```r
#libs
require("ROSE")
```

```
## Loading required package: ROSE
## Warning: package 'ROSE' was built under R version 3.3.3
## Loaded ROSE 0.0-3
```

```r
require("pROC")
```

```
## Loading required package: pROC
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
require("rpart")
```

```
## Loading required package: rpart
```

```r
require("rpart.plot")
```

```
## Loading required package: rpart.plot
## Warning: package 'rpart.plot' was built under R version 3.3.3
```

```r
require("caret")
```

```
## Loading required package: caret
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
require("randomForest")
```

```
## Loading required package: randomForest
## Warning: package 'randomForest' was built under R version 3.3.3
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
require("e1071")
```

```
## Loading required package: e1071
```

```r
#main file
cc <- data.frame(read.csv("data/cc.csv"))
cc <- cc[,c(2:31)]
```

```r
#check balance
fraud <- nrow(cc[cc$Class == 1,])
notFraud <- nrow(cc) - fraud
paste("fraud: ", fraud, "|| not fraud: ", notFraud)
```

```
## [1] "fraud:  492 || not fraud:  284315"
```

```r
#make the size smaller for easier use
set.seed(65)
cc_simp <- cc[sample(nrow(cc), 25000), ]
fraud <- nrow(cc_simp[cc_simp$Class == 1,])
notFraud <- nrow(cc_simp) - fraud
paste("fraud: ", fraud, "|| not fraud: ", notFraud)
```

```
## [1] "fraud:  43 || not fraud:  24957"
```

```r
#split data for testing models
trainLength <- floor(.7*nrow(cc_simp))
testLength <- nrow(cc_simp) - trainLength

train_model <- cc_simp[1:trainLength,]
train_eval <- cc_simp[(trainLength + 1):nrow(cc_simp),]

#undersample
fraud <- nrow(train_model[train_model$Class == 1,])
notFraud <- nrow(train_model) - fraud
paste("fraud: ", fraud, "|| not fraud: ", notFraud)
```

```
## [1] "fraud:  29 || not fraud:  17471"
```

```r
train_model <- ovun.sample(Class ~ ., data = train_model, method = "under",N = fraud*2, seed = 1)$data


#functions for sd and se
mysd <- function(predict, target) {
  diff_sq <- (predict - mean(target))^2
  return(mean(sqrt(diff_sq)))
}

myse <- function(predict, target) {
  diff_sq <- (predict - target)^2
  return(mean(sqrt(diff_sq)))
}


#Model1 - Multiple Linear Regression - Base Line
mlr1 <- glm(Class~., data = train_model)
BIC(mlr1)
```

```
## [1] 117.6208
```

```r
predict_mlr1 <- predict(mlr1, train_eval, type = 'response')
table(train_eval$Class, predict_mlr1 > 0.5)
```

```
##
##     FALSE TRUE
##   0  6715  771
```

```
##   1     3   11
```

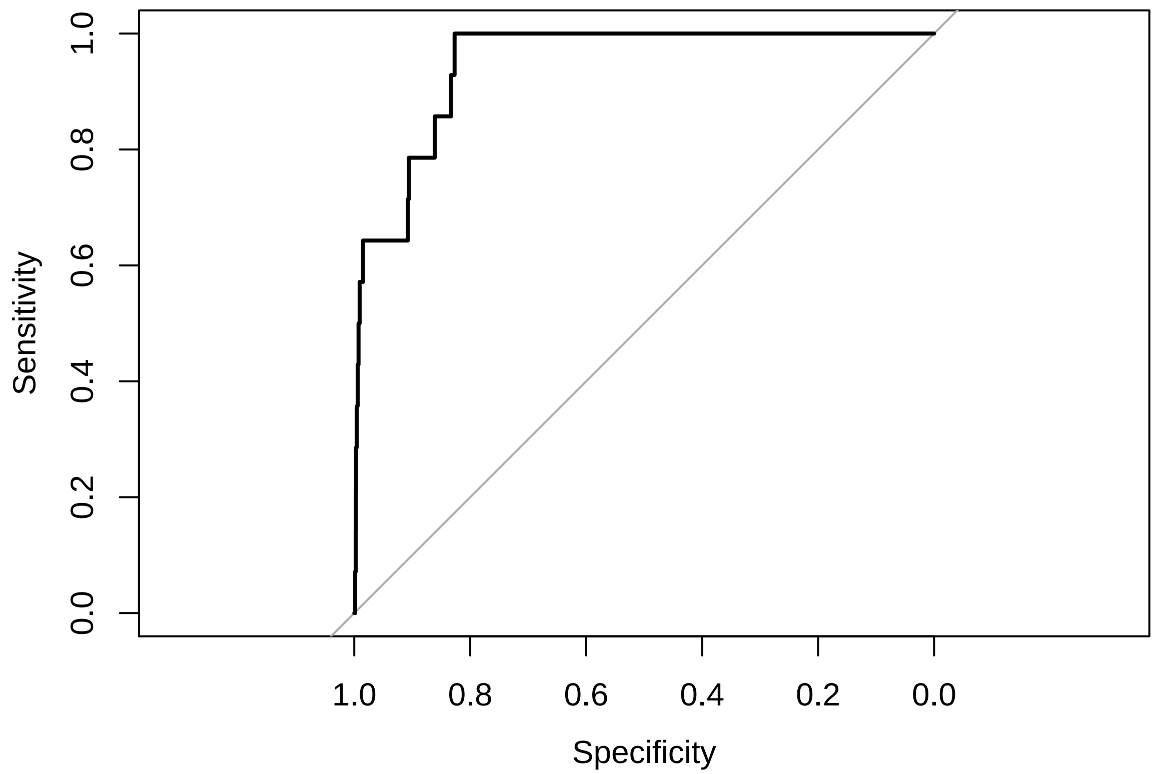
```r
mysd(predict_mlr1, train_eval$Class)
```

```
## [1] 0.2552219
```

```r
myse(predict_mlr1, train_eval$Class)
```

```
## [1] 0.255147
```

```r
auc_mlr1 <- roc(train_eval$Class, predict_mlr1)
plot(auc_mlr1)
```




```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_mlr1)
##
## Data: predict_mlr1 in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9488
```

```r
#Model2 - Poisson Model
poisson1 <- glm(Class ~ ., family = "poisson", data = train_model)
```

```
## Warning: glm.fit: fitted rates numerically 0 occurred
```

```r
BIC(poisson1)
```

```
## [1] 189.2826
```

```r
predict_poisson1 <- predict(poisson1, train_eval, type = 'response')
table(train_eval$Class, predict_poisson1 > 0.5)
```

```
##
##     FALSE TRUE
##   0  6488  998
##   1     6    8
```

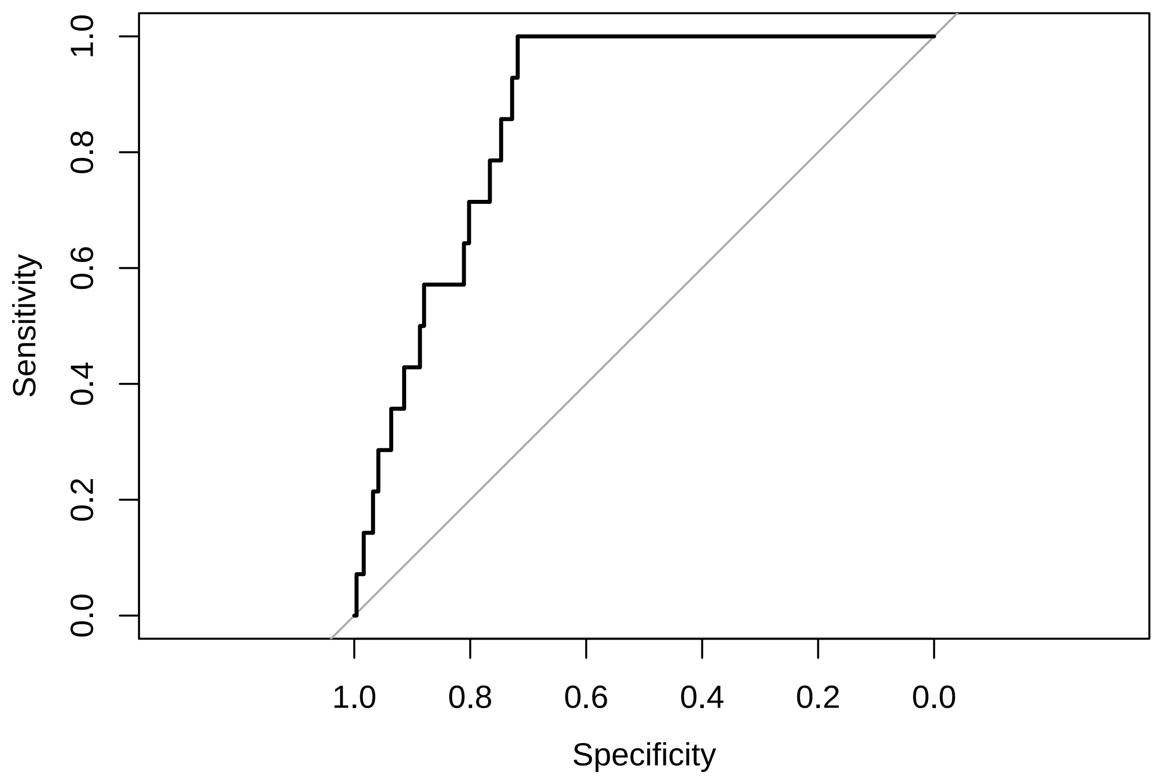
```r
mysd(predict_poisson1, train_eval$Class)
```

```
## [1] 72.3309
```

```r
myse(predict_poisson1, train_eval$Class)
```

```
## [1] 72.33149
```

```r
auc_poisson <- roc(train_eval$Class, predict_poisson1)
plot(auc_poisson)
```




```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_poisson1)
##
## Data: predict_poisson1 in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.8639
```

```r
#logit model
logit1 <- glm(Class ~., family = binomial(link='logit'), data = train_model)
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

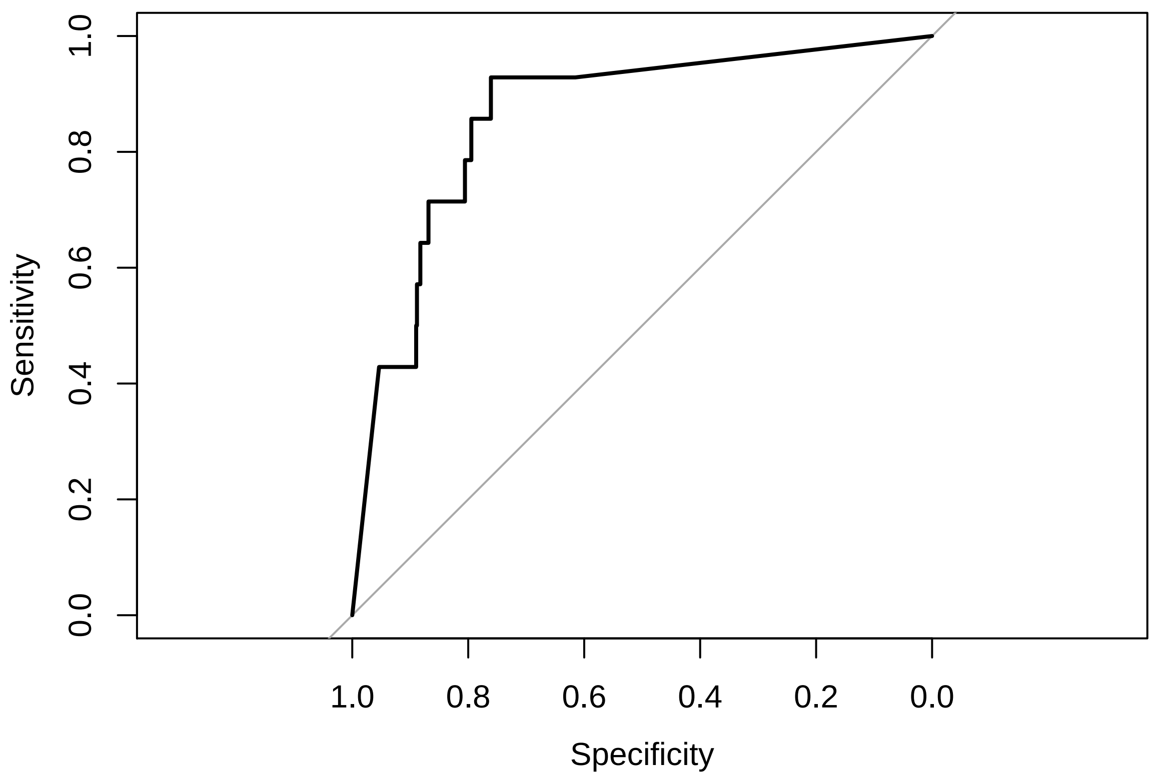
```r
BIC(logit1)
```

```
## [1] 121.8133
```

```r
predict_logit1 <- predict(logit1, train_eval, type = 'response')
table(train_eval$Class, predict_logit1 > 0.5)
```

```
##
##     FALSE TRUE
##   0  6478 1008
##   1     4   10
```

```r
auc_logit1 <- roc(train_eval$Class, predict_logit1)
plot(auc_logit1)
```




```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_logit1)
##
## Data: predict_logit1 in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.8614
```

```r
# backward stepwise
stepwise1 <- glm(Class ~ ., data = train_model)
backward <- step(stepwise1, trace = 0)
```

```r
BIC(backward)
```

```
## [1] 59.40368
```

```r
predict_backward <- predict(backward, train_eval, type = 'response')
table(train_eval$Class, predict_backward > 0.5)
```

```
##
##      FALSE TRUE
##    0  6619  867
##    1     4   10
```

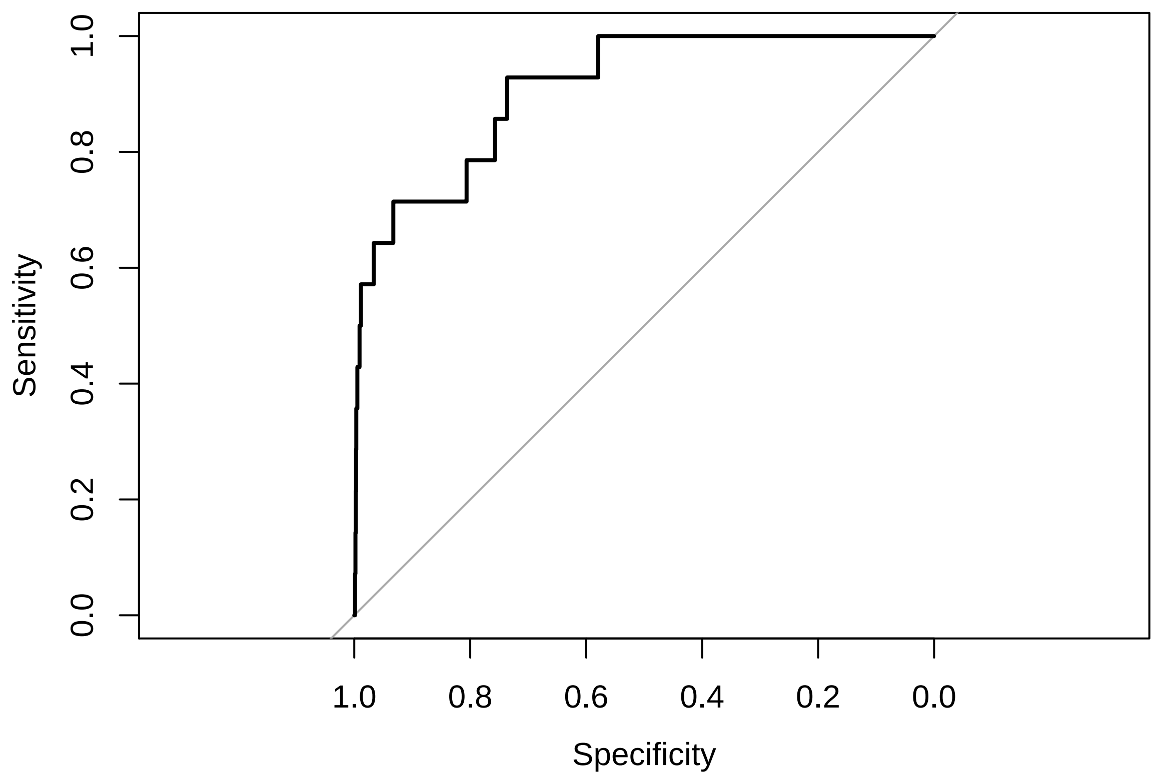
```r
mysd(predict_backward, train_eval$Class)
```

```
## [1] 0.2617563
```

```r
myse(predict_backward, train_eval$Class)
```

```
## [1] 0.2619181
```

```r
auc_backward <- roc(train_eval$Class, predict_backward)
plot(auc_backward)
```




```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_backward)
##
## Data: predict_backward in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.91
```

```r
#forward stepwise
stepwise2 <- glm(Class ~ 1,data = train_model)
forward <- step(stepwise2, scope = list(lower=formula(stepwise2), upper=formula(stepwise1)), direction =
BIC(forward)
```

```
## [1] 53.3026
```

```r
predict_forward <- predict(forward, train_eval, type = 'response')
table(train_eval$Class, predict_forward > 0.5)
```

```
##
##     FALSE TRUE
##   0  7170  316
##   1     3   11
```

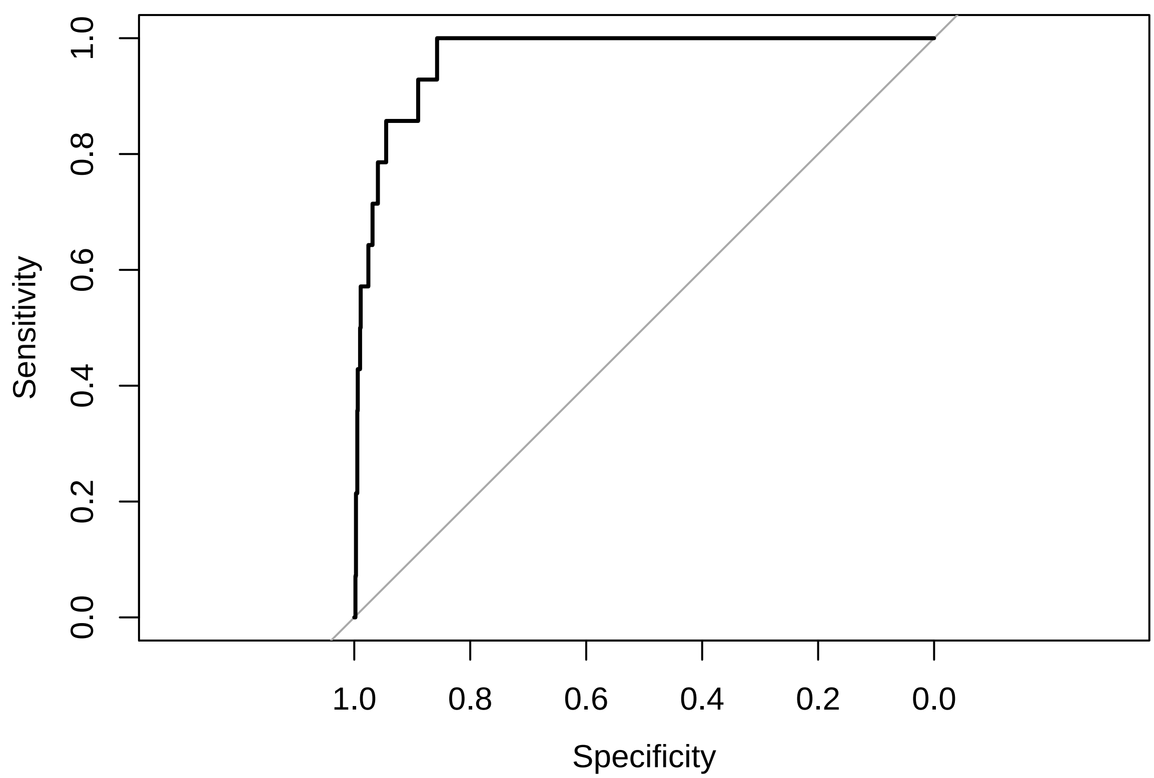
```r
mysd(predict_forward, train_eval$Class)
```

```
## [1] 0.2195392
```

```r
myse(predict_forward, train_eval$Class)
```

```
## [1] 0.220002
```

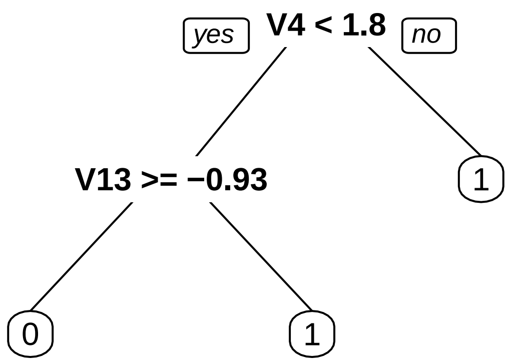
```r
auc_forward <- roc(train_eval$Class, predict_forward)
plot(auc_forward)
```




```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_forward)
```

```
##
## Data: predict_forward in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9679
```

```
# decision tree
decision <- rpart(Class ~ ., data = train_model, method = "class")
prp(decision)
```




```
predict_decision <- predict(decision, train_eval, type = "class")
confusionMatrix(train_eval$Class, predict_decision)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5663 1823
##          1    2   12
##
##                Accuracy : 0.7567
##                  95% CI : (0.7468, 0.7663)
##     No Information Rate : 0.7553
##     P-Value [Acc > NIR] : 0.4001
##
##                   Kappa : 0.0093
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99965
```

```
##                Specificity : 0.00654
##             Pos Pred Value : 0.75648
##             Neg Pred Value : 0.85714
##                 Prevalence : 0.75533
##             Detection Rate : 0.75507
##       Detection Prevalence : 0.99813
##          Balanced Accuracy : 0.50309
##
##           'Positive' Class : 0
##
```

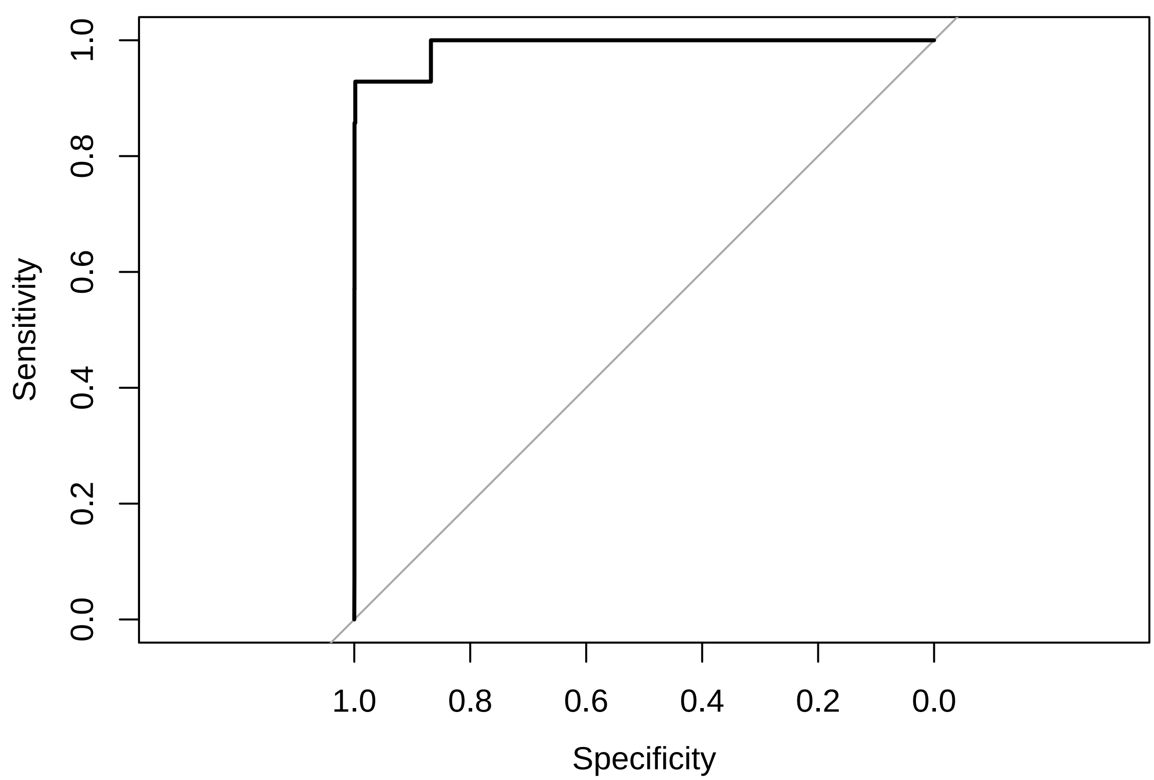
```r
#decision tree random forest (kind of broke with raw data)
rforest <- randomForest(Class ~ ., data = train_model)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```r
predict_rforest <- predict(rforest, train_eval)
table(train_eval$Class, predict_rforest > 0.5)
```

```
##
##      FALSE TRUE
##   0  7225  261
##   1     1   13
```

```r
auc_rforest <- roc(train_eval$Class, predict_rforest)
plot(auc_rforest)
```

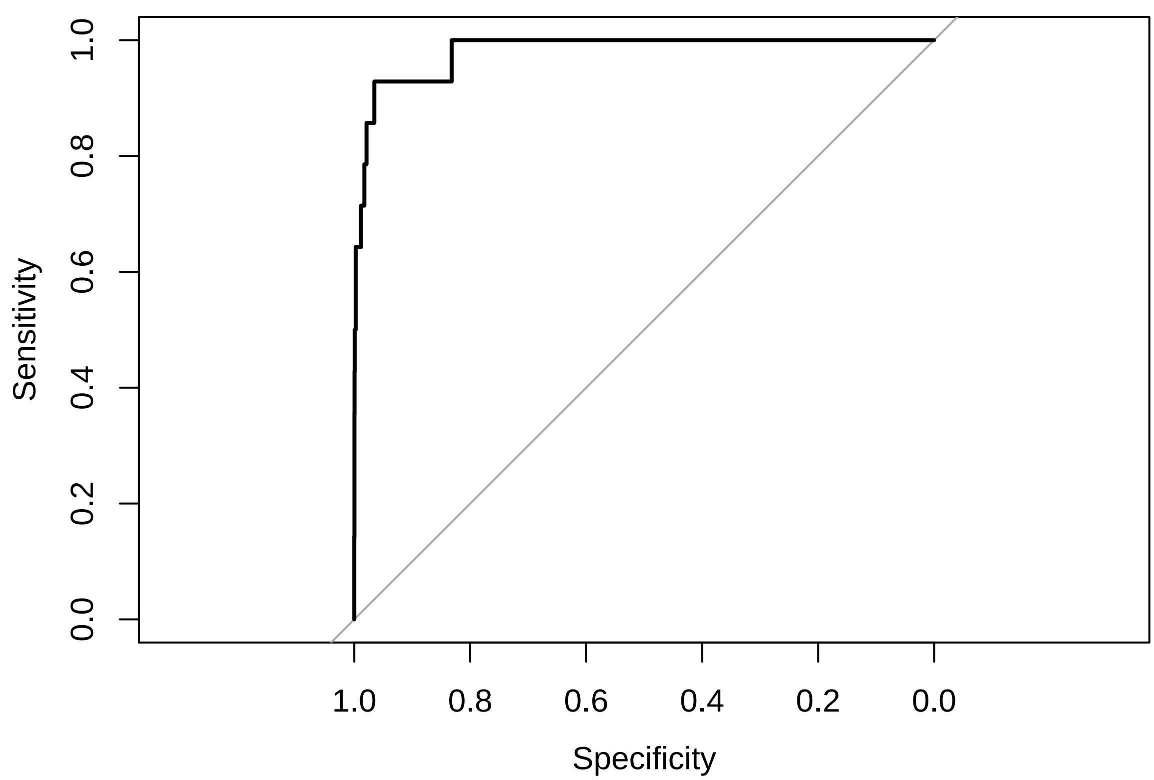


```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_rforest)
##
## Data: predict_rforest in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9902
```

```r
#svm (kind of broken with raw data)
svm <- svm(Class ~ ., data = train_model)
predict_svm <- predict(svm, train_eval)
table(train_eval$Class, predict_svm > 0.5)
```

```
##
##     FALSE TRUE
##   0  7297  189
##   1     2   12
```

```r
auc_svm <- roc(train_eval$Class, predict_svm)
plot(auc_svm)
```




```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_svm)
##
## Data: predict_svm in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9815
```

```r
#tables only
table(train_eval$Class, predict_mlr1 > 0.5)
```

```
##
##     FALSE TRUE
##   0  6715  771
##   1     3   11
```

```r
table(train_eval$Class, predict_poisson1 > 0.5)
```

```
##
##     FALSE TRUE
##   0  6488  998
##   1     6    8
```

```r
table(train_eval$Class, predict_logit1 > 0.5)
```

```
##
##     FALSE TRUE
##   0  6478 1008
##   1     4   10
```

```r
table(train_eval$Class, predict_backward > 0.5)
```

```
##
##     FALSE TRUE
##   0  6619  867
##   1     4   10
```

```r
table(train_eval$Class, predict_forward > 0.5)
```

```
##
##     FALSE TRUE
##   0  7170  316
##   1     3   11
```

```r
confusionMatrix(train_eval$Class, predict_decision)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5663 1823
##          1    2   12
##
##                Accuracy : 0.7567
##                  95% CI : (0.7468, 0.7663)
##     No Information Rate : 0.7553
##     P-Value [Acc > NIR] : 0.4001
##
##                   Kappa : 0.0093
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99965
##             Specificity : 0.00654
##          Pos Pred Value : 0.75648
##          Neg Pred Value : 0.85714
```

```
##               Prevalence : 0.75533
##           Detection Rate : 0.75507
##     Detection Prevalence : 0.99813
##        Balanced Accuracy : 0.50309
##
##         'Positive' Class : 0
##
```

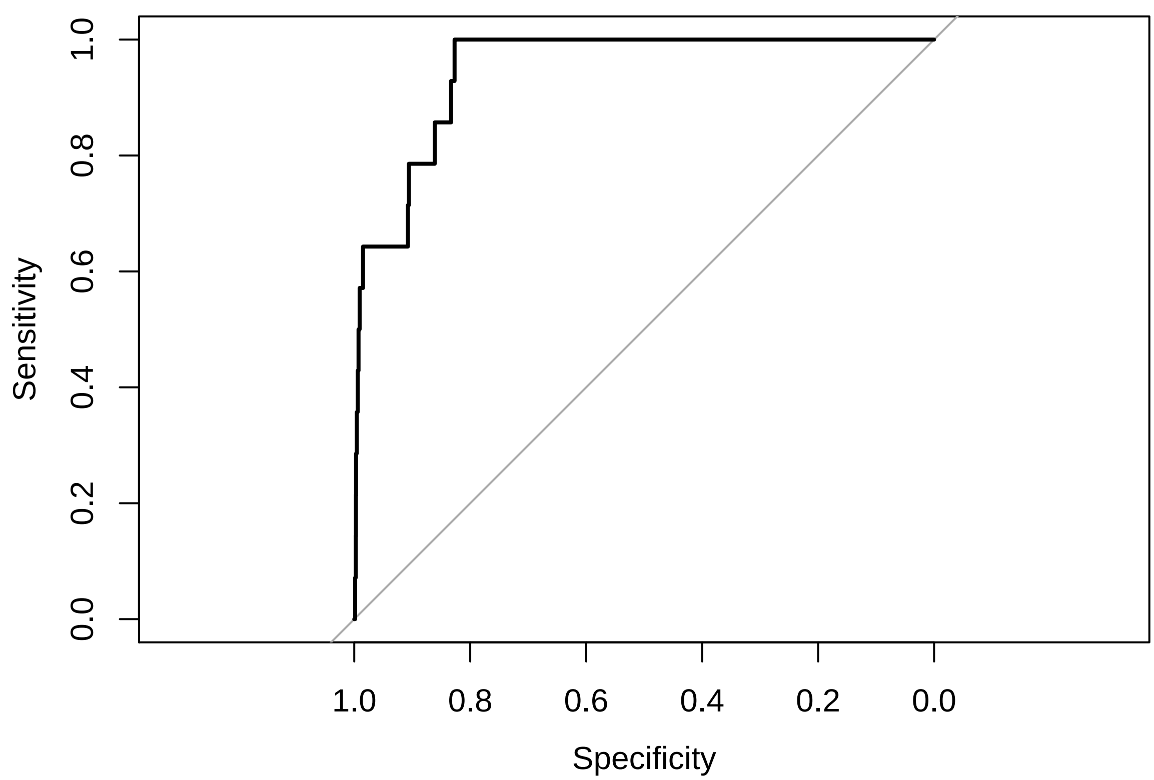
```r
table(train_eval$Class, predict_rforest > 0.5)
```

```
##
##     FALSE TRUE
##   0  7225  261
##   1     1   13
```

```r
table(train_eval$Class, predict_svm > 0.5)
```

```
##
##     FALSE TRUE
##   0  7297  189
##   1     2   12
```

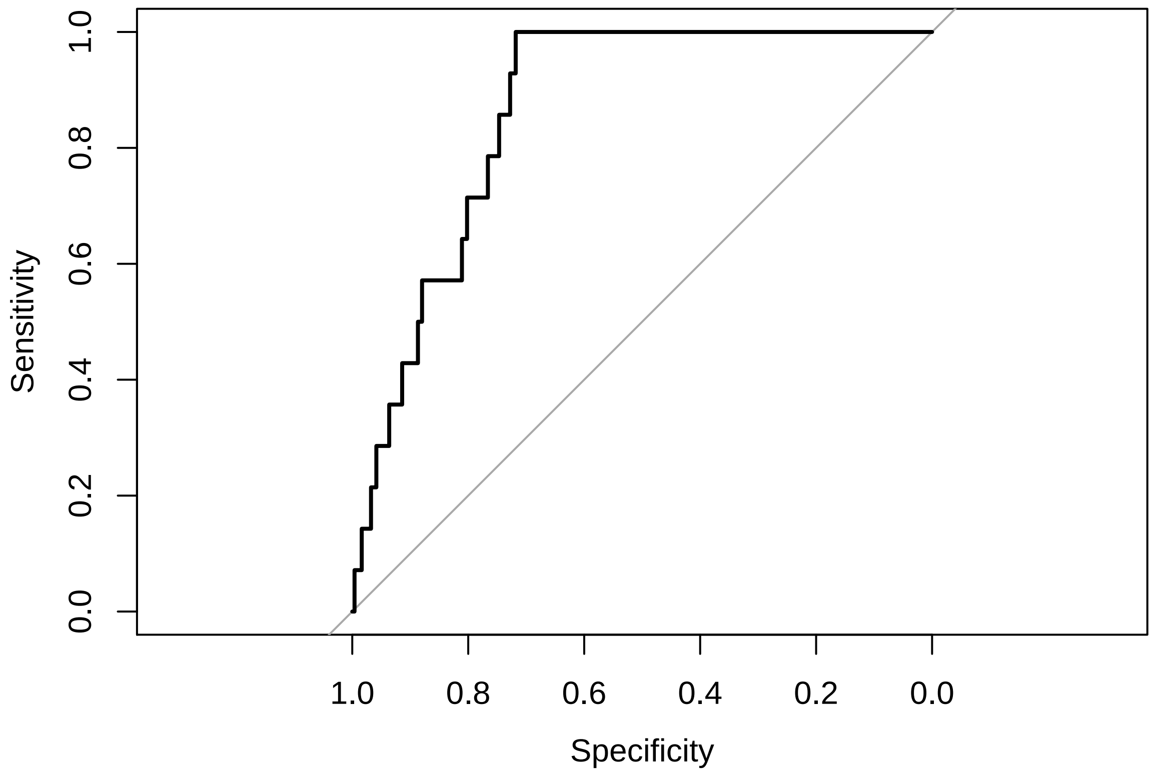
```r
#AUC plots only
plot(auc_mlr1)
```




```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_mlr1)
```

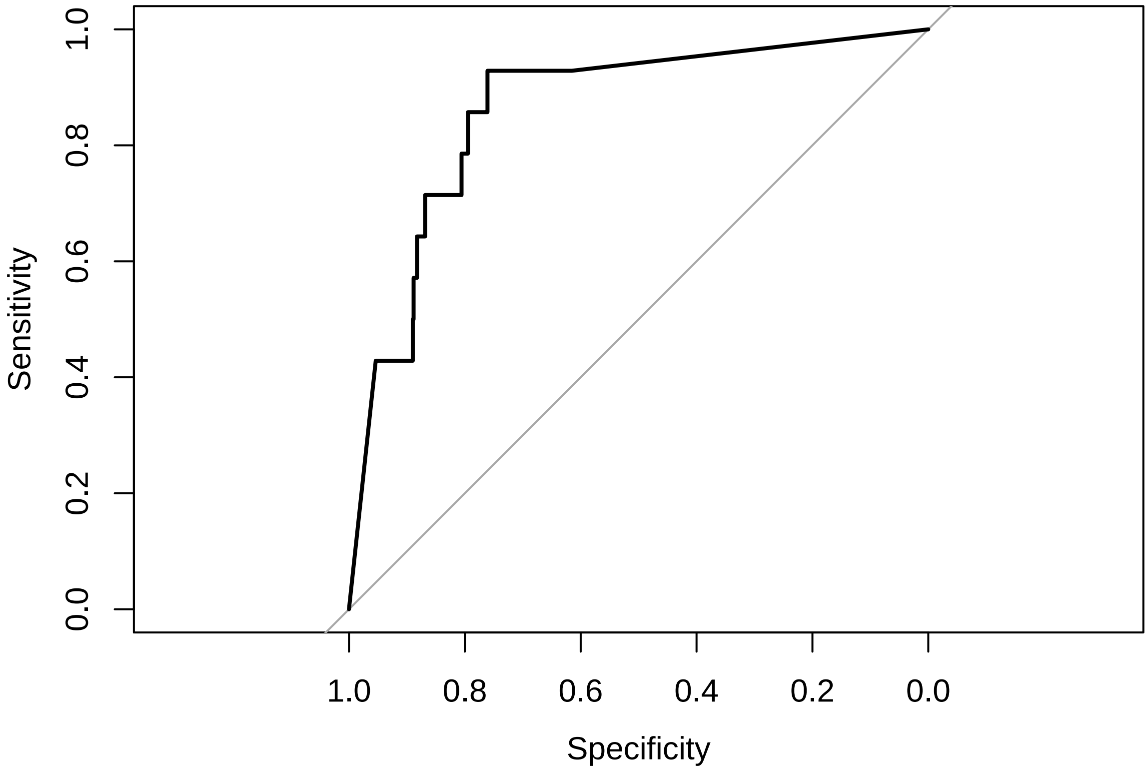
```
##
## Data: predict_mlr1 in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9488
```

```r
plot(auc_poisson)
```



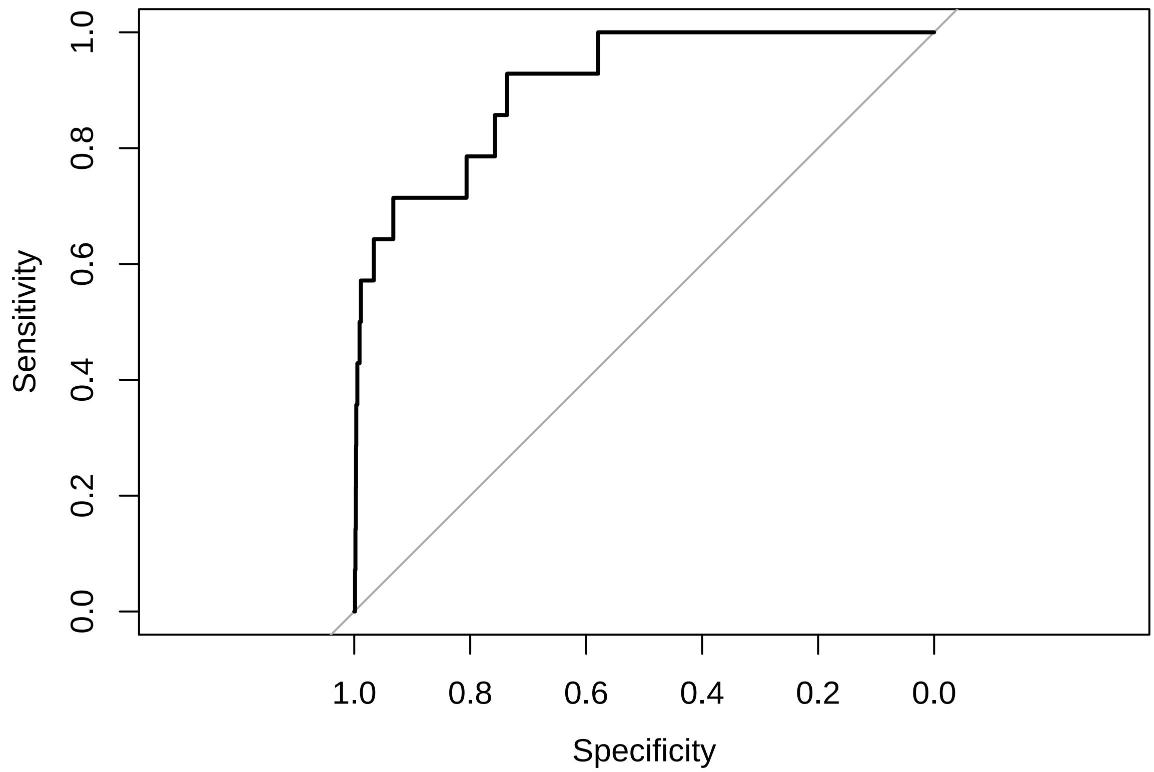

```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_poisson1)
##
## Data: predict_poisson1 in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.8639
```

```r
plot(auc_logit1)
```



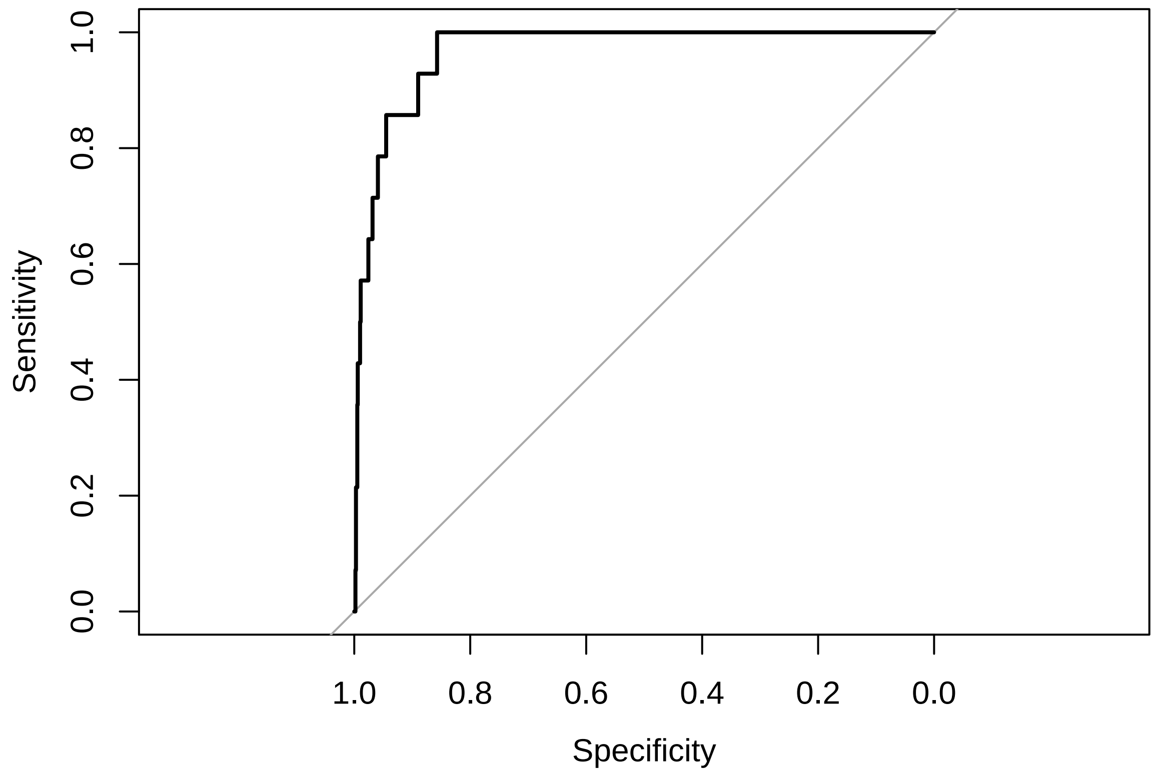

```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_logit1)
##
## Data: predict_logit1 in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.8614
```

```r
plot(auc_backward)
```



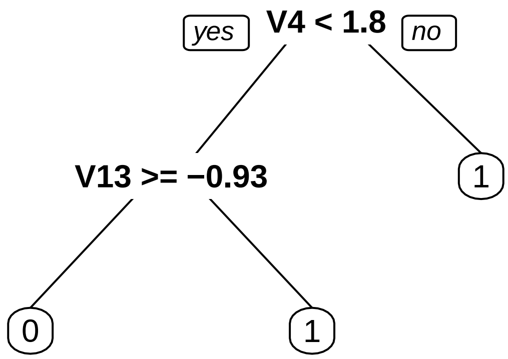

```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_backward)
##
## Data: predict_backward in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.91
```

```r
plot(auc_forward)
```

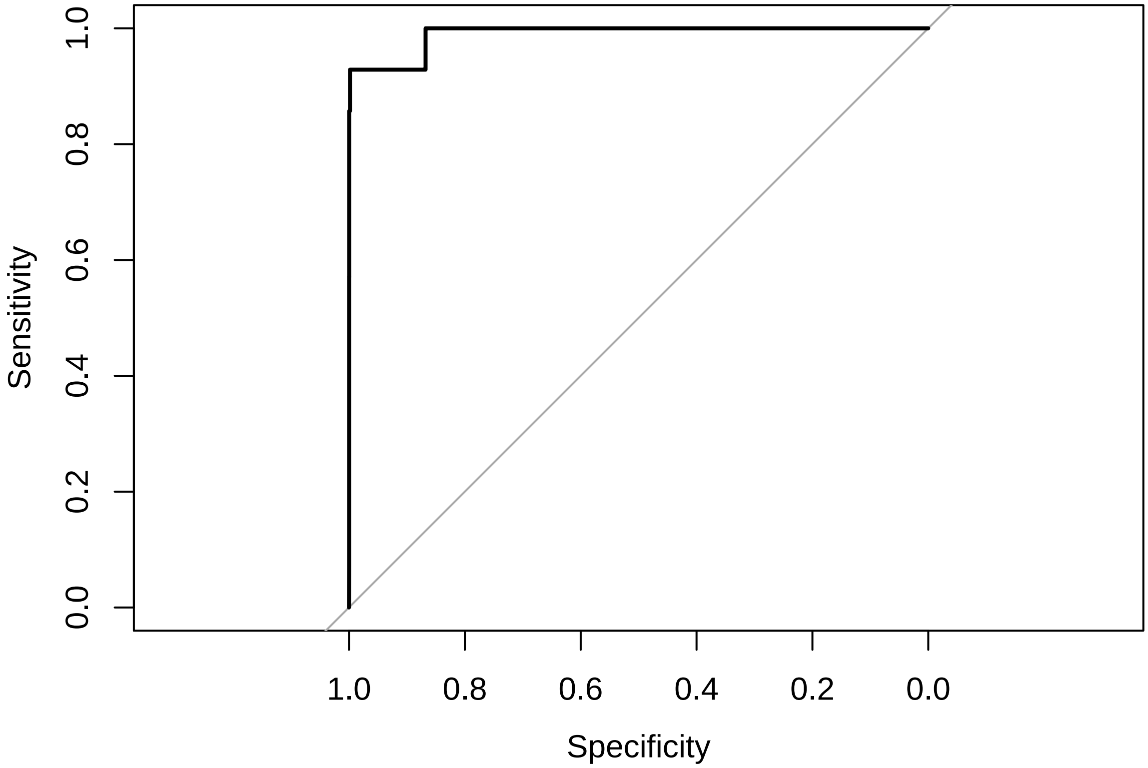



```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_forward)
##
## Data: predict_forward in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9679
```

```
prp(decision)
```

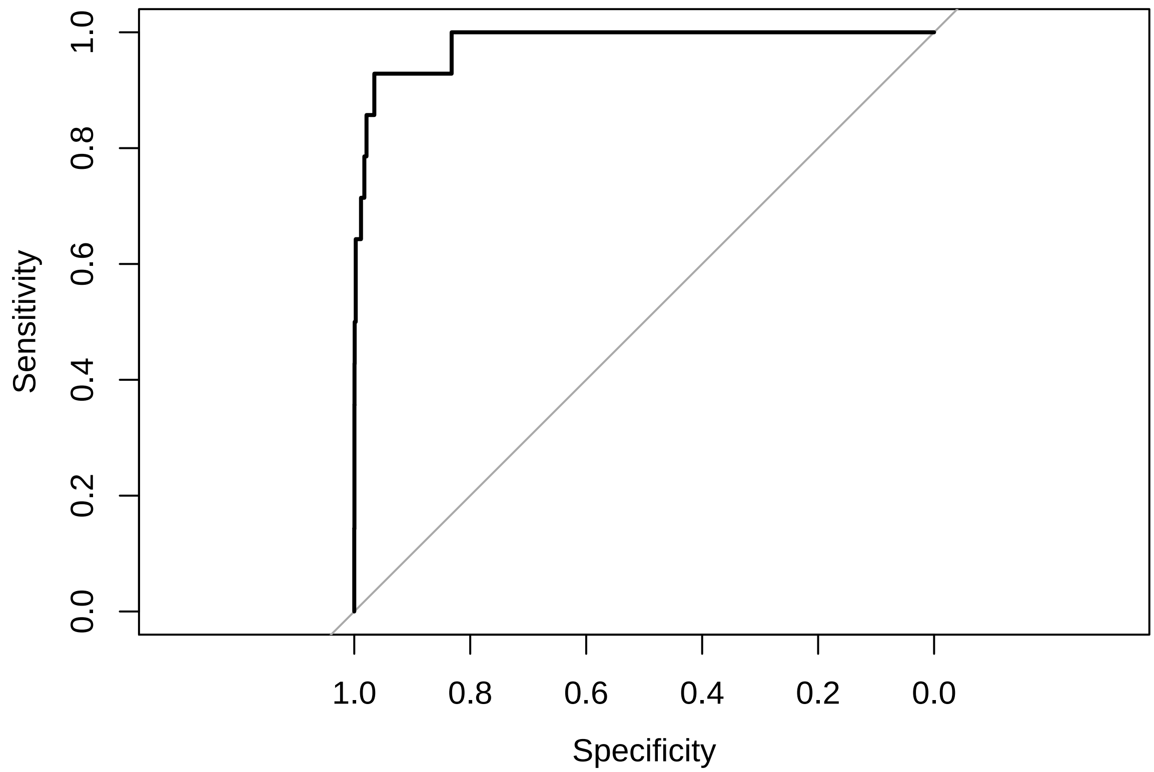



```r
plot(auc_rforest)
```




```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_rforest)
##
## Data: predict_rforest in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9902
```

```r
plot(auc_svm)
```




```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_svm)
##
## Data: predict_svm in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9815
```

```r
accuracy.meas(train_eval$Class, predict_mlr1)
```

```
##
## Call:
## accuracy.meas(response = train_eval$Class, predicted = predict_mlr1)
##
## Examples are labelled as positive when predicted is greater than 0.5
##
## precision: 0.014
## recall: 0.786
## F: 0.014
```

```r
accuracy.meas(train_eval$Class, predict_poisson1)
```

```
##
## Call:
## accuracy.meas(response = train_eval$Class, predicted = predict_poisson1)
##
## Examples are labelled as positive when predicted is greater than 0.5
##
## precision: 0.008
```

```
## recall: 0.571
## F: 0.008
```

```r
accuracy.meas(train_eval$Class, predict_logit1)
```

```
##
## Call:
## accuracy.meas(response = train_eval$Class, predicted = predict_logit1)
##
## Examples are labelled as positive when predicted is greater than 0.5
##
## precision: 0.010
## recall: 0.714
## F: 0.010
```

```r
accuracy.meas(train_eval$Class, predict_backward)
```

```
##
## Call:
## accuracy.meas(response = train_eval$Class, predicted = predict_backward)
##
## Examples are labelled as positive when predicted is greater than 0.5
##
## precision: 0.011
## recall: 0.714
## F: 0.011
```

```r
accuracy.meas(train_eval$Class, predict_forward)
```

```
##
## Call:
## accuracy.meas(response = train_eval$Class, predicted = predict_forward)
##
## Examples are labelled as positive when predicted is greater than 0.5
##
## precision: 0.034
## recall: 0.786
## F: 0.032
```

```r
accuracy.meas(train_eval$Class, predict_decision)
```

```
##
## Call:
## accuracy.meas(response = train_eval$Class, predicted = predict_decision)
##
## Examples are labelled as positive when predicted is greater than 0.5
##
## precision: 0.002
## recall: 1.000
## F: 0.002
```

```r
accuracy.meas(train_eval$Class, predict_rforest)
```

```
##
## Call:
## accuracy.meas(response = train_eval$Class, predicted = predict_rforest)
##
## Examples are labelled as positive when predicted is greater than 0.5
```

```
##
## precision: 0.047
## recall: 0.929
## F: 0.045
```

```r
accuracy.meas(train_eval$Class, predict_svm)
```

```
##
## Call:
## accuracy.meas(response = train_eval$Class, predicted = predict_svm)
##
## Examples are labelled as positive when predicted is greater than 0.5
##
## precision: 0.060
## recall: 0.857
## F: 0.056
```