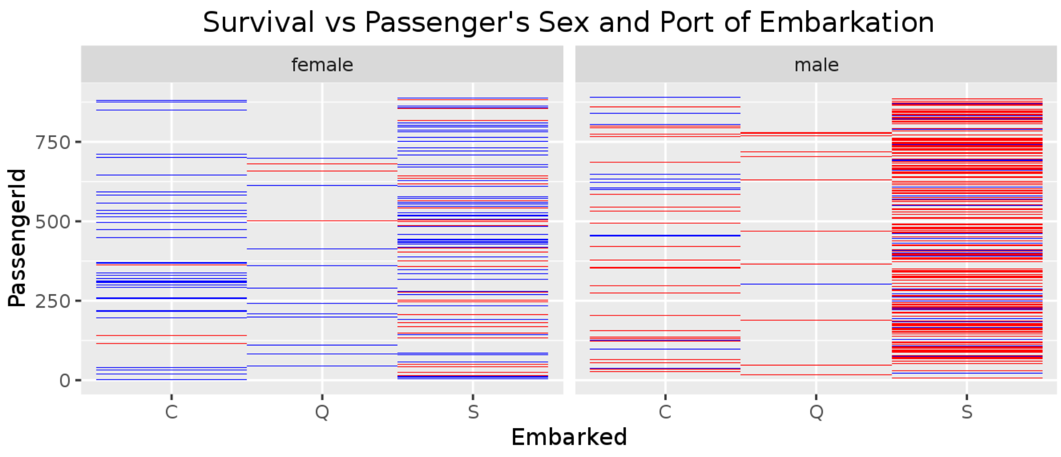
```
ggplot(titanic.train[titanic.train$Embarked != "",], aes(x=Embarked, y=PassengerId)) +
  geom_tile(aes(fill = as.factor(Survived))) +
  facet_grid(. ~ Pclass) +
  ggtitle("Survival vs Passenger's Pclass and Port of Embarkation")+
  theme(legend.position = "none")+
  scale_fill_manual(values = c("#FF0000","#0000FF"))
```





Most of the passengers who perished had embarked from port ‘S’ and were travelling 3rd class.

```r
ggplot(titanic.train[titanic.train$Embarked != "",], aes(x=Embarked, y=PassengerId)) +
  geom_tile(aes(fill = as.factor(Survived))) +
  facet_grid(. ~ Sex) +
  ggtitle("Survival vs Passenger's Sex and Port of Embarkation")+
  theme(legend.position = "none")+
  scale_fill_manual(values = c("#FF0000","#0000FF"))
```

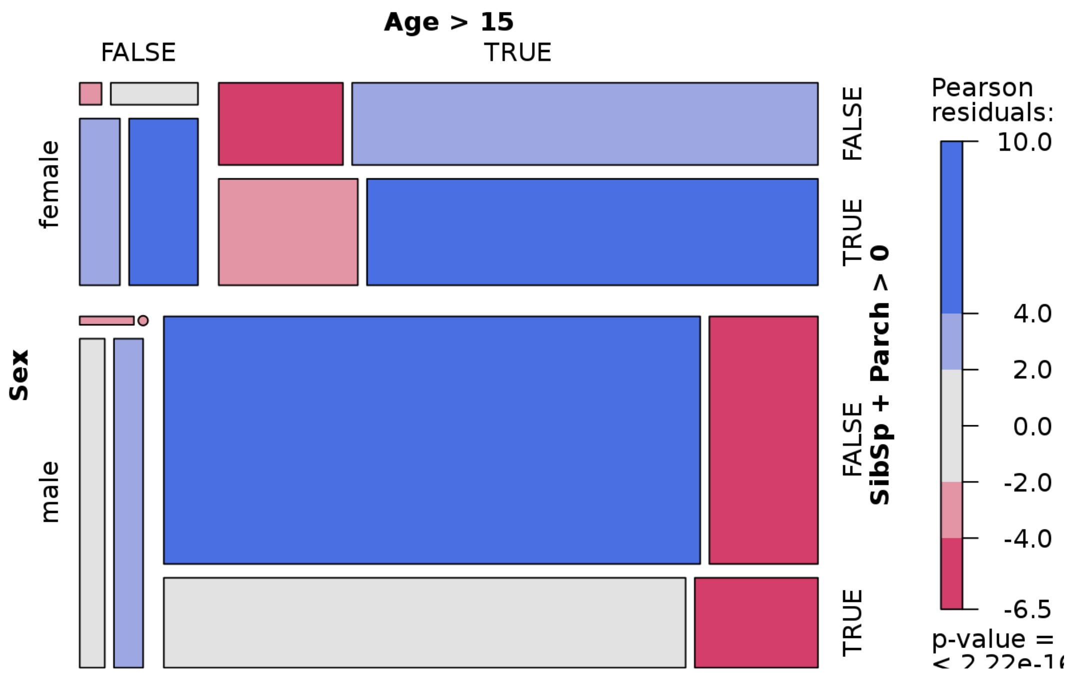


Looks like most of the unfortunate passengers from port ‘S’ travelling 3rd class were male, while most of the casualty among females also happened among passengers from the port ‘S’.

Did travelling with the family mattered?

The vcd package, which provides a variety of methods for visualizing multivariate categorical data, has been used here to look at possible correlation between survival of a passengers based on their gender, age and number of family member accompanying them.

```r
mosaic(~ Sex + (Age > 15) + (SibSp + Parch > 0) + Survived, data = titanic.train[complete.cases(titanic.train),],
       shade=T, legend=T)
```



Pearson residuals of 2 and higher suggest inter-dependece between the variables under consideration.

Finally, a combined dataframe of training and test data is created, with ‘Survived’ feature removed from the train dataset and saved separately. This will be added back to the train dataset after further processing of the combined data set.

```
Survived = titanic.train$Survived
titanic.test$Survived = NA
all = rbind(titanic.train, titanic.test)
```

Feature engineering

Title

Names of the passengers consist of titles ascribed to each individual as per their gender, age and social status, which can be extracted and maximum and minimum age for each category can be enumerated.

```
all$Title = sapply(all$Name,function(x) strsplit(x,', ')[[1]][2])
all$Title = sapply(all$Title,function(x) strsplit(x,'\\. ')[[1]][1])

as.data.frame(
  cbind("Title" = unique(all$Title),
        "No_of_passengers" = sapply(unique(all$Title), function(x) nrow(all[all$Title == x,])),
        "Age_missing" = sapply(unique(all$Title), function(x) nrow(all[all$Title == x & is.na(all$Age),])),
        "Minimum_Age" = sapply(unique(all$Title), function(x) min(all[all$Title == x,'Age'], na.rm = TRUE)),
        "Maximum_Age" = sapply(unique(all$Title), function(x) max(all[all$Title == x,'Age'], na.rm = TRUE))), row.name
s = F)
```

```
##           Title No_of_passengers Age_missing Minimum_Age Maximum_Age
## 1            Mr              757         176          11          80
## 2           Mrs              197          27          14          76
## 3          Miss              260          50        0.17          63
## 4        Master               61           8        0.33        14.5
## 5           Don                1           0          40          40
## 6           Rev                8           0          27          57
## 7            Dr                8           1          23          54
## 8           Mme                1           0          24          24
## 9            Ms                2           1          28          28
## 10        Major                2           0          45          52
## 11         Lady                1           0          48          48
## 12          Sir                1           0          49          49
## 13         Mlle                2           0          24          24
## 14          Col                4           0          47          60
## 15         Capt                1           0          70          70
## 16 the Countess                1           0          33          33
## 17     Jonkheer                1           0          38          38
## 18         Dona                1           0          39          39
```

These 18 categories can be combined into more manageable 5 categories as per their gender and age:

```
#   Mr:     For men above 14.5 years
#   Master: For boys below and equal to 14.5 years
#   Miss:   For girls below and equal to 14.5 years
#   Ms:     For women above 14.5 years, maybe unmarried
#   Mrs:    For married women above 14.5 years
```

All those who do not have missing age can be put into appropriate Title category on the basis of their age and gender as follows:

```
all[(all$Title == "Mr" & all$Age <= 14.5 & !is.na(all$Age)),]$Title = "Master"

all[all$Title == "Capt"|
    all$Title == "Col"|
    all$Title == "Don"|
    all$Title == "Major"|
    all$Title == "Rev"|
    all$Title == "Jonkheer"|
    all$Title == "Sir",]$Title = "Mr"
```

```
# None of these women are travelling with family, hence can be categorised as single women for this analysis
all[all$Title == "Dona"|
      all$Title == "Mlle"|
      all$Title == "Mme",]$Title = "Ms"

# Categories Lady and Countess as a married woman
all[all$Title == "Lady"| all$Title == "the Countess",]$Title = "Mrs"

# Categorise doctors as per their sex
all[all$Title == "Dr" & all$Sex == "female",]$Title = "Ms"
all[all$Title == "Dr" & all$Sex == "male",]$Title = "Mr"
```

All the titles have been successfully categorised into five defined categories excepting for observations in category Miss. Since Age feature is missing for these observations, they will be handled after imputing the missing age. Also, there is one women who is below 14.5 years of age and is married and hence has title Mrs, which is ignored.

```
all$Title = as.factor(all$Title)
all$Title <- droplevels(all$Title)
summary(all$Title)
```

```
## Master   Miss     Mr    Mrs     Ms
##     66    260    777    199      7
```

FamilySize

FamilySize is created based on the number of family member travelling with a passenger.

```
all$FamilySize = ifelse(all$SibSp + all$Parch + 1 <= 3, 1,0) # Small = 1, Big = 0
```

Mother

Identify the ladies travelling with their children.

```
all$Mother = ifelse(all$Title=="Mrs" & all$Parch > 0, 1,0)
```

Single

Identify people travelling solo.

```
all$Single = ifelse(all$SibSp + all$Parch + 1 == 1, 1,0) # People travelling alone
```

FamilyName

Family name of each individual can be extracted from their names.

```
all$FamilyName = sapply(all$Name,function(x) strsplit(x,', ')[[1]][1])
```

Since there are possibly many people sharing same family name, it is necessary to distinguish each family separately.

```
Family.Ticket = all[all$Single == 0,c("FamilyName", "Ticket")]
Family.Ticket = Family.Ticket[order(Family.Ticket$FamilyName),]
head(Family.Ticket)
```

```
##      FamilyName    Ticket
## 280      Abbott C.A. 2673
## 747      Abbott C.A. 2673
## 1284     Abbott C.A. 2673
## 309     Abelson P/PP 3381
## 875     Abelson P/PP 3381
## 41        Ahlin      7546
```

Baring few exceptions, in general, a family shared the same ticket number. This can be a good way of identifying families. Here, last three digits of the ticket is extracted and attached to family names, thereby creating unique family names for each family.

```
all$FamilyName  = paste(all$FamilyName , str_sub(all$Ticket,-3,-1), sep="")
```

FamilySurvived

Based on the exploratory analysis, a feature representing the survival of family can be created. One would hope that families travelling together would have tried to escape together and their survival must be closely tied to each other.

```
all$FamilySurvived = 0
# Dataset of passengers with family
Families = all[(all$Parch+all$SibSp) > 0,]

# Group families by their family name and number of survivals in the family
Survival.GroupByFamilyName = aggregate(as.numeric(Families$Survived), by=list("FamilyName" = Families$FamilyName), FUN
=sum, na.rm=TRUE)

# Family is considered to have survived if atleast one member survived
FamilyWithSurvival = Survival.GroupByFamilyName[Survival.GroupByFamilyName$x > 0,]$FamilyName
all[apply(all, 1, function(x){ifelse(x["FamilyName"] %in% FamilyWithSurvival,TRUE,FALSE)}),]$FamilySurvived = 1
```

AgeClass

We can categorise Age into four classes Class 1: Below 10 Class 2: between 10 to 20 Class 3: between 20 to 35 Class 4: Above 35

```
all$AgeClass = ifelse(all$Age<=10,1,
                      ifelse(all$Age>10 & all$Age<=20,2,
                             ifelse(all$Age>20 & all$Age<=35,3,4)))
all$AgeClass = as.factor(all$AgeClass)
```

Since Age is missing for many observations, there will be some data missing here too, which can be calculated after imputing Age.

Imputing missing data

The Multiple Imputation by Chained Equations (MICE) package is used for multiple imputation through predictive mean matching method, specifically for missing Age data, which ensures that imputed values are plausible.

```
all$Pclass = as.factor(all$Pclass)
all$Sex = as.factor(all$Sex)
all[all$Embarked == "",]$Embarked = NA
all$Embarked = as.factor(all$Embarked)
all[all$Cabin == "",]$Cabin = NA
all$Cabin = as.factor(all$Cabin)
all$FamilySize = as.factor(all$FamilySize)
all$Mother = as.factor(all$Mother)
all$Single = as.factor(all$Single)
all$FamilyName = as.factor(all$FamilyName)

md.pattern(all[,!names(all) %in% c("Survived", "Name", "PassengerId", "Ticket", "AgeClass")])
```

```
##     Pclass Sex SibSp Parch Title FamilySize Mother Single FamilyName
## 270      1   1     1     1     1          1      1      1          1
##  23      1   1     1     1     1          1      1      1          1
## 773      1   1     1     1     1          1      1      1          1
##   2      1   1     1     1     1          1      1      1          1
## 240      1   1     1     1     1          1      1      1          1
##   1      1   1     1     1     1          1      1      1          1
##          0   0     0     0     0          0      0      0          0
##     FamilySurvived Fare Embarked Age Cabin
## 270              1    1        1   1     1    0
##  23              1    1        1   0     1    1
## 773              1    1        1   1     0    1
##   2              1    1        0   1     1    1
## 240              1    1        1   0     0    2
##   1              1    0        1   1     0    2
##                  0    1        2 263  1014 1280
```

There are 263 observations Age feature missing from the dataset, 1 Observation with Fare, 2 observations with Embarked and 1014 observations have Cabin missing.

Embarked

Two of the passengers have the information about their port of embarkation missing. These two observations belong to passengers, both female, both survived and both were in B Type cabin travelling first class. It is assumed they embarked at S, as most of the female from first class who survived embarked either from S or C.

```
all$Embarked[is.na(all$Embarked)] = 'S'
```

Fare

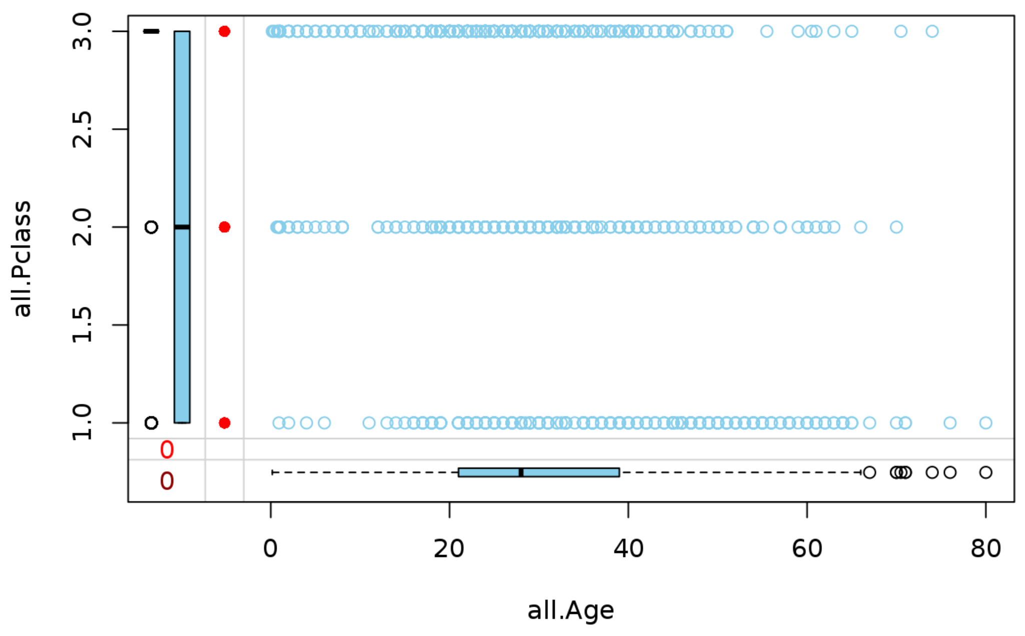
R package rpart is used to predict one missing fare data, using other features as predictors.

```
fit.Fare = rpart(Fare ~ Pclass + SibSp + Parch + Age + Embarked + Title,
                 data = all[!is.na(all$Fare),],
                 method = "anova")
all$Fare[is.na(all$Fare)] = predict(fit.Fare, newdata = all[is.na(all$Fare), ])
```

Age

It is observed that Age of 263 passengers is missing from the combined dataset, which needs to be imputed suitably.

```
marginplot(data.frame(all$Age, all$Pclass))
```



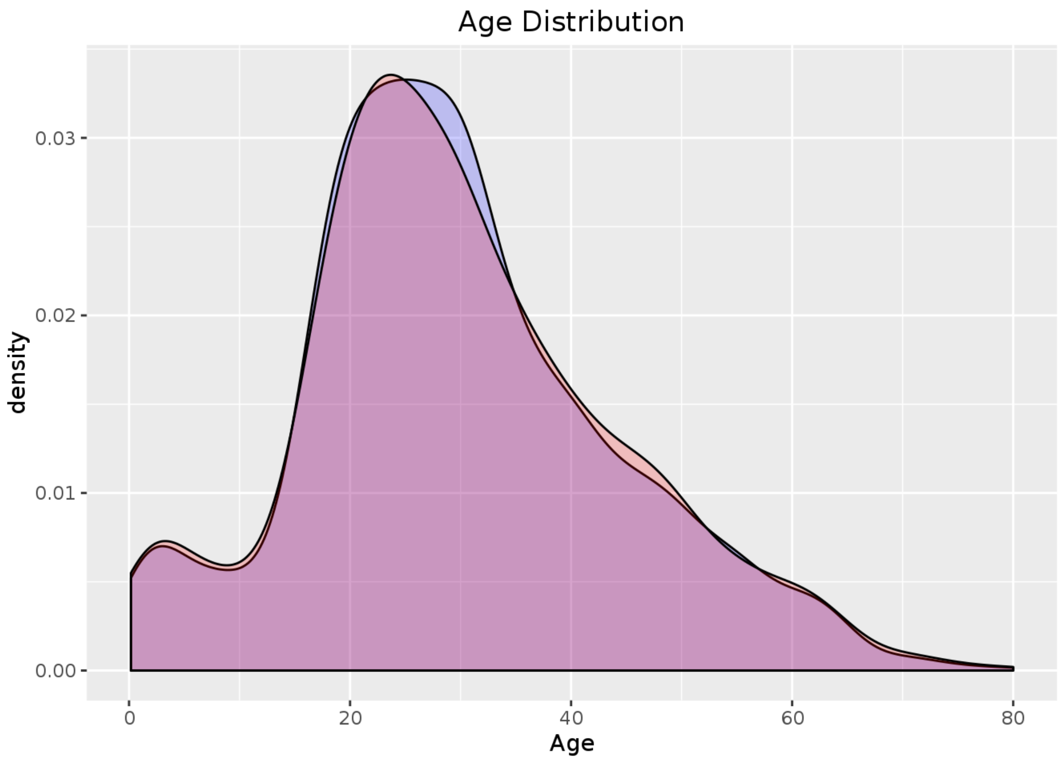
The missing age is not randomly distributed across all classes, but is rather concentrated amongst the passengers from 3rd class, indicating a missing at random(MAR) problem. Mice package can be used to impute these data using pmm or predictive mean matching method.

```
ageData <- mice(all[, !names(all) %in% c("Survived", "Name", "PassengerId", "Ticket", "AgeClass", "Cabin", "FamilyNam
e")],m=8,maxit=8,meth='pmm',seed=251863)
```

```
# Check out the imputed data
head(ageData$imp$Age)
```

```
##     1  2  3  4  5  6  7    8
## 6  24 55 22 29 23 36 57 28.0
## 18 39 18 21 35 41 21 25 36.0
## 20 36 19 66 45 22 15 43 36.5
## 27 28 49 31 27 37 20 32 40.0
## 29 24 18 39 21 18 31 14 28.0
## 30 28 16 28 19 22 31 40 34.0
```

```
# Check if the imputed data distribution follows the existing age distribution.
ggplot(all,aes(x=Age)) +
  geom_density(data=data.frame(all$PassengerId, complete(ageData,6)), alpha = 0.2, fill = "blue")+
  geom_density(data=all, alpha = 0.2, fill = "Red")+
  labs(title="Age Distribution")+
  labs(x="Age")
```



Imputed data seem to be acceptable and can be used for filling in the missing values in AgeClass and Title

```
# Sixth imputed data is picked up for further analysis based on the density distribution
all.imp <- data.frame(all$PassengerId, complete(ageData,6))

all$Age = all.imp$Age

all[is.na(all$AgeClass),]$AgeClass = ifelse(all[is.na(all$AgeClass),]$Age<=10,1,
                      ifelse(all[is.na(all$AgeClass),]$Age>10 & all[is.na(all$AgeClass),]$Age<=20,2,
                             ifelse(all[is.na(all$AgeClass),]$Age>20 & all[is.na(all$AgeClass),]$Age<=35,3,4)))

# All women above age of 14.5, with title Miss are to be recategorised as Ms.
all[all$Title == "Miss" & all$Age > 14.5,]$Title = "Ms"

# Check if titles and age are as required.
table(all$Title, all$Age > 14.5)
```

```
##
##          FALSE TRUE
##   Master    66    0
##   Miss      63    0
##   Mr         0  777
##   Mrs        1  198
##   Ms         0  204
```

Cabin

Missing cabin data can be imputed for people from same family, assuming they shared the cabins or had cabins nearby.

```
# Extract single alphabet prefixed to each cabin number provided. Each of these letters represent the part of the deck
 were these cabins were located.
all$CabinNo = sapply(all$Cabin,function(x) substr(x,1,1))
all$CabinNo[all$CabinNo == ""] = NA
table(is.na(all$CabinNo))
```

```
##
## FALSE  TRUE
##   295  1014
```

```r
# Dataset of all families with cabin data
familyWithCabinNo = unique(all[!is.na(all$CabinNo) & all$SibSp + all$Parch > 0,c("FamilyName", "CabinNo")])
head(familyWithCabinNo)
```

```
##      FamilyName CabinNo
## 2    Cumings599       C
## 4   Futrelle803       C
## 11 Sandstrom549       G
## 28   Fortune950       C
## 32   Spencer569       B
## 53    Harper572       D
```

```r
# Function to check if these people are travelling with family
checkIfHasCabin <- function(familyName, CabinNo){
  ifelse (familyName %in% familyWithCabinNo$FamilyName, familyWithCabinNo$CabinNo, CabinNo)
}

# Assign same cabin number to those members of a single family, whose cabin number is missing
all[is.na(all$CabinNo),]$CabinNo = apply(all[ is.na(all$CabinNo),c("FamilyName", "CabinNo")], 1, function(y) checkIfHa
sCabin(y["FamilyName"], y["CabinNo"]))

table(is.na(all$CabinNo))
```

```
##
## FALSE  TRUE
##   299  1010
```

```r
table(all$CabinNo, all$Pclass)
```

```
##
##      1  2  3
##   A 22  0  0
##   B 65  0  0
##   C 96  0  2
##   D 40  6  0
##   E 34  4  3
##   F  0 13  8
##   G  0  0  5
##   T  1  0  0
```

There are still some missing cabin data. Divide the unknown observations for cabin no into cabins for each Pclass in the same ratio as currently available.

```r
# Note: This procedure has been taken from script submitted on Kaggle.

# for first class obs
A.1 = round(22/(323-65) * 65)
B.1 = round(65/(323-65) * 65)
C.1 = round(96/(323-65) * 65)
D.1 = round(40/(323-65) * 65)
E.1 = 65 - (A.1+B.1+C.1+D.1)
# for second class
D.2 = round(6/(277-254) * 254)
E.2 = round(4/(277-254) * 254)
F.2 = 254 - (D.2+E.2)
# for third class
E.3 = round(3/(709-691) * 691)
F.3 = round(8/(709-691) * 691)
G.3 = 691 - (E.3+F.3)

set.seed(0)
all[ sample( which( all$Pclass==1 & is.na(all$CabinNo)), A.1 ) , "CabinNo"] <- rep("A", A.1)
all[ sample( which( all$Pclass==1 & is.na(all$CabinNo)), B.1 ) , "CabinNo"] <- rep("B", B.1)
all[ sample( which( all$Pclass==1 & is.na(all$CabinNo)), C.1 ) , "CabinNo"] <- rep("C", C.1)
all[ sample( which( all$Pclass==1 & is.na(all$CabinNo)), D.1 ) , "CabinNo"] <- rep("D", D.1)
all[ sample( which( all$Pclass==1 & is.na(all$CabinNo)), E.1 ) , "CabinNo"] <- rep("E", E.1)

set.seed(0)
all[ sample( which( all$Pclass==2 & is.na(all$CabinNo)), D.2 ) , "CabinNo"] <- rep("D", D.2)
all[ sample( which( all$Pclass==2 & is.na(all$CabinNo)), E.2 ) , "CabinNo"] <- rep("E", E.2)
```

```
all[ sample( which( all$Pclass==2 & is.na(all$CabinNo)), F.2 ) , "CabinNo"] <- rep("F", F.2)

set.seed(0)
all[ sample( which( all$Pclass==3 & is.na(all$CabinNo)), E.3 ) , "CabinNo"] <- rep("E", E.3)
all[ sample( which( all$Pclass==3 & is.na(all$CabinNo)), F.3 ) , "CabinNo"] <- rep("F", F.3)
all[ sample( which( all$Pclass==3 & is.na(all$CabinNo)), G.3 ) , "CabinNo"] <- rep("G", G.3)

all$CabinNo = as.factor(all$CabinNo)
table(all$CabinNo, all$Pclass)
```

```
## 
##       1   2   3
##   A  28   0   0
##   B  81   0   0
##   C 120   0   2
##   D  50  72   0
##   E  43  48 118
##   F   0 157 315
##   G   0   0 274
##   T   1   0   0
```

Finally, dataset is divided back into training and test set for further analysis.

```
all$Ticket = NULL
all$Name = NULL
all$Cabin = NULL

summary(all)
```

```
##   PassengerId      Survived      Pclass      Sex           Age       
##  Min.   :   1   Min.   :0.0000   1:323   female:466   Min.   : 0.17  
##  1st Qu.: 328   1st Qu.:0.0000   2:277   male  :843   1st Qu.:21.00  
##  Median : 655   Median :0.0000   3:709                Median :28.00  
##  Mean   : 655   Mean   :0.3838                        Mean   :29.74  
##  3rd Qu.: 982   3rd Qu.:1.0000                        3rd Qu.:38.00  
##  Max.   :1309   Max.   :1.0000                        Max.   :80.00  
##                 NA's   :418                                          
##      SibSp            Parch            Fare         Embarked    Title    
##  Min.   :0.0000   Min.   :0.000   Min.   :  0.000   C:270    Master: 66  
##  1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:  7.896   Q:123    Miss  : 63  
##  Median :0.0000   Median :0.000   Median : 14.454   S:916    Mr    :777  
##  Mean   :0.4989   Mean   :0.385   Mean   : 33.279            Mrs   :199  
##  3rd Qu.:1.0000   3rd Qu.:0.000   3rd Qu.: 31.275            Ms    :204  
##  Max.   :8.0000   Max.   :9.000   Max.   :512.329                        
##                                                                          
##  FamilySize Mother   Single         FamilyName   FamilySurvived   AgeClass
##  0: 125     0:1222   0:519    Sage343     :  11   Min.   :0.0000   1:101   
##  1:1184     1:  87   1:790    Goodwin144  :   8   1st Qu.:0.0000   2:219   
##                               Andersson082:   7   Median :0.0000   3:602   
##                               Asplund077  :   7   Mean   :0.2223   4:387   
##                               Fortune950  :   6   3rd Qu.:0.0000           
##                               Panula295   :   6   Max.   :1.0000           
##                               (Other)     :1264                            
##     CabinNo   
##  F      :472  
##  G      :274  
##  E      :209  
##  C      :122  
##  D      :122  
##  B      : 81  
##  (Other): 29
```

```
train = all[1:891,]
test = all[892:1309,]

train$Survived <- as.factor(Survived)
train$Survived <- as.factor(mapvalues(train$Survived, c("0", "1"), c("No","Yes")))
train$PassengerId = NULL
```

Lasso and Ridge Models

Training and validation sets

Dataset is split into separate male and female datasets, with respective Training and validation sets created using sample.split function. Levels and variables that are no longer meaningful in the newly created datasets are dropped.

```r
train.male = subset(train, train$Sex == "male")
train.female = subset(train, train$Sex == "female")
test.male = subset(test, test$Sex == "male")
test.female = subset(test, test$Sex == "female")

train.male$Sex = NULL
train.male$Mother = NULL
train.male$Title = droplevels(train.male$Title)

train.female$Sex = NULL
train.female$Title = droplevels(train.female$Title)

test.male$Sex = NULL
test.male$Mother = NULL
test.male$Title = droplevels(test.male$Title)

test.female$Sex = NULL
test.female$Title = droplevels(test.female$Title)

# MALE
set.seed(100)
splt.m = sample.split(train.male, 0.75)
cv.train.m = train.male[splt.m,]
cv.test.m = train.male[!splt.m,]

# FEMALE
set.seed(100)
splt.f = sample.split(train.female, 0.75)
cv.train.f = train.female[splt.f,]
cv.test.f = train.female[!splt.f,]
```

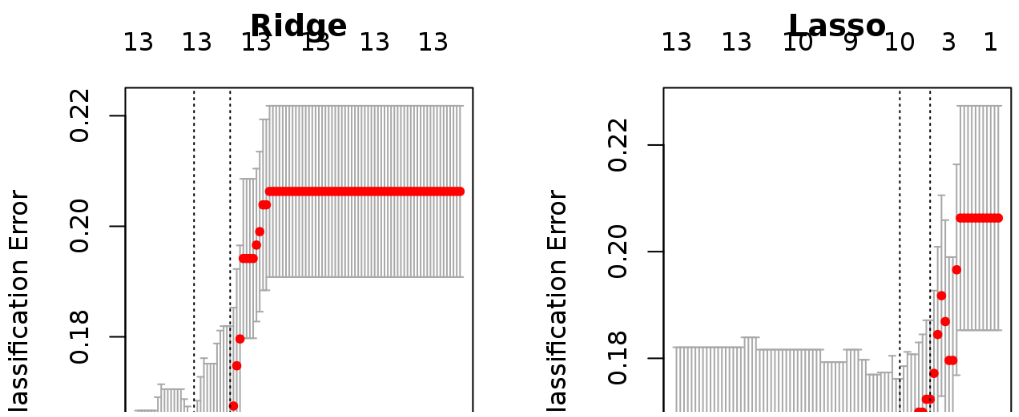
Model Training and Prediction

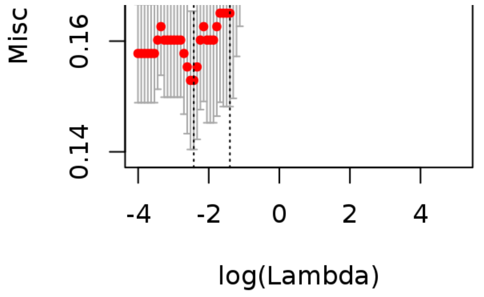
Male Dataset

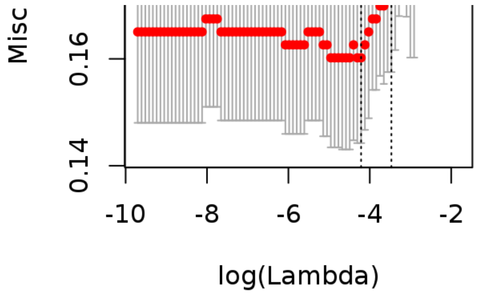
```r
x.m = data.matrix(cv.train.m[,2:14])
y.m = cv.train.m$Survived

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

cvfit.m.lasso = cv.glmnet(x.m, y.m,
                  family = "binomial",
                  alpha = 1,
                  type.measure = "class")
par(mfrow=c(1,2))
plot(cvfit.m.ridge, main = "Ridge")
plot(cvfit.m.lasso, main = "Lasso")
```







```
coef(cvfit.m.ridge, s = "lambda.min")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)     1.8153284542
## Pclass         -0.3058592985
## Age            -0.0055253738
## SibSp          -0.1930184111
## Parch          -0.0250353620
## Fare            0.0003377844
## Embarked       -0.0886130535
## Title          -1.2158127516
## FamilySize      0.5107429176
## Single          0.1376714899
## FamilyName     -0.0004935318
## FamilySurvived  1.2652525382
## AgeClass       -0.1582063951
## CabinNo        -0.0835574090
```

Ridge model gives a better misclassification error than the Lasso model, hence former is used for prediction.

```
# Prediction on training set
PredTrain.M = predict(cvfit.m.ridge, newx=x.m, type="class")
table(cv.train.m$Survived, PredTrain.M, cv.train.m$Title)
```

```
## , ,  = Master
## 
##      PredTrain.M
##        No Yes
##   No   14   0
##   Yes   3  16
## 
## , ,  = Mr
## 
##      PredTrain.M
##        No Yes
##   No  312   1
##   Yes  64   2
```

```
# Prediction on validation set
PredTest.M = predict(cvfit.m.ridge, newx=data.matrix(cv.test.m[,2:14]), type="class")
table(cv.test.m$Survived, PredTest.M, cv.test.m$Title)
```

```
## , ,  = Master
## 
##      PredTest.M
##        No Yes
##   No    6   0
##   Yes   0   4
## 
## , ,  = Mr
## 
##      PredTest.M
##        No Yes
##   No  135   0
##   Yes  20   0
```

```
# Prediction on test set
PredTest.M = predict(cvfit.m.ridge, newx=data.matrix(test.male[,3:15]), type="class")
```

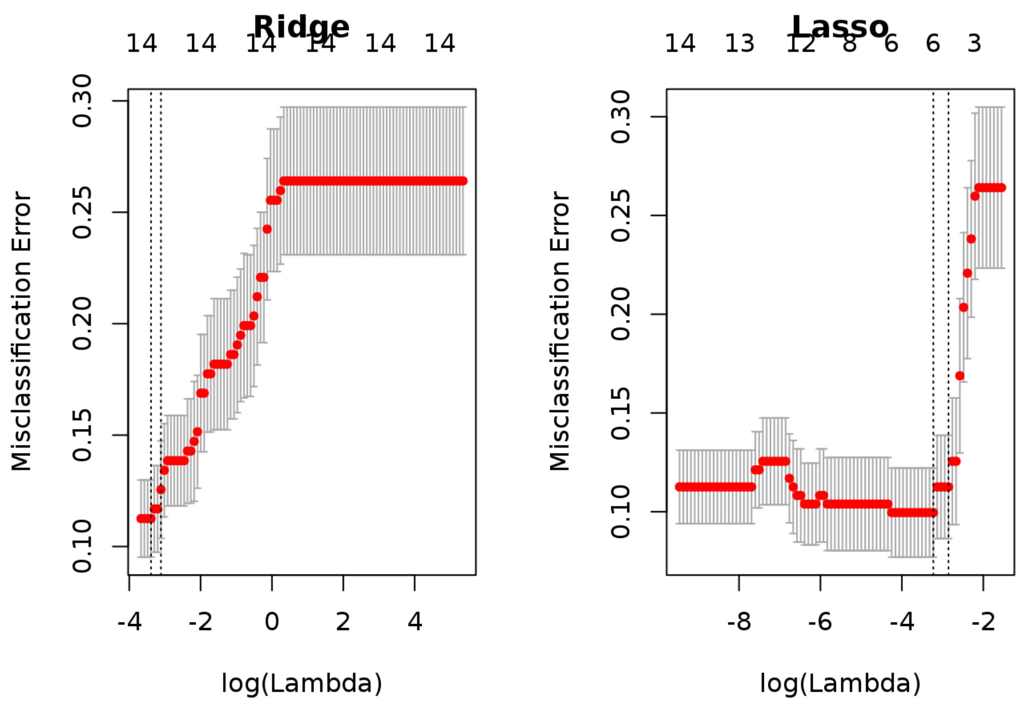
```
table(PredTest.M, test.male$Title)
```

```
##
## PredTest.M Master  Mr
##        No      15 243
##        Yes      8   0
```

Although ridge model does a fairly good job at predicting survivors among male below 14.5 years of age with very high precision and recall, i.e. for Master, it fails to predict even a single surviving adult male.

Female Dataset

```
                  family = "binomial",
                  alpha = 0,
                  type.measure = "class")
cvfit.f.lasso = cv.glmnet(x.f, y.f,
                  family = "binomial",
                  alpha = 1,
                  type.measure = "class")
par(mfrow=c(1,2))
plot(cvfit.f.ridge, main = "Ridge")
plot(cvfit.f.lasso, main = "Lasso")
```



```
coef(cvfit.f.ridge, s = "lambda.min")
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)     0.1984359713
## Pclass         -0.8003186702
## Age             0.0008458793
## SibSp          -0.1812623182
## Parch          -0.0430531135
## Fare            0.0003299235
## Embarked       -0.3034556467
## Title           0.2200749926
## FamilySize      0.6171064829
## Mother         -0.1155250147
## Single          1.1499282937
## FamilyName      0.0001647856
## FamilySurvived  2.8926840207
## AgeClass       -0.0281684980
## CabinNo        -0.0875213684
```

Although, lasso model gives much better misclassification error than the ridge model in case of female dataset, kappa statistics on the training set is better for ridge model.

```
# Ridge Model
# Prediction on training set
PredTrain.F = predict(cvfit.f.ridge, newx=x.f, type="class")
table(cv.train.f$Survived, PredTrain.F, cv.train.f$Title)
```

```
##      = Miss
```

```
##
## , ,  = Mrs
##
##      PredTrain.F
##       No Yes
##   No  16   3
##   Yes  0  73
##
## , ,  = Ms
##
##      PredTrain.F
##       No Yes
##   No   8  17
##   Yes  0  79
```

```
confusionMatrix(cv.train.f$Survived, PredTrain.F)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No   37  24
##        Yes   0 170
##
##                Accuracy : 0.8961
##                  95% CI : (0.8494, 0.9323)
##     No Information Rate : 0.8398
##     P-Value [Acc > NIR] : 0.009602
##
##                   Kappa : 0.6941
##  Mcnemar's Test P-Value : 2.668e-06
##
##             Sensitivity : 1.0000
##             Specificity : 0.8763
##          Pos Pred Value : 0.6066
##          Neg Pred Value : 1.0000
##              Prevalence : 0.1602
##          Detection Rate : 0.1602
##    Detection Prevalence : 0.2641
##       Balanced Accuracy : 0.9381
##
##        'Positive' Class : No
##
```

```
# Prediction on validation set
PredTest.F = predict(cvfit.f.ridge, newx=data.matrix(cv.test.f[,2:15]), type="class")
table(cv.test.f$Survived, PredTest.F, cv.test.f$Title)
```

```
## , ,  = Miss
##
##      PredTest.F
##       No Yes
##   No   5   0
##   Yes  0   8
##
```

```
## , ,  = Mrs
##
##      PredTest.F
##       No Yes
##   No   7   0
##   Yes  0  28
##
## , ,  = Ms
##
##      PredTest.F
##       No Yes
##   No   2   6
##   Yes  0  27
```

```
##
##           Reference
## Prediction No Yes
##        No  14   6
##        Yes  0  63
##
##                Accuracy : 0.9277
##                  95% CI : (0.8493, 0.973)
##     No Information Rate : 0.8313
##     P-Value [Acc > NIR] : 0.00885
##
##                   Kappa : 0.7798
##  Mcnemar's Test P-Value : 0.04123
##
##             Sensitivity : 1.0000
##             Specificity : 0.9130
##          Pos Pred Value : 0.7000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.1687
##          Detection Rate : 0.1687
##    Detection Prevalence : 0.2410
##       Balanced Accuracy : 0.9565
##
##        'Positive' Class : No
##
```

```
# Ridge Model
# Prediction on training set
PredTrain.F = predict(cvfit.f.lasso, newx=x.f, type="class")
table(cv.train.f$Survived, PredTrain.F, cv.train.f$Title)
```

```
## , ,  = Miss
##
##      PredTrain.F
##       No Yes
##   No  13   4
##   Yes  0  18
##
## , ,  = Mrs
##
##      PredTrain.F
##       No Yes
##   No  13   6
##   Yes  0  73
##
## , ,  = Ms
##
##      PredTrain.F
##       No Yes
##   No   9  16
##   Yes  0  79
```

```
confusionMatrix(cv.train.f$Survived, PredTrain.F)
```

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

```
##
##           Reference
## Prediction  No Yes
##        No   35  26
##        Yes   0 170
##
##                Accuracy : 0.8874
##                  95% CI : (0.8394, 0.9251)
##     No Information Rate : 0.8485
##     P-Value [Acc > NIR] : 0.05531
##
##                   Kappa : 0.6646
##  Mcnemar's Test P-Value : 9.443e-07
```

```
##              Prevalence : 0.1515
##          Detection Rate : 0.1515
##    Detection Prevalence : 0.2641
##       Balanced Accuracy : 0.9337
##
##        'Positive' Class : No
##
```

```
# Prediction on validation set
PredTest.F = predict(cvfit.f.lasso, newx=data.matrix(cv.test.f[,2:15]), type="class")
table(cv.test.f$Survived, PredTest.F, cv.test.f$Title)
```

```
## , ,  = Miss
##
##      PredTest.F
##       No Yes
##   No   5   0
##   Yes  0   8
##
## , ,  = Mrs
##
##      PredTest.F
##       No Yes
##   No   7   0
##   Yes  0  28
##
## , ,  = Ms
##
##      PredTest.F
##       No Yes
##   No   2   6
##   Yes  0  27
```

```
confusionMatrix(cv.test.f$Survived, PredTest.F)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  14   6
##        Yes  0  63
##
##                Accuracy : 0.9277
##                  95% CI : (0.8493, 0.973)
##     No Information Rate : 0.8313
##     P-Value [Acc > NIR] : 0.00885
##
##                   Kappa : 0.7798
##  Mcnemar's Test P-Value : 0.04123
##
##             Sensitivity : 1.0000
##             Specificity : 0.9130
##          Pos Pred Value : 0.7000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.1687
##         Detection Rate : 0.1687
##   Detection Prevalence : 0.2410
##      Balanced Accuracy : 0.9565
##
##       'Positive' Class : No
##
```

```
# Prediction on test set
PredTest.F = predict(cvfit.f.ridge, newx=data.matrix(test.female[,3:16]), type="class")
table(PredTest.F, test.female$Title)
```

```
##
```

but it predicts some false positives, predicting more survivors than there actually.

Conclusion

Ridge regression seems to predict on female dataset with higher precision and recall than on the male. Since the number of survivors in the male dataset is much less than that in female dataset, it fails to predict survivors especially among the adult male.

Kaggle Results

```
MySubmission.F = data.frame(PassengerId = test.female$PassengerId, Survived = ifelse(PredTest.F == "Yes",1,0))
MySubmission.M = data.frame(PassengerId = test.male$PassengerId, Survived = ifelse(PredTest.M == "Yes",1,0))

MySubmission = rbind(MySubmission.F, MySubmission.M)
MySubmission = MySubmission[order(MySubmission$PassengerId),]
names(MySubmission) = c("PassengerId","Survived")
table(MySubmission$Survived)
```

```
##
##   0   1
## 282 136
```

```
write.csv(MySubmission, file = 'Submission_RIDGE.csv', row.names = F)
```

In view of zero survival prediction among the adult male and too many false positives in case of females, this model did reasonably well with a score of 0.82297 on the public leaderboard.

What next?

You are currently viewing an old version of this script (17/18). View the latest version.