Classification learning Notebook

Again, first define the path where the notebooks reside.

```
rm(list = ls(all = TRUE)) #clean up workspace
path <- "" #type the full path between the quotes and use
           #forward slashes ('/') to separate directories.
# Example:
path <- "C:/Users/koenn/OneDrive/School/DAV/Notebook 3"
```

First, we load the data (a binary file).

```
load(file.path(path,"ads_prevENGs.RData"))
set.seed(13768) #for reproducability
dat <- dat[sample(1:nrow(dat), 4000,replace = FALSE),] #limit sample size for speed
dat <- na.omit(dat)
summary(dat)
```

```
##  REC4        SEX                   COUNTRYBIRTH       AGE
##  0:2607   man  :3267   Netherlands       :2766   Min.   :12.00
##  1:1380   vrouw: 720   Morocco           : 114   1st Qu.:24.00
##                        Dutch Antilles    : 134   Median :32.00
##                        Surinam           : 147   Mean   :34.73
##                        Turkey            : 107   3rd Qu.:43.00
##                        Other Western     : 429   Max.   :89.00
##                        Other non-Western: 290
##    AGE1STCASE                          CRIMETYPE      PREVCASES
##  Min.   :11.0   traffic                    :1115   Min.   :  0.000
##  1st Qu.:18.0   property without violence: 936   1st Qu.:  0.000
##  Median :23.0   misc                       : 636   Median :  1.000
##  Mean   :27.2   violence                   : 610   Mean   :  4.021
##  3rd Qu.:34.0   public order               : 371   3rd Qu.:  4.000
##  Max.   :84.0   drugs                      : 252   Max.   :145.000
##                 (Other)                    :  67
```

```
randnum <- runif(nrow(dat)) #uniformly distributed random numbers in the range [0,1]
train <- dat[randnum<=0.6, ]
val <- dat[randnum >0.6 & randnum <= 0.8, ]
test <- dat[randnum > 0.8,]  #keep this data in a safe for when the modelling phase is over.
```

The data consist of adult convicts in the Netherlands in 2010. The dependent variable is a reconviction within four years (yes = 1 / no = 0). Independent Variables are sex (SEX), country of birth (COUNTRYBIRTH), age in years (AGE), age at first conviction in years (AGE1STCASE), crime type of the index case (CRIMETYPE), number of previous convictions (PREVCASES).

As we now have a binary outcome, linear regression is not adequate anymore (see James et al., 2013, pp [PM])

Therefore, we fit a logistic regression, which can be fit in the glm framework with the binomial family and logit link.

```
mod1 <- glm(REC4 ~ ., data = train, family = binomial)
summary(mod1)
```

```
##
## Call:
## glm(formula = REC4 ~ ., family = binomial, data = train)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0865  -0.8671  -0.5825   1.0710   2.5855
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        0.413683   0.189681   2.181  0.02919
## SEXvrouw                           0.361734   0.132536   2.729  0.00635
## COUNTRYBIRTHMorocco               -0.111303   0.313645  -0.355  0.72269
## COUNTRYBIRTHDutch Antilles         0.671712   0.264525   2.539  0.01111
## COUNTRYBIRTHSurinam                0.028561   0.259307   0.110  0.91230
## COUNTRYBIRTHTurkey                 0.196329   0.316921   0.619  0.53559
## COUNTRYBIRTHOther Western          0.064859   0.158714   0.409  0.68279
## COUNTRYBIRTHOther non-Western      0.325023   0.189412   1.716  0.08617
## AGE                               -0.023163   0.006623  -3.497  0.00047
## AGE1STCASE                        -0.031843   0.008013  -3.974 7.06e-05
## CRIMETYPEsexual                   -0.372028   0.670113  -0.555  0.57878
## CRIMETYPEproperty with violence    1.323945   0.595260   2.224  0.02614
## CRIMETYPEproperty without violence 0.223138   0.157547   1.416  0.15668
## CRIMETYPEpublic order              0.117914   0.190189   0.620  0.53527
## CRIMETYPEdrugs                    -0.078458   0.226167  -0.347  0.72866
## CRIMETYPEtraffic                  -0.153384   0.150856  -1.017  0.30927
## CRIMETYPEmisc                     -0.232945   0.174882  -1.332  0.18286
## PREVCASES                          0.110015   0.012692   8.668  < 2e-16
##
## (Intercept)                       *
## SEXvrouw                          **
## COUNTRYBIRTHMorocco
## COUNTRYBIRTHDutch Antilles        *
## COUNTRYBIRTHSurinam
## COUNTRYBIRTHTurkey
## COUNTRYBIRTHOther Western
## COUNTRYBIRTHOther non-Western     .
## AGE                               ***
## AGE1STCASE                        ***
## CRIMETYPEsexual
## CRIMETYPEproperty with violence   *
## CRIMETYPEproperty without violence
## CRIMETYPEpublic order
## CRIMETYPEdrugs
## CRIMETYPEtraffic
## CRIMETYPEmisc
## PREVCASES                         ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3086.9  on 2403  degrees of freedom
## Residual deviance: 2630.3  on 2386  degrees of freedom
## AIC: 2666.3
##
## Number of Fisher Scoring iterations: 5
```

Now we evaluate the accuracy of predicting recidivists in the validation set. The accuracy is the complement of the error rate. The accuracy is simply the percentage of cases classified incorrectly. We classify someone as recidivist when his/her score is 0.5 or larger. This score is called the cutoff-score. Of course, other choices for this value are possible.

```
library(pROC)
```

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

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

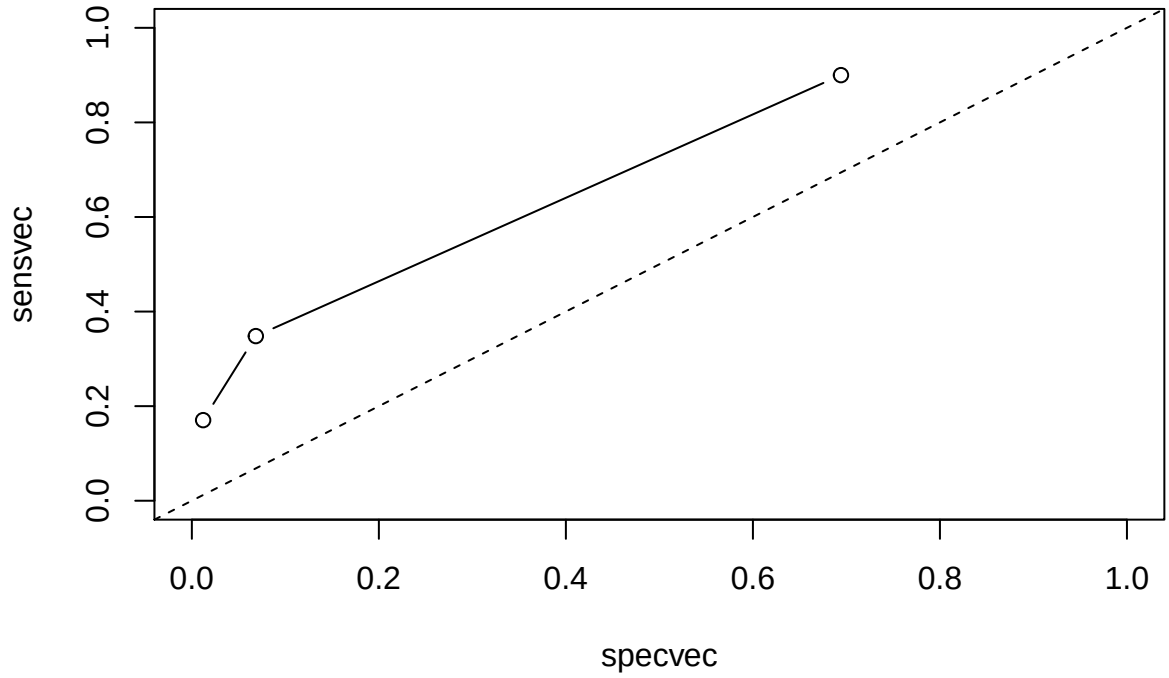
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
accuracy <- function(obs, pred){
  sum(obs == pred) / length(obs)
}
p1 <- predict(mod1, newdata = val, type = "response")

thres <- c(0.2, 0.5, 0.7) #vector of thresholds we want to try out
specvec <- character(length(thres))
sensvec <- character(length(thres))
for(i in 1:length(thres)){
pred <- ifelse(p1 >= thres[i], 1, 0) #predicted class at cutoff = 0.5

acc <- accuracy(val$REC4, pred)
TP <- sum(val$REC4==1&pred==1)
FN <- sum(val$REC4==1&pred==0)
TN <- sum(val$REC4==0&pred==0)
FP <- sum(val$REC4==0&pred==1)
sens <- TP/(FN+TP)
prec <- TP/(FP+TP)
recall <- TP/(FN+TP)
spec <- FP/(TN+FP)

cat("With threshold of ", thres[i], "\n")
cat("Accuracy is       ", acc, "\n")
cat("Error rate is     ", 1-acc, "\n")
cat("Sensitivity is    ", sens, "\n")
cat("Recall is         ", recall, "\n")
cat("Precision is      ", prec, "\n")
cat("Specificity is    ", spec, "\n\n")

specvec[i] <- spec
sensvec[i] <- sens
}
```

```
## With threshold of  0.2
## Accuracy is        0.5149935
## Error rate is      0.4850065
## Sensitivity is     0.9
## Recall is          0.9
## Precision is       0.4132653
## Specificity is     0.694165
```

```
##
## With threshold of  0.5
## Accuracy is        0.726206
## Error rate is      0.273794
## Sensitivity is     0.3481481
## Recall is          0.3481481
## Precision is       0.734375
## Specificity is     0.06841046
##
## With threshold of  0.7
## Accuracy is        0.7001304
## Error rate is      0.2998696
## Sensitivity is     0.1703704
## Recall is          0.1703704
## Precision is       0.8846154
## Specificity is     0.01207243
```

```
plot(specvec, sensvec, type = "b", xlim = c(0,1), ylim = c(0,1))
abline(0,1, lty = 2)
```



So, about 74% of the observations is classified correctly. let us look at the classification table (a.k.a. confusion matrix). The classification table crosses the observed labels with the predicted labels. The correct classifications are on the diagonal of the table.

```
table(val$REC4,pred)
```

```
##    pred
##       0   1
```

```
##   0 491   6
##   1 224  46
```

The model predicts a large portion of all of the observations as non-recidivists. When a model would classify all observations in the majority class, it could essentially ignore all covariate information. This special case is called the no information rate

```
max(sum(val$REC4==0),sum(val$REC4==1)) / nrow(val)
```

```
## [1] 0.6479791
```

Therefore, the model is performing better than the no information rate.

From previous research by Copas (1996), we know that the ratio of previous convictions divided by the criminal career length is an excellent predictor. This is an example of what machine learning calls 'feature engineering'. Moreover, as we already saw in an earlier notebook, the `PREVCASES` variable can better be logged

```
library(dplyr)
train <- mutate(train, copas = sqrt(PREVCASES / (AGE - AGE1STCASE + 1)))
val <- mutate(val, copas = sqrt(PREVCASES / (AGE - AGE1STCASE + 1)))
test <- mutate(test, copas = sqrt(PREVCASES / (AGE - AGE1STCASE + 1)))

mod2 <- glm(REC4 ~ SEX +  AGE + AGE1STCASE + COUNTRYBIRTH + CRIMETYPE + log(PREVCASES+1) + copas, data =
summary(mod2)
```

```
##
## Call:
## glm(formula = REC4 ~ SEX + AGE + AGE1STCASE + COUNTRYBIRTH +
##     CRIMETYPE + log(PREVCASES + 1) + copas, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6296  -0.7561  -0.5019   0.8314   2.5221
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -0.72317    0.21343  -3.388 0.000703
## SEXvrouw                            0.49193    0.13954   3.525 0.000423
## AGE                                -0.07278    0.01179  -6.172 6.73e-10
## AGE1STCASE                          0.03525    0.01290   2.733 0.006278
## COUNTRYBIRTHMorocco                -0.29677    0.32100  -0.925 0.355220
## COUNTRYBIRTHDutch Antilles          0.57535    0.27912   2.061 0.039279
## COUNTRYBIRTHSurinam                 0.01108    0.26413   0.042 0.966541
## COUNTRYBIRTHTurkey                  0.28284    0.33311   0.849 0.395839
## COUNTRYBIRTHOther Western           0.12001    0.16754   0.716 0.473789
## COUNTRYBIRTHOther non-Western       0.43129    0.19907   2.166 0.030274
## CRIMETYPEsexual                    -0.10566    0.68352  -0.155 0.877146
## CRIMETYPEproperty with violence     1.11022    0.63677   1.744 0.081244
## CRIMETYPEproperty without violence  0.29429    0.16560   1.777 0.075543
## CRIMETYPEpublic order               0.04920    0.20155   0.244 0.807136
## CRIMETYPEdrugs                     -0.07736    0.23975  -0.323 0.746939
## CRIMETYPEtraffic                   -0.09560    0.15850  -0.603 0.546417
## CRIMETYPEmisc                      -0.17395    0.18450  -0.943 0.345766
## log(PREVCASES + 1)                  1.23605    0.19028   6.496 8.25e-11
## copas                               0.34113    0.33409   1.021 0.307211
##
```

```
## (Intercept)                          ***
## SEXvrouw                             ***
## AGE                                  ***
## AGE1STCASE                           **
## COUNTRYBIRTHMorocco
## COUNTRYBIRTHDutch Antilles           *
## COUNTRYBIRTHSurinam
## COUNTRYBIRTHTurkey
## COUNTRYBIRTHOther Western
## COUNTRYBIRTHOther non-Western        *
## CRIMETYPEsexual
## CRIMETYPEproperty with violence      .
## CRIMETYPEproperty without violence   .
## CRIMETYPEpublic order
## CRIMETYPEdrugs
## CRIMETYPEtraffic
## CRIMETYPEmisc
## log(PREVCASES + 1)                   ***
## copas
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3086.9  on 2403  degrees of freedom
## Residual deviance: 2434.8  on 2385  degrees of freedom
## AIC: 2472.8
##
## Number of Fisher Scoring iterations: 4
```

```
p2 <- predict(mod2, newdata = val, type = "response")
pred <- ifelse(p2 >= 0.5, 1, 0) #predicted class at cutoff = 0.5
acc2 <- accuracy(val$REC4, pred)
cat("Accuracy is ", acc2)
```

```
## Accuracy is  0.7444589
```

A further improvement of the model accuracy.

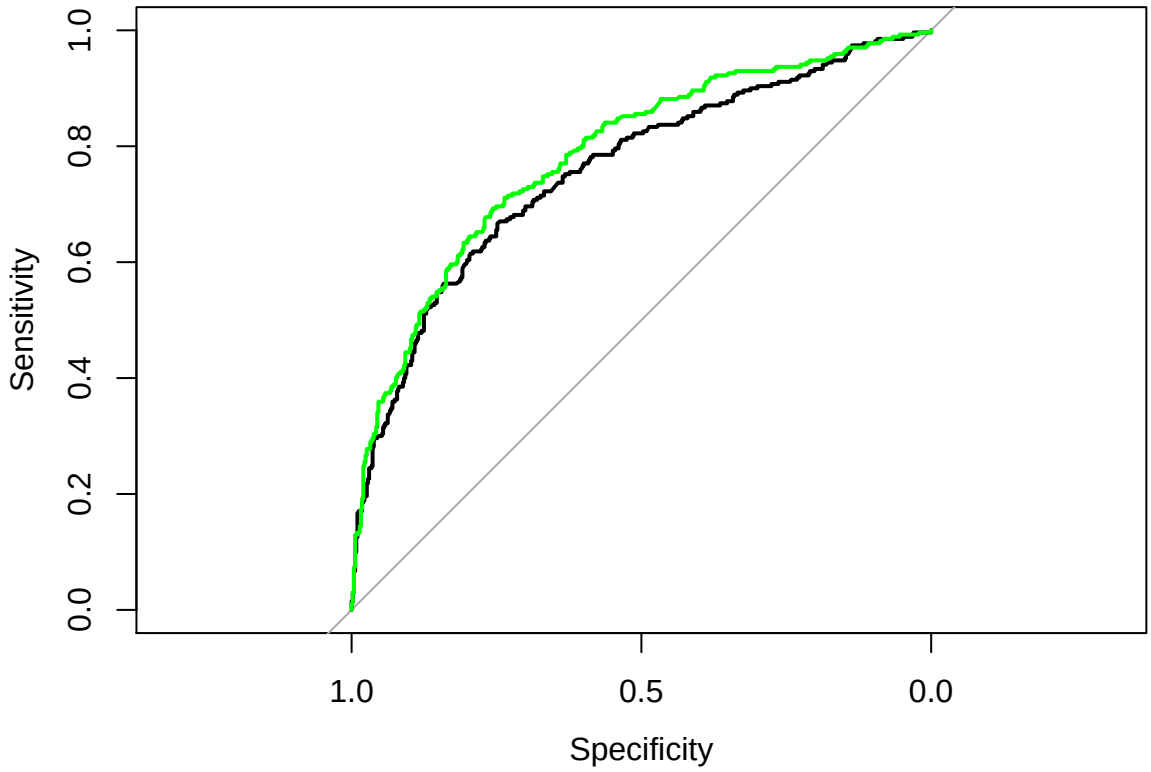
There are however other goals for prediction than classification of new observations.

Two other goals might be discrimination and calibration. Discrimination means that a model is able to correctly rank observations from low to high probability. Calibration means that the predicted probabilities correspond well to observed values.

First we look at the discrimination. A common measure for quantifying discrimination is the AUC (Area under the ROC-Curve). The ROC-curve is constructed by plotting the proportion of the positive class correctly classified (the sensitivity or true positive rate, TPR) against one minus the proportion of the negative class correctly classified (1 - specificity or false negative rate, FNR), over all values of cutoff-points. The AUC is the area under this curve and can be interpreted as the percentage of all negative-positive pairs that are ranked correctly on the predicted probability (i.e. risk score).

```
library(pROC)
r1 <- roc(val$REC4 ~ p1)
r2 <- roc(val$REC4 ~ p2)

plot(r1)
```

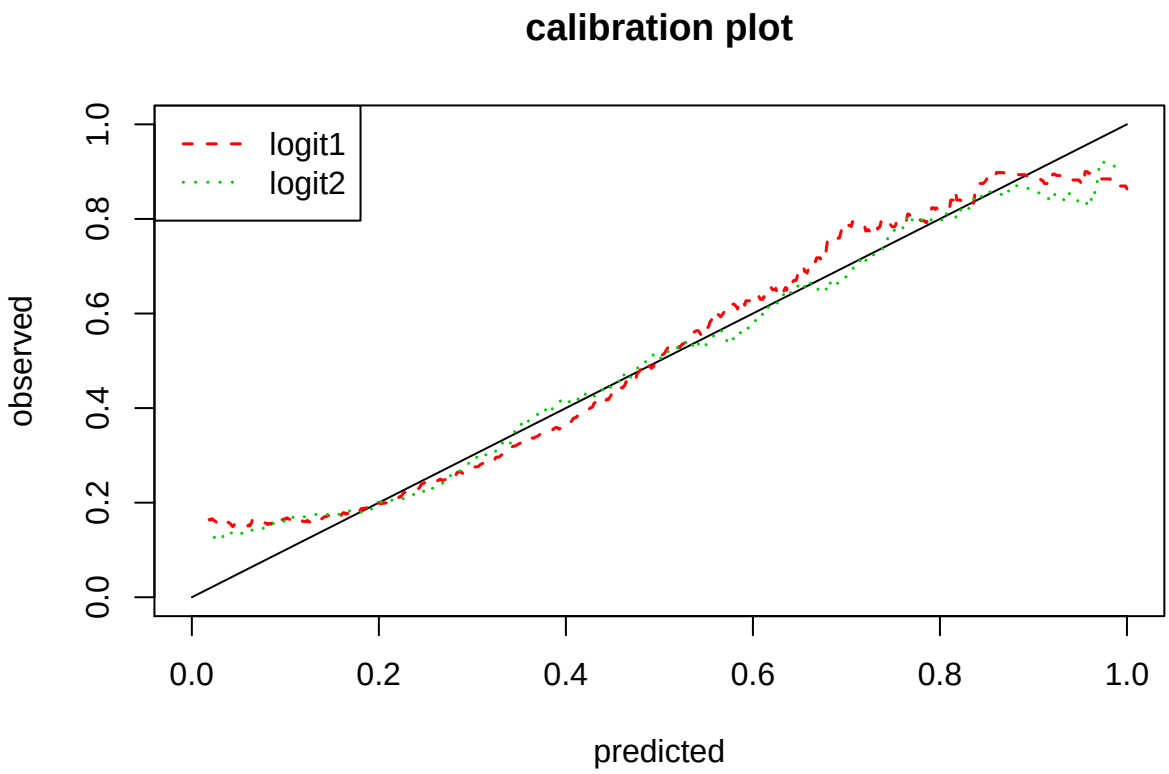
```
plot(r2, add = TRUE, col = "green")
```



The improvement of the new logit model is obvious.

Let's see how well calibrated the predicted probabilities are

```
#function for nonparametric calibration curve
calplot <- function(x, cl) {
  x <- as.matrix(x); if (is.factor(cl)) cl <- as.numeric(cl) - 1
  plot( c(0, 1), c(0, 1), type = "l", xlab = "predicted", ylab = "observed", main = "calibration plot"
  for (i in 1:ncol(x)) {
  lines( ksmooth(x[, i], cl, bandwidth = .3), lty = i + 1, lwd = 1.5, col = i + 1, type = "l" )
  }
}
calplot(cbind(p1,p2), val$REC4); legend("topleft", legend = c("logit1", "logit2"), lty = 2:3, lwd = 1.5
```



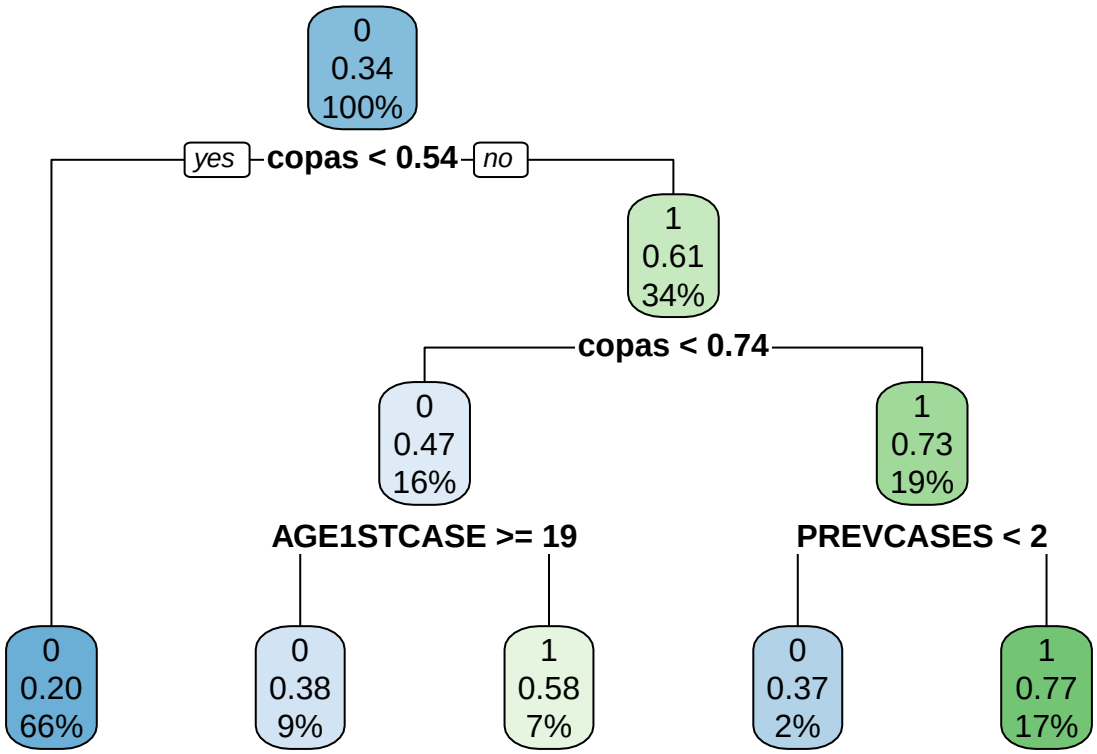
The second logistic model is also better calibrated. The first logit model generates some overprediction from predicted values in the range 0.2. to 0.4, and underprediction of the actual proportion in the region of about 0.55 to 0.9. Both models however have a hard time predicting the lower probabilities. The actual proportions are systematically larger.

A drawback of linear statistical models is that you have to manually specify which interactions to include in the model and specify transformation if the functional form is not linear.

Decision trees (or for continuous outcomes, regression trees) do not have this drawback. They are insensitive to transformations and implicitly model interaction effects by their recursive data splitting.

We now fit a single decision tree

```
library(rpart)
form <- factor(REC4) ~ SEX + PREVCASES + AGE + AGE1STCASE + COUNTRYBIRTH + CRIMETYPE + copas
mod3 <- rpart(form, data = train, control = rpart.control(cp = 0.008))
library(rpart.plot)
rpart.plot(mod3)
```

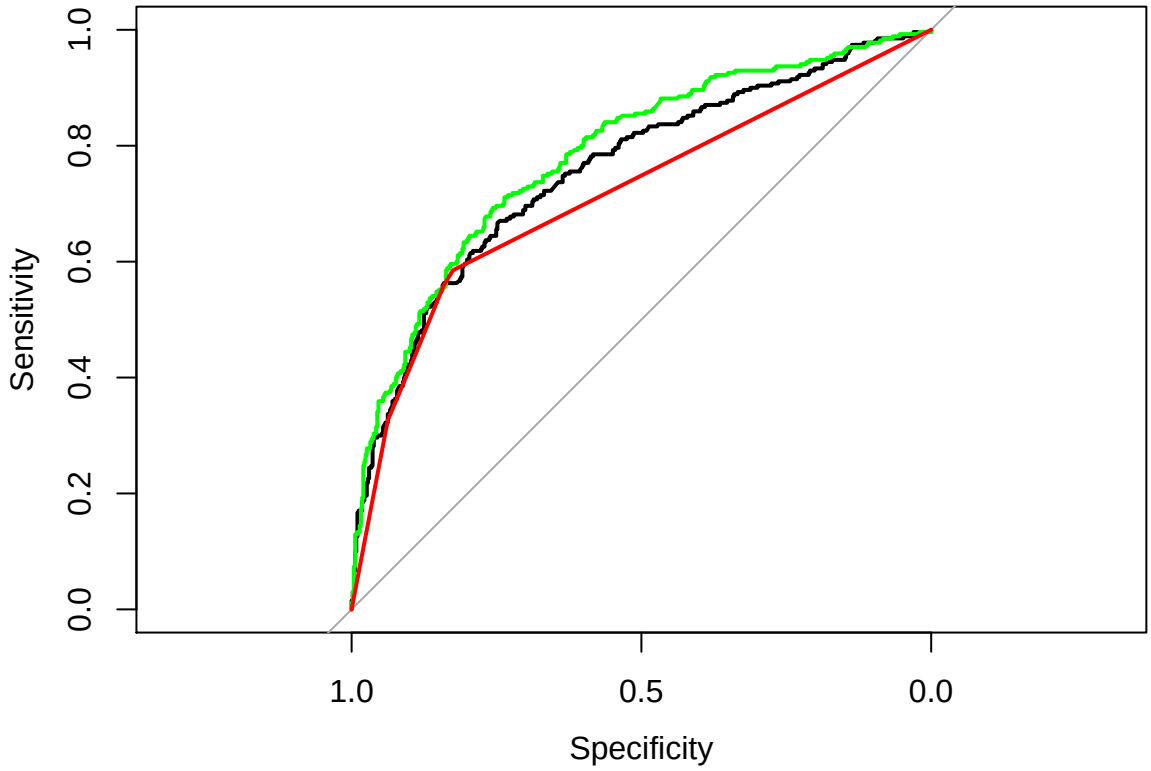


With the current settings, the tree model ends up with five terminal nodes. This means it can only predict eight distinct predicted probabilities (You can try out different values for the complexity parameter (`cp`, i.e. the minimal change in Gini index by allowing a split) and see how this affects the tree). How well does this model predict?

```
pred <- predict(mod3, type = "class", newdata = val)
p3 <- predict(mod3, type = "prob", newdata = val)[,2]
acc3 <- accuracy(val$REC4,pred); cat("Accuracy is ", acc3)
```

```
## Accuracy is  0.7314211
```

```
r3 <- roc(val$REC4 ~ p3)
plot(r1)
plot(r2, add = TRUE, col = "green")
plot(r3, add = TRUE, col = "red")
```



The accuracy is ok, but the AUC is lower than our initial logistic regression model.

Single classification trees (and regression trees also) are, besides being somewhat crude, very sensitive to small perturbations in the input data (and are thus high variance models). To counteract this property, one can apply bootstrap aggregation (bagging) of regression trees on bootstrap samples.

```
#Bagging demonstration.
set.seed(37729)
B <- 100 #set the number of bootstrap samples to z100
b <- nrow(train) #bootstrap sample size, usually as large as the training data
predmat <- matrix(0, nrow(val), B)

for (j in 1:B) {
  samp <- sample(1:nrow(train), size = b, replace = TRUE) #sample *with* replacement
  mod <- rpart( form, data = train[samp,],  #train on bootstrap sample
              control = rpart.control(cp = 0.008)   )
  predmat[,j] <- as.numeric(predict(mod, newdata = val, type = "class")) - 1 #validate on validation sa
  }
Mode <- function(x){
  # you would not believe R does not have this functionality by default!
  as.numeric(names(sort(- table(x)))[1])
}

#predicted class is majority vote
pred <- apply(predmat,1,Mode)

#averaged predicted class is pseudo-probability
p4 <- apply(predmat,1,mean)
```

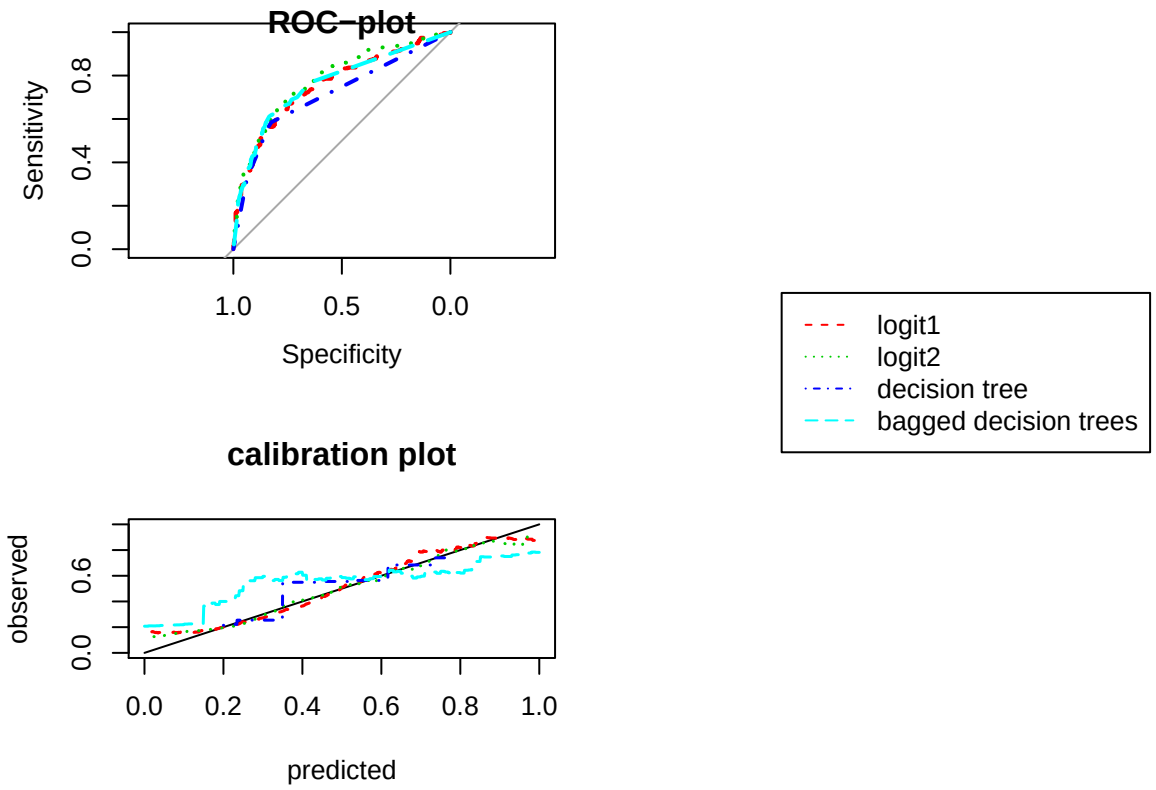
```
acc4 <- accuracy(val$REC4, ifelse(p4 < 0.5, 0, 1)); cat("Accuracy is ", acc4)
```

```
## Accuracy is  0.7470665
```

```
r4 <- roc(val$REC4, p4)
```

Indeed an improval in accuracy over a single regression tree. Does this also hold for the AUC?

```
layout(rbind(c(2, 1), c(3,1)))  #the numbers refer to the plot number in the grid
plot(1, type="n", axes=FALSE, xlab="", ylab="")
legend(1,1, legend = c("logit1","logit2","decision tree","bagged decision trees"),
       lty = 2:5, col = 2:5, xjust = 0.5, yjust = 0.5)
plot(r1, lty = 2, col = 2)
plot(r2, add = TRUE, col = 3, lty = 3)
plot(r3, add = TRUE, col = 4, lty = 4)
plot(r4, add = TRUE, col = 5, lty = 5)
title('ROC-plot')
calplot(cbind(p1,p2,p3,p4), val$REC4)
```



Indeed, a dramatic improvement over single trees, but bagging destroys the calibration, because it averages the class decisions instead of the individual tree probabilities.

You can speed up bagging by using the optimized program for random forests: when `mtry` (i.e. the number of sampled predictors at each tree node) equals the total number of variables, random forests are equivalent to bagging trees. Unfortunately, in this implementation you cannot set the complexity parameter, so the results will not be the same.

```
form.rf <- factor(REC4) ~ SEX + PREVCASES + AGE + AGE1STCASE + COUNTRYBIRTH + CRIMETYPE + copas
library(randomForest)
```

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

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

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

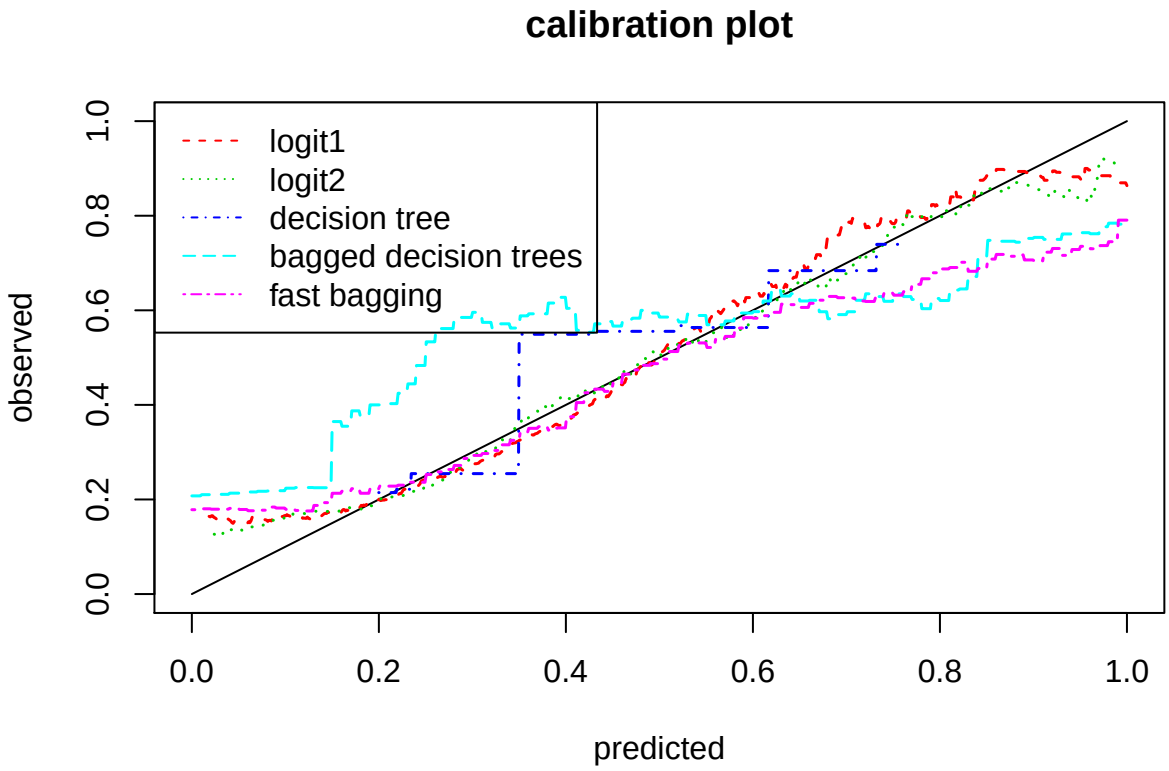
```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
mod5 <- randomForest(form.rf, data = train, mtry = 7, ntree = 100)
p5 <- predict(mod5, newdata = val, type = "prob")[,2]
acc5 <- accuracy(val$REC4, ifelse(p5<0.5,0,1)); acc5
```

```
## [1] 0.7288136
```

Note that rerunning the random forest (or bagging) will give different results, due to the random sampling of observations and variables (features).

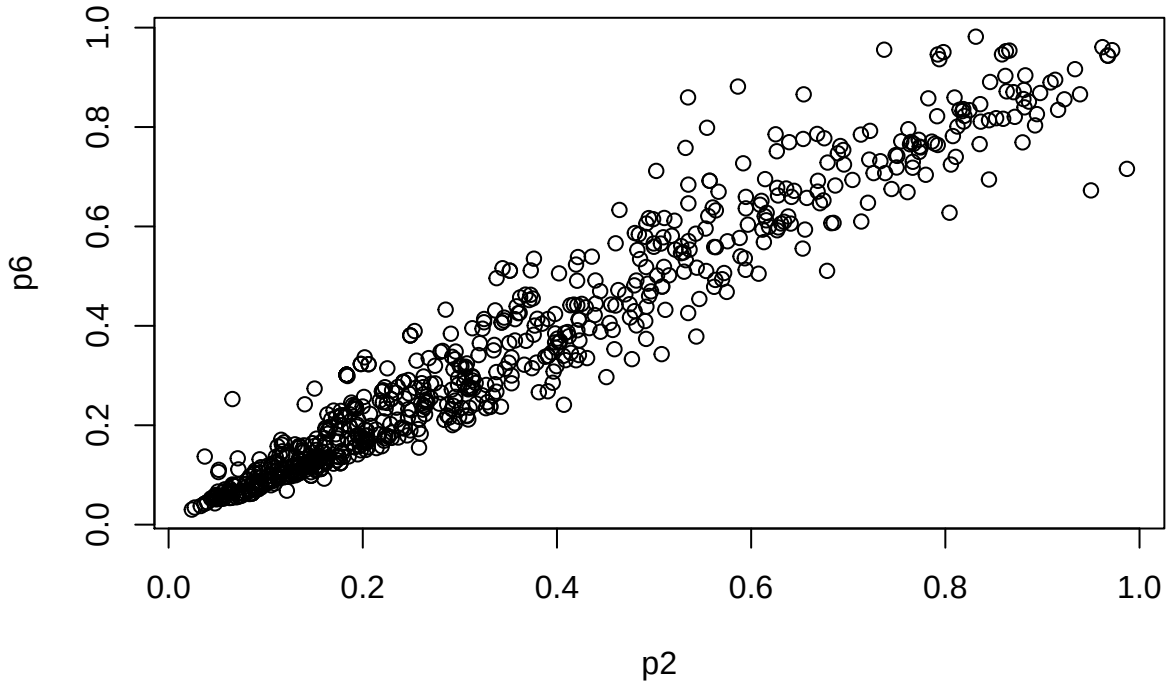
```
calplot(cbind(p1,p2,p3,p4,p5), val$REC4)
legend("topleft", legend = c("logit1","logit2","decision tree","bagged decision trees","fast bagging"),
```



The variability of bagging across runs can be diminished by fitting many more trees.

Bagging only works on high variance models. Therefore, when bagging on logistic regression (a low variance model) the performance should not improve.

```r
#Bagging demonstration on low variance model.
set.seed(37729)
B <- 100 #set the number of bootstrap samples to 100
b <- nrow(train) #bootstrap sample size, usually as large as the training data
predmat <- matrix(0, nrow(val), B)

library(splines)
form <- factor(REC4) ~
SEX + ns(PREVCASES, df = 10) +
ns(AGE, df = 10) +
ns(AGE1STCASE, df = 10) +
COUNTRYBIRTH +
CRIMETYPE +
ns(copas, df = 10)

for (j in 1:B) {
  samp <- sample(1:nrow(train), size = b, replace = TRUE) #sample *with* replacement
  mod <- glm( form, data = train[samp,],  #train on bootstrap sample
              family = binomial)
  predmat[,j] <- predict(mod, newdata = val, type = "response") #validate on validation sample
  } #averaged predictions
p6 <- apply(predmat,1,mean)
plot(p2,p6)
```



the predicted probabilities are basically the same, so the performance will also be the same. A useful byproduct of tree ensemble methods is variable importance measures. For decision trees, the variable importance of

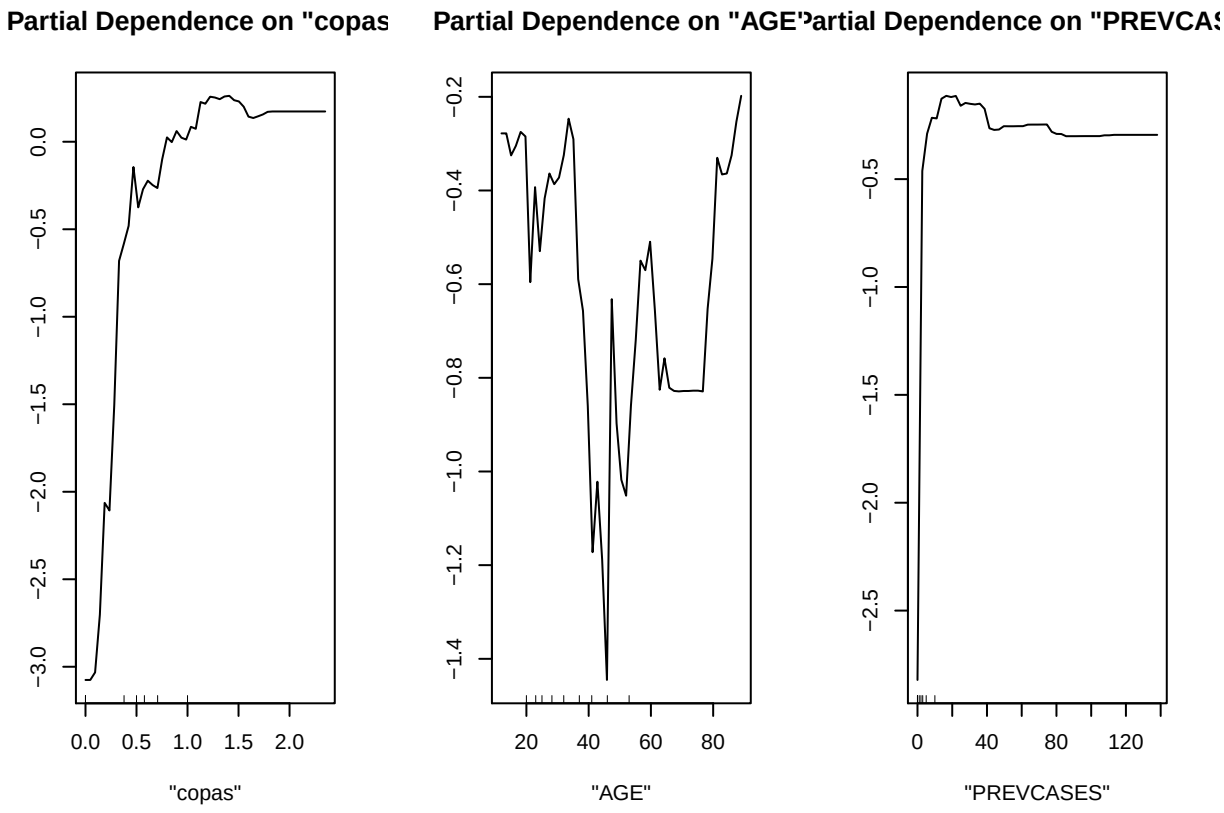
a predictor is the summed total amount that the Gini index decreased whenever there was a split on that predictor.

```
library(randomForest)
importance(mod5)
```

```
##               MeanDecreaseGini
## SEX                    27.92495
## PREVCASES              80.73990
## AGE                   173.68199
## AGE1STCASE            162.70421
## COUNTRYBIRTH           84.03978
## CRIMETYPE             123.51517
## copas                 340.43342
```

apparently, the Copas rate is by far the most important. Age and crime type are the next most important. The partial effect of variables on the outcome can be visualized by partial dependence plots.

```
par(mfrow = c(1,3))
partialPlot(mod5, pred.data = train, x.var="copas", which.class = 1)
partialPlot(mod5, pred.data = train, x.var="AGE", which.class = 1)
partialPlot(mod5, pred.data = train, x.var="PREVCASES", which.class = 1)
```



Note that partial dependence plots only make sense when there is not much interaction present in the data, as the effects of the other predictors are averaged out. This can be checked by individual conditional expectation plots (for the interested reader, see https://arxiv.org/abs/1309.6392, and R package `iceBox`). Finally, we compare the test error rates of our logistic regressions, decision tree and bagged decision trees with those of

- Naive Bayes

- Linear discriminant analysis
- K-nearest neighbors classification
- Stochastic gradient boosting

Both Naive Bayes and linear discriminant analysis do not need a validation data set or cross validation, as both do not have 'tuning parameters' that need to be set in advance.

```
library(e1071)
mod6 <- naiveBayes(factor(REC4) ~ SEX + PREVCASES + AGE + AGE1STCASE + COUNTRYBIRTH + CRIMETYPE + copas
mod7 <- MASS::lda(REC4 ~ SEX +  AGE + AGE1STCASE + COUNTRYBIRTH + CRIMETYPE + log(PREVCASES+1) + copas,

#for k-nn, we choose the best value of k (number of neighbors) on the validation data set using accurac
vars <- c("SEX","PREVCASES","AGE","AGE1STCASE","COUNTRYBIRTH","CRIMETYPE","copas")
kvec <- c(1,3,5,7,9,11,13,15,17)   #considered values for k
accvec <- rep(0,length(kvec)) #vector for storing accuracy values
library(class)
for (i in 1:length(kvec)) {
  mod <- knn( train = data.matrix(train[, vars]),
                                  test = data.matrix(val[, vars]),
                                  cl = train$REC4,
                                  k = kvec[i],
                                  prob = TRUE
  )
  accvec[i] <-
  accuracy(val$REC4, ifelse(1 - attributes(mod)$prob >= 0.5, 1, 0)) #one minus because model predicts c
}
cbind(kvec,accvec)
```

```
##       kvec    accvec
##  [1,]    1 0.6401565
##  [2,]    3 0.6414602
##  [3,]    5 0.6505867
##  [4,]    7 0.6492829
##  [5,]    9 0.6466754
##  [6,]   11 0.6414602
##  [7,]   13 0.6466754
##  [8,]   15 0.6479791
##  [9,]   17 0.6466754
```

```
k <- kvec[which.max(accvec)]
cat('chose k of ', k)
```

```
## chose k of  5
```

```
set.seed(577298)
library(gbm)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:rpart':
##
##     solder
```

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

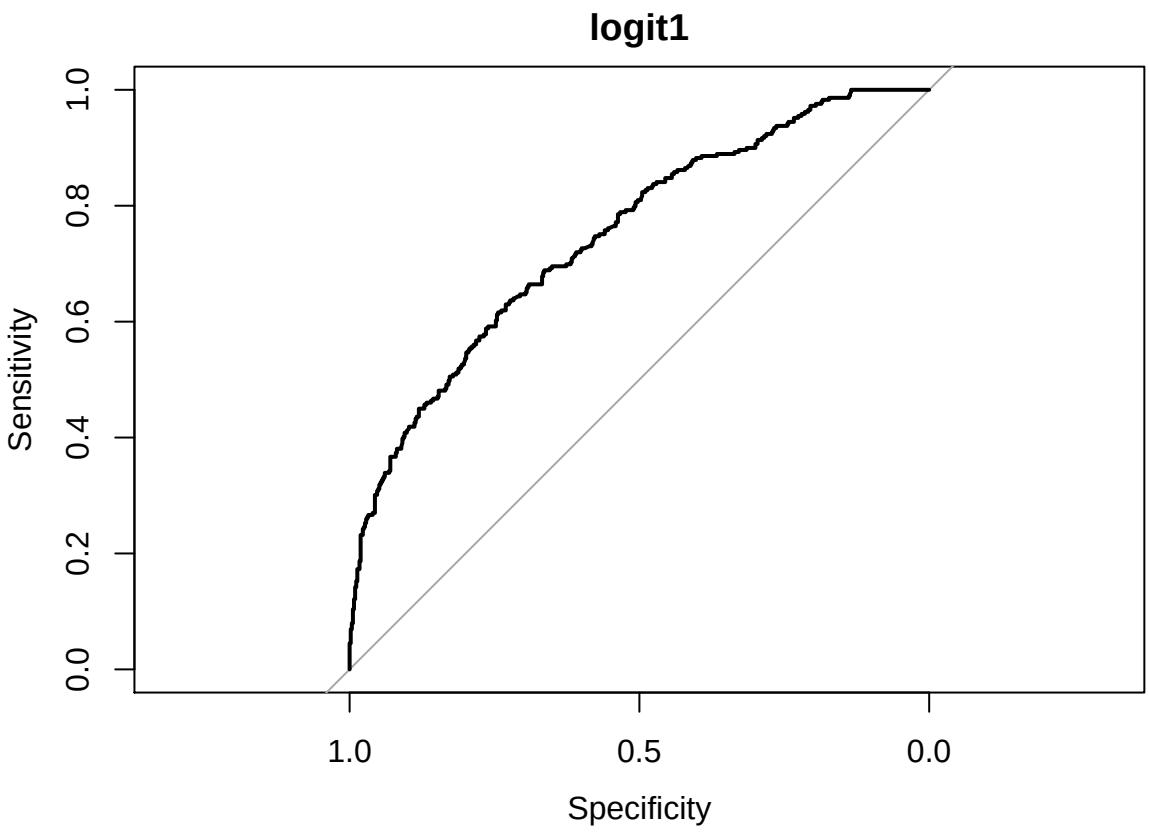
```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

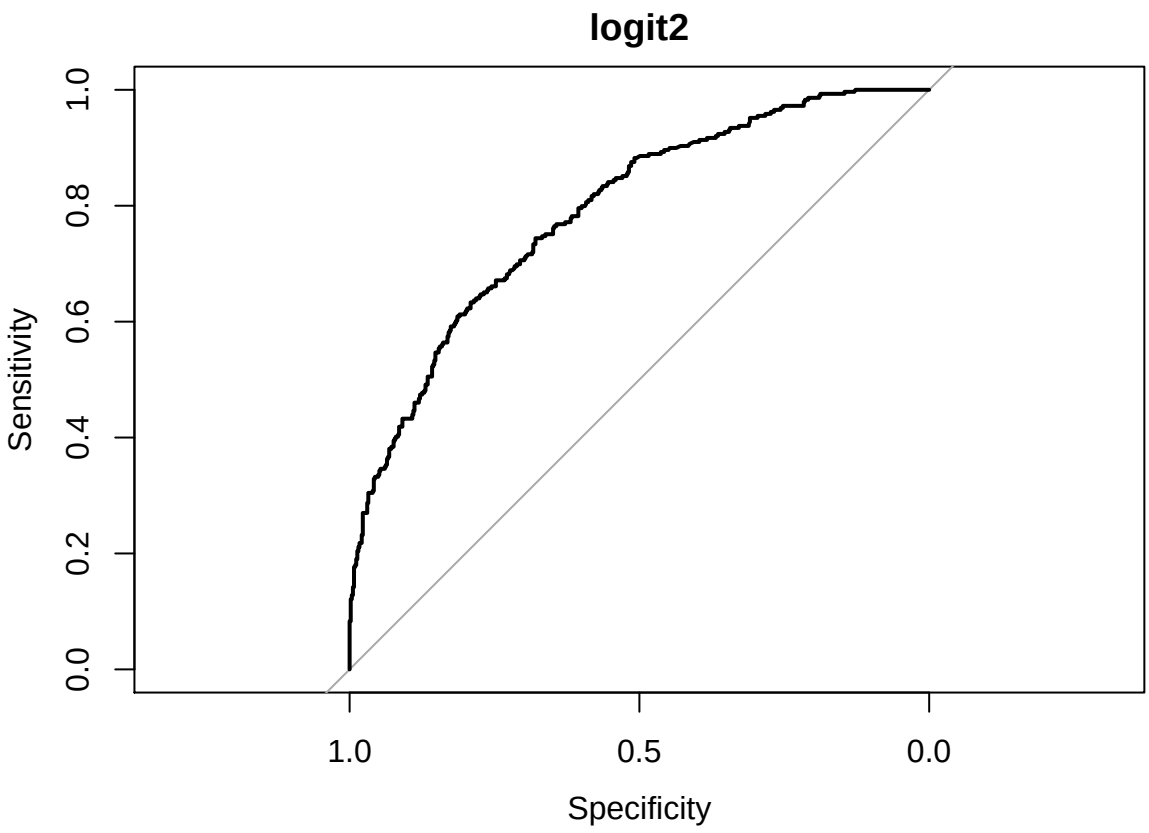
```
mod9 <- gbm(as.character(REC4) ~ ., data = train, distribution = "bernoulli",
            n.trees = 5000, shrinkage = 0.001, interaction.depth = 2, train.fraction = 0.5)
mod9
```

```
## gbm(formula = as.character(REC4) ~ ., distribution = "bernoulli",
##     data = train, n.trees = 5000, interaction.depth = 2, shrinkage = 0.001,
##     train.fraction = 0.5)
## A gradient boosted model with bernoulli loss function.
## 5000 iterations were performed.
## The best test-set iteration was 4994.
## There were 7 predictors of which 7 had non-zero influence.
```

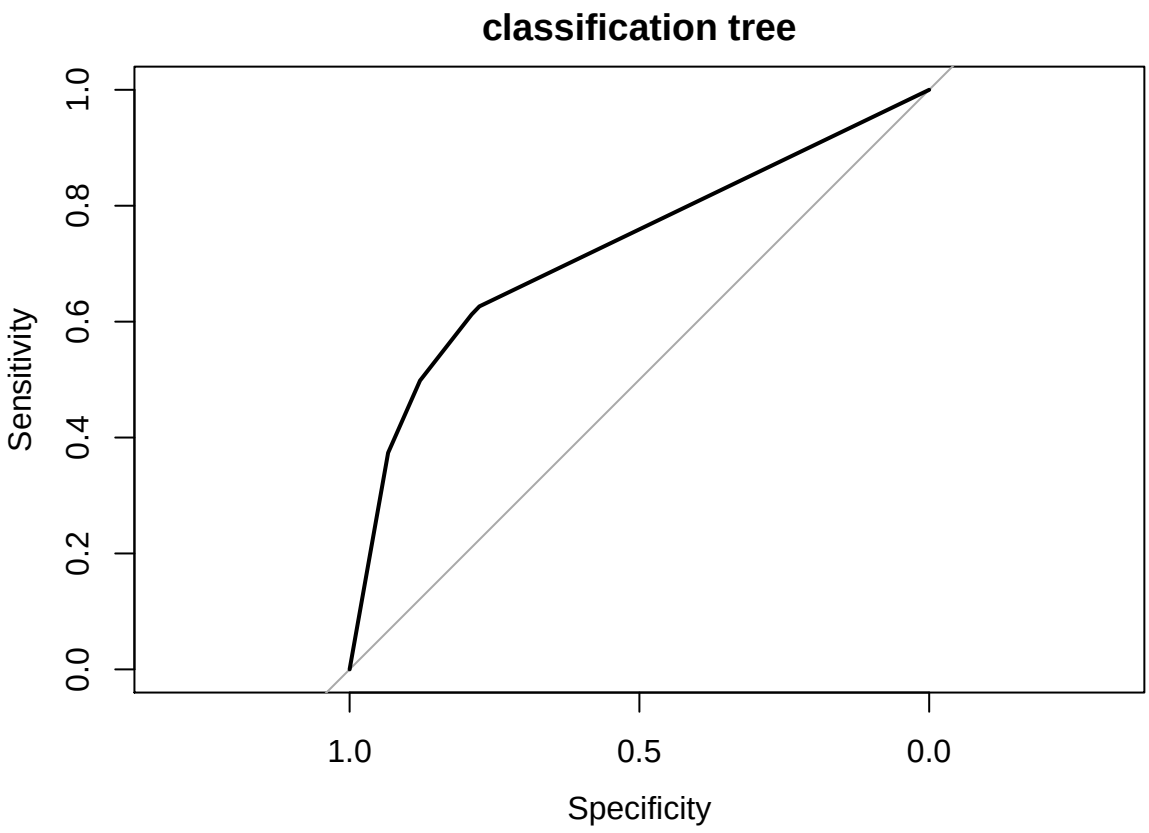
```
#modelling phase is over, now bring in the test data.

#final verdict on test data

#first get predicted probabilities on test set
p1 <- predict(mod1, newdata = test, type = "response")
p2 <- predict(mod2, newdata = test, type = "response")
p3 <- predict(mod3, newdata = test, type = "prob")[,2]
p5 <- predict(mod5, newdata = test, type = "prob")[,2]
p6 <- predict(mod6, newdata = test, type = "raw")[,2]
p7 <- predict(mod7, newdata = test)$posterior[,2]
#obtain 'probabilities' for k-nn
mod8 <- knn(train = data.matrix(train[, vars]), test = data.matrix(test[, vars]), cl = train$REC4, k = 
p8 <- 1 - attributes(mod8)$prob
p9 <- predict(mod9, newdata = test, type = "response", n.trees = 5000)
#compute accuracies on test set
acc1t <- accuracy(test$REC4, ifelse(p1>=0.5,1,0))
acc2t <- accuracy(test$REC4, ifelse(p2>=0.5,1,0))
acc3t <- accuracy(test$REC4, ifelse(p3>=0.5,1,0))
acc5t <- accuracy(test$REC4, ifelse(p5>=0.5,1,0))
acc6t <- accuracy(test$REC4, ifelse(p6>=0.5,1,0))
acc7t <- accuracy(test$REC4, ifelse(p7>=0.5,1,0))
acc8t <- accuracy(test$REC4, ifelse(p8>=0.5,1,0))
acc9t <- accuracy(test$REC4, ifelse(p9>=0.5,1,0))
#compute AUC's on test set
library(pROC)
r1t <- roc(test$REC4, p1); plot(r1t, main = "logit1")
```



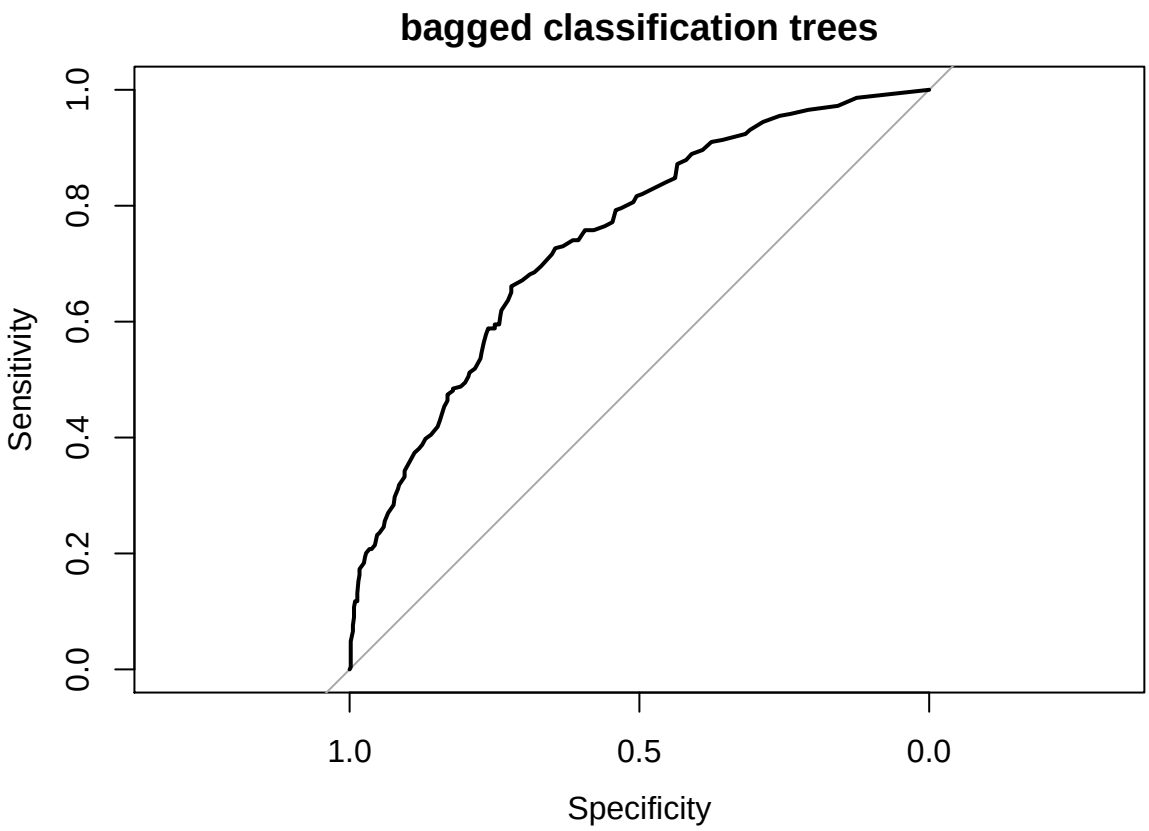

```
r2t <- roc(test$REC4, p2); plot(r2t, main = "logit2")
```



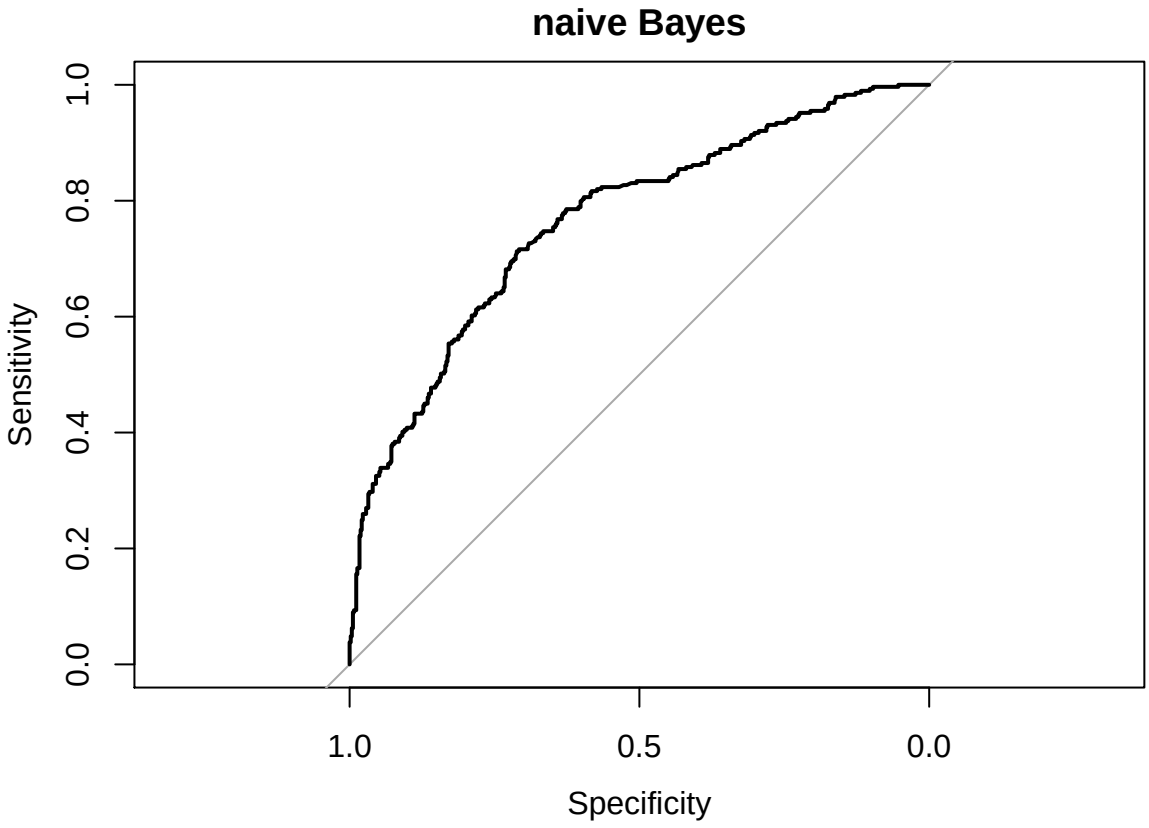

```
r3t <- roc(test$REC4, p3); plot(r3t, main = "classification tree")
```



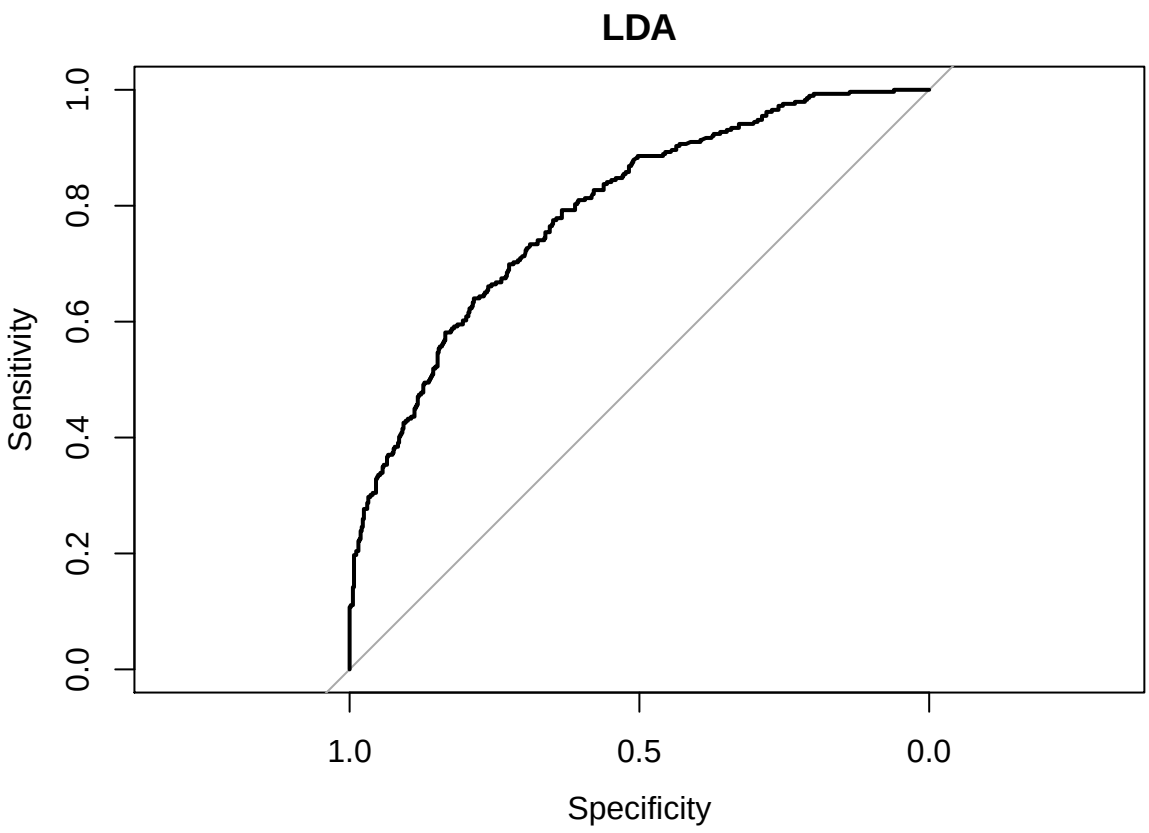

```
r5t <- roc(test$REC4, p5); plot(r5t, main = "bagged classification trees")
```



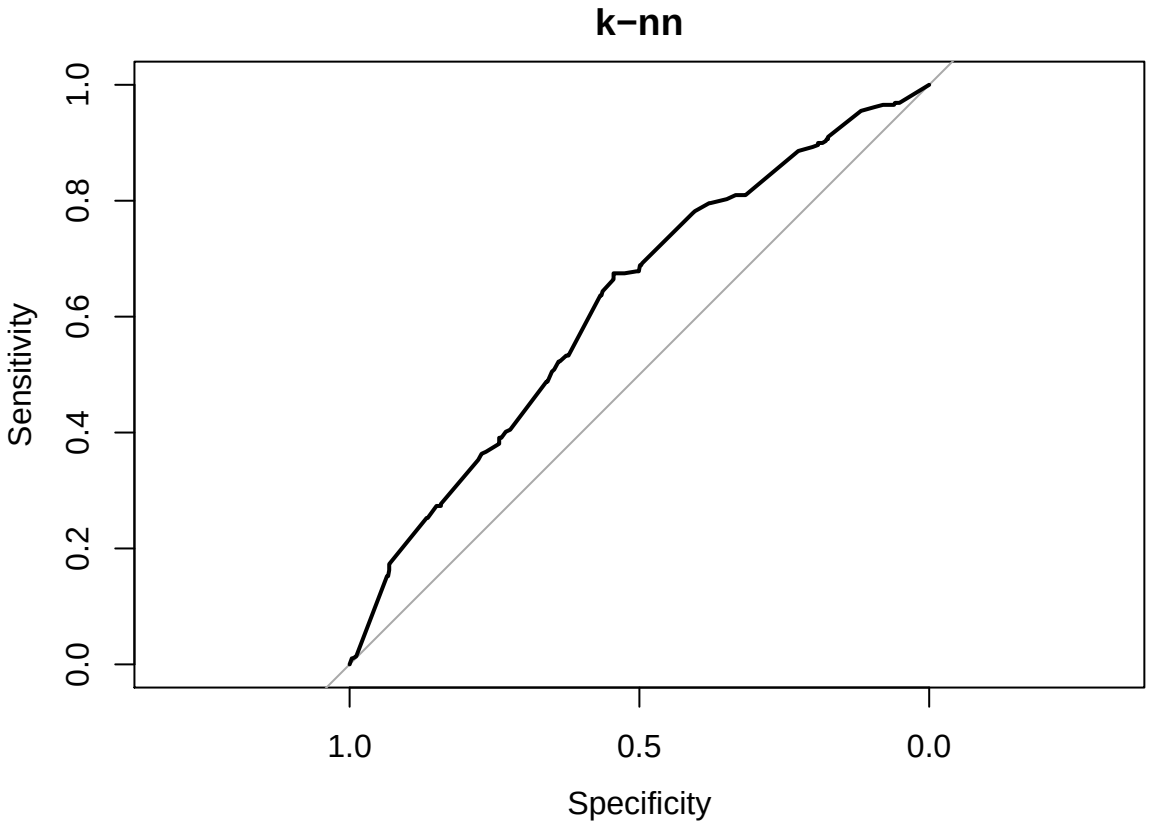

```
r6t <- roc(test$REC4, p6); plot(r6t, main = "naive Bayes")
```



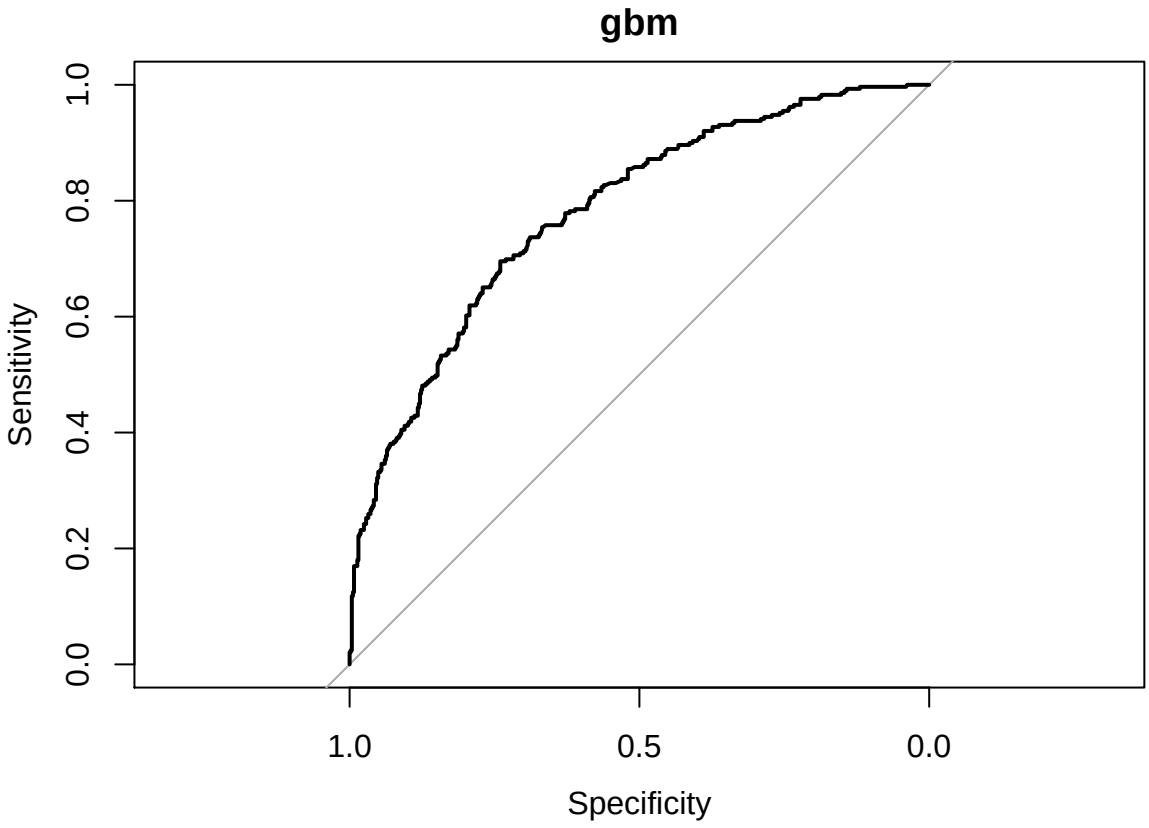

```
r7t <- roc(test$REC4, p7); plot(r7t, main = "LDA")
```



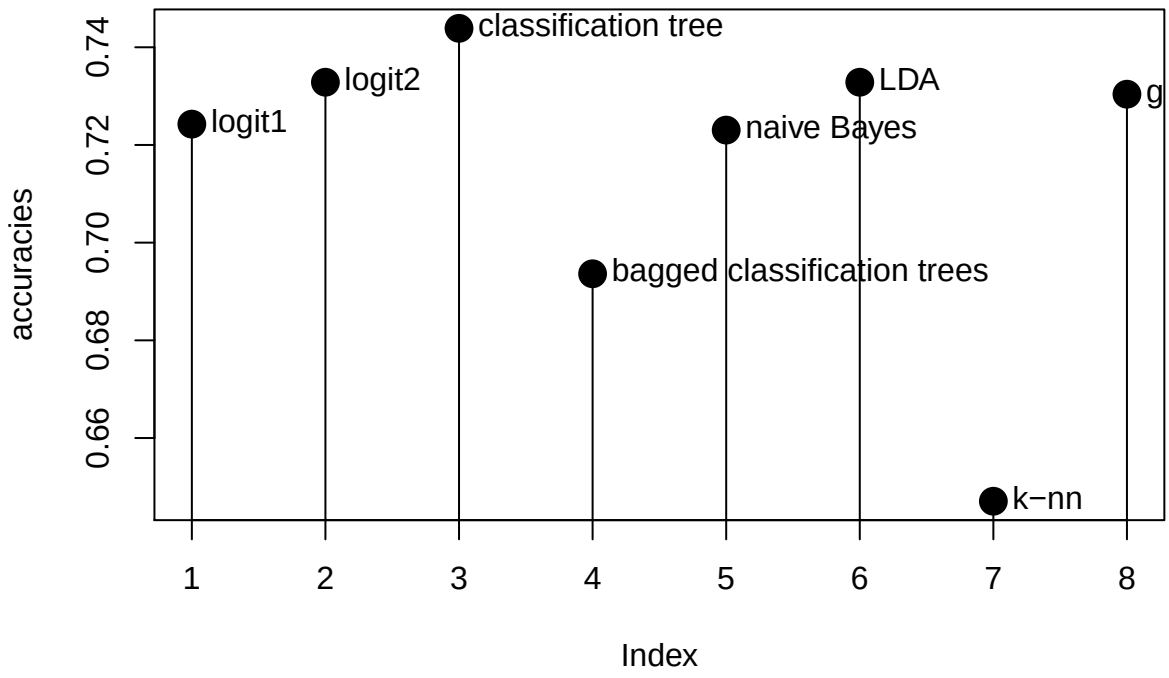

```
r8t <- roc(test$REC4, p8); plot(r8t, main = "k-nn")
```




```
r9t <- roc(test$REC4, p9); plot(r9t, main = "gbm")
```




```
accuracies <- c(acc1t,acc2t,acc3t,acc5t,acc6t,acc7t,acc8t,acc9t)
names(accuracies) <- c("logit1","logit2","classification tree","bagged classification trees","naive Bay
plot(accuracies, type = 'h'); text(accuracies, names(accuracies), pos = 4)
points(accuracies, pch = 16, cex = 2)
```



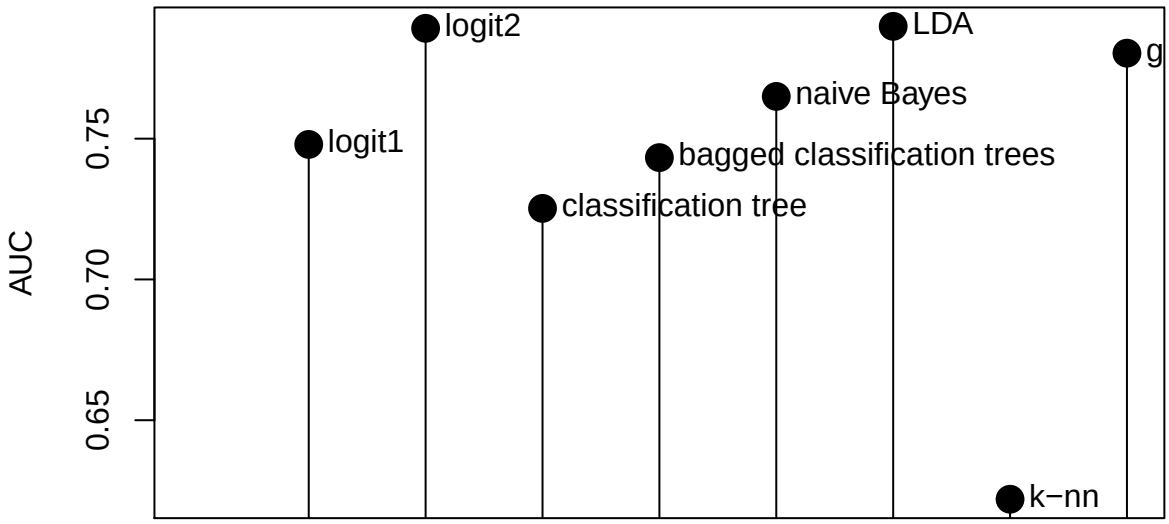

```
cat("highest ACC on test set:\n")
```

```
## highest ACC on test set:
```

```
accuracies[which.max(accuracies)]
```

```
## classification tree
##           0.7438725
```

```
aucs <- c(r1t$auc,r2t$auc,r3t$auc,r5t$auc,r6t$auc,r7t$auc,r8t$auc,r9t$auc)
names(aucs) <- c("logit1","logit2","classification tree","bagged classification trees","naive Bayes","LI
plot(aucs, type = 'h', xaxt='n', xlab = "", ylab = "AUC", xlim = c(0,8));
points(aucs, pch = 16, cex = 2)
text(aucs, names(aucs), pos = 4)
```




```
cat("highest AUC on test set:\n")
```

```
## highest AUC on test set:
```

```
aucs[which.max(aucs)]
```

```
##       LDA
## 0.7898892
```

LDA the highest AUC, and the classification tree has the highest accuracy on these data.

One criterion to quantify calibration in one number is the Brier score. It is defined as the mean squared difference between the observed class and the predicted probability. It is known from weather prediction. The lower the Brier score, the better calibrated the model is.

```
library(tidyverse)
brier <- function(obs, pr){
  if(is.factor(obs)) obs <- as.numeric(obs) - 1 # convert back to numeric
  mean((pr-obs)^2)
}
test <- mutate(test, REC4 = as.numeric(REC4)-1)
bs1t <- brier(test$REC4, p1)
bs2t <- brier(test$REC4, p2)
bs3t <- brier(test$REC4, p3)
bs5t <- brier(test$REC4, p5)
bs6t <- brier(test$REC4, p6)
bs7t <- brier(test$REC4, p7)
bs8t <- brier(test$REC4, p8)
bs9t <- brier(test$REC4, p9)
```

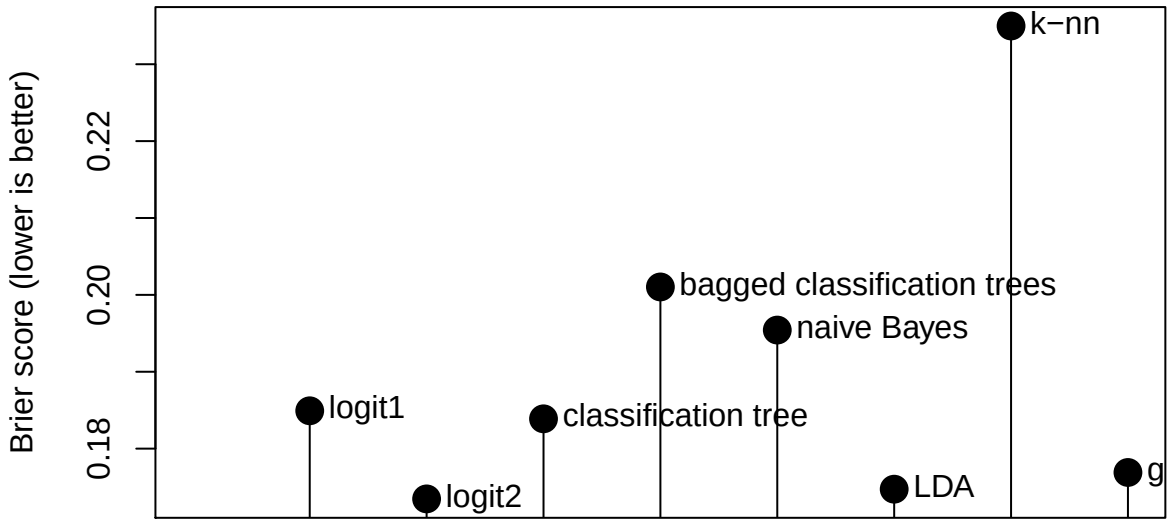
```
bss <- c(bs1t,bs2t,bs3t,bs5t,bs6t,bs7t,bs8t,bs9t)
names(bss) <- c("logit1","logit2","classification tree","bagged classification trees","naive Bayes","LDA
cat("lowest brier score on test set:\n")
```

```
## lowest brier score on test set:
```

```
bss[which.min(bss)]
```

```
##   logit2
## 0.173461
```

```
plot(bss, type = "h", xaxt='n', xlab = "", ylab = "Brier score (lower is better)", xlim = c(0,8))
text(bss, names(bss), pos = 4)
points(bss, pch = 16, cex = 2)
```



Conclusion: concerning the calibration error, overall the logit2 generalizes the best, and k-nn generalizes worst.

Up until now, we have concentrated on just 3 criteria for model adequacy.

With the command `confusionMatrix`, a whole range of criteria can be calculated, all based on the frequencies in a confusion matrix. The following figure lists them all with their formula's.

	Reference	
Predicted	Event	No Event
Event	A	B
No Event	C	D

The formulas used here are:

$$
\begin{aligned}
Sensitivity &= \frac{A}{A+C} \\
Specificity &= \frac{D}{B+D} \\
Prevalence &= \frac{A+C}{A+B+C+D} \\
PPV &= \frac{sensitivity \times prevalence}{((sensitivity \times prevalence) + ((1-specificity) \times (1-prevalence))} \\
NPV &= \frac{specificity \times (1-prevalence)}{((1-sensitivity) \times prevalence) + ((specificity) \times (1-prevalence))} \\
Detection\ Rate &= \frac{A}{A+B+C+D} \\
Detection\ Prevalence &= \frac{A+B}{A+B+C+D} \\
Balanced\ Accuracy &= (sensitivity + specificity)/2
\end{aligned}
$$

$$
\begin{aligned}
Precision &= \frac{A}{A+B} \\
Recall &= \frac{A}{A+C} \\
F1 &= \frac{(1+\beta^2) \times precision \times recall}{(\beta^2 \times precision) + recall}
\end{aligned}
$$

Figure 1: List of statistics for prediction models

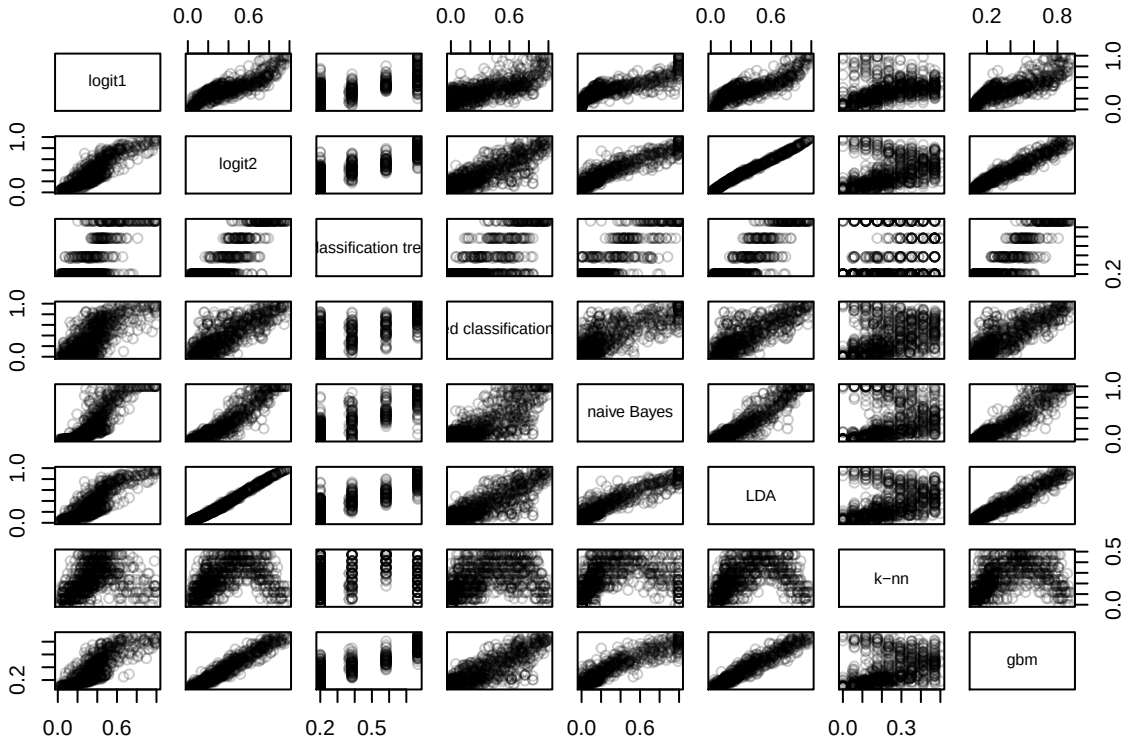
caret::confusionMatrix(data = ifelse(p9 >= .5, 1, 0), reference = test$REC4)

“‘

similarities of individual predictions

On an aggregate level, thing may not seem to vary a lot. How similar are the individual predictions generated by the models? First, we make a scatterplot matrix with `pairs`. Then, we use the unsupervised learning techniques cluster analysis and PCA (these will be explained in a later lecture) to visualize similarities

```
pmat <- data.matrix(t(cbind(p1,p2,p3,p5,p6,p7,p8,p9)))
row.names(pmat) <- c("logit1","logit2","classification tree","bagged classification trees","naive Bayes
pairs(t(pmat),col=rgb(0,0,0,alpha=0.2))
```



Can you reason why some of them are more alike?