Tree-based models

Khanin Sisaengsuwanchai

2022-06-07

In this practice, I will compare different methods for generating a classification tree.

a) Randomly split the titanic dataset into test and train components.

```r
# Load Titanic dataset
load(url("https://stodden.net/StatData/titanic.rda"))

# Data exploration
summary(titanic)
```

```
##      pclass         survived          name               sex           
##  Min.   :1.000   Min.   :0.000   Length:1309        Length:1309       
##  1st Qu.:2.000   1st Qu.:0.000   Class :character   Class :character  
##  Median :3.000   Median :0.000   Mode  :character   Mode  :character  
##  Mean   :2.295   Mean   :0.382                                        
##  3rd Qu.:3.000   3rd Qu.:1.000                                        
##  Max.   :3.000   Max.   :1.000                                        
##                                                                       
##       age            sibsp            parch          ticket         
##  Min.   : 0.17   Min.   :0.0000   Min.   :0.000   Length:1309       
##  1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.000   Class :character  
##  Median :28.00   Median :0.0000   Median :0.000   Mode  :character  
##  Mean   :29.88   Mean   :0.4989   Mean   :0.385                     
##  3rd Qu.:39.00   3rd Qu.:1.0000   3rd Qu.:0.000                     
##  Max.   :80.00   Max.   :8.0000   Max.   :9.000                     
##  NA's   :263                                                        
##       fare             cabin             embarked             boat          
##  Min.   :  0.000   Length:1309        Length:1309        Length:1309       
##  1st Qu.:  7.896   Class :character   Class :character   Class :character  
##  Median : 14.454   Mode  :character   Mode  :character   Mode  :character  
##  Mean   : 33.295                                                           
##  3rd Qu.: 31.275                                                           
##  Max.   :512.329                                                           
##  NA's   :1                                                                 
##       body        home.dest        
##  Min.   :  1.0   Length:1309       
##  1st Qu.: 72.0   Class :character  
##  Median :155.0   Mode  :character  
##  Mean   :160.8                     
##  3rd Qu.:256.0                     
## Max.   :328.0
## NA's   :1188
```

```
# Remarks:
# 1. Many columns are integer, not categorical --> need to be changed
# 2. age and fare have NAs --> Fill the missing age with its mean, remove an NA of fare, and remove the

# Data pre-processing
titanic$survived <- as.factor(titanic$survived) # Change the type of survived
titanic$pclass   <- as.factor(titanic$pclass) # Change the type of survived
titanic$sex   <- as.factor(titanic$sex) # Change the type of survived
titanic$sibsp   <- as.factor(titanic$sibsp) # Change the type of survived
titanic$parch   <- as.factor(titanic$parch) # Change the type of survived
titanic$embarked   <- as.factor(titanic$embarked) # Change the type of survived


titanic$age[is.na(titanic$age)] <- mean(titanic$age, na.rm = T) # fill the missing values of age
titanic = titanic[, -c(3, 8, 10, 12, 13, 14)] # Remove the name, ticket, cabin, boat, body, and home.de
titanic = titanic[complete.cases(titanic), ] # Remove remaining 1 NA rows

# Recheck the changes
summary(titanic)
```

```
##  pclass  survived     sex           age          sibsp       parch
##  1:323   0:808    female:466   Min.   : 0.17   0:890   0      :1001
##  2:277   1:500    male  :842   1st Qu.:22.00   1:319   1      : 170
##  3:708                         Median :29.88   2: 42   2      : 113
##                                Mean   :29.86   3: 20   3      :   8
##                                3rd Qu.:35.00   4: 22   4      :   6
##                                Max.   :80.00   5:  6   5      :   6
##                                                8:  9   (Other):   4
##       fare          embarked
##  Min.   :  0.000    :  2
##  1st Qu.:  7.896   C:270
##  Median : 14.454   Q:123
##  Mean   : 33.295   S:913
##  3rd Qu.: 31.275
##  Max.   :512.329
##
```

```
set.seed(1)
# I would split training and test data into 80:20
train <- sample(1:nrow(titanic), nrow(titanic) * 0.8)
titanic.test <- titanic[-train, ]
survived.test <- titanic$survived[-train]


head(titanic.test)
```

```
##    pclass survived    sex      age sibsp parch     fare embarked
## 2       1        1   male  0.92000     1     2 151.5500        S
## 4       1        0   male 30.00000     1     2 151.5500        S
## 10      1        0   male 71.00000     0     0  49.5042        C
## 12      1        1 female 18.00000     1     0 227.5250        C
## 16      1        0   male 29.88114     0     0  25.9250        S
## 18      1        1 female 50.00000     0     1 247.5208        C
```

```
head(survived.test)
```

```
## [1] 1 0 0 1 0 1
## Levels: 0 1
```

b) Fit a classification tree to predict survival of the titanic accident. Using a full tree and a pruned tree.

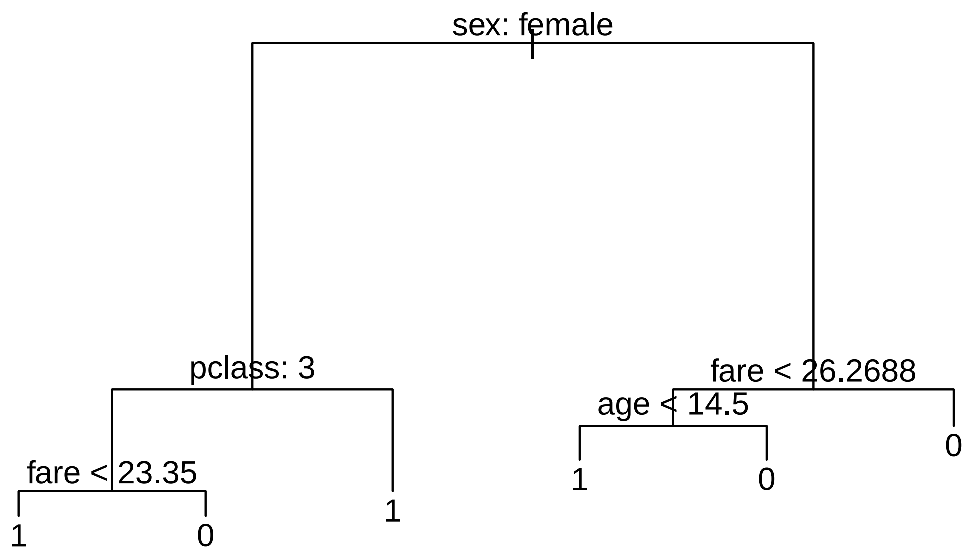
```
library(tree)
# Fit the full tree
set.seed(2)
tree.titanic <- tree(survived ~ ., titanic,
                      subset = train)
tree.pred <- predict(tree.titanic, titanic.test,
                     type = "class")
table(tree.pred, survived.test)
```

```
##          survived.test
## tree.pred   0   1
##         0 139  28
##         1  25  70
```

```
mean(tree.pred != survived.test) # Test error rates
```

```
## [1] 0.2022901
```

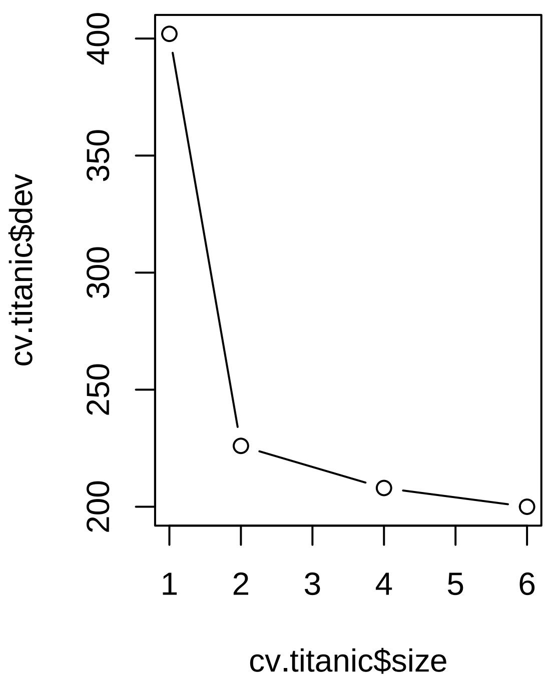
```
plot(tree.titanic)
text(tree.titanic, pretty = 0)
```

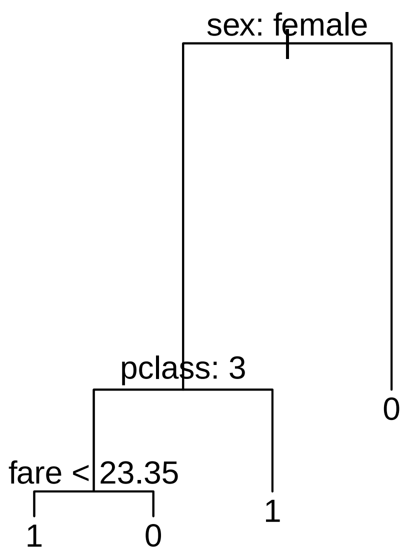


```
# Fit the pruned tree
set.seed(7)
cv.titanic <- cv.tree(tree.titanic, FUN = prune.misclass) # Prune with cross-validation errors

# Plot the number of terminal node with the cv errors
par(mfrow = c(1, 2))
plot(cv.titanic$size, cv.titanic$dev, type = "b")
# When we prune the trees, it introduces bias but decreases variance.

prune.titanic <- prune.misclass(tree.titanic, best = 4) # Select # of tree from the cv where the errors
plot(prune.titanic)
text(prune.titanic, pretty = 0)
```





```
# Predict and find errors
prune.pred <- predict(prune.titanic, titanic.test, type = "class")
table(prune.pred, survived.test)
```

```
##           survived.test
## prune.pred   0   1
##          0 140  33
##          1  24  65
```

```
mean(prune.pred != survived.test) # Test error rates
```

```
## [1] 0.2175573
```

```
# After pruning, the test errors slightly increase from 20.22% to 21.76%
```

c) Now try bagging to fit a classification tree.

```
library(randomForest)
```

```
## randomForest 4.7-1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
bag.titanic <- randomForest(survived ~ ., data = titanic, subset = train, mtry = 7, importance = TRUE)
# mtry = 7 means all predictors are being used, which are bagging.
bag.titanic
```

```
##
## Call:
##  randomForest(formula = survived ~ ., data = titanic, mtry = 7,      importance = TRUE, subset = tra
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 21.32%
## Confusion matrix:
##     0   1 class.error
## 0 545  99   0.1537267
## 1 124 278   0.3084577
```

```
bag.pred <- predict(bag.titanic, titanic.test,
                    type = "class")
table(bag.pred, survived.test)
```

```
##         survived.test
## bag.pred   0   1
##        0 135  24
##        1  29  74
```

```
mean(bag.pred != survived.test) # Test error rates
```

```
## [1] 0.2022901
```

```
# After fitting bagging, the test errors are the same as the full tree (20.23%).
# This is because the trees generated from bagging might be very similar leading to local optima, not g
```

d) Do the same as c) using the Boosting technique.

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
## Loading required package: lattice
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
library(stats)
library(pROC)
```

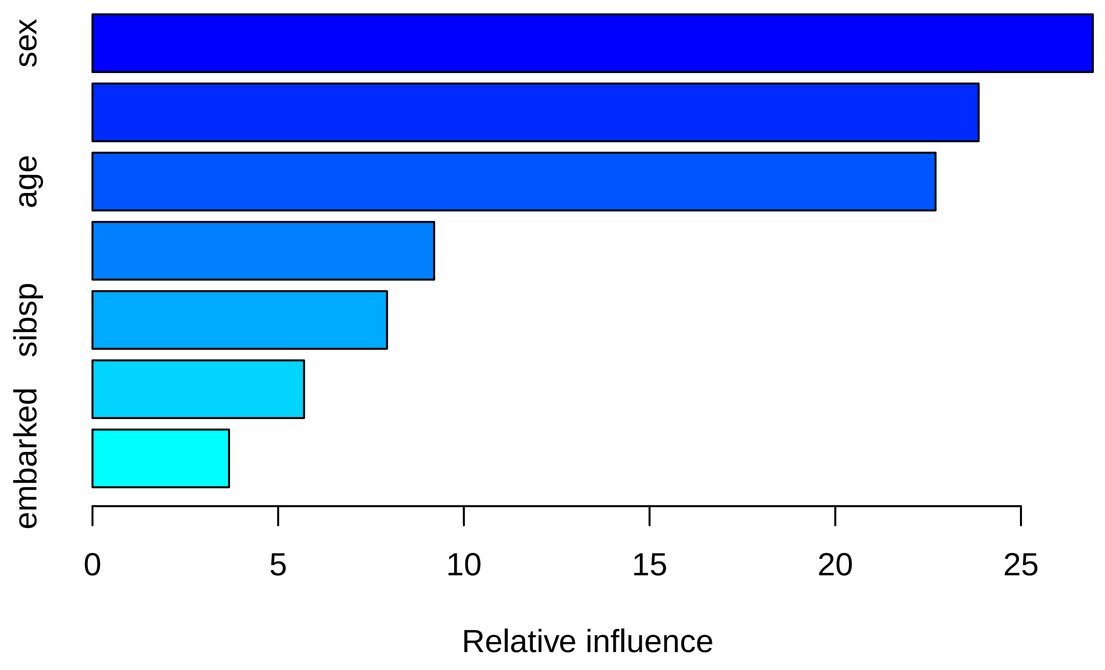
```
## Type 'citation("pROC")' for a citation.
```

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

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

```r
set.seed(1)
boost.titanic <- gbm(as.integer(survived) - 1 ~ ., data = titanic[train, ],
                     shrinkage = .02, distribution = 'bernoulli', n.trees = 5000,
                     verbose = FALSE)
# For distribution = 'bernoulli', gbm expects y to be 1 and 0 (integer)
# Note:
# A regression problem: distribution = "gaussian"
# A binary classification problem: distribution = "bernoulli"
# n.trees = 5000 indicates that we want 5000 trees
# interaction.depth = 4 limits the depth of each tree

summary(boost.titanic)
```



```
##               var   rel.inf
## sex           sex 26.935735
## fare         fare 23.861184
## age           age 22.700380
## pclass     pclass  9.199577
## sibsp       sibsp  7.930256
## parch       parch  5.696137
## embarked embarked  3.676732
```

```
# sex, fare, and age are by far the most important variables.

boost.pred <- predict(boost.titanic,
                      newdata = titanic.test, n.trees = 5000)

# Calculate cut points of boost.pred using ROC
boost.roc = roc(survived.test,boost.pred)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
coords(boost.roc,"best") # The best cutpoint is -0.3425848.
```

```
##    threshold specificity sensitivity
## 1 -0.3425848   0.7987805   0.7653061
```

```r
boost.pred_label = as.factor(ifelse(boost.pred>-0.3425848,1,0))

head(boost.pred_label)
```

```
## [1] 1 0 0 1 0 1
## Levels: 0 1
```

```r
table(boost.pred_label, survived.test)
```

```
##                  survived.test
## boost.pred_label   0   1
##                0 131  23
##                1  33  75
```

```r
mean(boost.pred_label != survived.test) # Test error rates
```

```
## [1] 0.2137405
```

```r
# After fitting a boosting tree, the test errors slightly increase from the full tree to be 20.23%.

# Show accuracy
confusionMatrix(boost.pred_label, survived.test)$overall[1]
```

```
##  Accuracy
## 0.7862595
```

e) Use Random Forests to fit a classification tree.

```r
set.seed(1)
rf.titanic <- randomForest(survived ~ ., data = titanic, subset = train, importance = TRUE)
rf.titanic
```

```
##
## Call:
##  randomForest(formula = survived ~ ., data = titanic, importance = TRUE,      subset = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 19.6%
## Confusion matrix:
##     0   1 class.error
## 0 580  64  0.09937888
## 1 141 261  0.35074627
```

```r
rf.pred <- predict(rf.titanic, titanic.test,
                     type = "class")
table(rf.pred, survived.test)
```

```
##         survived.test
## rf.pred   0   1
##       0 144  29
##       1  20  69
```

```
mean(rf.pred != survived.test) # Test error rates
```

```
## [1] 0.1870229
```

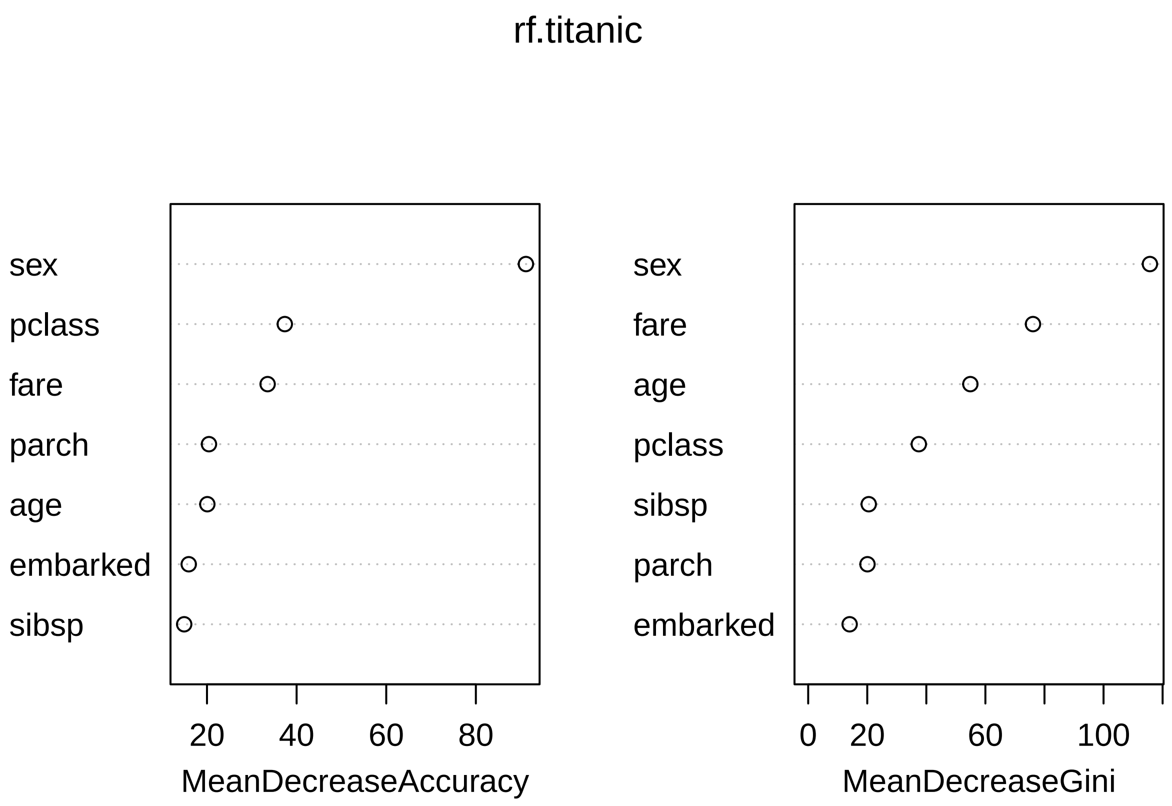
```
# For random forest classification trees, I pick # of trees(B) to be a default values 500
# and # of predictor = sqrt(p) = sqrt(7).

# The test errors are decreased significantly to 18.7%.

# Feature important
importance(rf.titanic)
```

```
##                 0         1 MeanDecreaseAccuracy MeanDecreaseGini
## pclass   19.95606 30.954176             37.35965         37.46594
## sex      66.07029 86.437024             91.16974        115.70716
## age      16.98026  8.846402             20.07660         54.90396
## sibsp    14.30944  3.093422             14.90993         20.51004
## parch    17.75063  5.739275             20.43979         20.06794
## fare     20.13790 23.851896             33.52815         76.11196
## embarked 11.96793  8.048888             15.93094         14.04192
```

```
varImpPlot(rf.titanic)
```



f) What characteristics of the dataset and/or the research question that might be driving the differences in misclassification error you are observing in parts b) through e)?

According to the models, the random forest model is the best model with about 19% of errors, which slightly lower than other models. The main differences of each method from b) through e) are the following.
1. The full classification tree fits on a single dataset, which might lead to high variance. 2. The pruned classification tree still fits on a single dataset but it removes some leaf nodes increasing bias but reducing variance. 3. For the bagging trees, the trees are grown independently on random samples of the observations. Therefore, the trees tend to be quite similar to each other, and thus more likely to get caught in local optima and can fail to thoroughly explore the model space. 4. For random forests, similar to bagging, the trees are grown independently on random samples of the observations. However, each split on each tree is performed using a random subset of the features, thereby decorre- lating the trees, and leading to a more thorough exploration of model space relative to bagging. 5. In boosting, we only use the original data, and do not draw any random samples. The trees are grown successively, using a “slow” learning approach: each new tree is fit to the signal that is left over from the earlier trees, and shrunken down before it is used.

g) Fit a logistic model to predict survival on the titanic.

```
# A logistic model
glm.fits <- glm(
  survived ~ ., data = titanic,
  family = binomial
)
summary(glm.fits)
```

```
##
```

```
## Call:
## glm(formula = survived ~ ., family = binomial, data = titanic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6069  -0.6367  -0.4378   0.6229   2.5920
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.815e+01  1.670e+03   0.011  0.99133
## pclass2     -1.031e+00  2.473e-01  -4.166 3.10e-05 ***
## pclass3     -1.840e+00  2.447e-01  -7.520 5.46e-14 ***
## sexmale     -2.542e+00  1.617e-01 -15.717  < 2e-16 ***
## age         -3.236e-02  6.713e-03  -4.820 1.43e-06 ***
## sibsp1      -8.606e-02  1.829e-01  -0.471  0.63790
## sibsp2      -2.289e-01  4.291e-01  -0.533  0.59376
## sibsp3      -1.757e+00  6.159e-01  -2.853  0.00433 **
## sibsp4      -1.971e+00  7.159e-01  -2.753  0.00590 **
## sibsp5      -1.650e+01  8.432e+02  -0.020  0.98439
## sibsp8      -1.617e+01  6.788e+02  -0.024  0.98100
## parch1       6.891e-01  2.393e-01   2.880  0.00398 **
## parch2       3.321e-01  3.026e-01   1.097  0.27244
## parch3       3.793e-01  8.557e-01   0.443  0.65761
## parch4      -1.632e+00  1.193e+00  -1.368  0.17133
## parch5      -1.119e+00  1.153e+00  -0.970  0.33207
## parch6      -1.540e+01  1.442e+03  -0.011  0.99148
## parch9      -1.582e+01  1.430e+03  -0.011  0.99117
## fare         9.085e-04  1.926e-03   0.472  0.63716
## embarkedC   -1.462e+01  1.670e+03  -0.009  0.99302
## embarkedQ   -1.496e+01  1.670e+03  -0.009  0.99285
## embarkedS   -1.511e+01  1.670e+03  -0.009  0.99278
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1740.1  on 1307  degrees of freedom
## Residual deviance: 1171.9  on 1286  degrees of freedom
## AIC: 1215.9
##
## Number of Fisher Scoring iterations: 15
```

```r
# Predict glm
glm.pred_prop <- predict(glm.fits, newdata = titanic.test, type = "response")

glm.pred <- as.factor(ifelse(glm.pred_prop < 0.5 , "0", "1")) # < 0.5 puts it as "0" else "1"

table(glm.pred, survived.test)
```

```
##         survived.test
## glm.pred   0   1
##        0 135  26
##        1  29  72
```

```r
mean(glm.pred != survived.test) # Test error rates
```

```
## [1] 0.2099237
```

```r
# Logistic regression is originated by linear regression, but used logistic function to give smooth cur

# According to the p-values of the logistic regression model, embark has a very high p-values
# , therefore, I am going to drop the embarked variable and rerun the model.


# A logistic model with variable selection
glm.fits <- glm(
  survived ~ . - embarked, data = titanic,
  family = binomial
)
summary(glm.fits)
```

```
##
## Call:
## glm(formula = survived ~ . - embarked, family = binomial, data = titanic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7021  -0.6419  -0.4592   0.6408   2.6062
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.205e+00  3.681e-01   8.707  < 2e-16 ***
## pclass2     -1.148e+00  2.429e-01  -4.726 2.29e-06 ***
## pclass3     -1.892e+00  2.378e-01  -7.958 1.75e-15 ***
## sexmale     -2.550e+00  1.578e-01 -16.167  < 2e-16 ***
## age         -3.238e-02  6.669e-03  -4.855 1.20e-06 ***
## sibsp1      -8.683e-02  1.819e-01  -0.477  0.63305
## sibsp2      -2.690e-01  4.283e-01  -0.628  0.52995
## sibsp3      -1.941e+00  6.181e-01  -3.140  0.00169 **
## sibsp4      -2.075e+00  7.159e-01  -2.899  0.00374 **
## sibsp5      -1.662e+01  8.417e+02  -0.020  0.98424
## sibsp8      -1.631e+01  6.773e+02  -0.024  0.98079
## parch1       7.184e-01  2.382e-01   3.016  0.00256 **
## parch2       3.177e-01  3.000e-01   1.059  0.28953
## parch3       4.007e-01  8.461e-01   0.474  0.63579
## parch4      -1.850e+00  1.203e+00  -1.538  0.12397
## parch5      -1.214e+00  1.152e+00  -1.054  0.29188
## parch6      -1.554e+01  1.439e+03  -0.011  0.99138
## parch9      -1.598e+01  1.426e+03  -0.011  0.99106
## fare         1.669e-03  1.909e-03   0.874  0.38185
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1740.1  on 1307  degrees of freedom
## Residual deviance: 1179.2  on 1289  degrees of freedom
## AIC: 1217.2
##
## Number of Fisher Scoring iterations: 15
```

```
# Predict glm
glm.pred_prop <- predict(glm.fits, newdata = titanic.test, type = "response")

glm.pred <- as.factor(ifelse(glm.pred_prop < 0.5 , "0", "1")) # < 0.5 puts it as "0" else "1"

table(glm.pred, survived.test)
```

```
##         survived.test
## glm.pred   0   1
##        0 137  27
##        1  27  71
```

```
mean(glm.pred != survived.test) # Test error rates
```

```
## [1] 0.2061069
```

```
# As a result, the test errors are slightly decreased from 20.99% to 20.61%.
```