DATA698 - Data Cleaning and Look Through

Max Wagner

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

```
## Warning: package 'ROSE' was built under R version 3.3.3
```

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

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

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

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```r
require("e1071")


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


#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),]


#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] -71401.37
```

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

```
##
##     FALSE TRUE
##   0  7485    1
##   1    10    4
```

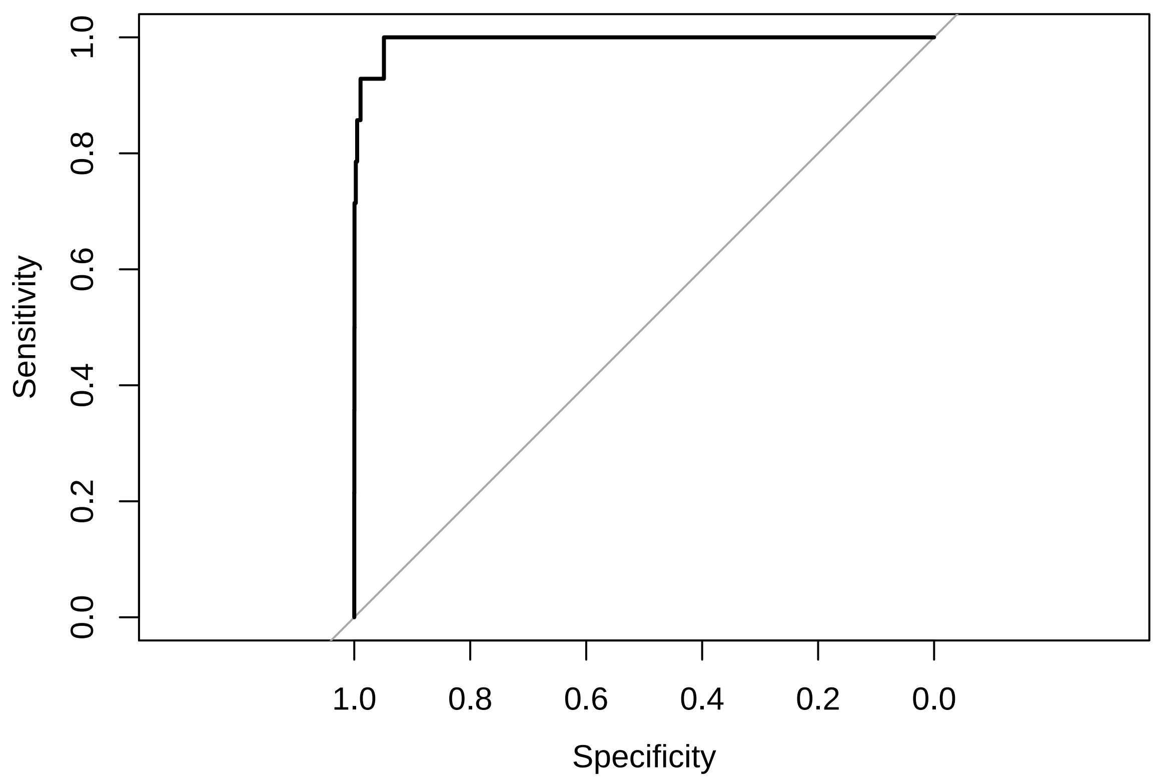
```
mysd(predict_mlr1, train_eval$Class)
```

```
## [1] 0.00414942
```

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

```
## [1] 0.004008179
```

```
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: 0.9949
```

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

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

```r
BIC(poisson1)
```

```
## [1] 487.1279
```

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

```
##
##     FALSE TRUE
##   0  7484    2
##   1    12    2
```

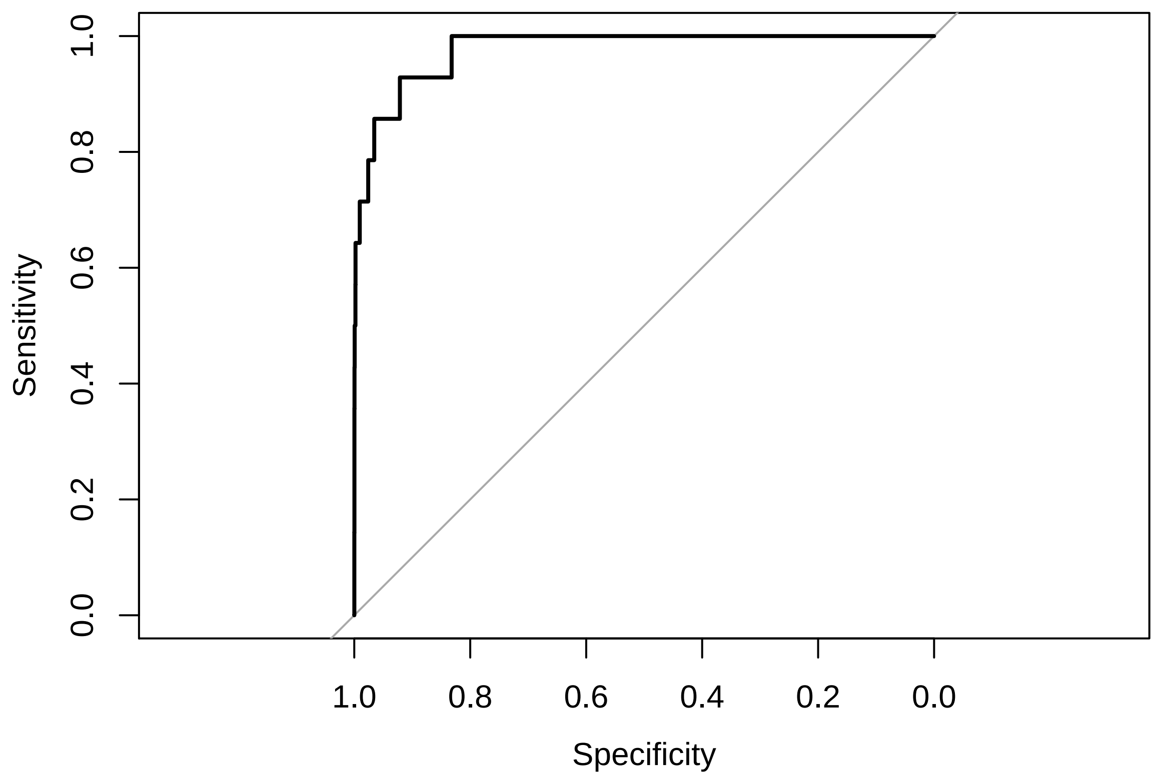
```r
mysd(predict_poisson1, train_eval$Class)
```

```
## [1] 0.002839854
```

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

```
## [1] 0.002619012
```

```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: 0.9771
```

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

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

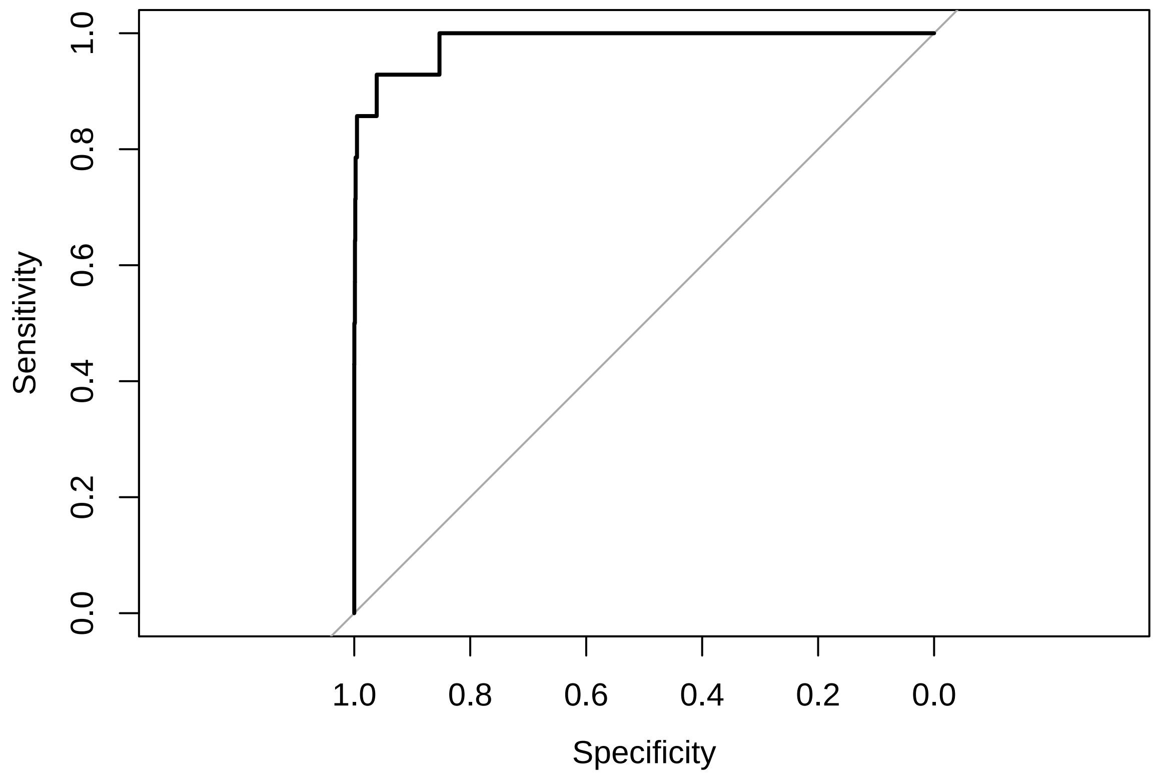
```r
BIC(logit1)
```

```
## [1] 440.7449
```

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

```
##
##     FALSE TRUE
##   0  7484    2
##   1     7    7
```

```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: 0.9859
```

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

```
## [1] -71455.25
```

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

```
##
##     FALSE TRUE
##   0  7485    1
##   1    10    4
```

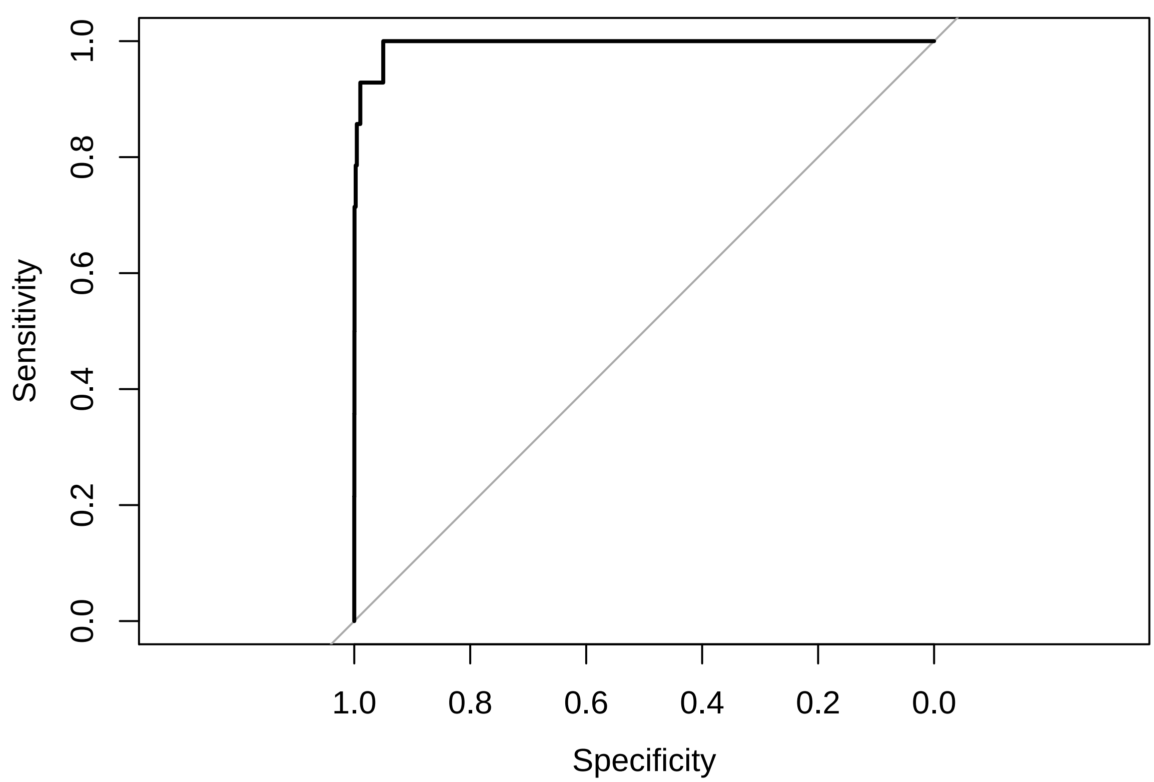
```r
mysd(predict_backward, train_eval$Class)
```

```
## [1] 0.0041126
```

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

```
## [1] 0.003986638
```

```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: 0.9951
```

```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] -71455.25
```

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

```
##
##     FALSE TRUE
##   0  7485    1
##   1    10    4
```

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

```
## [1] 0.0041126
```

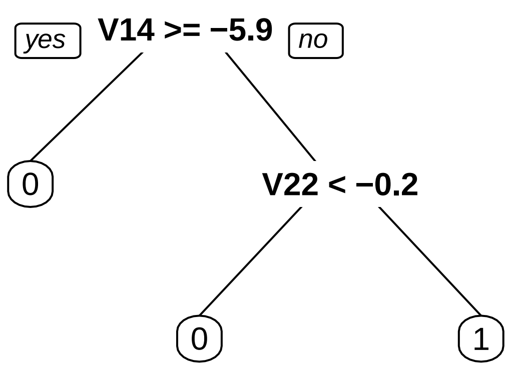
```r
myse(predict_forward, train_eval$Class)
```

```
## [1] 0.003986638
```

```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: 0.9951
```

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




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

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

```
## Prediction    0    1
##          0 7484    2
##          1    8    6
##
##                Accuracy : 0.9987
##                  95% CI : (0.9975, 0.9994)
##     No Information Rate : 0.9989
##     P-Value [Acc > NIR] : 0.8160
##
##                   Kappa : 0.5448
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.9989
##             Specificity : 0.7500
##          Pos Pred Value : 0.9997
##          Neg Pred Value : 0.4286
##              Prevalence : 0.9989
##          Detection Rate : 0.9979
##    Detection Prevalence : 0.9981
##       Balanced Accuracy : 0.8745
##
##        'Positive' Class : 0
##
```

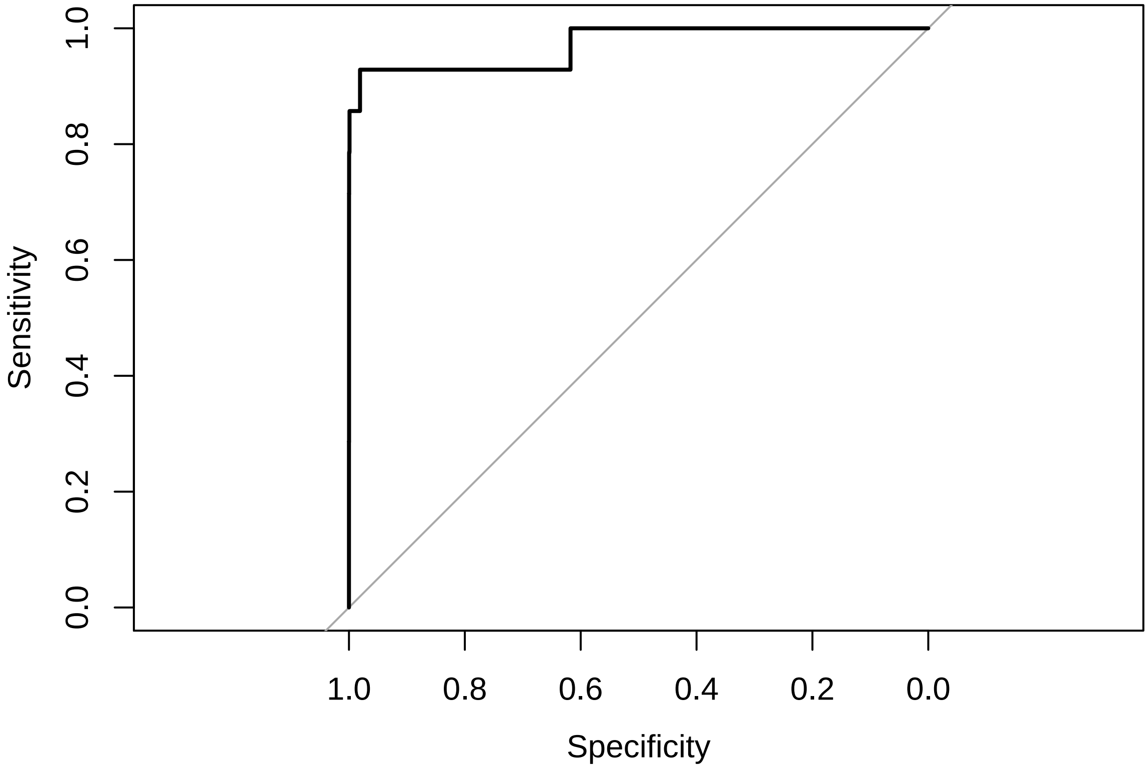
```r
#decision tree random forest (kind of broke with raw data)
rforest <- randomForest(Class ~ ., data = train_model)
```

```
## 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  7485    1
##   1     6    8
```

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

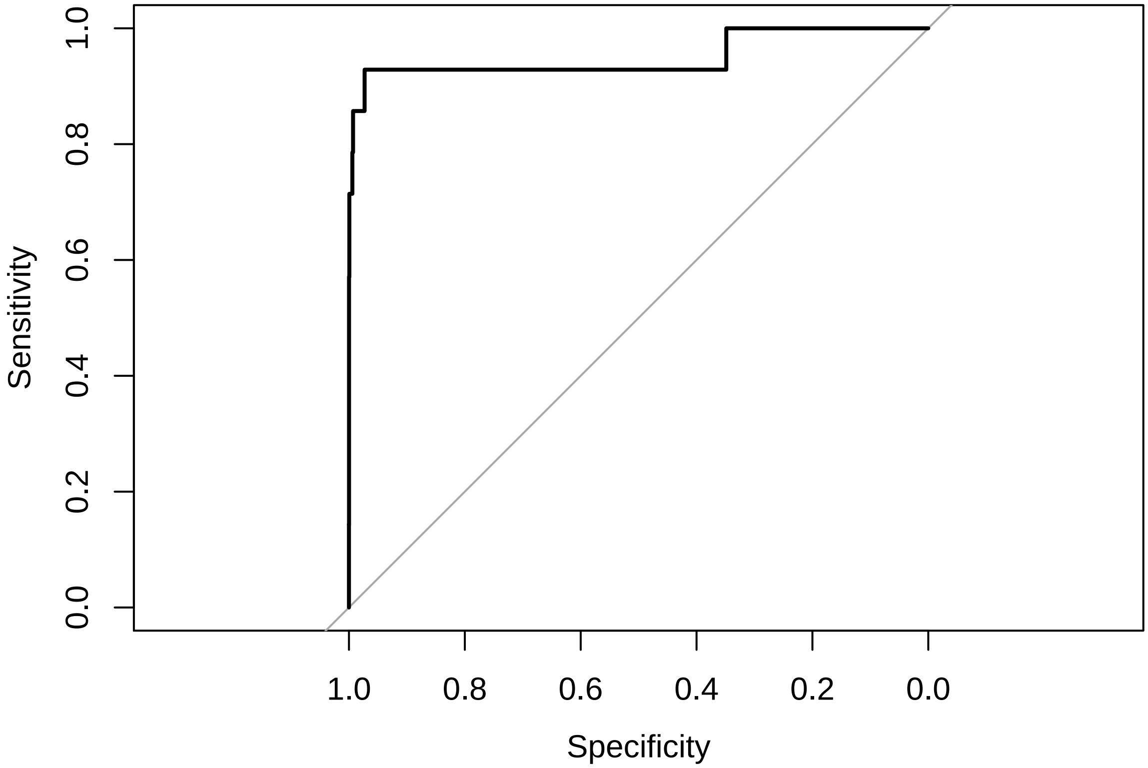



```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_rforest)
##
## Data: predict_rforest in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9712
```

```
#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
##   0  7486
##   1    14
```

```
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: 0.9505
```

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

```
##
##     FALSE TRUE
##   0  7485    1
##   1    10    4
```

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

```
##
##     FALSE TRUE
##   0  7484    2
##   1    12    2
```

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

```
##
##     FALSE TRUE
##   0  7484    2
##   1     7    7
```

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

```
##
##     FALSE TRUE
##   0  7485    1
##   1    10    4
```

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

```
##
##     FALSE TRUE
##   0  7485    1
##   1    10    4
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7484    2
##          1    8    6
##
##                Accuracy : 0.9987
##                  95% CI : (0.9975, 0.9994)
##     No Information Rate : 0.9989
##     P-Value [Acc > NIR] : 0.8160
##
##                   Kappa : 0.5448
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.9989
##             Specificity : 0.7500
##          Pos Pred Value : 0.9997
##          Neg Pred Value : 0.4286
##              Prevalence : 0.9989
##          Detection Rate : 0.9979
##    Detection Prevalence : 0.9981
##       Balanced Accuracy : 0.8745
##
##        'Positive' Class : 0
##
```

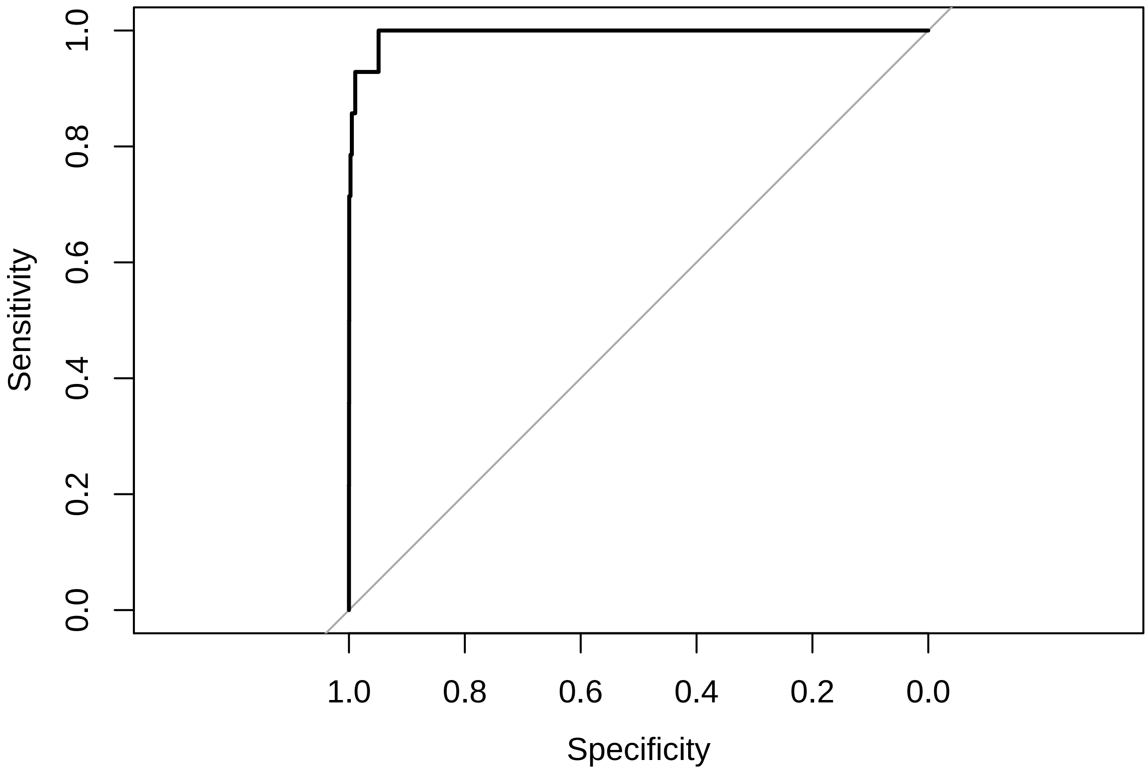
```r
table(train_eval$Class, predict_rforest > 0.5)
```

```
##
##     FALSE TRUE
##   0  7485    1
##   1     6    8
```

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

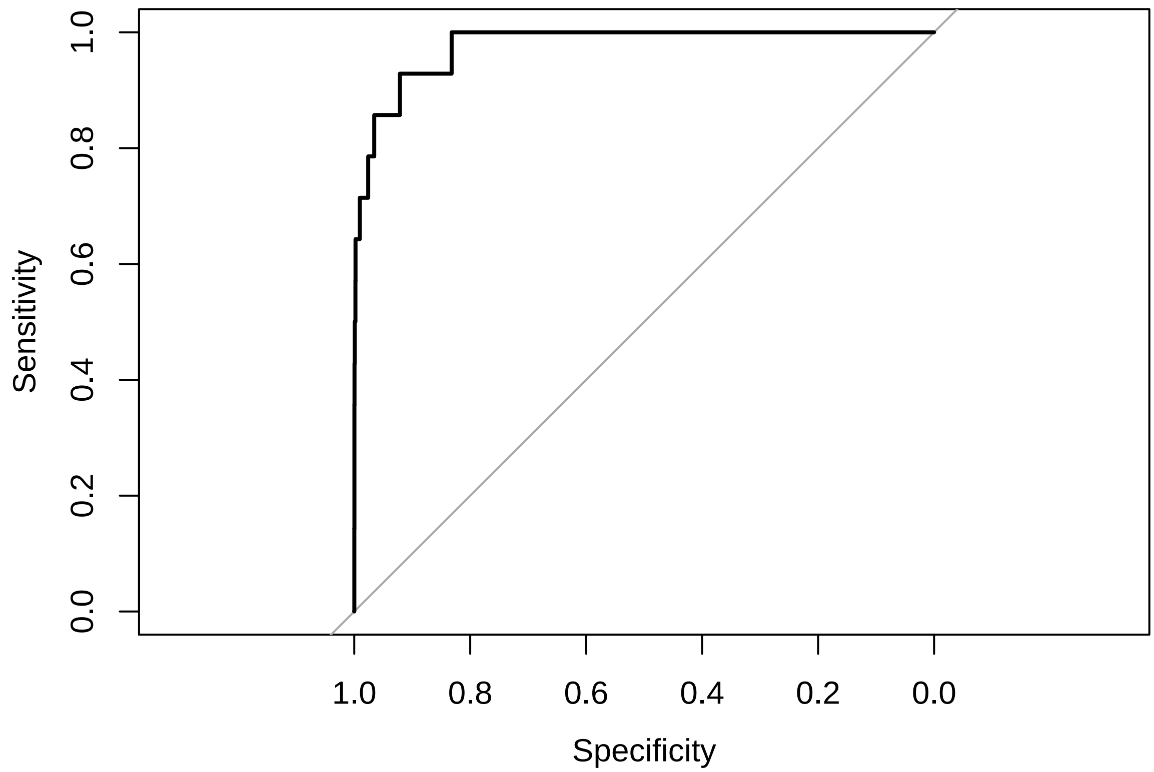
```
##
##     FALSE
##   0  7486
##   1    14
```

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



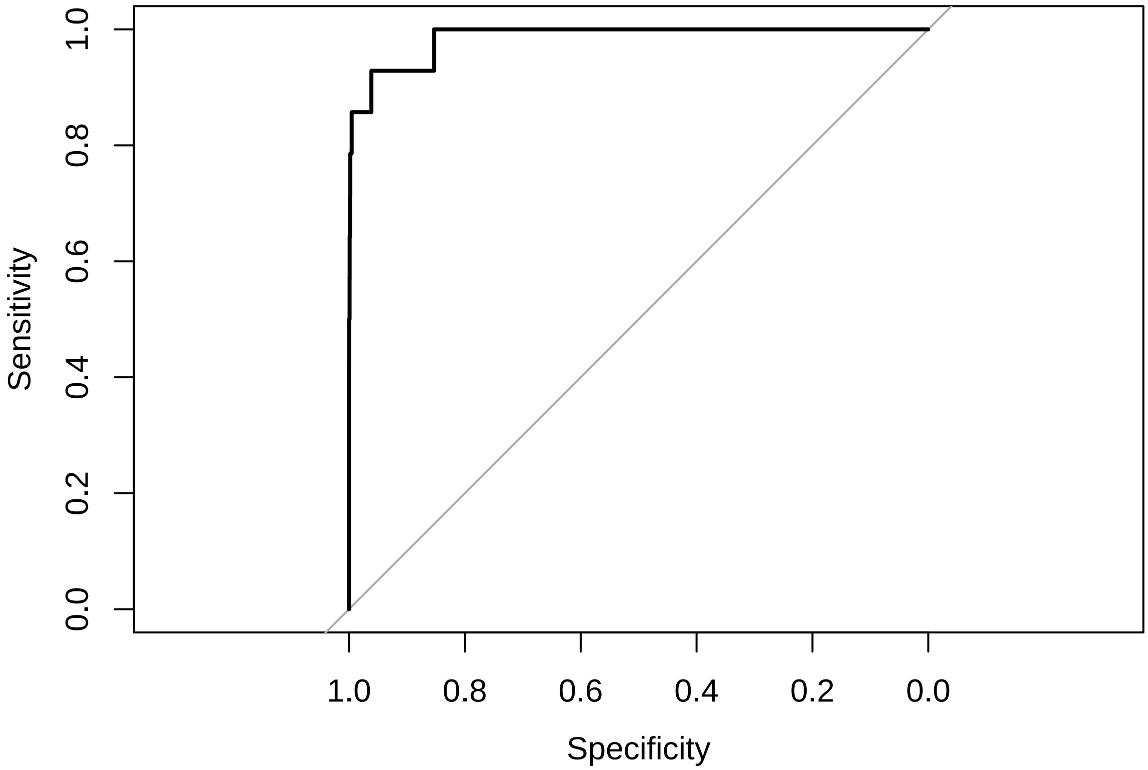

```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_mlr1)
##
## Data: predict_mlr1 in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9949
```

```r
plot(auc_poisson)
```



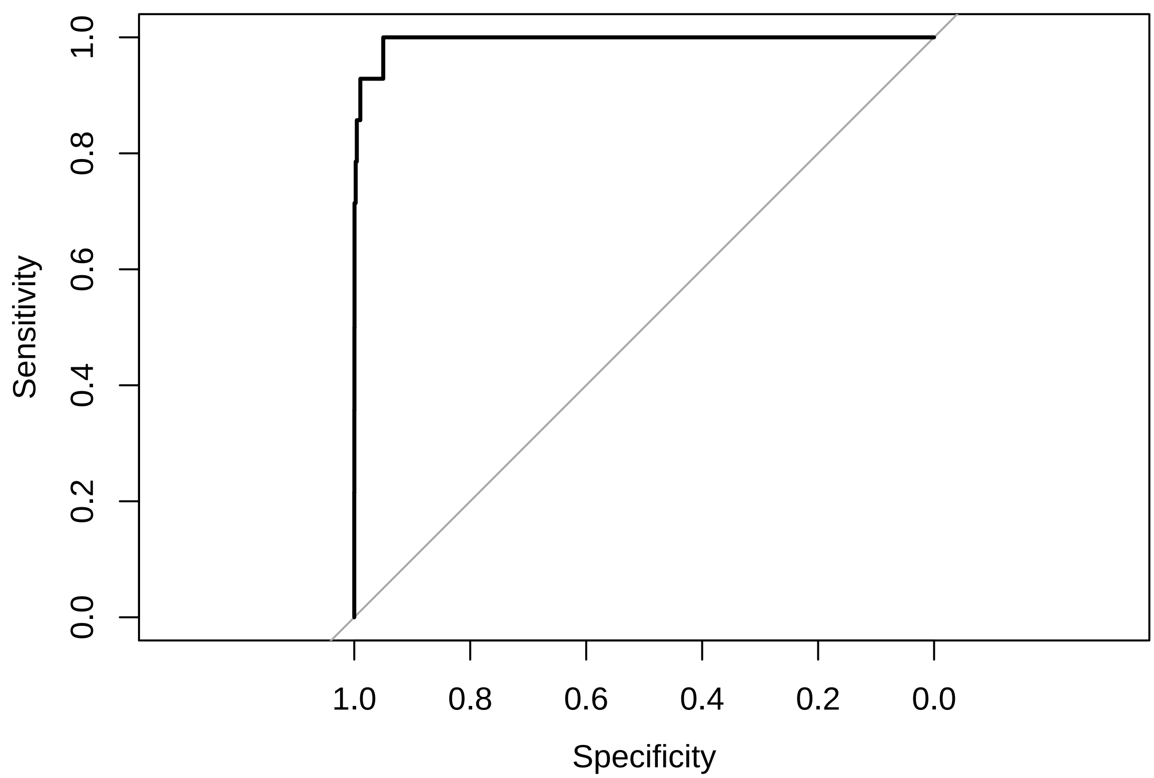

```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_poisson1)
##
## Data: predict_poisson1 in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9771
```

```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: 0.9859
```

```r
plot(auc_backward)
```

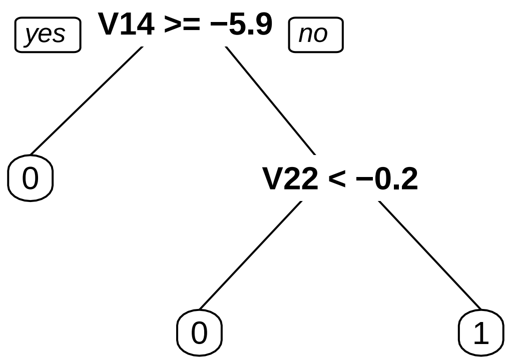



```
##
## Call:
## roc.default(response = train_eval$Class, predictor = predict_backward)
##
## Data: predict_backward in 7486 controls (train_eval$Class 0) < 14 cases (train_eval$Class 1).
## Area under the curve: 0.9951
```

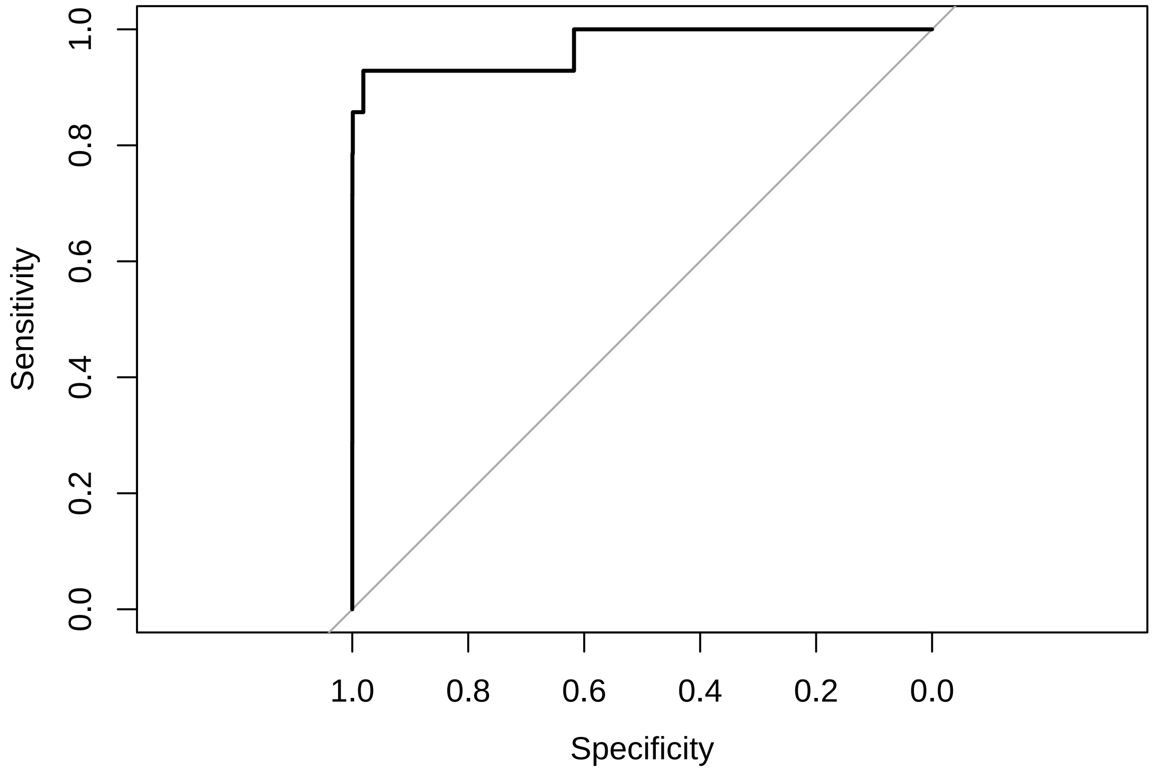
```
plot(auc_forward)
```

```
##
## 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: 0.9951
```

```
prp(decision)
```

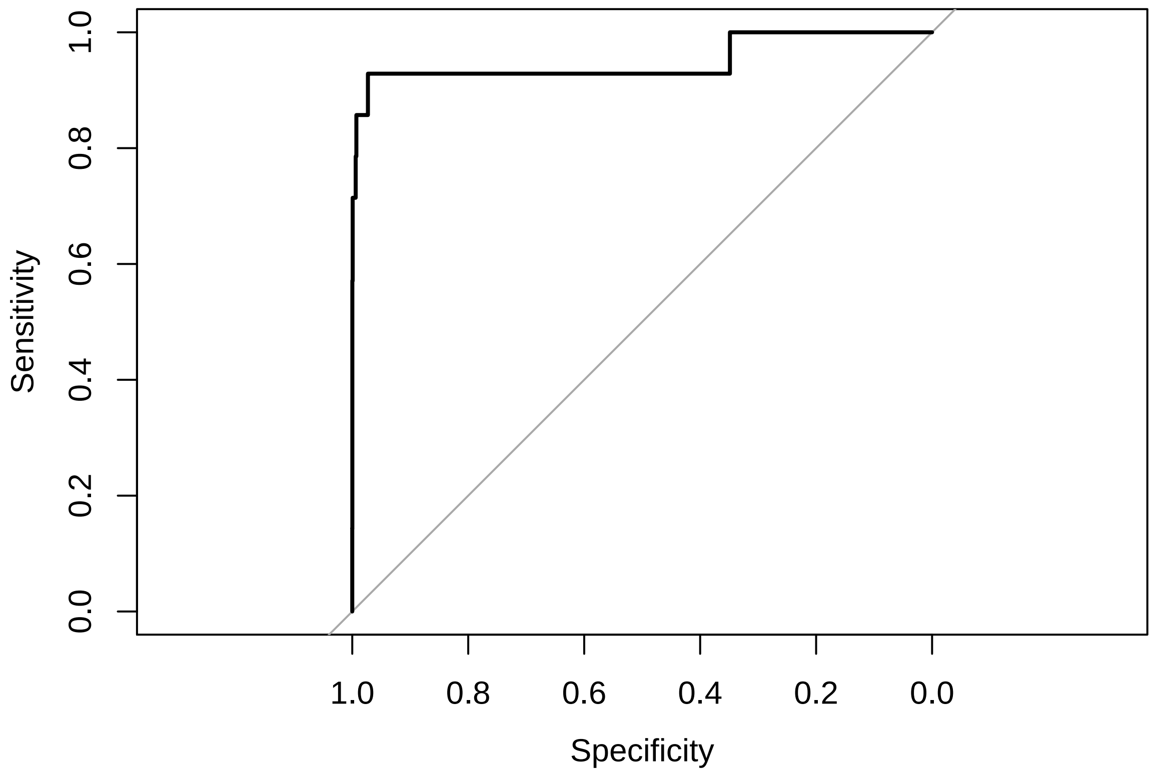



```r
plot(auc_rforest)
```




```
##
## 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: 0.9712
```

```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: 0.9505
```

```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: 0.800
## recall: 0.286
## F: 0.211
```

```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: 0.500
```

```
## recall: 0.143
## F: 0.111
```

```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: 0.778
## recall: 0.500
## F: 0.304
```

```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: 0.800
## recall: 0.286
## F: 0.211
```

```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: 0.800
## recall: 0.286
## F: 0.211
```

```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: 0.889
## recall: 0.571
## F: 0.348
```

```
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: NaN
## recall: 0.000
## F: NaN
```

```
varImpPlot(rforest)
```