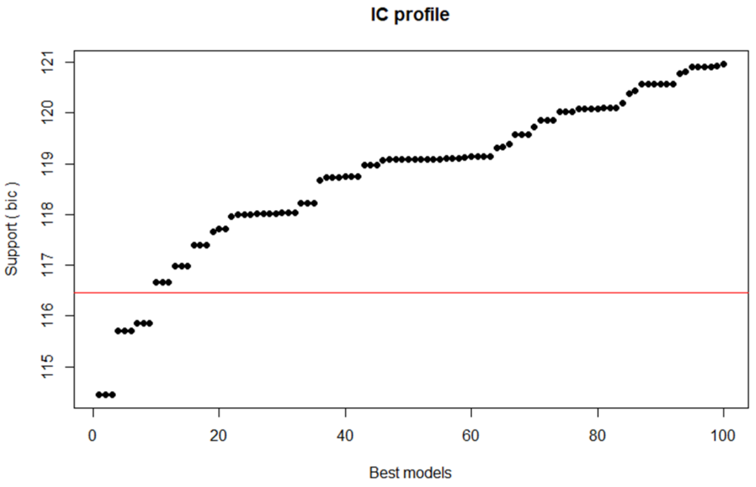
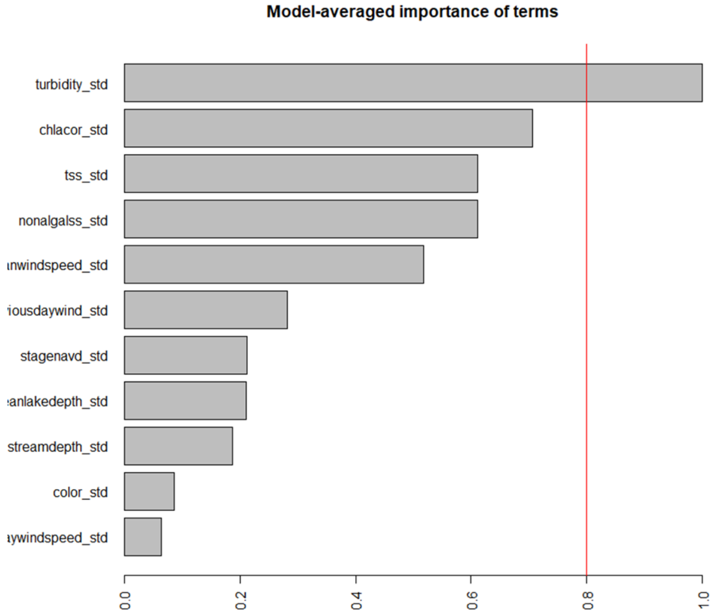
Advanced R: Statistical Machine Learning

Dan Schmutz, MS
Chief Environmental Scientist

Zoom Workshop for SJRWMD
September 24, 2020

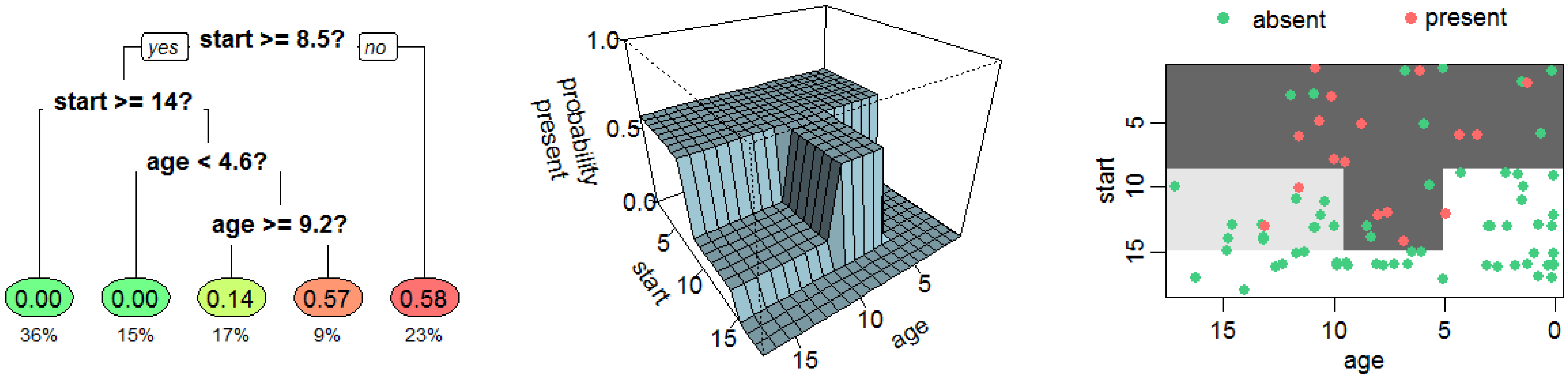




Decision Trees

recursive binary splits

Starting with all data, recursive, greedy binary splits are made using the Gini impurity measure (classification) or RMSE (regression).



Fitting tree (rpart) model to train

- notice we don't need to preprocess nominal variables using one-hot encoding because trees handle mixed variable types easily

```
rp_ttrain4cl <- rpart(
  formula = Survived2 ~ .,
  data    = ttrain4,
  method  = "class"
)
```

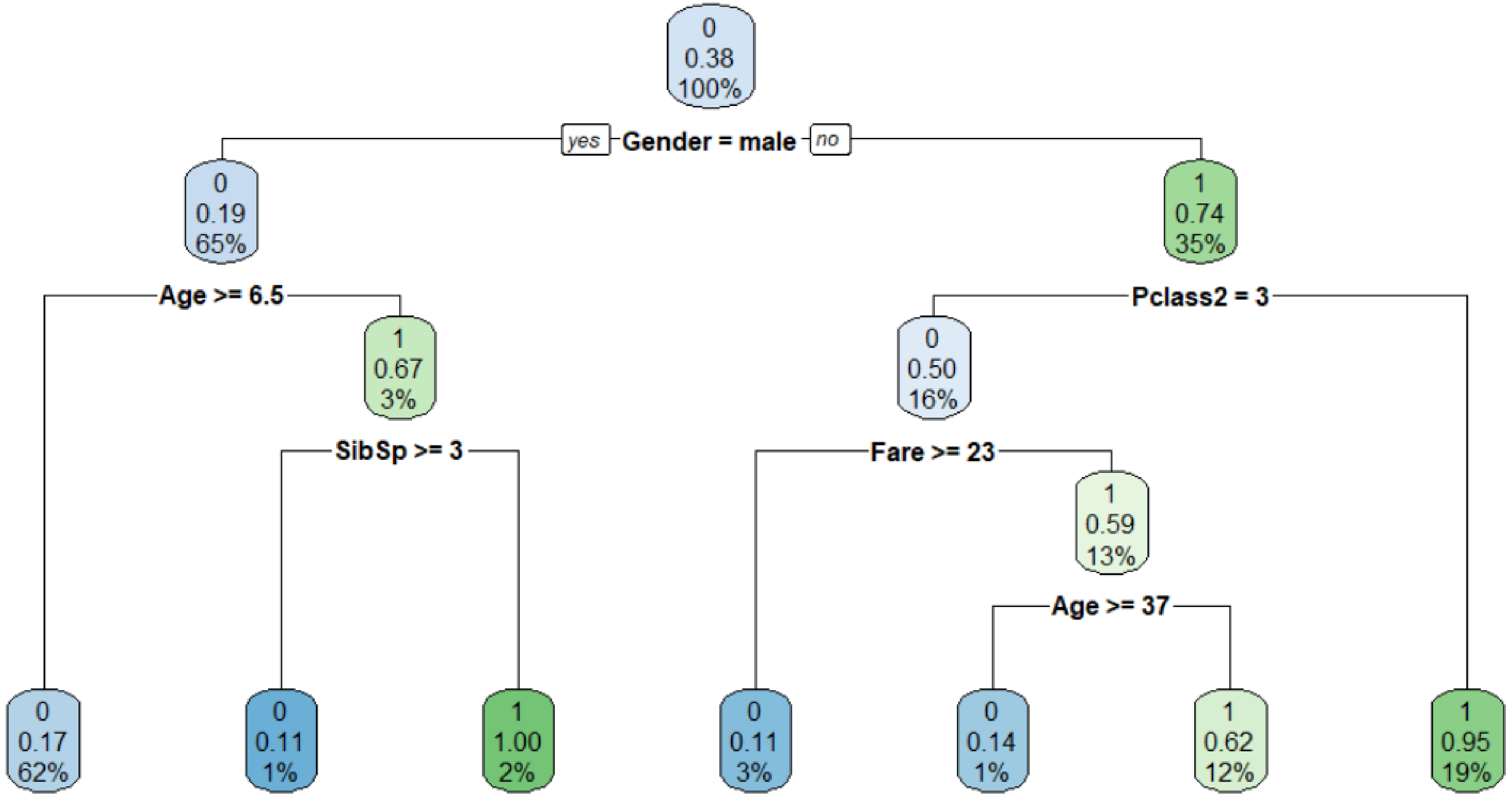
summary of rpart object

```
> summary(rp_ttrain4cl)
Call:
rpart(formula = Survived2 ~ ., data = ttrain4, method = "class")
  n= 891

          CP nsplit rel error    xerror       xstd
1 0.44444444      0 1.0000000 1.0000000 0.04244576
2 0.03070175      1 0.5555556 0.5555556 0.03574957
3 0.02339181      3 0.4941520 0.4941520 0.03421740
4 0.02046784      4 0.4707602 0.4941520 0.03421740
5 0.01461988      5 0.4502924 0.5146199 0.03474917
6 0.01000000      6 0.4356725 0.4970760 0.03429471

Variable importance
  Gender     Fare  Pclass2      Age    Parch    SibSp Embarked 
      47       16       13        9        6        6        3 

Node number 1: 891 observations,    complexity param=0.4444444
  predicted class=0  expected loss=0.3838384  P(node) =1
    class counts:   549   342
   probabilities: 0.616 0.384 
  left son=2 (577 obs) right son=3 (314 obs)
  Primary splits:
      Gender   splits as  RL,            improve=124.42630, (0 missing)
      Pclass2  splits as  LRR,           improve= 43.78183, (0 missing)
      Fare     < 10.48125 to the left,   improve= 37.94194, (0 missing)
      Embarked splits as  RLL,           improve= 11.92920, (0 missing)
      Age      < 6.5      to the right,  improve= 10.05326, (0 missing)
  Surrogate splits:
      Fare  < 77.6229  to the left,  agree=0.679, adj=0.089, (0 split)
      Parch < 0.5      to the left,  agree=0.678, adj=0.086, (0 split)
      Age   < 15.5     to the right, agree=0.651, adj=0.010, (0 split)
```

GPI

Plot of decision tree: succinct rules-based categorization

rpart.plot(rp_ttrain4cl)

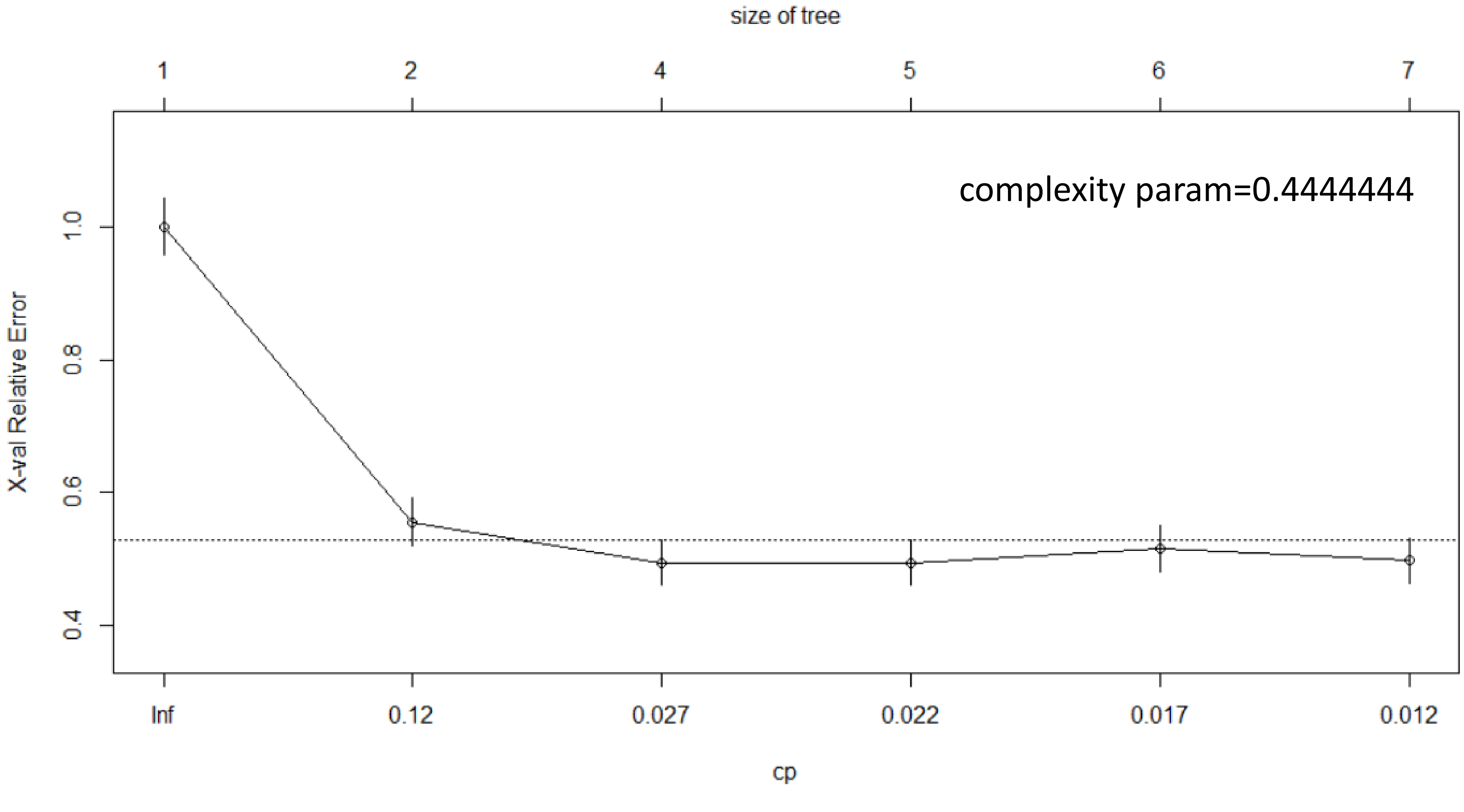


Blue boxes on bottom are terminal nodes of those who died. Green survived.

GPI

Behind the scenes rpart is using 10-fold cv to select optimal complexity parameter

plotcp(rp_ttrain4cl)



GPI

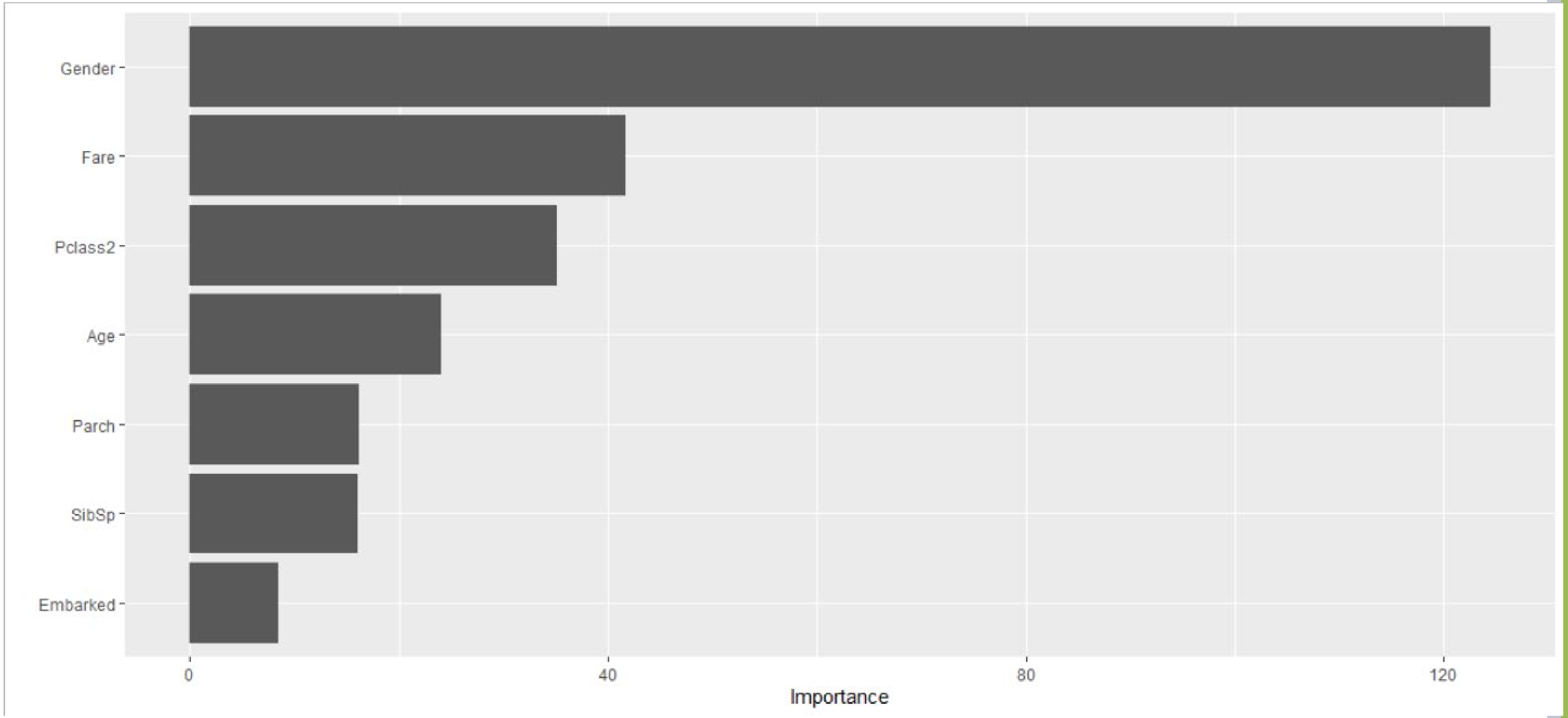
Rules for classifying future observations, with probabilities

```
> rpart.rules(rp_ttrain4cl, extra=4)
 Survived2     0    1
         0 [ .89  .11] when Gender is   male & Age <   6.5                                  & SibSp >= 3
         0 [ .89  .11] when Gender is female               & Pclass2 is      3 & Fare >= 23
         0 [ .86  .14] when Gender is female & Age >= 36.5 & Pclass2 is      3 & Fare <  23
         0 [ .83  .17] when Gender is   male & Age >=  6.5
         1 [ .38  .62] when Gender is female & Age <  36.5 & Pclass2 is      3 & Fare <  23
         1 [ .05  .95] when Gender is female               & Pclass2 is 2 or 1
         1 [ .00 1.00] when Gender is   male & Age <   6.5                                  & SibSp <  3
```

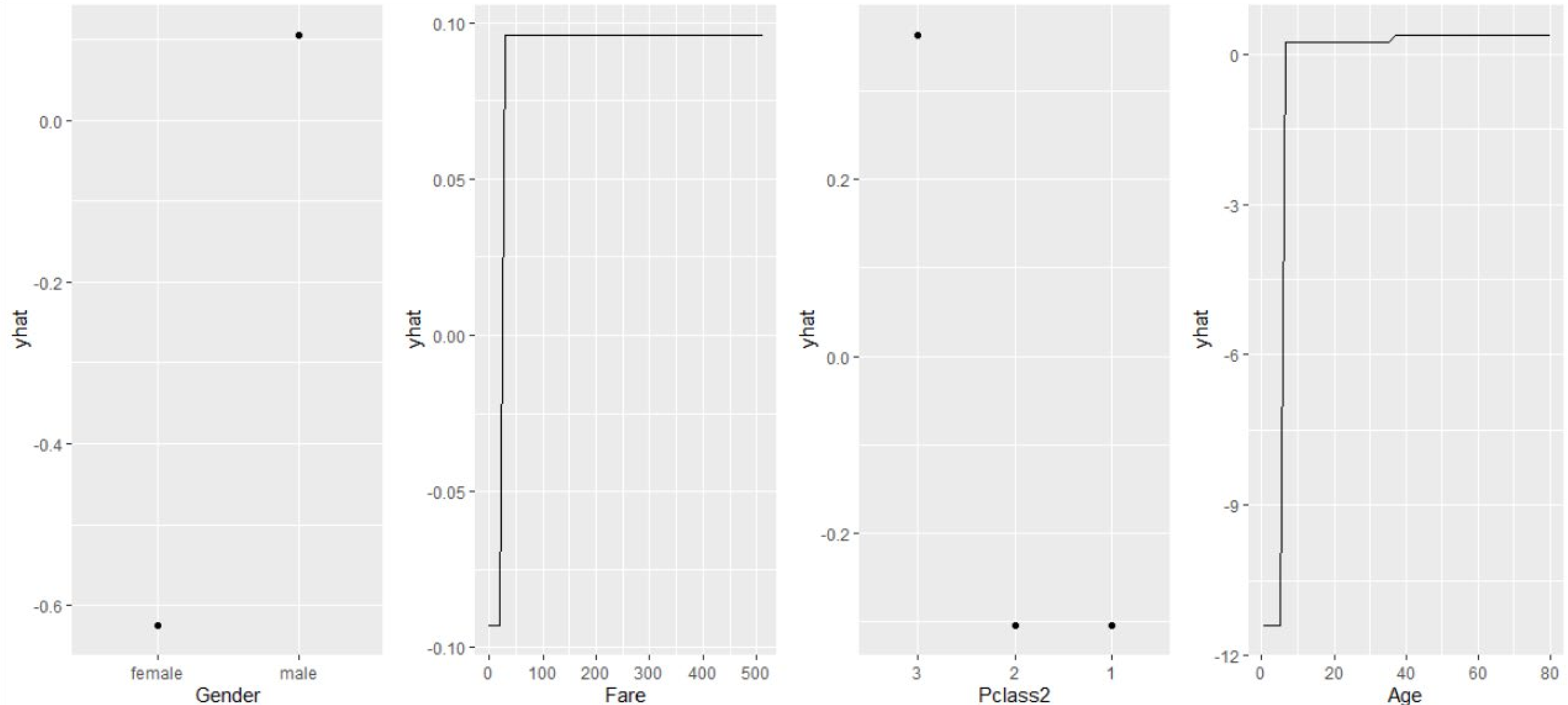
GPI

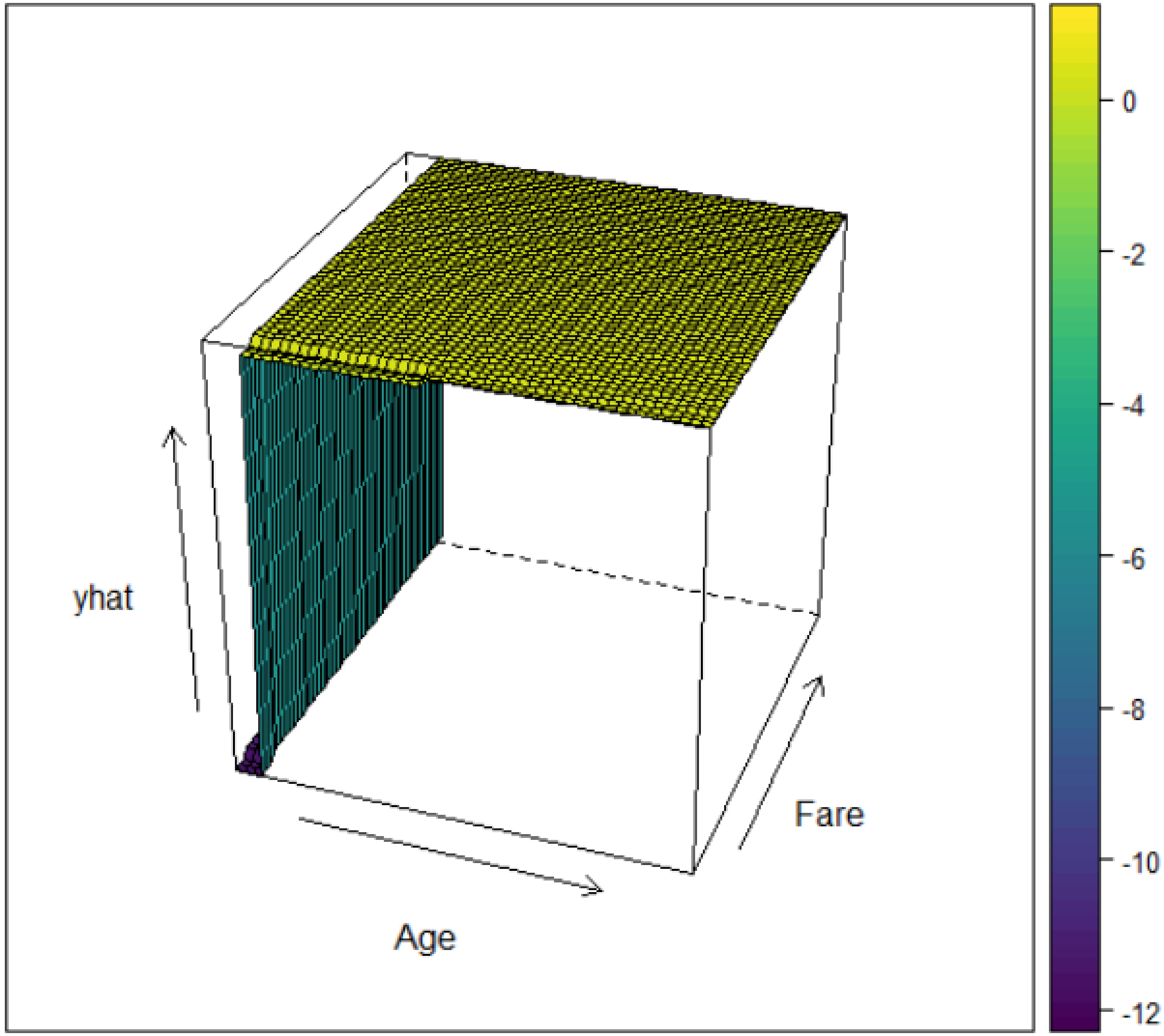
Variable importance to building the tree

vip(rp_ttrain4cl)



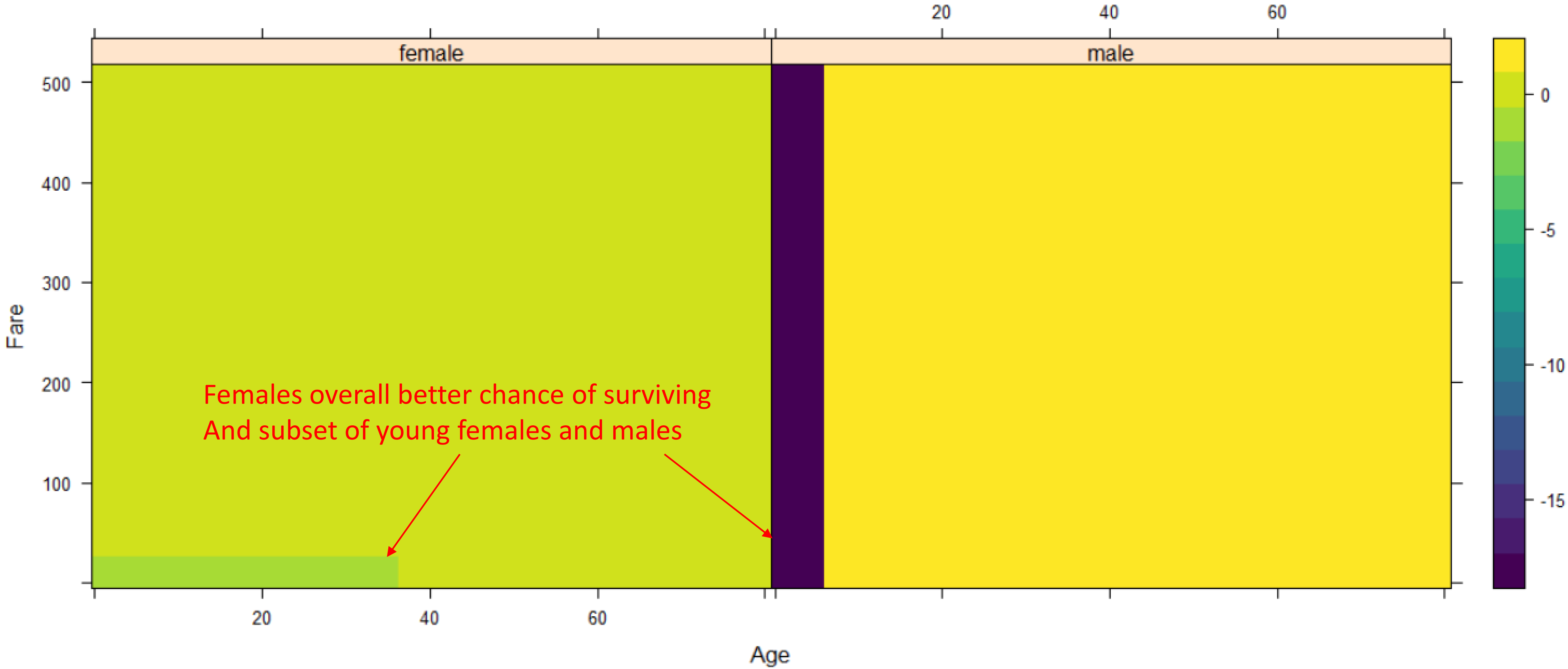
Partial dependence plots





yhat
Age
Fare
0
-2
-4
-6
-8
-10
-12

Partial dependence plot using 3 variables



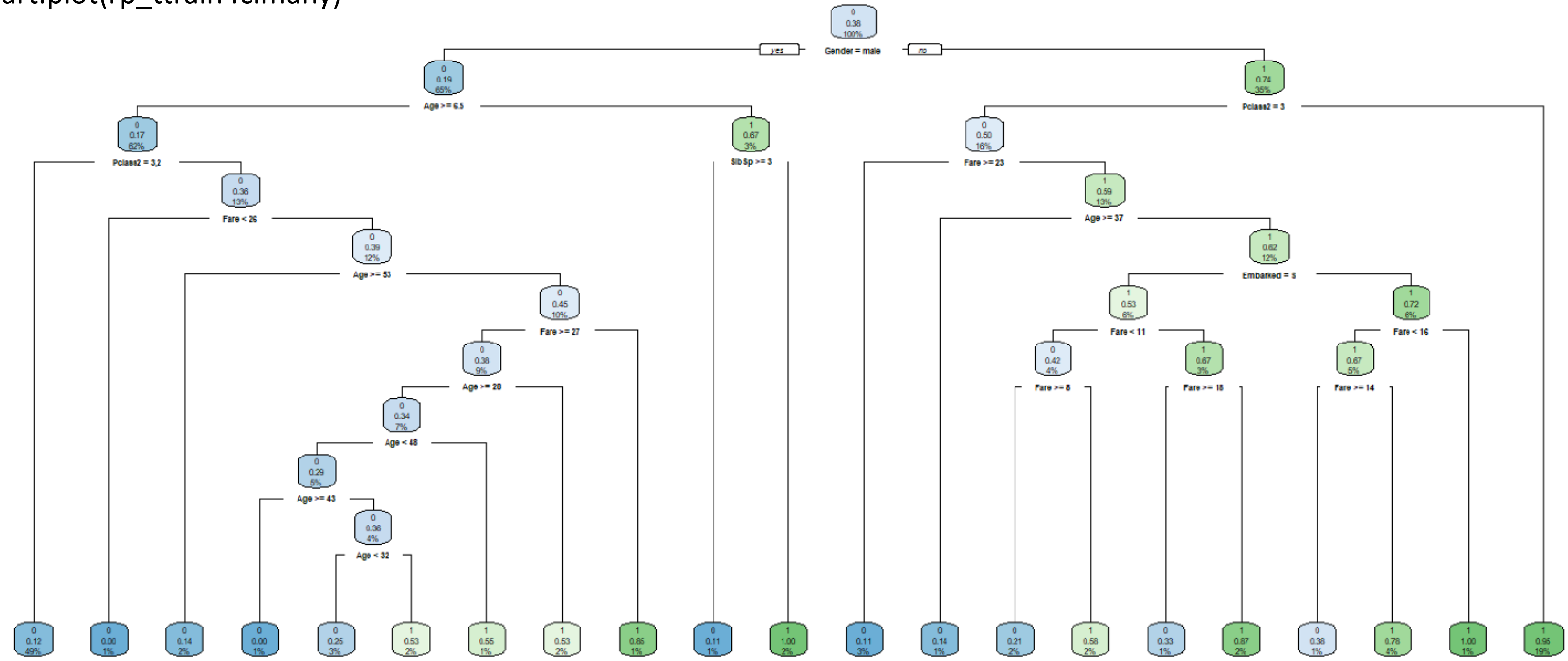
What happens if we relax the complexity parameter?

```
rp_ttrain4clmany <- rpart(
  formula = Survived2 ~ .,
  data    = ttrain4,
  method  = "class",
  cp = 0.000001
)
rpart.plot(rp_ttrain4clmany)
```

GPI

With cp = 0.000001 we grow an unnecessarily complex tree

rpart.plot(rp_ttrain4clmany)



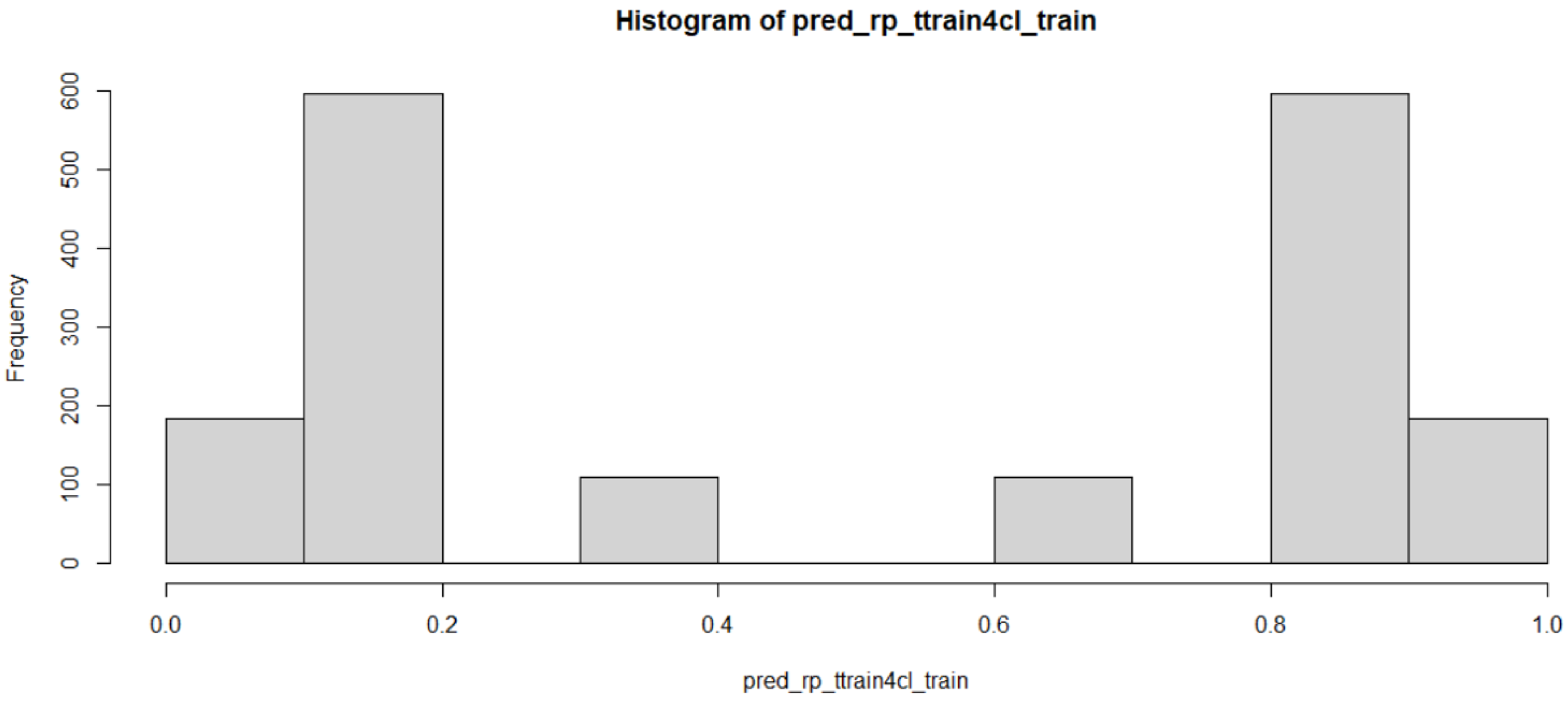
How do our trees perform on train and test?

```
# evaluating parsimonious and overfit models on train and test
pred_rp_ttrain4cl_train<-predict(rp_ttrain4cl, newdata=ttrain4)
pred_rp_ttrain4cl_test<-predict(rp_ttrain4cl, newdata=ttest4)
pred_rp_ttrain4clmany_train<-predict(rp_ttrain4clmany, newdata=ttrain4)
pred_rp_ttrain4clmany_test<-predict(rp_ttrain4clmany, newdata=ttest4)
```

GPI

Output from the predictions is again a probability

```
hist(pred_rp_ttrain4cl_train) # base r histogram works here for quick look at the data
```



GPI

Evaluation comparison

- Plot ROC curves and AUC results for each
- Calculate overall accuracy for each

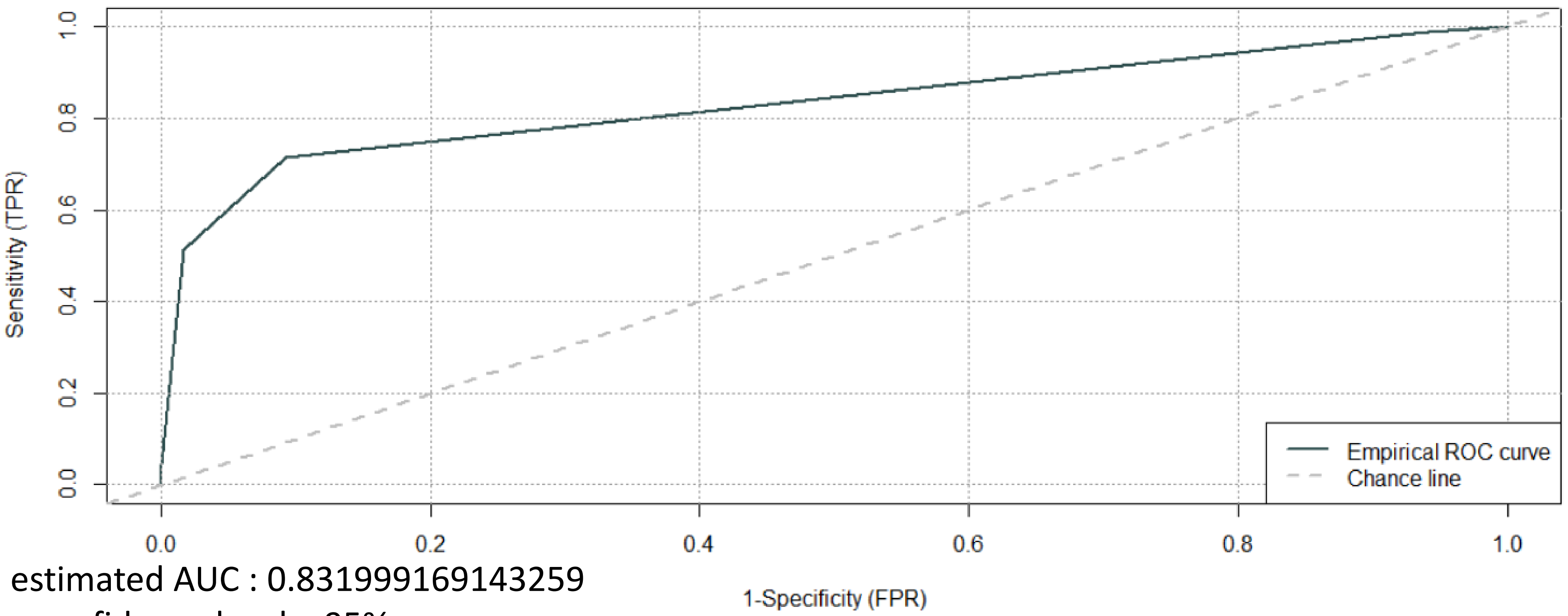
```
> summary(pred_rp_ttrain4cl_train)
       0                1
 Min.   :0.0000   Min.   :0.1111
 1st Qu.:0.3818   1st Qu.:0.1682
 Median :0.8318   Median :0.1682
 Mean   :0.6162   Mean   :0.3838
 3rd Qu.:0.8318   3rd Qu.:0.6182
 Max.   :0.8889   Max.   :1.0000
> str(pred_rp_ttrain4cl_train)
 num [1:891, 1:2] 0.8318 0.0529 0.3818 0.0529 0.8318 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:891] "1" "2" "3" "4" ...
  ..$ : chr [1:2] "0" "1"
> 
```

Output from rpart is a matrix, so need to extract the probability of 1 only for evaluation
pred_rp_ttrain4cl_train[,2] to extract the second column

GPI

ROC, AUC on train for parsimonious model

pred_rp_ttrain4cl_train[,2]



estimated AUC : 0.831999169143259

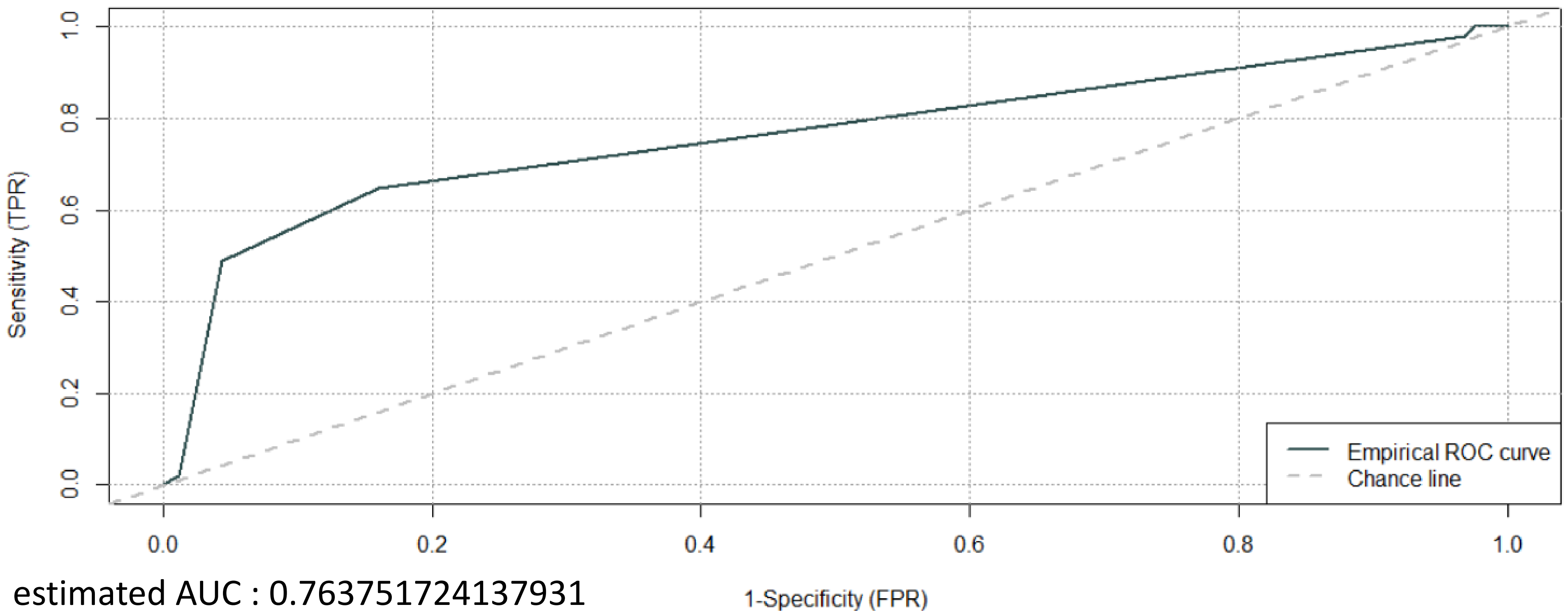
confidence level = 95%

lower = 0.802761380932999 upper = 0.861236957353519

ROC, AUC on test for parsimonious model

pred_rp_ttrain4cl_test[,2]

Slightly degraded performance on the test as expected due to slight overfitting



estimated AUC : 0.763751724137931

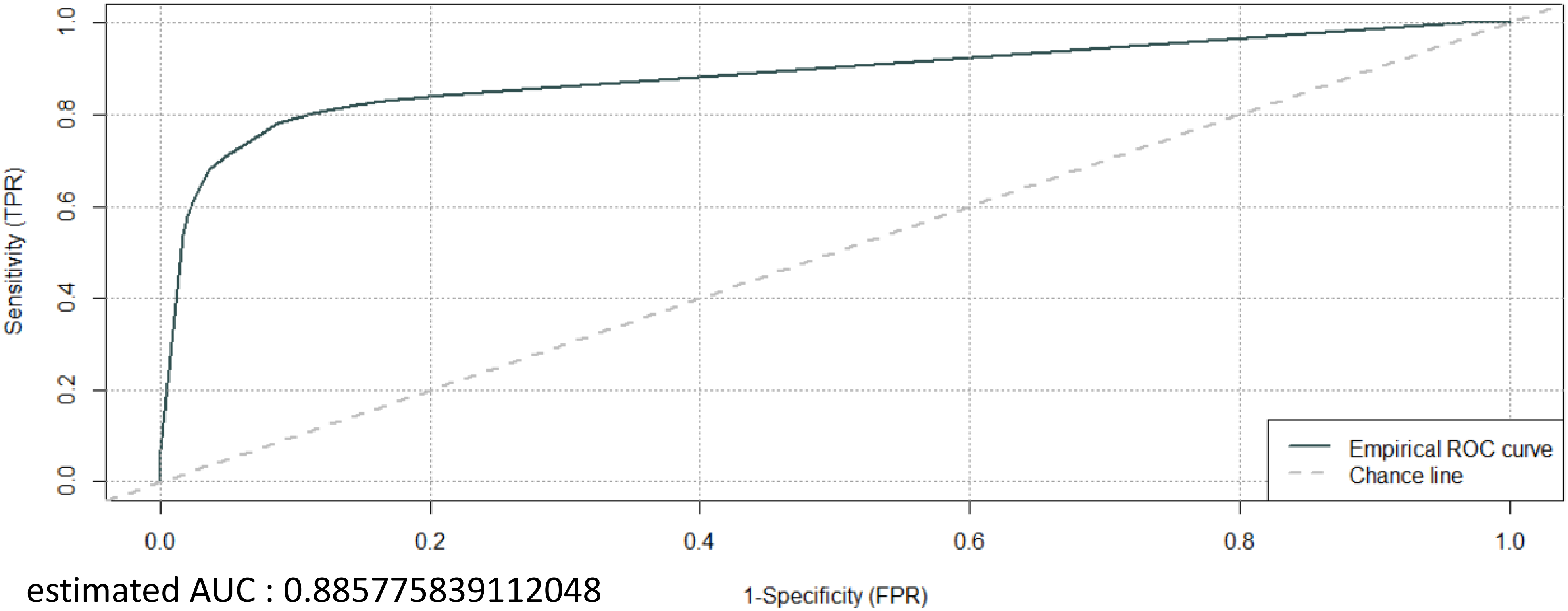
confidence level = 95%

lower = 0.712693033076003 upper = 0.814810415199859

ROC, AUC on train for overfit model

pred_rp_ttrain4clmany_train[,2]

The overfit model does better on train than our parsimonious model



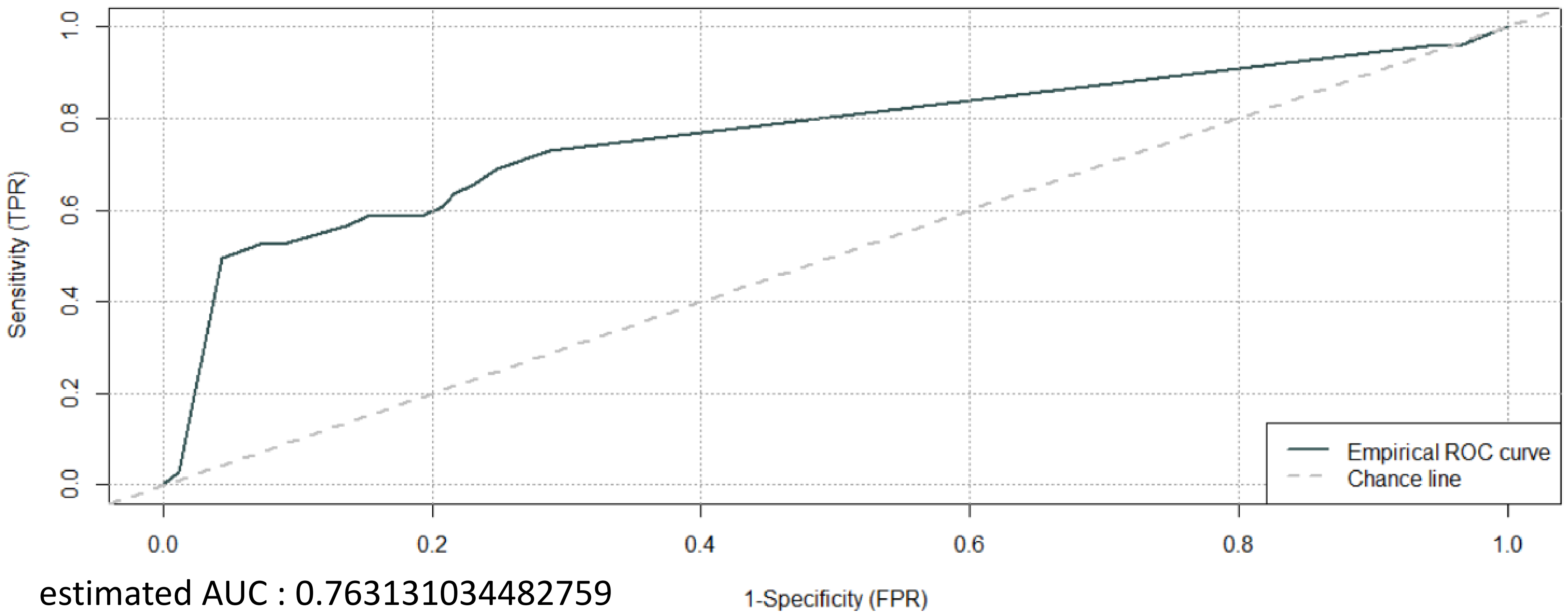
estimated AUC : 0.885775839112048
confidence level = 95%
lower = 0.86113299286047 upper = 0.910418685363627

GPI

ROC, AUC on test for overfit model

pred_rp_ttrain4clmany_test[,2]

The overfit model actually doesn't do any worse than the parsimonious model in this case, indicating robustness of trees



estimated AUC : 0.763131034482759
confidence level = 95%
lower = 0.712024453005344 upper = 0.814237615960173

confusionMatrix for parsimonious model (train and test)

```
> confusionMatrix(factor(predclass$predclas
s),ttrain4$Survived2)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 498  98
         1  51 244

               Accuracy : 0.8328
                 95% CI : (0.8066, 0.8567)
    No Information Rate : 0.6162
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6371

 Mcnemar's Test P-Value : 0.0001643

            Sensitivity : 0.9071
            Specificity : 0.7135
         Pos Pred Value : 0.8356
         Neg Pred Value : 0.8271
             Prevalence : 0.6162
         Detection Rate : 0.5589
   Detection Prevalence : 0.6689
      Balanced Accuracy : 0.8103

       'Positive' Class : 0
```

```
> confusionMatrix(factor(predclass$predclas
s),ttest4$Survived2)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 210  51
         1  40  94

               Accuracy : 0.7696
                 95% CI : (0.7249, 0.8103)
    No Information Rate : 0.6329
    P-Value [Acc > NIR] : 3.776e-09

                  Kappa : 0.4962

 Mcnemar's Test P-Value : 0.2945

            Sensitivity : 0.8400
            Specificity : 0.6483
         Pos Pred Value : 0.8046
         Neg Pred Value : 0.7015
             Prevalence : 0.6329
         Detection Rate : 0.5316
   Detection Prevalence : 0.6608
      Balanced Accuracy : 0.7441

       'Positive' Class : 0
```

confusionMatrix for overfit model (train and test)

```
> confusionMatrix(factor(predclass$predclas
s),ttrain4$Survived2)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 501  75
         1  48 267

               Accuracy : 0.862
                 95% CI : (0.8375, 0.8839)
    No Information Rate : 0.6162
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.7037

 Mcnemar's Test P-Value : 0.01906

            Sensitivity : 0.9126
            Specificity : 0.7807
         Pos Pred Value : 0.8698
         Neg Pred Value : 0.8476
             Prevalence : 0.6162
         Detection Rate : 0.5623
   Detection Prevalence : 0.6465
      Balanced Accuracy : 0.8466

       'Positive' Class : 0
```

```
> confusionMatrix(factor(predclass$predclas
s),ttest4$Survived2)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 198  57
         1  52  88

               Accuracy : 0.7241
                 95% CI : (0.6771, 0.7676)
    No Information Rate : 0.6329
    P-Value [Acc > NIR] : 7.882e-05

                  Kappa : 0.4018

 Mcnemar's Test P-Value : 0.7016

            Sensitivity : 0.7920
            Specificity : 0.6069
         Pos Pred Value : 0.7765
         Neg Pred Value : 0.6286
             Prevalence : 0.6329
         Detection Rate : 0.5013
   Detection Prevalence : 0.6456
      Balanced Accuracy : 0.6994

       'Positive' Class : 0
```

GPI