Kaggle Titanic

Titanic Survivor Prediction

Data loading and organization

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
train <- read.csv("Datasets/train.csv", stringsAsFactors = TRUE, na.strings = "")
test <- read.csv("Datasets/test.csv", stringsAsFactors = TRUE)

test$Survived <- NA

dat <- rbind(train,test)

summary(dat)
```

```
##   PassengerId      Survived          Pclass
##  Min.   :   1   Min.   :0.0000   Min.   :1.000
##  1st Qu.: 328   1st Qu.:0.0000   1st Qu.:2.000
##  Median : 655   Median :0.0000   Median :3.000
##  Mean   : 655   Mean   :0.3838   Mean   :2.295
##  3rd Qu.: 982   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :1309   Max.   :1.0000   Max.   :3.000
##                   NA's    :418
##                                  Name             Sex             Age
##  Connolly, Miss. Kate              :   2   female:466   Min.   : 0.17
##  Kelly, Mr. James                  :   2   male  :843   1st Qu.:21.00
##  Abbing, Mr. Anthony               :   1                Median :28.00
##  Abbott, Mr. Rossmore Edward       :   1                Mean   :29.88
##  Abbott, Mrs. Stanton (Rosa Hunt):   1                3rd Qu.:39.00
##  Abelson, Mr. Samuel               :   1                Max.   :80.00
##  (Other)                           :1301                NA's   :263
##      SibSp            Parch            Ticket          Fare
##  Min.   :0.0000   Min.   :0.000   CA. 2343:  11   Min.   :  0.000
##  1st Qu.:0.0000   1st Qu.:0.000   1601    :   8   1st Qu.:  7.896
##  Median :0.0000   Median :0.000   CA 2144 :   8   Median : 14.454
##  Mean   :0.4989   Mean   :0.385   3101295 :   7   Mean   : 33.295
##  3rd Qu.:1.0000   3rd Qu.:0.000   347077  :   7   3rd Qu.: 31.275
##  Max.   :8.0000   Max.   :9.000   347082  :   7   Max.   :512.329
##                                   (Other) :1261   NA's   :1
##             Cabin      Embarked
##                :327   C   :270
##  C23 C25 C27    :  6   Q   :123
##  B57 B59 B63 B66:  5   S   :914
##  G6             :  5   NA's:  2
##  B96 B98        :  4
##  (Other)        :275
##  NA's           :687
```

```
#convert survived and pclass to factor variable
dat$Survived <- as.factor(dat$Survived)
dat$Pclass <- as.factor(dat$Pclass)

#Cabin Na values -> 0, otherwise 1
dat$Cabin <- as.factor(ifelse(is.na(dat$Cabin), 0, 1))

#Gender -> male = 0, female = 1
dat$Sex <- as.factor(ifelse(dat$Sex == "male", 0, 1))

#NA values -> mean
dat$Age[is.na(dat$Age)] <- mean(dat$Age, na.rm=TRUE)
dat$Fare[is.na(dat$Fare)] <- mean(dat$Fare, na.rm=TRUE)
dat$Embarked[is.na(dat$Embarked)] <- "S"


#family size (if family = 1, then it's alone)
dat$family <- dat$SibSp + dat$Parch + 1


#converting names
name <- dat$Name
name <- sub(".*, ", "", name)
name <- sub("\\..*", "", name)
head(name)
```

```
## [1] "Mr"   "Mrs"  "Miss" "Mrs"  "Mr"   "Mr"
```

```
dat$Name <- as.character(name)

dat$Name[dat$Name %in% c("Capt","Col","Don","Dr","Jonkheer", "Lady", "Major","Mlle","Mme","Ms","Rev","S

dat$Name <- as.factor(dat$Name)
table(dat$Name)
```

```
##
##  Master    Miss      Mr     Mrs unknown
##      61     260     757     197      34
```

```
#dropping variables
dat <- subset(dat, select = -c(PassengerId, SibSp, Parch, Ticket))

names(dat)
```

```
## [1] "Survived" "Pclass"   "Name"     "Sex"      "Age"      "Fare"
## [7] "Cabin"    "Embarked" "family"
```

```
summary(dat)
```

```
##  Survived   Pclass       Name     Sex          Age
##  0   :549   1:323   Master : 61   0:843   Min.   : 0.17
##  1   :342   2:277   Miss   :260   1:466   1st Qu.:22.00
##  NA's:418   3:709   Mr     :757           Median :29.88
##                     Mrs    :197           Mean   :29.88
##                     unknown: 34           3rd Qu.:35.00
##                                           Max.   :80.00
##       Fare          Cabin   Embarked     family
##  Min.   :  0.000   0:687   C:270    Min.   : 1.000
##  1st Qu.:  7.896   1:622   Q:123    1st Qu.: 1.000
##  Median : 14.454           S:916    Median : 1.000
##  Mean   : 33.295                    Mean   : 1.884
##  3rd Qu.: 31.275                    3rd Qu.: 2.000
##  Max.   :512.329                    Max.   :11.000
```

```
train1 <- dat[1:891,] #train.csv
test1 <- dat[892:1309,] #test.csv


set.seed(125)

split<-createDataPartition(y=train1$Survived,p=0.7,list=FALSE)

training <- train1[split,] #training set of train data
testing <- train1[-split,] #test set of train data, which is validation set

#test1 is the test dataset
```

Data Exploration

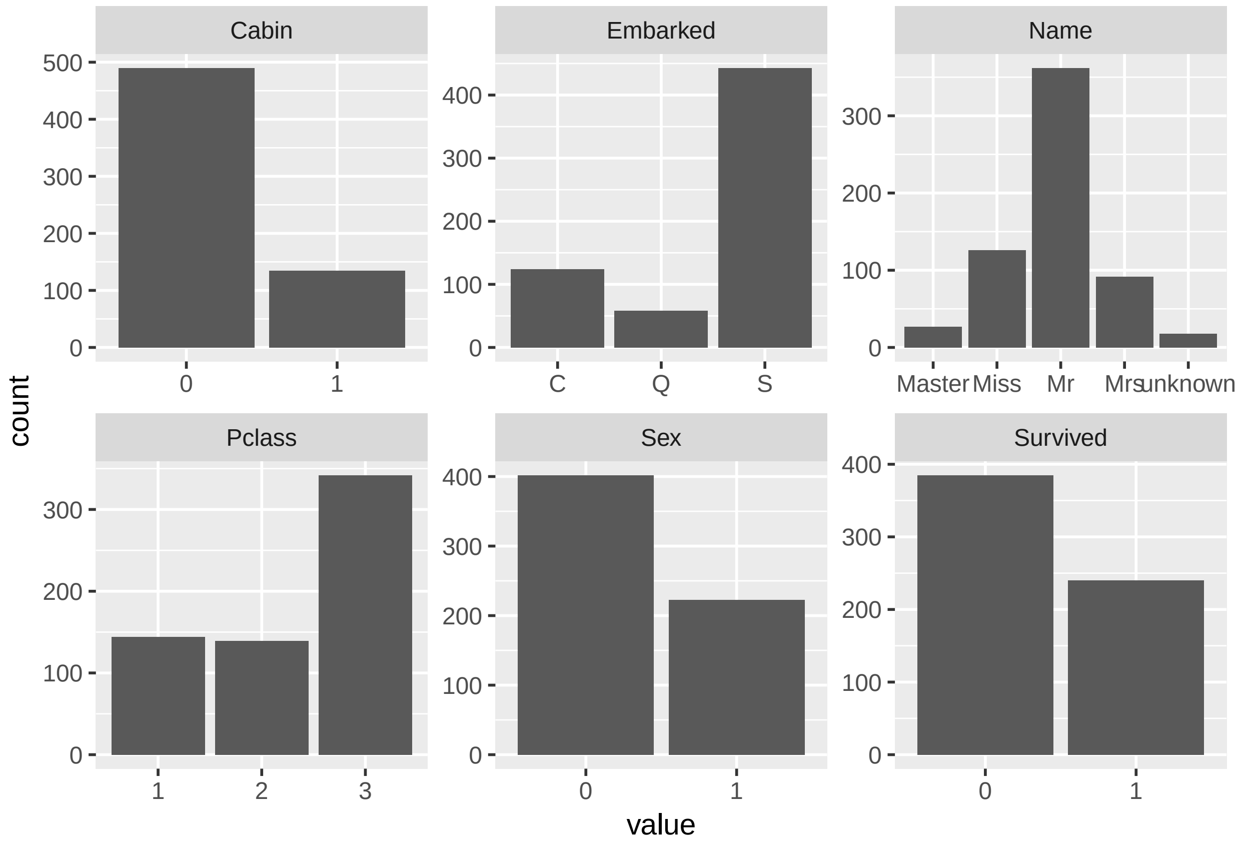
```
library(dplyr)
library(purrr)
```

```
##
```

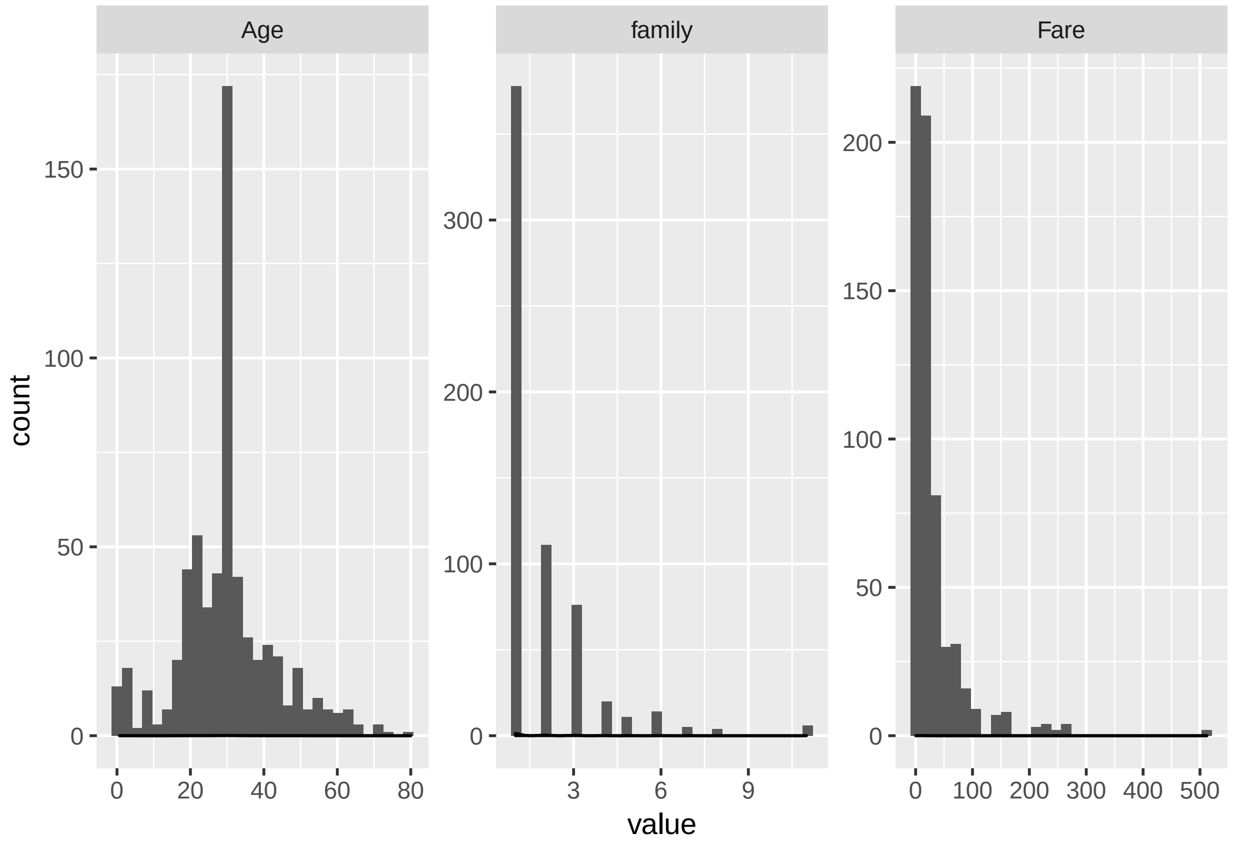
```
## Attaching package: 'purrr'

## The following object is masked from 'package:caret':
##
##     lift
```

```
library(tidyr)
library(ggplot2)


###all variables
training %>% keep(is.factor) %>% gather() %>%
  ggplot(aes(value)) +
  facet_wrap(~key, scales="free")+
  geom_bar()
```

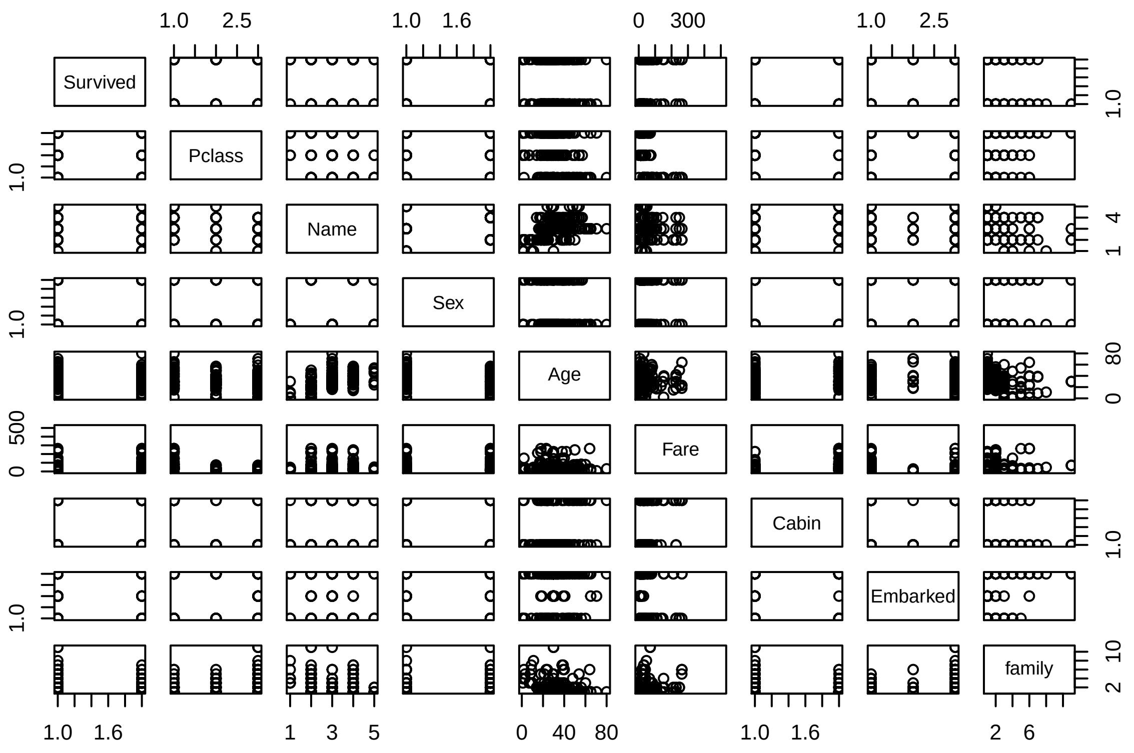
```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```



```
training %>% keep(is.numeric) %>% gather() %>%
  ggplot(aes(value)) +
  facet_wrap(~key, scales="free")+
  geom_histogram(bins=30)+
  geom_density()
```



```r
pairs(training,col=train$Survived)
```



Logistic Regression

```r
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma
```

```r
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
logistic.fit <- glm(Survived ~., data=training, family=binomial)

summary(logistic.fit)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial, data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3109  -0.5738  -0.4029   0.5231   2.4922
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   4.052945   0.891576   4.546 5.47e-06 ***
## Pclass2      -0.283878   0.498261  -0.570 0.568856
## Pclass3      -1.295398   0.499055  -2.596 0.009440 **
## NameMiss    -16.581174 639.499709  -0.026 0.979314
## NameMr       -3.749260   0.651305  -5.757 8.59e-09 ***
## NameMrs     -15.786362 639.499776  -0.025 0.980306
## Nameunknown  -3.735382   0.973263  -3.838 0.000124 ***
## Sex1         15.702310 639.499405   0.025 0.980411
## Age          -0.019111   0.010537  -1.814 0.069724 .
## Fare          0.003516   0.003241   1.085 0.278094
## Cabin1        0.860558   0.425625   2.022 0.043190 *
## EmbarkedQ    -0.072022   0.462585  -0.156 0.876273
## EmbarkedS    -0.392620   0.299850  -1.309 0.190403
## family       -0.502213   0.107321  -4.680 2.88e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 832.49  on 624  degrees of freedom
## Residual deviance: 512.66  on 611  degrees of freedom
## AIC: 540.66
##
## Number of Fisher Scoring iterations: 14
```

```
logistic.probs <- predict(logistic.fit, newdata = testing, type="response")

logistic.pred <- ifelse(logistic.probs > 0.5, 1, 0)

test.pred <- testing$Survived

table(logistic.pred, test.pred)
```

```
##                test.pred
## logistic.pred   0   1
##               0 148  26
##               1  16  76
```

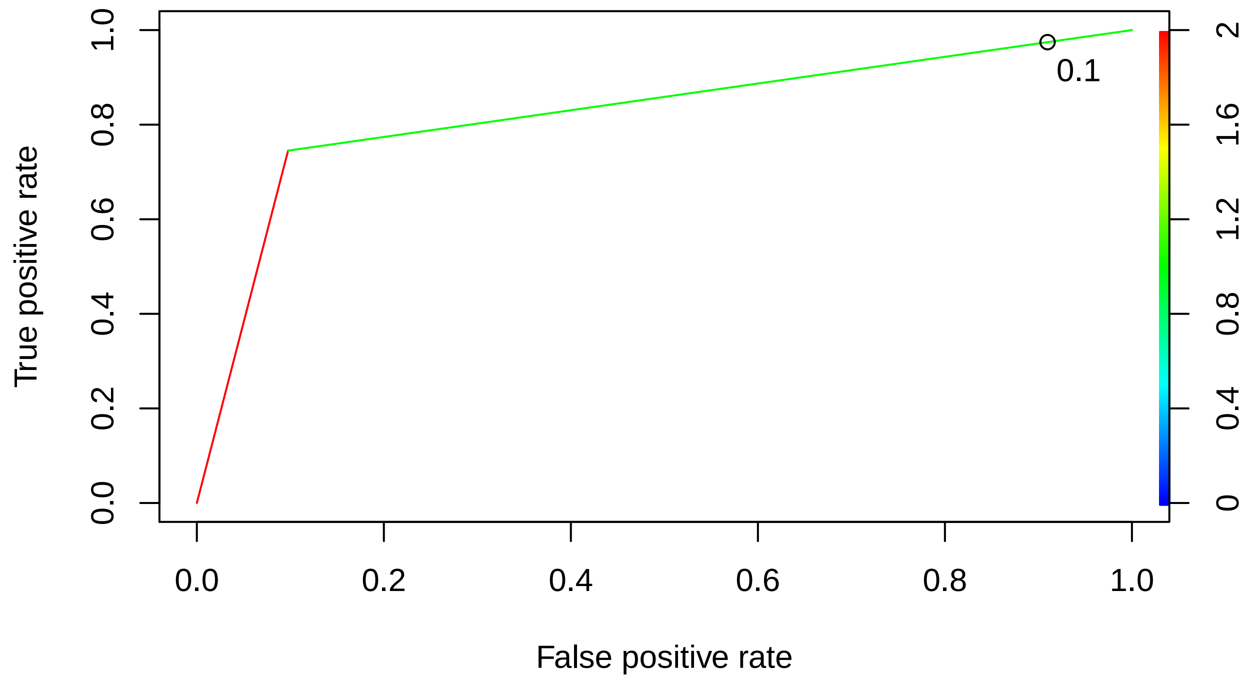
```
sum(diag(table(logistic.pred, test.pred)))/nrow(testing)
```

```
## [1] 0.8421053
```

```
#accuracy = 84.21%

#cv error with 10 folds
cv.glm(training, logistic.fit, K=10)$delta[1]
```

```
## [1] 0.1336628
```

```
#around 13% error rate

ROCpred <- prediction(logistic.pred, test.pred)
ROCperf <- performance(ROCpred, "tpr", "fpr")
plot(ROCperf, colorize = TRUE, print.cutoffs.at = seq(0.1,0.1), text.adj = c(-0.2,1.7))
```



```r
performance(ROCpred, "auc")@y.values
```

```
## [[1]]
## [1] 0.8237685
```

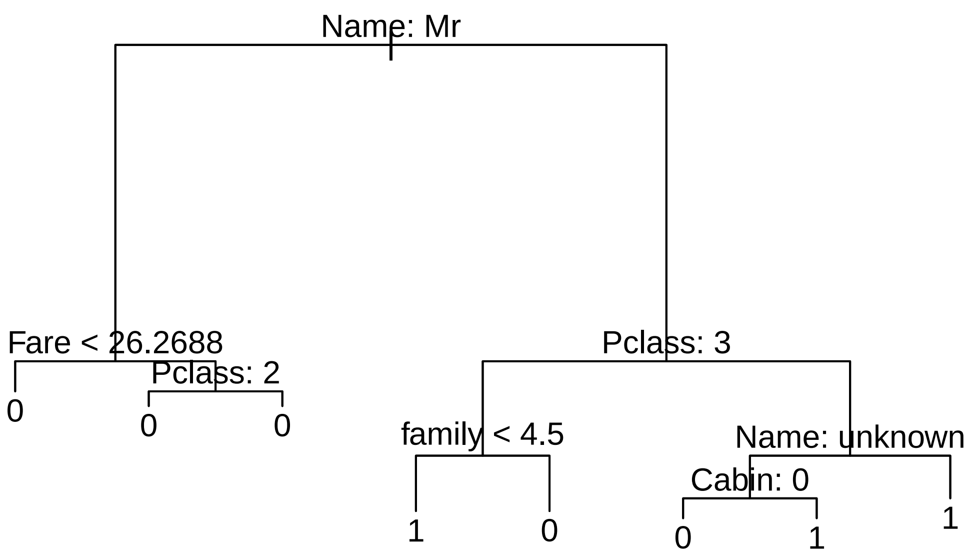
```r
#AUC value = 0.8237

confusionMatrix(as.factor(logistic.pred), test.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 148  26
##          1  16  76
##
##                Accuracy : 0.8421
##                  95% CI : (0.7926, 0.8838)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 6.804e-16
##
##                   Kappa : 0.6598
##
##  Mcnemar's Test P-Value : 0.1649
##
##             Sensitivity : 0.9024
##             Specificity : 0.7451
##          Pos Pred Value : 0.8506
##          Neg Pred Value : 0.8261
##              Prevalence : 0.6165
##          Detection Rate : 0.5564
##    Detection Prevalence : 0.6541
##       Balanced Accuracy : 0.8238
##
##        'Positive' Class : 0
##
```

Logistic Regression Accuracy -> 84.21%

Decision Tree

```r
library(tree)

tree.training <- tree(Survived~., training)
plot(tree.training);text(tree.training,pretty=0)
```



```r
summary(tree.training)
```

```
##
## Classification tree:
## tree(formula = Survived ~ ., data = training)
## Variables actually used in tree construction:
## [1] "Name"   "Fare"   "Pclass" "family" "Cabin"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7791 = 480.7 / 617
## Misclassification error rate: 0.1632 = 102 / 625
```

```r
tree.pred=predict(tree.training,testing,type="class") #with test datasets

table(tree.pred , testing$Survived)
```

```
##
## tree.pred   0   1
##         0 147  28
##         1  17  74
```

```r
sum(diag(table(tree.pred, testing$Survived)))/nrow(testing)
```

```
## [1] 0.8308271
```

```r
#accuracy 83.08%

deviance(tree.training)
```

```
## [1] 480.6808
```

```r
misclass.tree(tree.training)
```

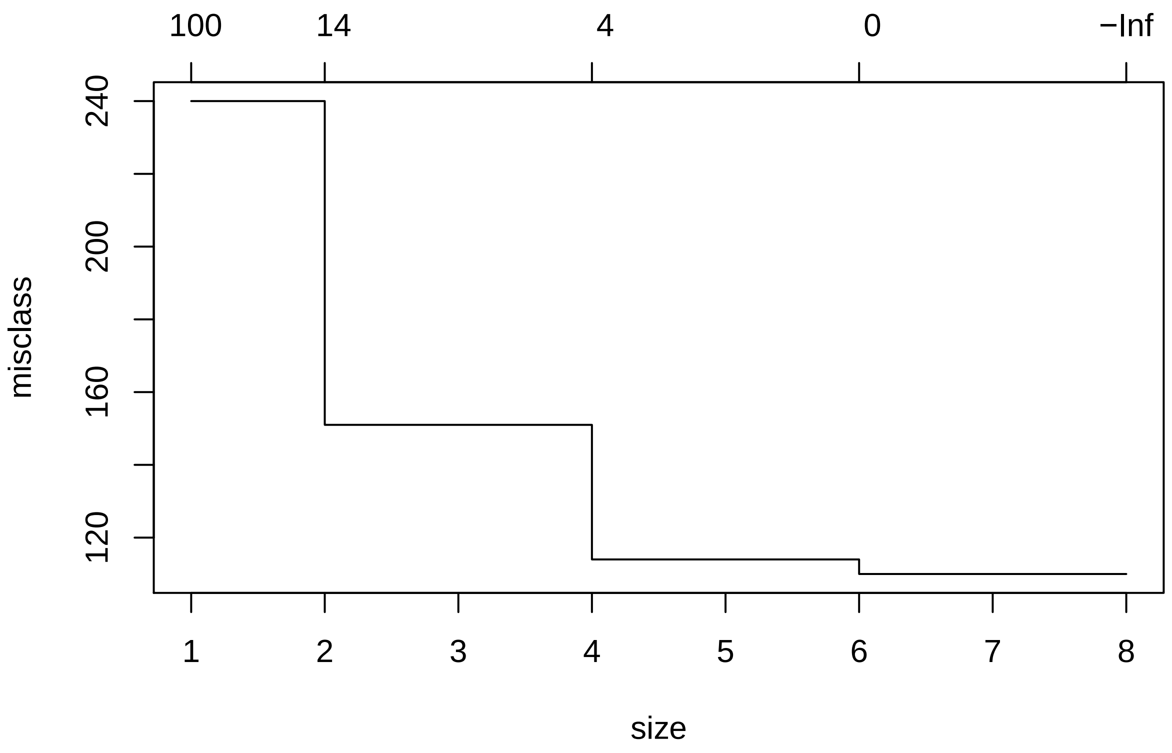
```
## [1] 102
```

```r
#pruning via cv
cv.training=cv.tree(tree.training,FUN=prune.misclass)
cv.training
```

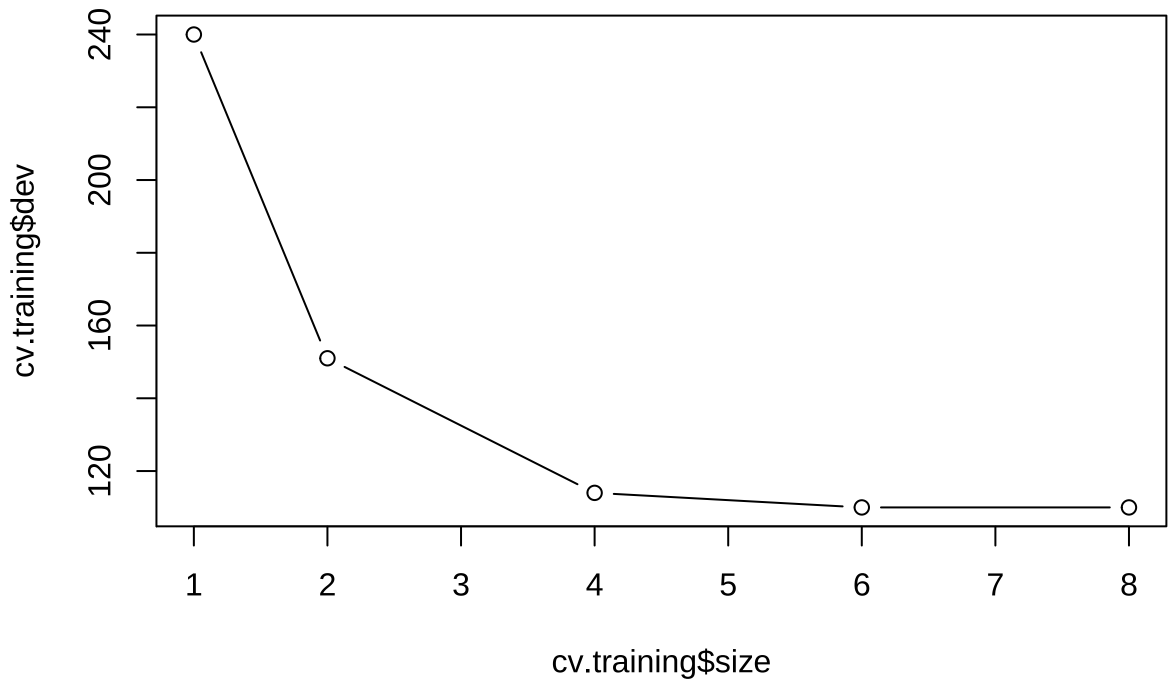
```
## $size
## [1] 8 6 4 2 1
```

```
##
## $dev
## [1] 110 110 114 151 240
##
## $k
## [1]  -Inf   0.0   4.0  13.5 103.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

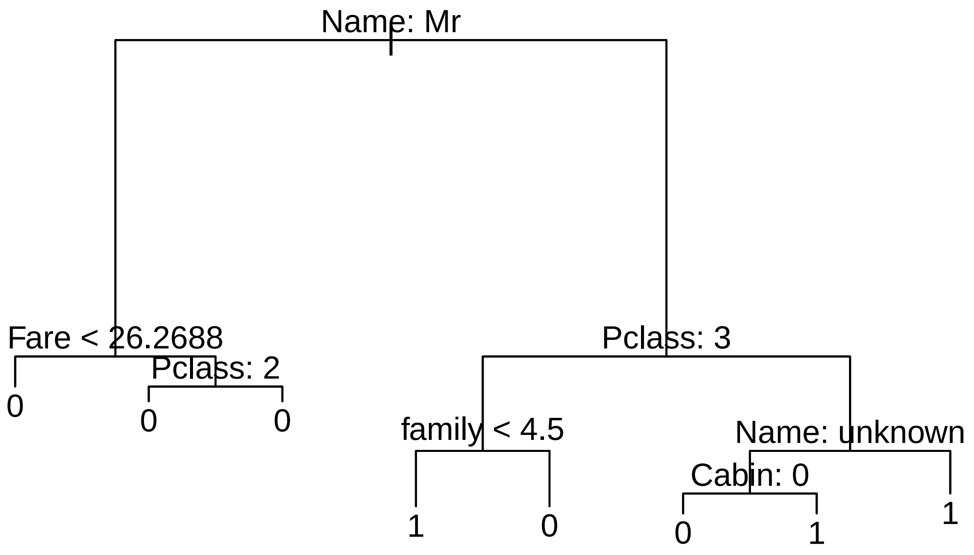
```
plot(cv.training)
```



```
plot(cv.training$size, cv.training$dev, type="b")
```



```
prune.training <- prune.misclass(tree.training,best=7)#actually same with tree
plot(prune.training);text(prune.training,pretty=0)
```



```
#pruned prediction
tree.pred <- predict(prune.training, testing, type="class")
table(tree.pred, testing$Survived)
```

```
##
## tree.pred   0   1
##         0 147  28
##         1  17  74
```

```
sum(diag(table(tree.pred, testing$Survived)))/nrow(testing)
```

```
## [1] 0.8308271
```

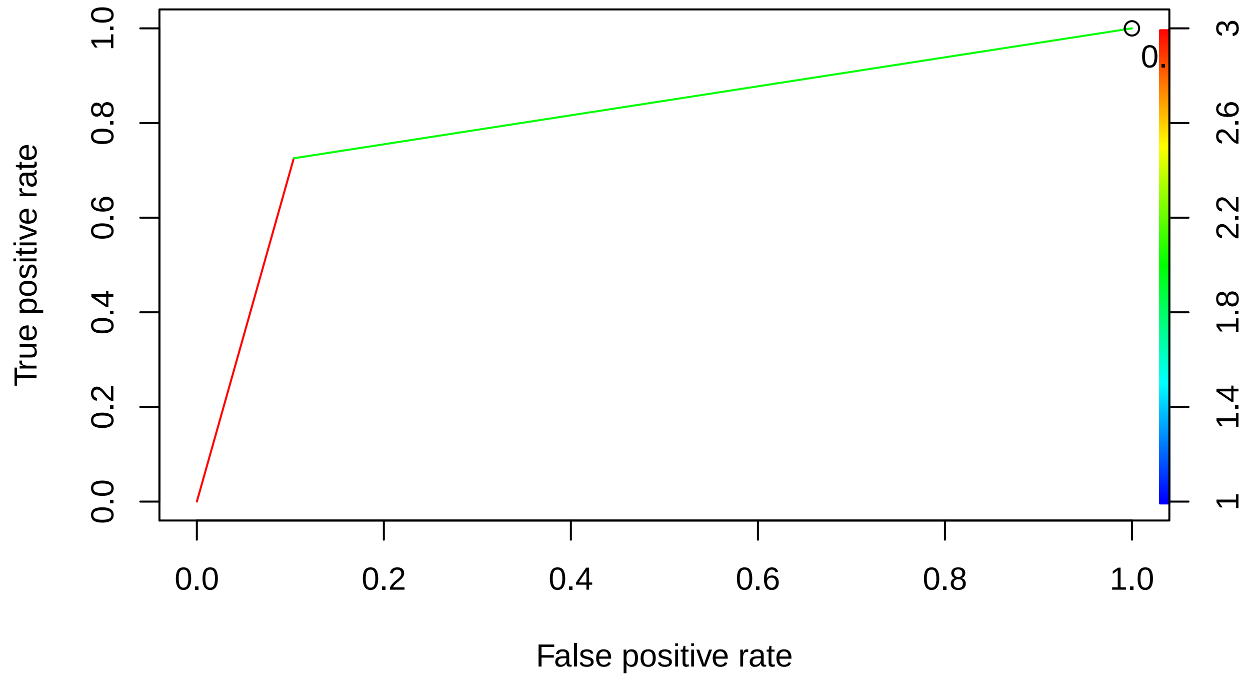
```r
#83.08%

class(tree.pred)
```

```
## [1] "factor"
```

```r
class(test.pred)
```

```
## [1] "factor"
```

```r
ROCpred <- prediction(as.numeric(tree.pred), test.pred)
ROCperf <- performance(ROCpred, "tpr", "fpr")
plot(ROCperf, colorize = TRUE, print.cutoffs.at = seq(0.1,0.1), text.adj = c(-0.2,1.7))
```



```r
performance(ROCpred, "auc")@y.values
```

```
## [[1]]
## [1] 0.8109158
```

```r
#auc = 0.8109

confusionMatrix(tree.pred, test.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 147  28
##          1  17  74
##
##                Accuracy : 0.8308
##                  95% CI : (0.7803, 0.8738)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 2.211e-14
##
```

```
##                  Kappa : 0.6348
##
##  Mcnemar's Test P-Value : 0.136
##
##            Sensitivity : 0.8963
##            Specificity : 0.7255
##         Pos Pred Value : 0.8400
##         Neg Pred Value : 0.8132
##             Prevalence : 0.6165
##         Detection Rate : 0.5526
##   Detection Prevalence : 0.6579
##      Balanced Accuracy : 0.8109
##
##       'Positive' Class : 0
##
```

Decision Tree Accuracy -> 83.08%

Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

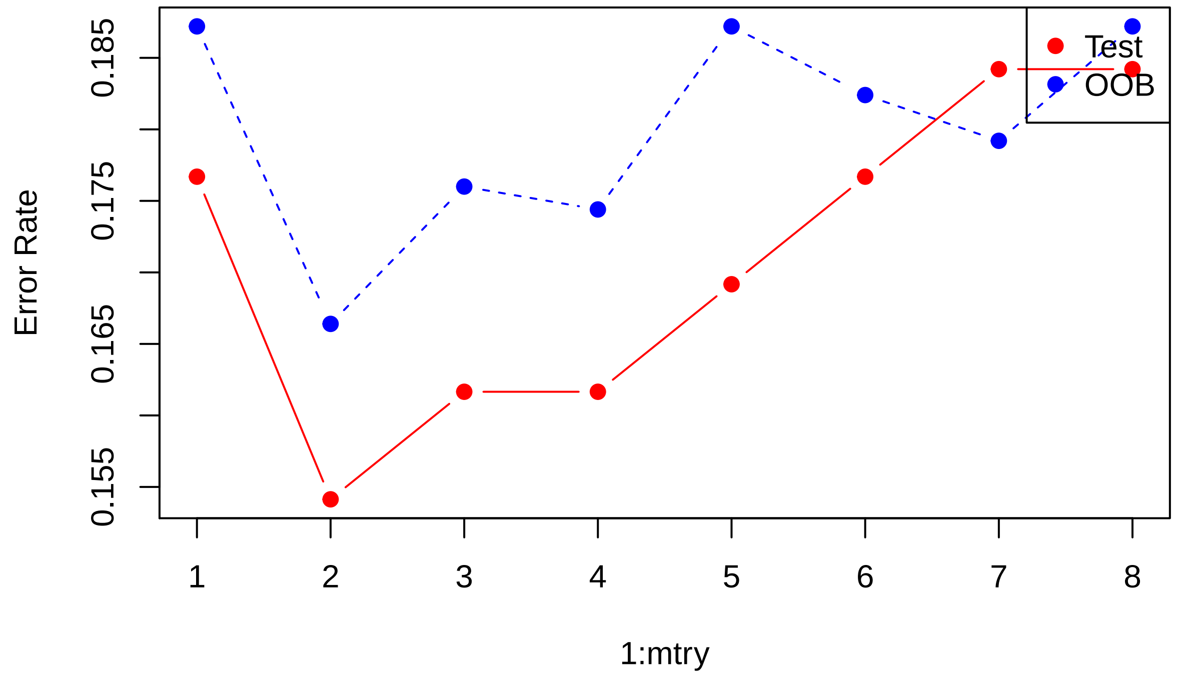
```
library(MASS)



#base random forest
rf.training <- randomForest(Survived~., data=training, importance=TRUE)
rf.training
```

```
##
## Call:
##  randomForest(formula = Survived ~ ., data = training, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 17.44%
## Confusion matrix:
##     0   1 class.error
## 0 348  37   0.0961039
## 1  72 168   0.3000000
```

```
#2 mtry - classification defaults = sqrt(p) = sqrt(8) ~= 2, n.tree = 500, oob error rate = aroun 17%

#find best mtry
oob.err=double(8)
test.err=double(8)

for(mtry in 1:8){
  fit <- randomForest(Survived~., data=training, mtry=mtry, ntree=500)
  oob.err[mtry] <- 1-sum(diag(table(training$Survived,fit$predicted)))/nrow(training)
  pred <- predict(fit, testing)
  test.err[mtry] <- 1-sum(diag(table(testing$Survived,pred)))/nrow(testing)
  cat(mtry," ")
}
```

```
## 1  2  3  4  5  6  7  8
```

```
matplot(1:mtry,cbind(test.err,oob.err),pch=19,col=c("red","blue"),type="b",ylab="Error Rate")
legend("topright",legend=c("Test","OOB"),pch=19,col=c("red","blue"))
```



```
#2 mtry might be the best

#updated random forest
rf.training.update <- randomForest(Survived~., data=training, mtry=2, importance=TRUE, ntree=500)
rf.training.update
```

```
##
## Call:
##  randomForest(formula = Survived ~ ., data = training, mtry = 2,      importance = TRUE, ntree = 500
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
```

```
##         OOB estimate of  error rate: 17.44%
## Confusion matrix:
##     0   1 class.error
## 0 349  36  0.09350649
## 1  73 167  0.30416667
```

```
rf.pred <- predict(rf.training.update, testing)
table(testing$Survived, rf.pred)
```

```
##    rf.pred
##       0   1
##   0 153  11
##   1  30  72
```

```
sum(diag(table(testing$Survived,rf.pred)))/nrow(testing)
```

```
## [1] 0.8458647
```

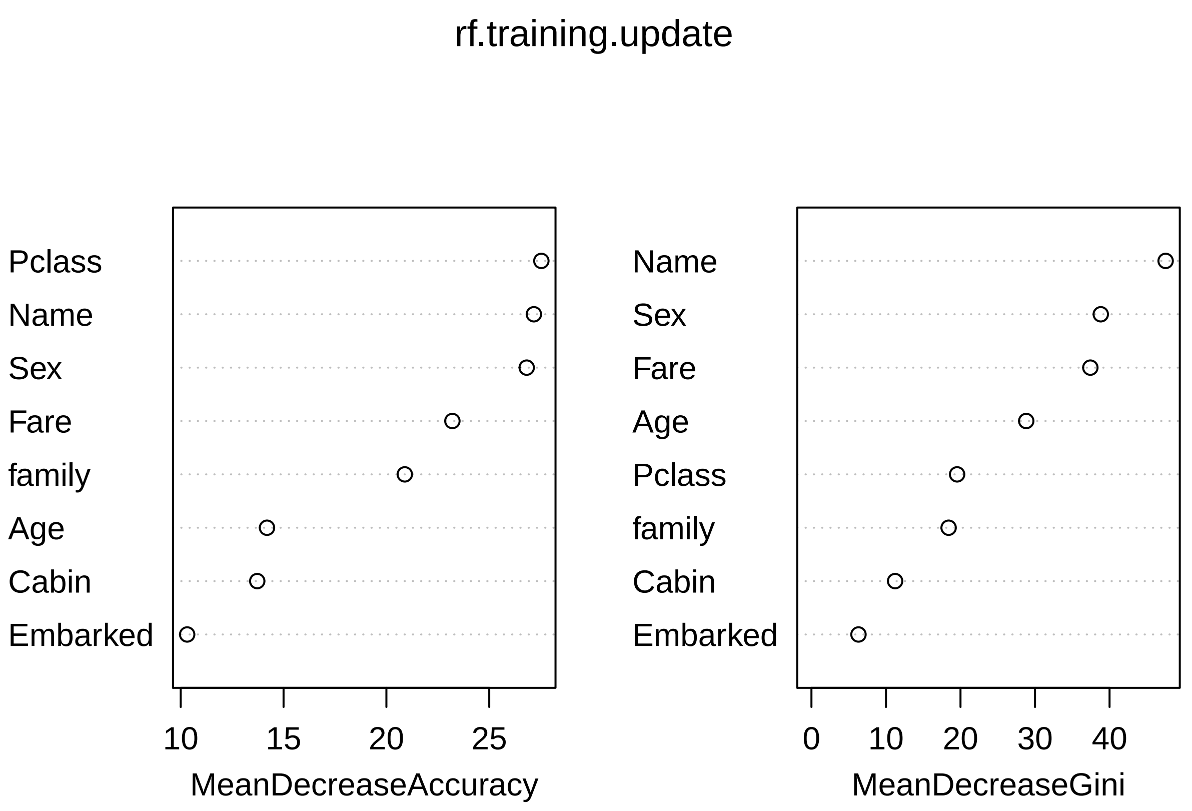
```
#around 84.58% accuracy
#since random forest provides different accuracy in every try, I made a function to get a mean value of

mean.acc <- function(training, testing, mtry, ntree, try, formula){
  acc <- NULL

  for(i in 1:try){
  randomFtrain <- randomForest(formula, data=training, mtry=mtry, importance=TRUE, ntree=ntree)
  randomFpred <- predict(randomFtrain, testing)
  acc[i] <- sum(diag(table(testing$Survived, randomFpred)))/nrow(testing)
  }

  return(mean(acc))
}

#2 mtry, 500 ntree, 10 tries
mean.acc(training = training, testing = testing, mtry = 2, ntree = 500, try = 10, formula = as.formula(
```

```
## [1] 0.8454887
```

```
#around 84.5% accuracy

importance(rf.training.update)
```

```
##                  0         1 MeanDecreaseAccuracy MeanDecreaseGini
## Pclass   18.134499 20.011172             27.53240        19.545816
## Name     23.064960 20.999153             27.16987        47.525815
## Sex      23.792787 20.190242             26.82020        38.822613
## Age      11.570976  7.040736             14.19494        28.820272
## Fare     14.556035 15.535657             23.20695        37.421976
## Cabin    12.280560  1.776714             13.72102        11.226495
## Embarked  6.638619  7.246604             10.31469         6.311426
## family   21.015530  5.218679             20.89440        18.407316
```

```
varImpPlot(rf.training.update)
```



```r
#we can improve our RF model by removing Embarked
rf.update <- randomForest(Survived~.-Embarked, data=training, mtry=2, importance=TRUE, ntree=500)
rf.update
```

```
##
## Call:
##  randomForest(formula = Survived ~ . - Embarked, data = training,      mtry = 2, importance = TRUE, 
##               Type of random forest: classification
##                     Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 16.8%
## Confusion matrix:
##     0   1 class.error
## 0 353  32  0.08311688
## 1  73 167  0.30416667
```

```r
rf.pred <- predict(rf.update, testing)
table(testing$Survived, rf.pred)
```

```
##    rf.pred
##       0   1
##   0 152  12
##   1  30  72
```

```r
sum(diag(table(testing$Survived,rf.pred)))/nrow(testing)
```

```
## [1] 0.8421053
```

```r
#around 84%
```

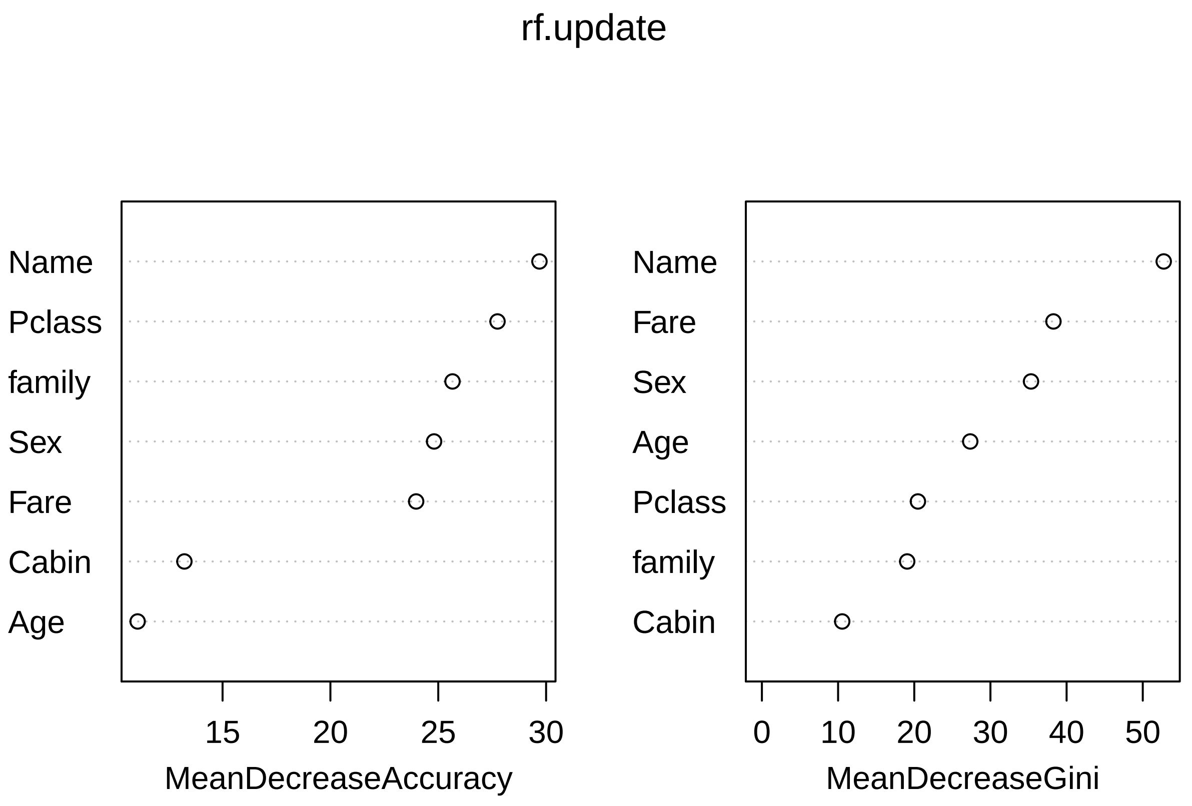
```
mean.acc(training, testing, mtry = 2, ntree = 500, try = 10, formula = as.formula(Survived~.-Embarked))
```

```
## [1] 0.8421053
```

```
#accruacy around 84.3%

importance(rf.update)
```

```
##                0         1 MeanDecreaseAccuracy MeanDecreaseGini
## Pclass 19.281870 20.462255             27.74701         20.47984
## Name   26.192644 23.155114             29.69098         52.75031
## Sex    22.439350 19.452844             24.80557         35.32776
## Age     7.367302  8.111400             11.06260         27.34985
## Fare   14.755373 17.864728             23.97459         38.27450
## Cabin  12.514360  1.968030             13.22990         10.53989
## family 24.767579  7.058626             25.66010         19.07239
```

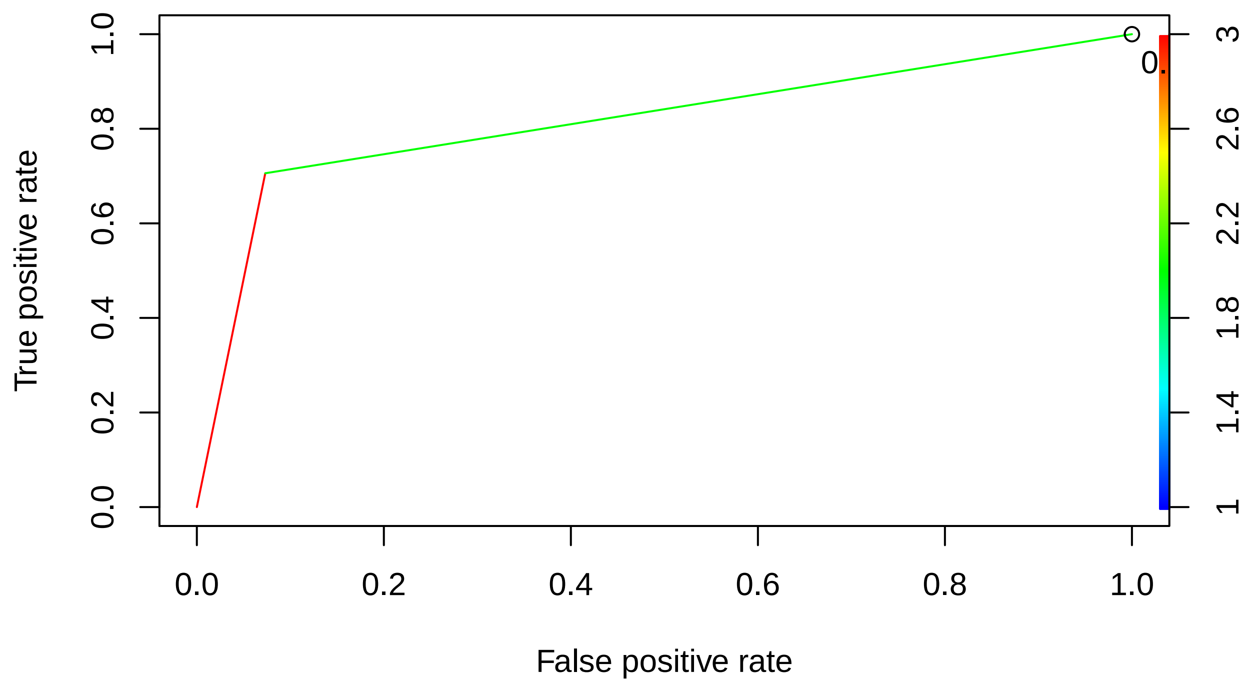
```
varImpPlot(rf.update)
```



```
class(rf.pred)
```

```
## [1] "factor"
```

```
ROCpred <- prediction(as.numeric(rf.pred), test.pred)
ROCperf <- performance(ROCpred, "tpr", "fpr")
plot(ROCperf, colorize = TRUE, print.cutoffs.at = seq(0.1,0.1), text.adj = c(-0.2,1.7))
```



```r
performance(ROCpred, "auc")@y.values
```

```
## [[1]]
## [1] 0.8163558
```

```r
#auc ~= around 0.8212

confusionMatrix(rf.pred, test.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 152  30
##          1  12  72
##
##                Accuracy : 0.8421
##                  95% CI : (0.7926, 0.8838)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 6.804e-16
##
##                   Kappa : 0.6545
##
##  Mcnemar's Test P-Value : 0.008712
##
##             Sensitivity : 0.9268
##             Specificity : 0.7059
##          Pos Pred Value : 0.8352
##          Neg Pred Value : 0.8571
##              Prevalence : 0.6165
##          Detection Rate : 0.5714
##    Detection Prevalence : 0.6842
##       Balanced Accuracy : 0.8164
```

```
##
##        'Positive' Class : 0
##
```

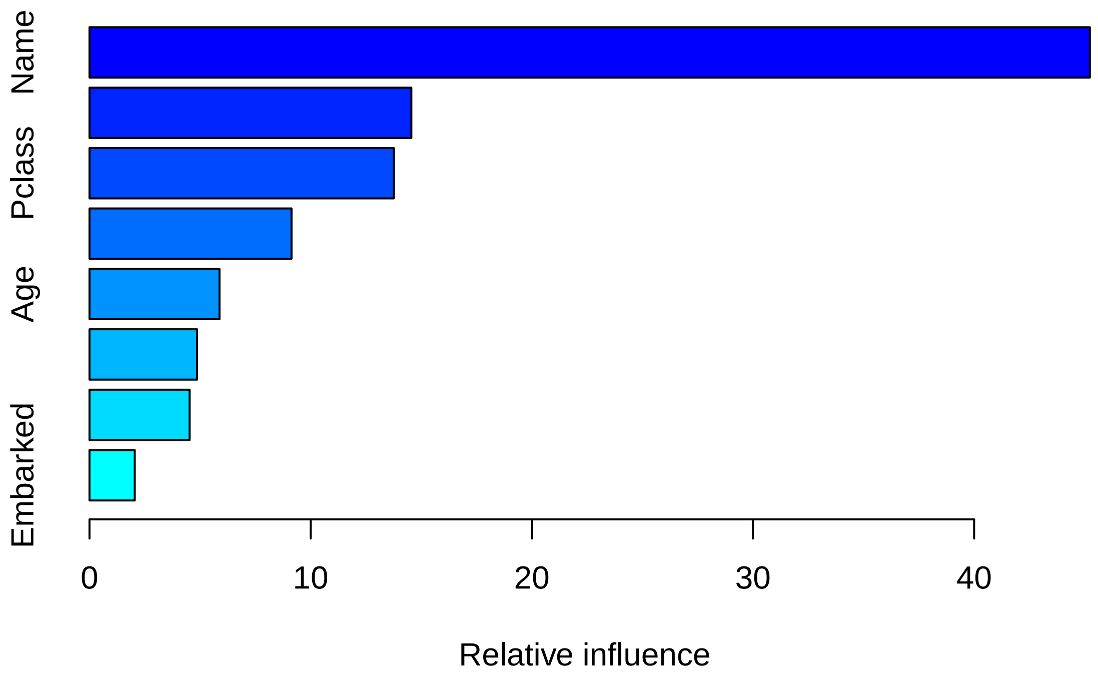
Random Forest Accuracy -> around 84.5%

boosting

```r
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```r
set.seed(123)
boost.training <- gbm(as.character(Survived)~., data=training,
                      distribution = "bernoulli", n.trees = 500,
                      interaction.depth = 3, shrinkage = 0.01, cv.folds=5)

summary(boost.training)
```



```
##               var   rel.inf
## Name         Name 45.234382
## Fare         Fare 14.554858
## Pclass     Pclass 13.760218
## family     family  9.136101
## Age           Age  5.878103
## Sex           Sex  4.864853
## Cabin       Cabin  4.525450
## Embarked Embarked  2.046035
```

```r
#test on the training set
boost.pred <- predict(boost.training, newdata = training, n.trees = 500, type = "response") #prob
boost.pred <- ifelse(boost.pred > 0.5, 1, 0)
```

```r
table(training$Survived, boost.pred)
```

```
##    boost.pred
##       0   1
##   0 356  29
##   1  54 186
```

```r
sum(diag(table(training$Survived, boost.pred)))/nrow(training)
```

```
## [1] 0.8672
```

```r
#86.08% accuracy


#test on the test set
boost.pred <- predict(boost.training, newdata = testing, n.trees = 500, type = "response")

boost.pred <- ifelse(boost.pred > 0.5, 1, 0)

table(testing$Survived, boost.pred)
```

```
##    boost.pred
##       0   1
##   0 148  16
##   1  29  73
```

```r
sum(diag(table(testing$Survived, boost.pred)))/nrow(testing)
```

```
## [1] 0.8308271
```

```r
#83.83% accuracy


#cross validation to get the best ntree
folds <- sample(rep(1:5, length = nrow(training)))
folds
```

```
##   [1] 4 2 1 3 3 2 4 2 4 3 3 1 2 1 2 3 2 5 1 3 2 5 1 3 1 3 1 5 4 2 2 3 5 1 4
##  [36] 5 2 2 5 2 1 4 1 3 4 3 1 3 5 4 2 5 2 3 1 1 1 1 3 5 1 4 5 4 1 4 1 4 1 1
##  [71] 5 5 3 2 2 2 3 1 2 5 2 1 2 1 4 5 2 5 4 5 1 4 2 5 3 4 3 1 5 1 2 3 1 4 1
## [106] 3 2 5 3 3 2 5 4 5 5 5 5 5 3 3 2 3 2 3 2 4 5 1 4 1 5 1 5 4 3 5 1 3 3 5
## [141] 4 5 2 5 3 3 1 4 1 1 2 2 4 5 1 2 3 1 3 2 4 5 1 3 4 5 5 5 1 3 3 1 4 5 1
## [176] 4 1 4 1 5 4 2 1 5 1 2 3 3 2 5 4 2 2 2 4 1 3 5 3 5 2 1 5 1 1 4 4 5 2 4
## [211] 4 4 4 2 1 4 2 5 4 4 4 3 1 5 3 1 4 1 1 5 4 2 5 3 3 2 2 5 4 2 3 1 2 5 1
## [246] 2 3 4 3 4 4 5 5 5 2 3 2 4 3 1 5 3 1 4 5 5 2 1 3 2 3 3 3 4 2 5 5 2 4 5
## [281] 5 3 2 2 1 4 5 2 1 4 1 3 5 1 5 1 4 3 3 3 4 3 5 4 4 1 1 1 3 3 2 5 4 4 1
## [316] 5 3 3 4 4 2 4 1 4 3 1 3 3 3 5 4 5 4 3 1 3 4 4 1 4 1 5 4 2 2 5 1 5 4 3
## [351] 3 3 5 2 2 5 3 1 3 5 4 1 1 3 5 3 3 2 2 2 3 5 4 4 2 3 2 5 4 2 3 1 1 2 4
## [386] 2 5 5 3 4 4 4 1 2 5 2 3 5 1 2 1 3 3 1 4 2 5 1 2 3 4 4 3 1 3 2 3 2 3 5
## [421] 1 2 3 4 1 1 1 5 1 3 1 2 5 3 5 3 1 2 2 3 5 2 4 5 2 3 1 3 4 2 3 2 3 1 1
## [456] 5 1 4 4 4 3 4 1 2 2 5 2 1 4 3 1 3 1 5 4 5 5 4 2 2 2 2 4 5 5 5 2 3 5 3
## [491] 3 4 3 1 2 2 4 4 5 4 5 2 1 2 3 4 2 3 1 2 1 2 5 1 3 2 2 1 4 5 3 2 2 4 5
## [526] 2 4 5 2 5 2 1 5 5 1 1 2 4 5 2 4 4 5 1 3 1 4 2 5 1 4 4 3 3 3 4 1 1 1 4
## [561] 5 2 2 1 4 1 1 5 4 3 4 3 4 5 2 3 3 2 4 3 1 1 2 5 2 3 5 5 4 5 4 1 4 5 1
## [596] 5 4 4 4 1 2 1 1 5 2 3 4 3 4 5 2 2 4 2 5 2 5 3 3 5 4 5 2 3 4
```

```r
table(folds)
```

```
## folds
##   1   2   3   4   5
## 125 125 125 125 125
```

```r
ntree <- seq(500, 4000, by=500)
cv.errors <- matrix(NA, length(ntree),5)
cv.errors <- cbind(ntree, cv.errors)
cv.errors
```

```
##      ntree
## [1,]   500 NA NA NA NA NA
## [2,]  1000 NA NA NA NA NA
## [3,]  1500 NA NA NA NA NA
## [4,]  2000 NA NA NA NA NA
## [5,]  2500 NA NA NA NA NA
## [6,]  3000 NA NA NA NA NA
## [7,]  3500 NA NA NA NA NA
## [8,]  4000 NA NA NA NA NA
```

```r
for(i in 1:5){
  for(j in ntree){
    boost.fit <- gbm(as.character(Survived)~., data = training[folds != i,], distribution = "bernoulli"
    boost.pred <- predict(boost.fit, training[folds == i,], n.trees = j)
    boost.pred <- ifelse(boost.pred > 0.5, 1, 0)
    cv.errors[which(j==ntree), i+1] = 1- sum(diag(table(training$Survived[folds==i], boost.pred)))/nrow
  }
}

cv.errors
```

```
##      ntree
## [1,]   500 0.240 0.144 0.200 0.184 0.152
## [2,]  1000 0.232 0.128 0.208 0.176 0.160
## [3,]  1500 0.248 0.136 0.208 0.160 0.168
## [4,]  2000 0.232 0.136 0.208 0.160 0.168
## [5,]  2500 0.256 0.144 0.216 0.160 0.160
## [6,]  3000 0.248 0.128 0.216 0.168 0.144
## [7,]  3500 0.256 0.136 0.232 0.176 0.152
## [8,]  4000 0.256 0.144 0.224 0.176 0.168
```

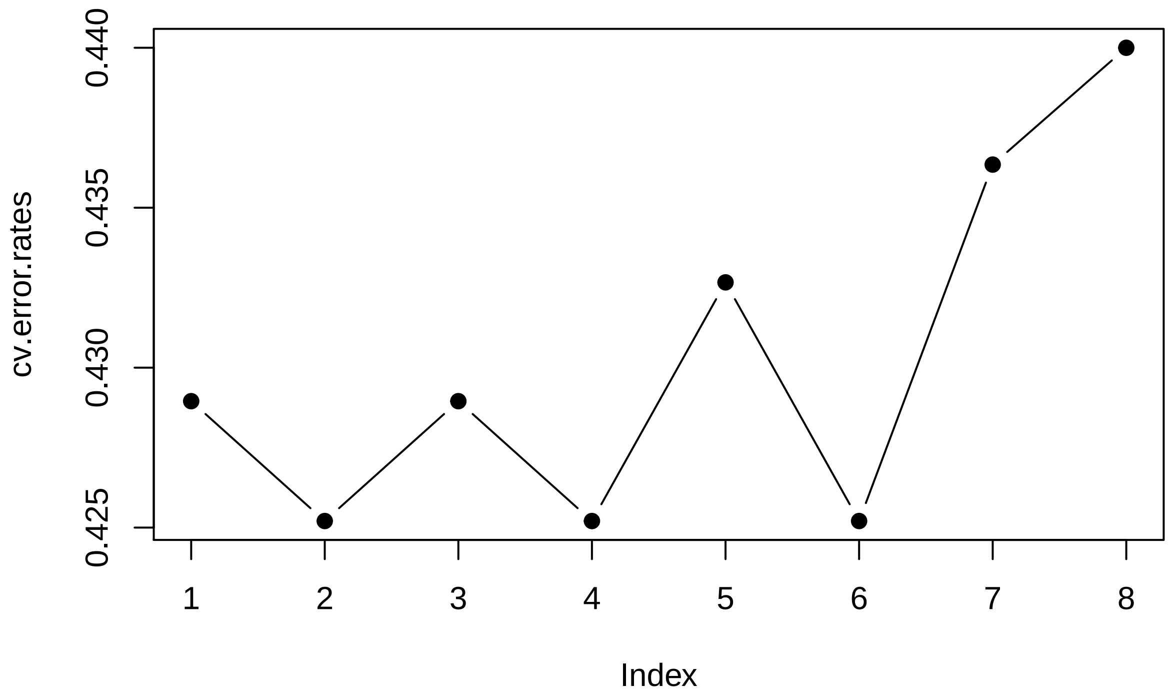
```r
cv.errors1 <- cv.errors[,-1]
cv.errors1
```

```
##
## [1,] 0.240 0.144 0.200 0.184 0.152
## [2,] 0.232 0.128 0.208 0.176 0.160
## [3,] 0.248 0.136 0.208 0.160 0.168
## [4,] 0.232 0.136 0.208 0.160 0.168
## [5,] 0.256 0.144 0.216 0.160 0.160
## [6,] 0.248 0.128 0.216 0.168 0.144
## [7,] 0.256 0.136 0.232 0.176 0.152
## [8,] 0.256 0.144 0.224 0.176 0.168
```

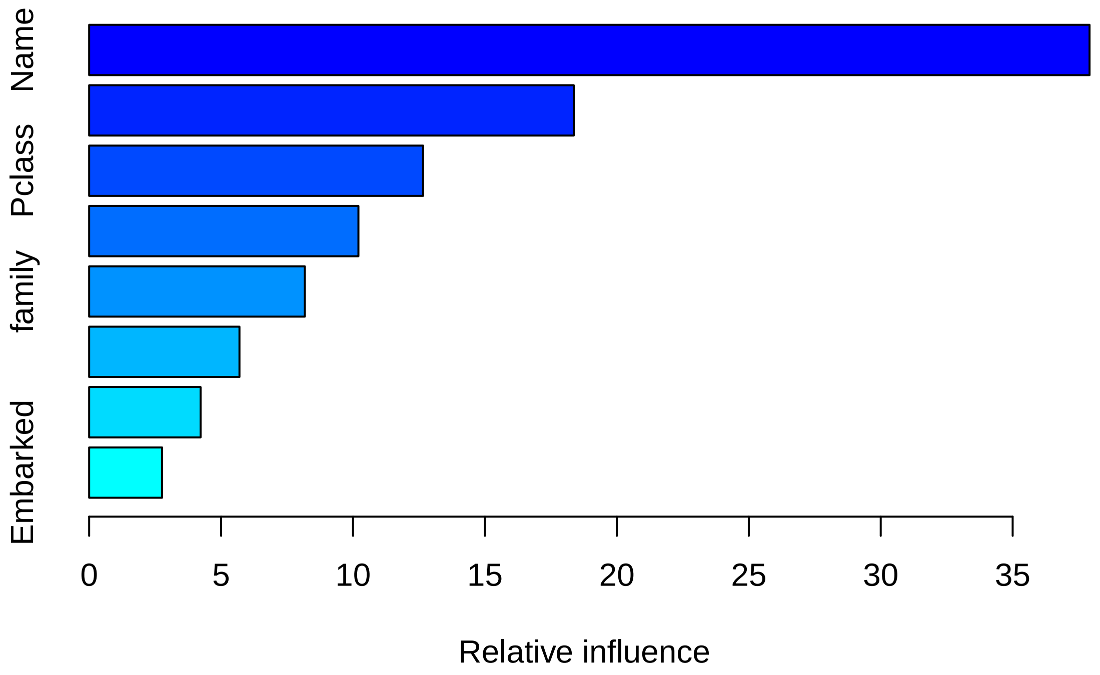
```r
cv.error.rates=sqrt(apply(cv.errors1,1,mean))
which.min(cv.error.rates)
```

```
## [1] 2
```

```r
plot(cv.error.rates,pch=19,type="b")
```



```r
boost.training <- gbm(as.character(Survived)~., data=training, distribution = "bernoulli", n.trees = 10

summary(boost.training)
```



```
##                 var   rel.inf
## Name           Name 37.914412
## Fare           Fare 18.366654
## Pclass       Pclass 12.655526
## Age           Age 10.205115
## family     family  8.173176
## Sex           Sex  5.696653
## Cabin       Cabin  4.223646
## Embarked Embarked  2.764818
```

```
#test on the test set
boost.pred <- predict(boost.training, newdata = testing, n.trees = 1000, type = "response")
boost.pred <- ifelse(boost.pred > 0.5, 1, 0)

table(testing$Survived, boost.pred)
```

```
##    boost.pred
##       0   1
##   0 147  17
##   1  25  77
```

```
sum(diag(table(testing$Survived, boost.pred)))/nrow(testing)
```

```
## [1] 0.8421053
```

```
#84.21% accuracy
```

Boosting -> 84.21%

SVM - linear

```
library(e1071)
set.seed(123)
svm.fit <- svm(Survived~., data=training, scale=FALSE, kernel="linear", cost=5)
print(svm.fit)
```

```
##
## Call:
## svm(formula = Survived ~ ., data = training, kernel = "linear",
##     cost = 5, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  5
##       gamma:  0.07142857
##
## Number of Support Vectors:  258
```

```
summary(svm.fit)
```

```
##
## Call:
## svm(formula = Survived ~ ., data = training, kernel = "linear",
##     cost = 5, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  5
##       gamma:  0.07142857
##
## Number of Support Vectors:  258
##
##  ( 130 128 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
tune.out <- tune(svm, Survived~., data=training, kernel = "linear", ranges = list(cost = c(0.001, 0.01,

summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.1646953
##
## - Detailed performance results:
##     cost     error dispersion
## 1  0.001 0.3808244 0.04926884
## 2  0.010 0.2030978 0.04264397
## 3  0.100 0.1967486 0.04079913
## 4  1.000 0.1710701 0.03838288
## 5  5.000 0.1695341 0.02972506
## 6 10.000 0.1646953 0.03213910
## 7 15.000 0.1646953 0.03213910
## 8 20.000 0.1646953 0.03213910
```

```
tune.out$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = Survived ~ ., data = training,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 15, 20)),
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.07142857
##
```

```
## Number of Support Vectors:  255
```

```
svm.best <- tune.out$best.model
summary(svm.best)
```

```
##
## Call:
## best.tune(method = svm, train.x = Survived ~ ., data = training,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 15, 20)),
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.07142857
##
## Number of Support Vectors:  255
##
##  ( 129 126 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

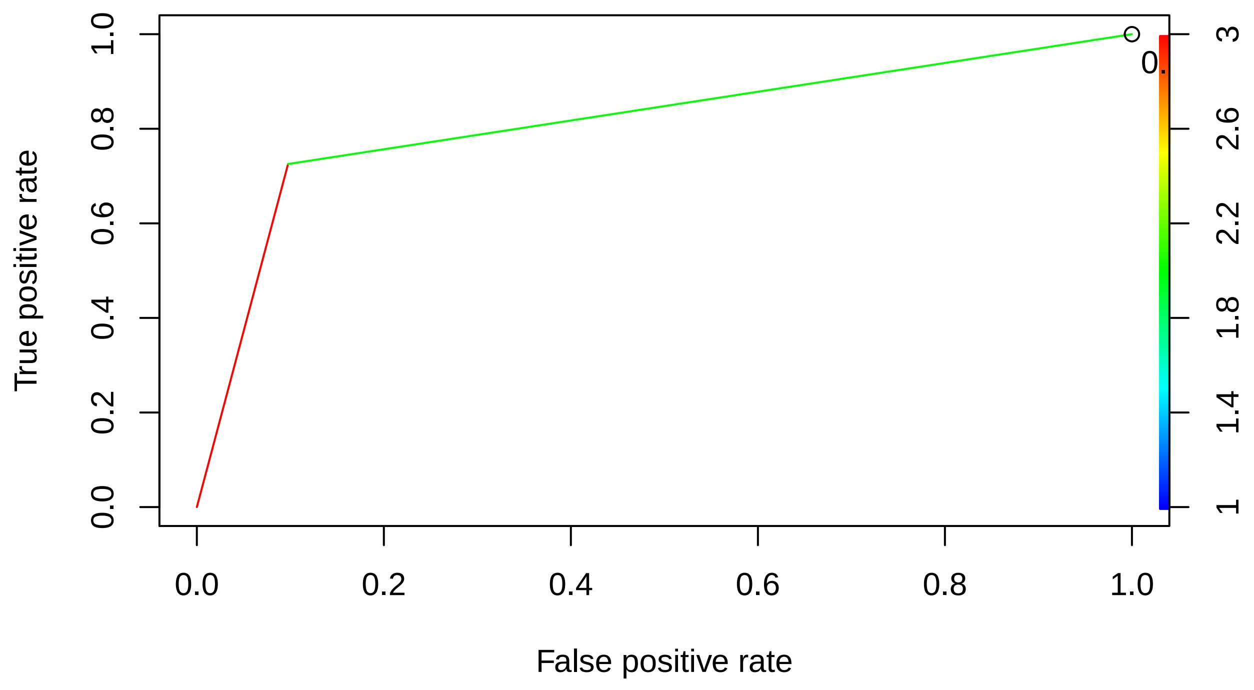
```
svm.pred <- predict(svm.best, testing, type="class")
table(predict = svm.pred, truth = testing$Survived)
```

```
##        truth
## predict   0   1
##       0 148  28
##       1  16  74
```

```
sum(diag(table(svm.pred, testing$Survived)))/nrow(testing)
```

```
## [1] 0.8345865
```

```
#83.46 accuracy


ROCpred <- prediction(as.numeric(svm.pred), test.pred)
ROCperf <- performance(ROCpred, "tpr", "fpr")
plot(ROCperf, colorize = TRUE, print.cutoffs.at = seq(0.1,0.1), text.adj = c(-0.2,1.7))
```



```r
performance(ROCpred, "auc")@y.values
```

```
## [[1]]
## [1] 0.8139646
```

```r
#auc = 0.8139

confusionMatrix(svm.pred, testing$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 148  28
##          1  16  74
##
##                Accuracy : 0.8346
##                  95% CI : (0.7844, 0.8772)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 7.118e-15
##
##                   Kappa : 0.6422
##
##  Mcnemar's Test P-Value : 0.09725
##
##             Sensitivity : 0.9024
##             Specificity : 0.7255
##          Pos Pred Value : 0.8409
##          Neg Pred Value : 0.8222
##              Prevalence : 0.6165
##          Detection Rate : 0.5564
##    Detection Prevalence : 0.6617
##       Balanced Accuracy : 0.8140
```

```
##
##        'Positive' Class : 0
##
```

SVM linear Accuracy -> 83.46%

SVM - radial

```
svm.fit <- svm(Survived~., data=training, scale=FALSE, kernel="radial", cost=5)
print(svm.fit)
```

```
##
## Call:
## svm(formula = Survived ~ ., data = training, kernel = "radial",
##     cost = 5, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##       gamma:  0.07142857
##
## Number of Support Vectors:  445
```

```
summary(svm.fit)
```

```
##
## Call:
## svm(formula = Survived ~ ., data = training, kernel = "radial",
##     cost = 5, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##       gamma:  0.07142857
##
## Number of Support Vectors:  445
##
##  ( 215 230 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
tune.out <- tune(svm, Survived~., data=training, kernel = "radial", ranges = list(cost = c(0.001, 0.01,

summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
```

```
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1664363
##
## - Detailed performance results:
##     cost     error dispersion
## 1  0.001 0.3841014 0.06229791
## 2  0.010 0.3841014 0.06229791
## 3  0.100 0.1935996 0.05389965
## 4  1.000 0.1664363 0.04209901
## 5  5.000 0.1711982 0.04505016
## 6 10.000 0.1711214 0.04533815
## 7 15.000 0.1790835 0.05147086
## 8 20.000 0.1742960 0.05290177
```

```r
tune.out$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = Survived ~ ., data = training,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 15, 20)),
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.07142857
##
## Number of Support Vectors:  299
```

```r
svm.best <- tune.out$best.model
summary(svm.best)
```

```
##
## Call:
## best.tune(method = svm, train.x = Survived ~ ., data = training,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 15, 20)),
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.07142857
##
## Number of Support Vectors:  299
##
```

```
##  ( 146 153 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

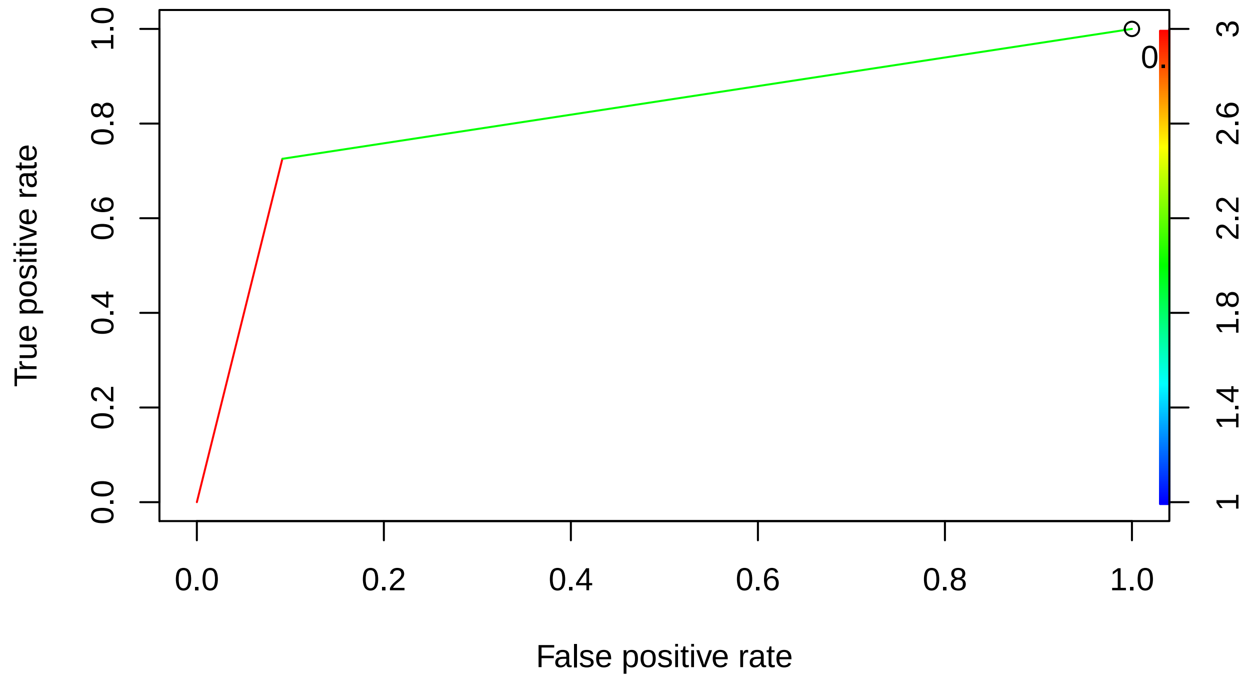
```
svm.pred <- predict(svm.best, testing, type="class")
table(predict = svm.pred, truth = testing$Survived)
```

```
##        truth
## predict   0   1
##       0 149  28
##       1  15  74
```

```
sum(diag(table(svm.pred, testing$Survived)))/nrow(testing)
```

```
## [1] 0.8383459
```

```
#83.83% accuracy


ROCpred <- prediction(as.numeric(svm.pred), test.pred)
ROCperf <- performance(ROCpred, "tpr", "fpr")
plot(ROCperf, colorize = TRUE, print.cutoffs.at = seq(0.1,0.1), text.adj = c(-0.2,1.7))
```



```
performance(ROCpred, "auc")@y.values
```

```
## [[1]]
## [1] 0.8170134
```

```
#auc = 0.8170

confusionMatrix(svm.pred, testing$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 149  28
##          1  15  74
##
##                Accuracy : 0.8383
##                  95% CI : (0.7885, 0.8805)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 2.231e-15
##
##                   Kappa : 0.6497
##
##  Mcnemar's Test P-Value : 0.06725
##
##             Sensitivity : 0.9085
##             Specificity : 0.7255
##          Pos Pred Value : 0.8418
##          Neg Pred Value : 0.8315
##              Prevalence : 0.6165
##          Detection Rate : 0.5602
##    Detection Prevalence : 0.6654
##       Balanced Accuracy : 0.8170
##
##        'Positive' Class : 0
##
```

SVM - radial Accuracy -> 83.83%

Logistic Regression Accuracy -> 84.21%

Decision Tree Accuracy -> 83.08%

Random Forest Accuracy -> around 84.5%

Boosting -> 84.21%

SVM linear Accuracy -> 83.46%

SVM radial Accuracy -> 83.83%