Comparing ROC Packages

June 19, 2020

1 Packages

[16]:
```
library(mdsr)
library(rpart)
library(nnet)
library(e1071)
library(randomForest)
library(ROCR)
library(plotROC)
library(ROCit)
```

2 Init

[2]:
```
census <- read.csv(
  "http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data",
  header = FALSE)
names(census) <- c("age", "workclass", "fnlwgt", "education",
                   "education.num", "marital.status", "occupation",
                   "relationship", "race", "sex", "capital.gain",
                   "capital.loss", "hours.per.week", "native.country",
                   "income")
set.seed(364)
n <- nrow(census)
test_idx <- sample.int(n, size = round(0.2 * n))
train <- census[-test_idx,]
test <- census[test_idx,]

form <- as.formula("income ~ age + workclass + education + marital.status +
  occupation + relationship + race + sex + capital.gain + capital.loss +
  hours.per.week")

cut_seq <- seq(0, 1, 0.1)
cut_seq_2 <- seq(0, 1, 0.25)
```

3 Models

3.1 Decision Tree

[3]:
```
mod_tree <- rpart(form, data = train)
income_tree_probs <- mod_tree %>% predict(newdata = test, type = "prob")
df_tree <- data.frame(predictions = income_tree_probs[, ' >50K'],
                      labels = ifelse(test$income == ' >50K', 1, 0))
```

3.2 Naive Bayes

[4]:
```
mod_nb <- naiveBayes(form, data = train)
income_nb_probs <- mod_nb %>% predict(newdata = test, type = "raw")
df_nb <- data.frame(predictions = income_nb_probs[, ' >50K'],
                    labels = ifelse(test$income == ' >50K', 1, 0))
```

3.3 Neural Network

[5]:
```
mod_nn <- nnet(form, data = train, size = 5)
income_nn_probs <- mod_nn %>% predict(newdata = test, type = "raw")
df_nn <- data.frame(predictions = income_nn_probs,
                    labels = ifelse(test$income == ' >50K', 1, 0))
```

```
# weights:  296
initial  value 17863.316757
iter  10 value 13160.102775
iter  20 value 11490.083360
iter  30 value 10492.265833
iter  40 value 9190.590748
iter  50 value 8868.952933
iter  60 value 8669.234503
iter  70 value 8582.637394
iter  80 value 8516.139198
iter  90 value 8442.790655
iter 100 value 8408.961720
final  value 8408.961720
stopped after 100 iterations
```

3.4 Random Forest

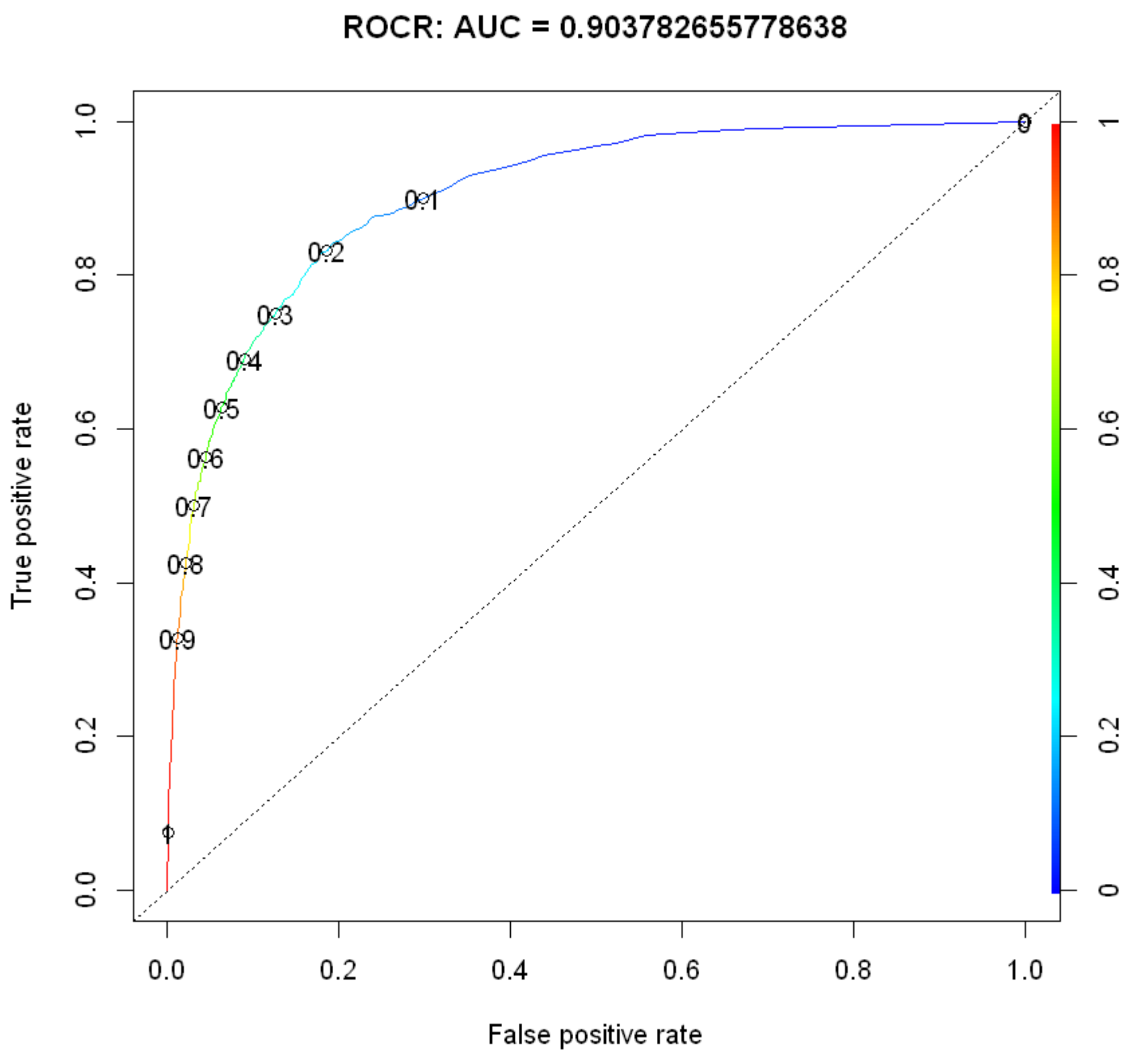
[6]:
```
mod_forest <- randomForest(form, data = train, ntree = 201, mtry = 3)
income_forest_probs <- mod_forest %>% predict(newdata = test, type = "prob")
df_forest <- data.frame(predictions = income_forest_probs[, ' >50K'],
                        labels = ifelse(test$income == ' >50K', 1, 0))
```

4 Comparing Packages

4.1 ROCR

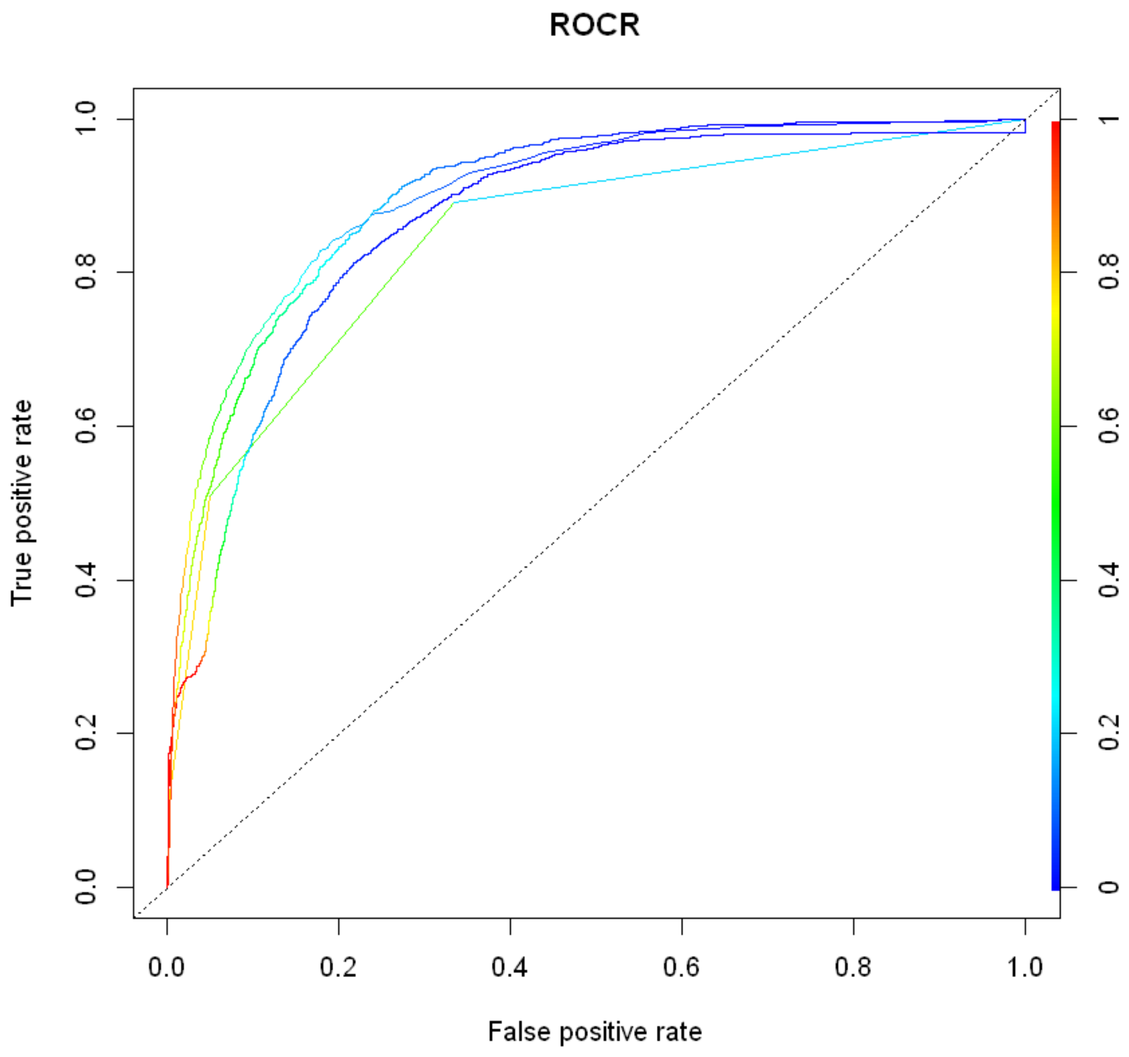
ROCR allows you to `colorize` a ROC curve according to threshold values. On top of that, you can also label a certain set of threshold values on the curve, giving you two different methods of identifying cutoffs.

[7]:
```r
rocr_pred_forest <- prediction(df_forest$predictions, df_forest$labels)
rocr_perf_forest <- performance(rocr_pred_forest, 'tpr', 'fpr')
rocr_perf_forest %>% plot(colorize = TRUE,
                          print.cutoffs.at = cut_seq)
rocr_auc_forest <- performance(rocr_pred_forest, 'auc')@y.values[[1]]
title(paste("ROCR: AUC =", rocr_auc_forest))
abline(0, 1, lty = 3)
```



Overlaying curves is actually fairly simple with ROCR. All you have to do is plot an initial curve, then every subsequent curve you plot must specify add = TRUE. This prevents the previous graph from going away. This, combined with colorize, provides an interesting way of comparing ROC curves and how their thresholds are distributed differently along them. The drawback of this approach is that there isn't an easy way to label or identify each line, other than remembering the shape of each one.

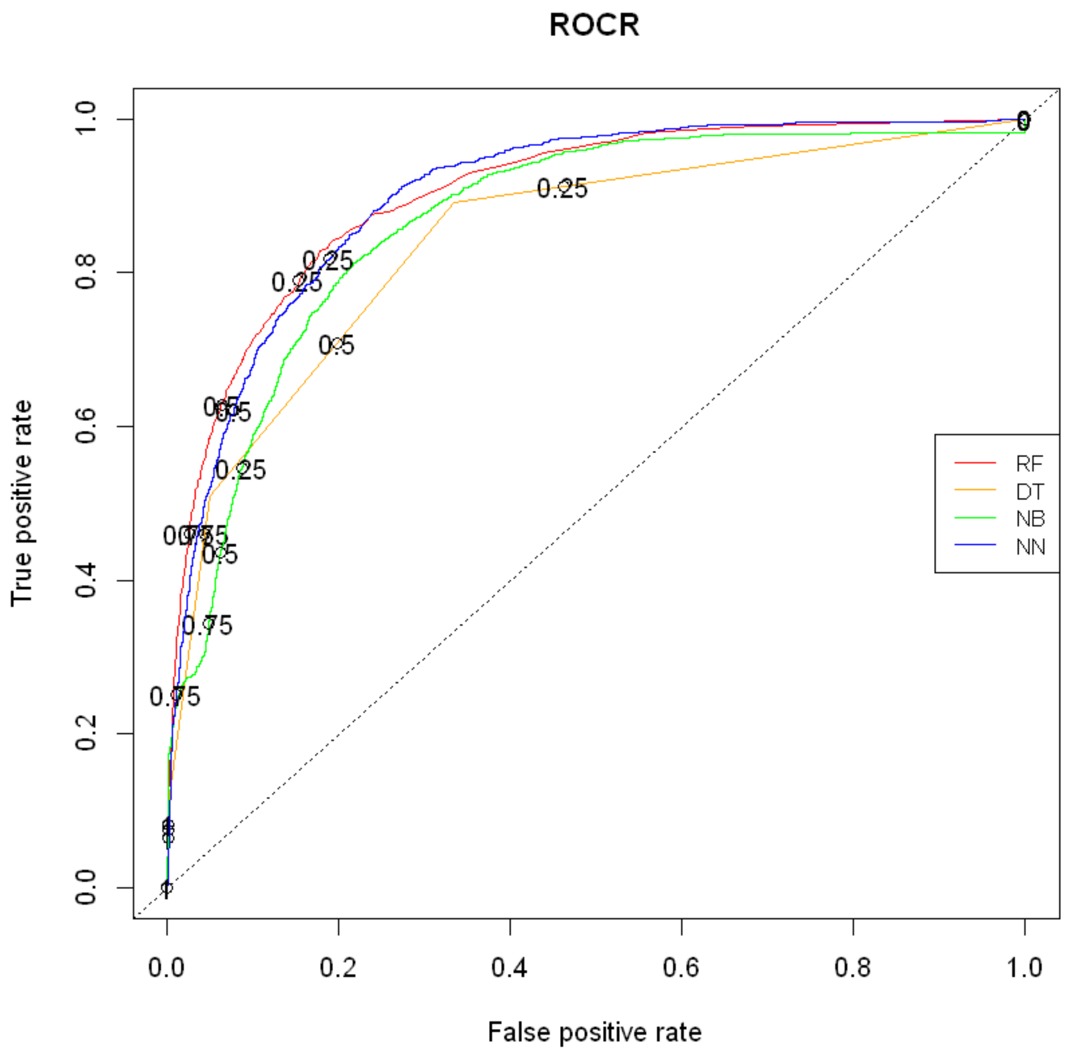
[8]:
```r
rocr_pred_tree <- prediction(df_tree$predictions, df_tree$labels)
rocr_pred_nb <- prediction(df_nb$predictions, df_nb$labels)
rocr_pred_nn <- prediction(df_nn$predictions, df_nn$labels)

rocr_perf_tree <- performance(rocr_pred_tree, 'tpr', 'fpr')
rocr_perf_nb <- performance(rocr_pred_nb, 'tpr', 'fpr')
rocr_perf_nn <- performance(rocr_pred_nn, 'tpr', 'fpr')

rocr_perf_forest %>% plot(colorize = TRUE)
rocr_perf_tree %>% plot(colorize = TRUE, add = TRUE)
rocr_perf_nb %>% plot(colorize = TRUE, add = TRUE)
rocr_perf_nn %>% plot(colorize = TRUE, add = TRUE)
abline(0, 1, lty = 3)
title("ROCR")
```



We can emulate the same idea as before but with identifiers for each curve. We lose the gradients in exchange for color identifiers and substitute in labels to identify threshold locations. The labels are all black, so it is easy to confuse nearby labels.

[9]:
```
rocr_perf_forest %>% plot(print.cutoffs.at = cut_seq_2,
                          col = "red")
rocr_perf_tree %>% plot(print.cutoffs.at = cut_seq_2,
                        col = "orange", add = TRUE)
rocr_perf_nb %>% plot(print.cutoffs.at = cut_seq_2,
                      col = "green", add = TRUE)
rocr_perf_nn %>% plot(print.cutoffs.at = cut_seq_2,
                      col = "blue", add = TRUE)
abline(0, 1, lty = 3)
```

```r
legend("right", legend = c("RF", "DT", "NB", "NN"),
       col = c("red", "orange", "green", "blue"),
       lty = 1, cex = 0.8)
title("ROCR")
```

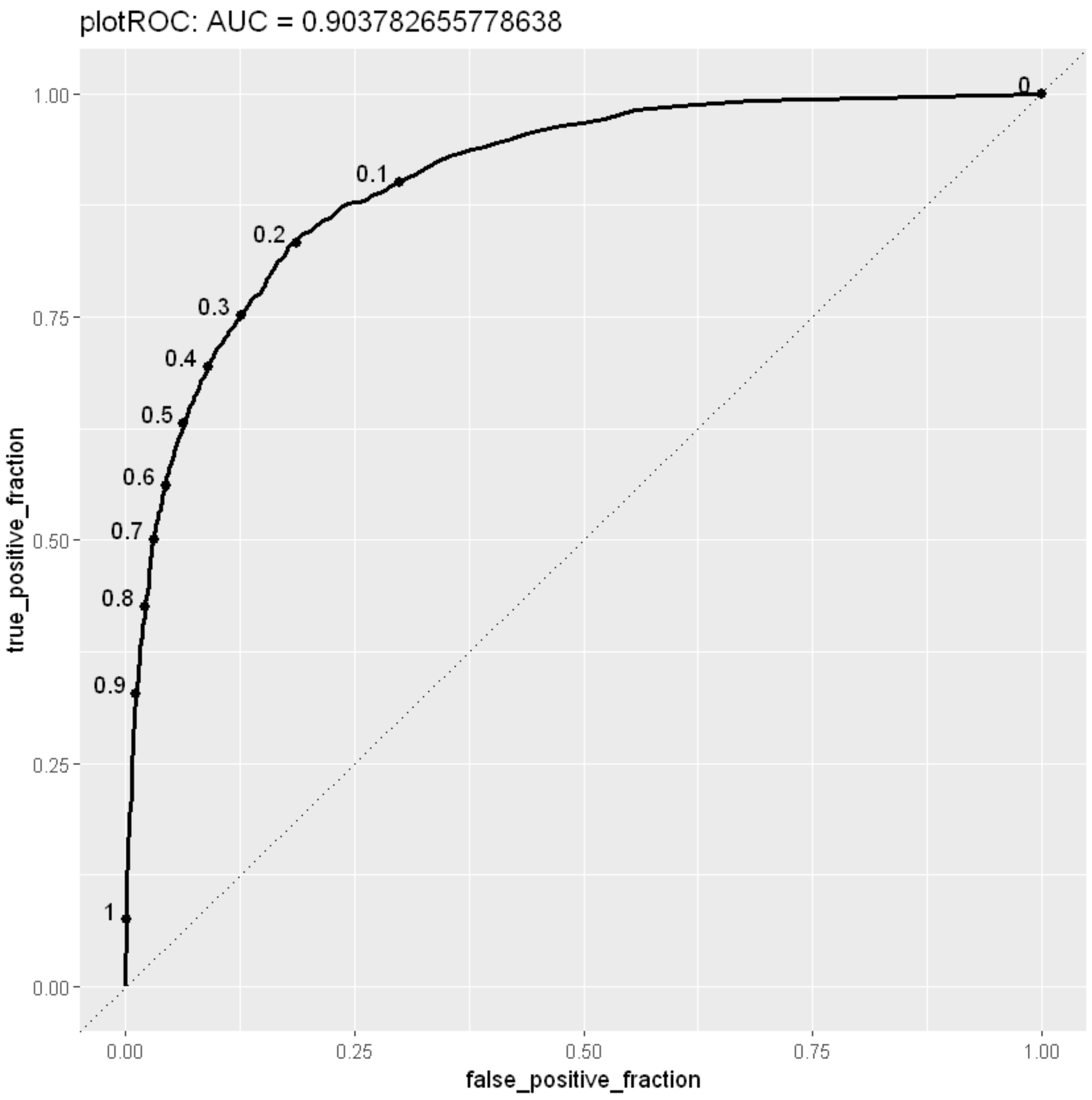


4.2 plotROC

As opposed to `ROCR`, which does not integrate with `ggplot` by default, `plotROC` does. This has its tradeoffs. In order to calculate AUC, you must first have a `ggplot` object with a `geom_roc` layer. This is because there is no `plotROC` object. It also means that there is no easy way of graphing a ROC curve with a color gradient, since the threshold values are calculated behind the scenes and never returned. However, the labeling is a bit nicer with `plotROC`, as they are offset a bit from the curve. Also, the AUC calculations in this package are sometimes different than those in `ROCR` and

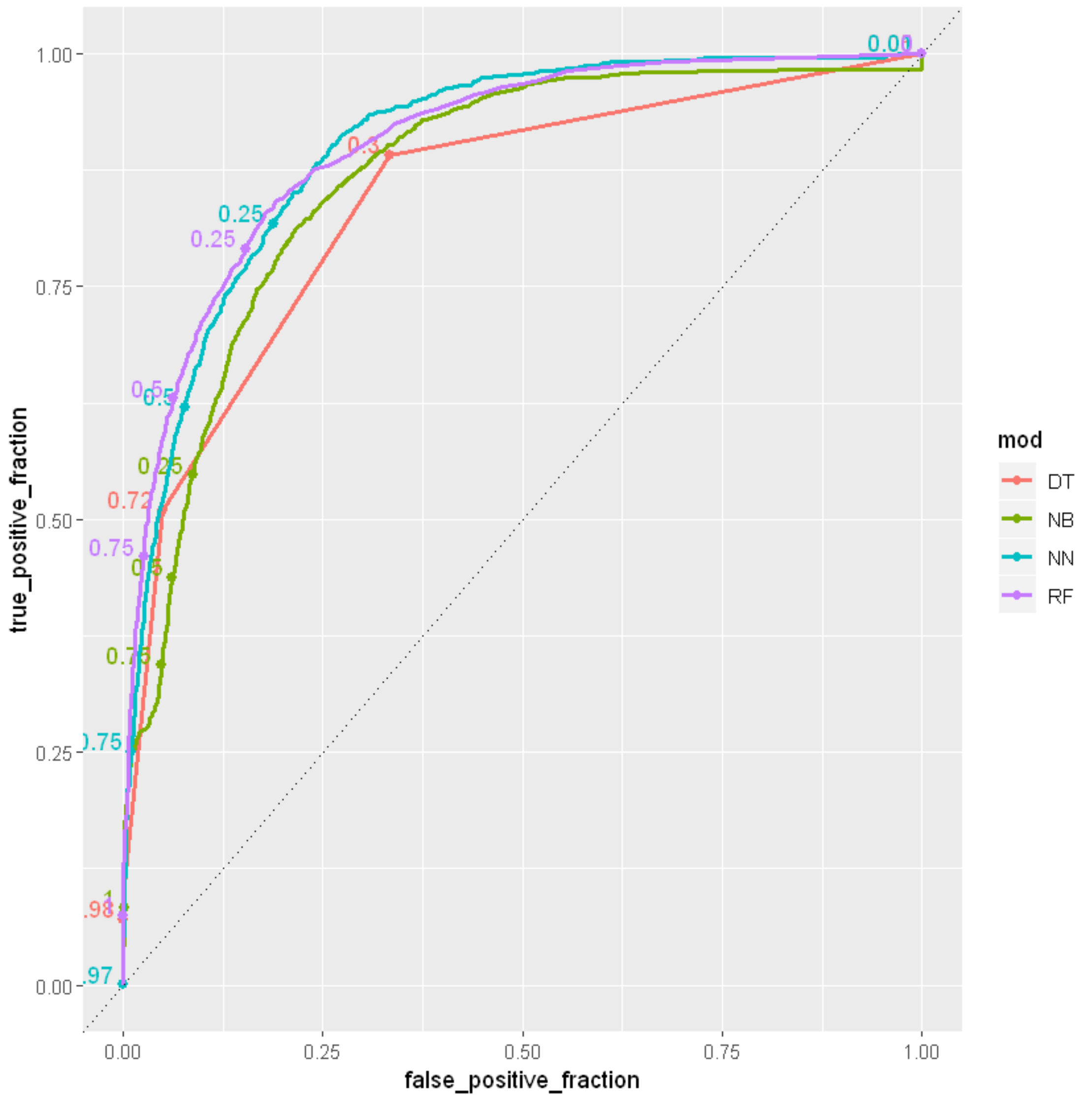
ROCit, sometimes starting as early as the fourth or fifth decimal place.

[10]:
```
plotroc_plot <- df_forest %>% ggplot(aes(m = predictions, d = labels)) +
  geom_roc(cutoffs.at = cut_seq) +
  geom_abline(slope = 1, intercept = 0, lty = 3)
plotroc_auc <- calc_auc(plotroc_plot)[['AUC']]
plotroc_ggplot <- plotroc_plot +
  ggtitle(paste("plotROC: AUC =", plotroc_auc))
plotroc_ggplot
```



Overlaying multiple curves is fairly simple. All you have to do is combine them all into one data frame and give them an identifier by model, then separate them by color on the plot. The labels are colored according to which model they belong to, making this graph easier to interpret and nicer to look at than the similar ROCR overlay.

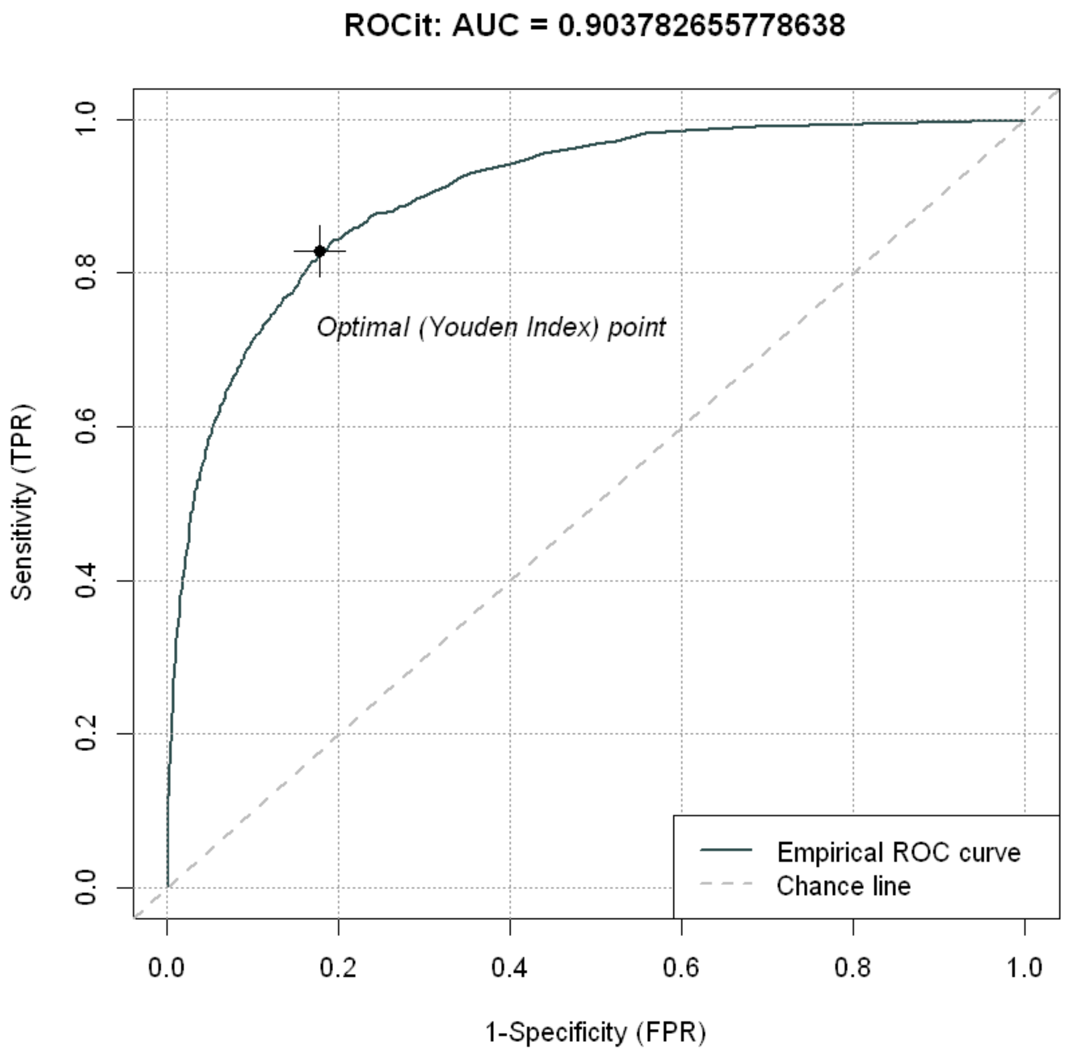
```
[11]: df_all <- rbind(mutate(df_forest, mod = "RF"),
                     mutate(df_tree, mod = "DT"),
                     mutate(df_nb, mod = "NB"),
                     mutate(df_nn, mod = "NN"))
      df_all %>% ggplot(aes(m = predictions, d = labels, color = mod)) +
        geom_roc(cutoffs.at = cut_seq_2, labelround = 2) +
        geom_abline(slope = 1, intercept = 0, lty = 3)
```



4.3 ROCit

`ROCit` is interesting because, by default, it plots an "Optimal (Youden Index) point" for each curve, which is the point on the curve that is the farthest from the Chance line vertically. This gives a useful but perhaps loose benchmark for optimality. The `rocit_plot` also silently has all the fpr, tpr, and cutoff values, but does not display the cutoff values at all.

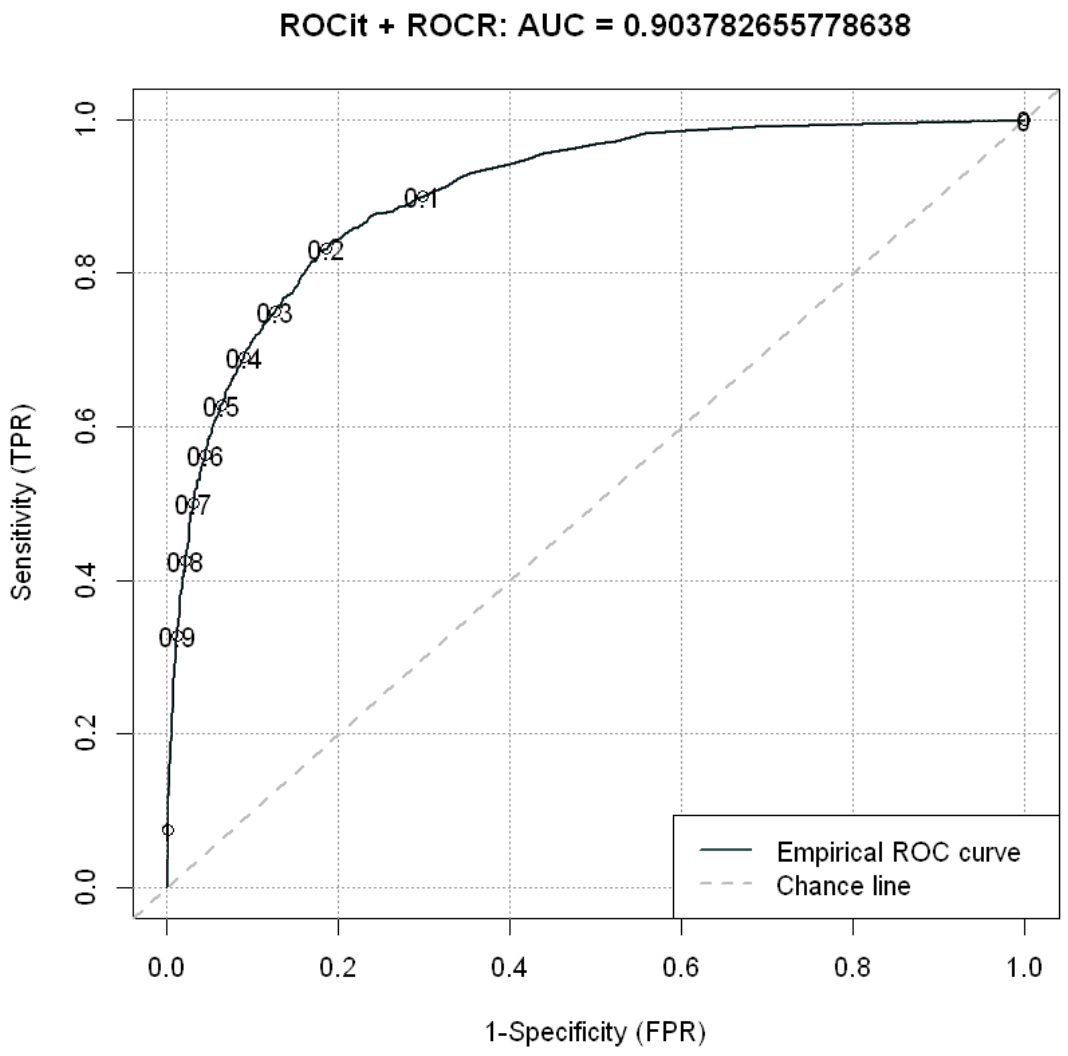
[12]:
```
rocit_obj <- rocit(df_forest$predictions, df_forest$labels)
rocit_plot <- rocit_obj %>% plot()
rocit_auc <- rocit_obj[['AUC']]
title(paste("ROCit: AUC =", rocit_auc))
```



4.4 ROCit and ROCR Combined

If you want the nice look of a `ROCit` plot with the labelling provided by `ROCR`, you can combine the two in the same way we combined the multiple `ROCR` plots from before, by specifying `add = TRUE`.

[15]:
```
rocit_obj %>% plot(YIndex = FALSE)
rocr_perf_forest %>% plot(print.cutoffs.at = cut_seq, add = TRUE)
title(paste("ROCit + ROCR: AUC =", rocit_auc))
```


ROCit + ROCR: AUC = 0.903782655778638
Sensitivity (TPR)
1-Specificity (FPR)
Empirical ROC curve
Chance line