plotROC Package Investigation

June 10, 2020

1 Packages

[2]:
```
library(mdsr)
library(tidyverse)
library(rpart)
library(randomForest)
library(e1071)
library(class)
library(nnet)
library(ROCR)
library(plotROC)
library(plotly)
library(gridExtra)
```

2 Init

[3]:
```
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")
```

We will need to convert our income labels into binary to use `plotROC` comfortably.

[4]:
```
test_income_binary <- ifelse(test$income == ' <=50K', 0, 1)
test_income_binary %>% head()
```

1. 0 2. 1 3. 0 4. 0 5. 0 6. 0

3 Models

3.1 K-Nearest Neighbors

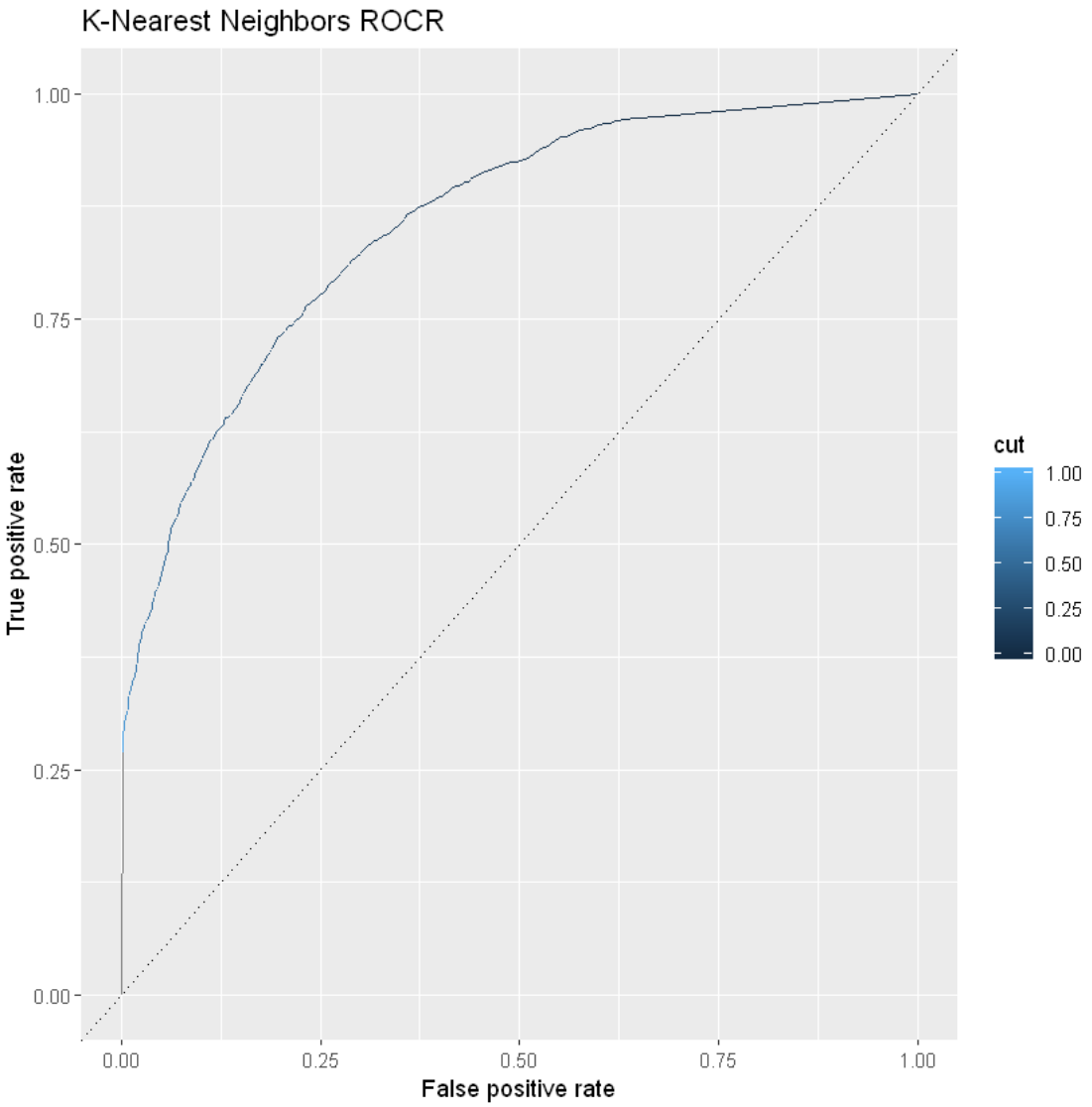
We will need to calculate the probabilities for each model, not only for `ROCR`, but for `plotROC` as well.

[5]:
```
train_q <- train %>%
  select(age, education.num, capital.gain, capital.loss, hours.per.week)
test_q <- test %>%
  select(age, education.num, capital.gain, capital.loss, hours.per.week)
income_knn <- knn(train_q, test = test_q, cl = train$income, k = 10, prob =␣
↪TRUE)
income_knn_probs <- ifelse(income_knn == ' >50K', attr(income_knn, 'prob'), 1 -␣
↪attr(income_knn, 'prob'))
income_knn_probs %>% head()
```

1. 0 2. 0.3 3. 0 4. 0 5. 0 6. 0

Below is the `ROCR` ROC curve as normal.

[6]:
```
pred_knn <- ROCR::prediction(income_knn_probs, test$income)
perf_knn <- ROCR::performance(pred_knn, 'tpr', 'fpr')
perf_knn_df <- data.frame(perf_knn@x.values, perf_knn@y.values, perf_knn@alpha.
↪values)
names(perf_knn_df) <- c("fpr", "tpr", "cut")

rocr_knn <- perf_knn_df %>% ggplot(aes(x = fpr, y = tpr, color = cut)) +
  geom_line() + geom_abline(intercept = 0, slope = 1, lty = 3) +
  ylab(perf_knn@y.name) + xlab(perf_knn@x.name) + ggtitle("K-Nearest Neighbors␣
↪ROCR")
rocr_knn
```



We can convert this `ROCR` implementation into `plotly` to become interactive. `plotly` currently has a bug where `geom_line` will not display if `color` is specified in the aesthetic. This is why `geom_point` is added.

[25]:
```
ggplotly(rocr_knn + geom_point())
```

```
HTML widgets cannot be represented in plain text (need html)
```

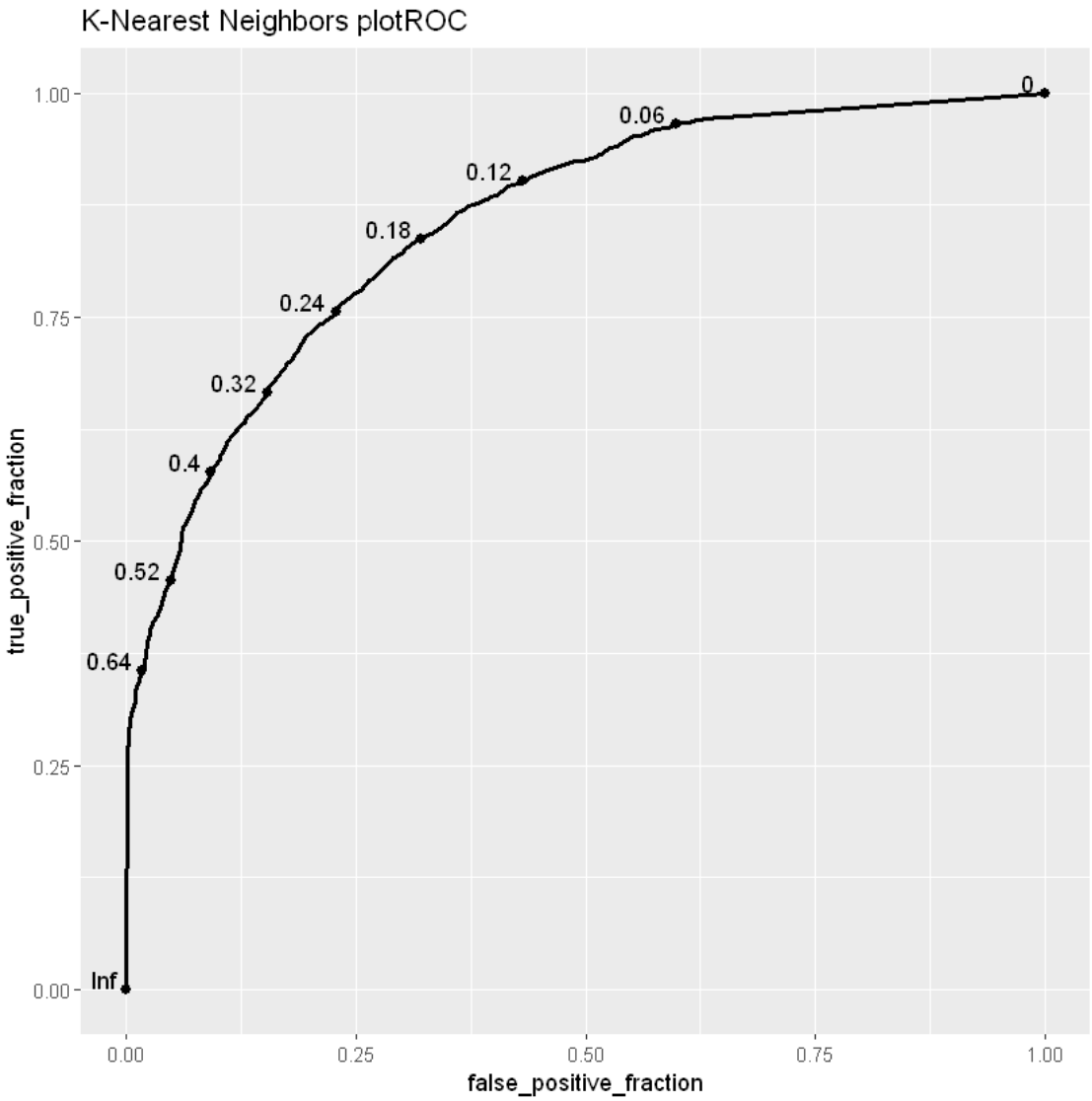
In order to use `plotROC`, you first need a data frame with a column for ground truth classifications (D) and a column for probabilities returned by your model (M). The D column should be in 0's and 1's, or else `plotROC` will make an educated guess as to which factor level will be mapped to 0 and which will be mapped to 1.

[9]:
```
plotroc_knn_df <- data.frame(test_income_binary, income_knn_probs)
names(plotroc_knn_df) <- c('D', 'M')
plotroc_knn_df %>% head()
```

D	M
0	0.0
1	0.3
0	0.0
0	0.0
0	0.0
0	0.0

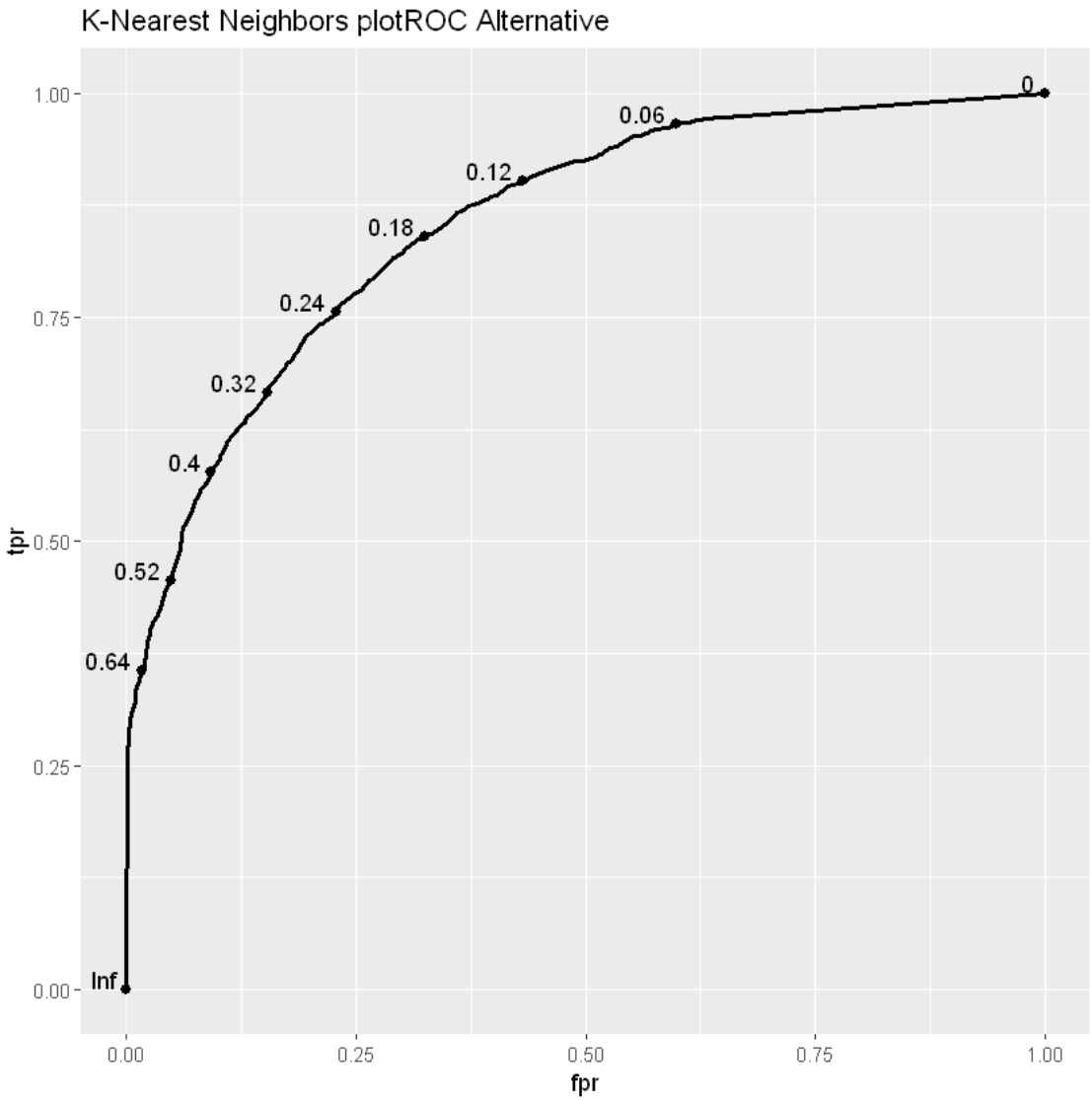
Once you have this data frame, you use `ggplot` with d equal to the D column and m equal to the M column in the initial aesthetic. Then you add a `geom_roc` label.

[10]:
```
plotroc_knn <- plotroc_knn_df %>% ggplot(aes(d = D, m = M)) +
  geom_roc(labelround = 2) + ggtitle("K-Nearest Neighbors plotROC")
plotroc_knn
```



Alternatively, you could simply use the fpr, tpr, and cut returned from the `ROCR::performance` function. Just set `x = fpr`, `y = tpr`, and `label = cut`. Then add a `geom_roc` layer with `stat = "identity"` inside.

[11]:
```
plotroc_knn_alt <- perf_knn_df %>% ggplot(aes(x = fpr, y = tpr, label = cut)) +
  geom_roc(stat = "identity", labelround = 2) + ggtitle("K-Nearest Neighbors␣
 ↪plotROC Alternative")
plotroc_knn_alt
```



The first `plotROC` ROC curve can be made into an interactive plot easily.

[12]:
```
plotroc_knn %>% plot_interactive_roc(labelround = 2)
```

However, the second curve cannot currently be made into an interactive plot.

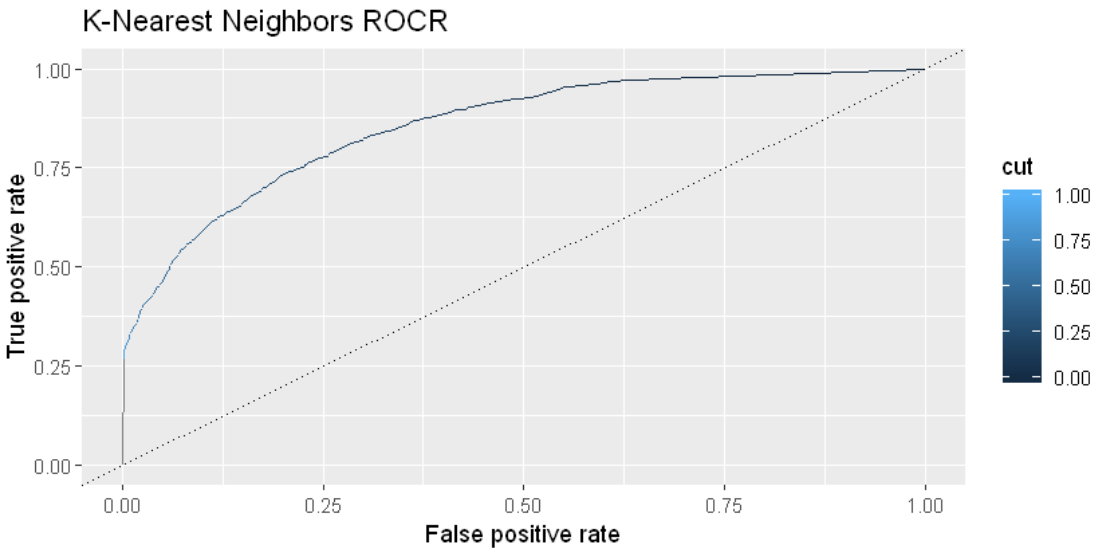
[13]:
```
plotroc_knn_alt %>% plot_interactive_roc()
```

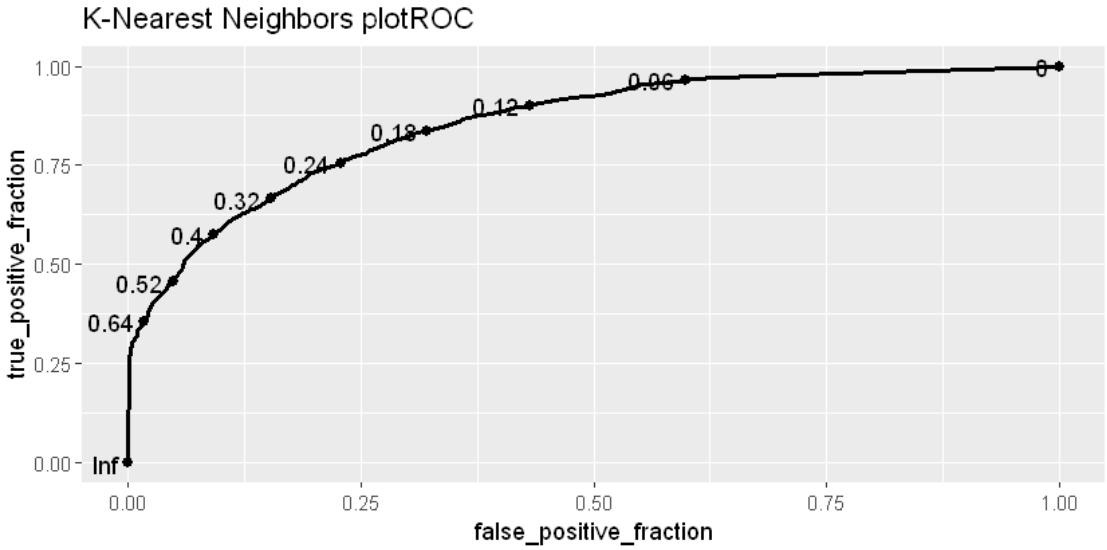
```
Error in `[.data.frame`(ggroc_p$data, , Mran.name): undefined columns␣
↪selected
Traceback:
```

1. plotroc_knn_alt %>% plot_interactive_roc()
2. withVisible(eval(quote(`_fseq`(`_lhs`)), env, env))
3. eval(quote(`_fseq`(`_lhs`)), env, env)
4. eval(quote(`_fseq`(`_lhs`)), env, env)
5. `_fseq`(`_lhs`)
6. freduce(value, `_function_list`)
7. withVisible(function_list[[k]](value))
8. function_list[[k]](value)
9. plot_interactive_roc(.)
10. export_interactive_roc(ggroc, …)
11. ggroc_p$data[, Mran.name]
12. `[.data.frame`(ggroc_p$data, , Mran.name)
13. stop("undefined columns selected")

Here is a look at the differences between the `ROCR` ROC curve and the `plotROC` ROC curve.

[14]:
```
grid.arrange(rocr_knn, plotroc_knn)
```

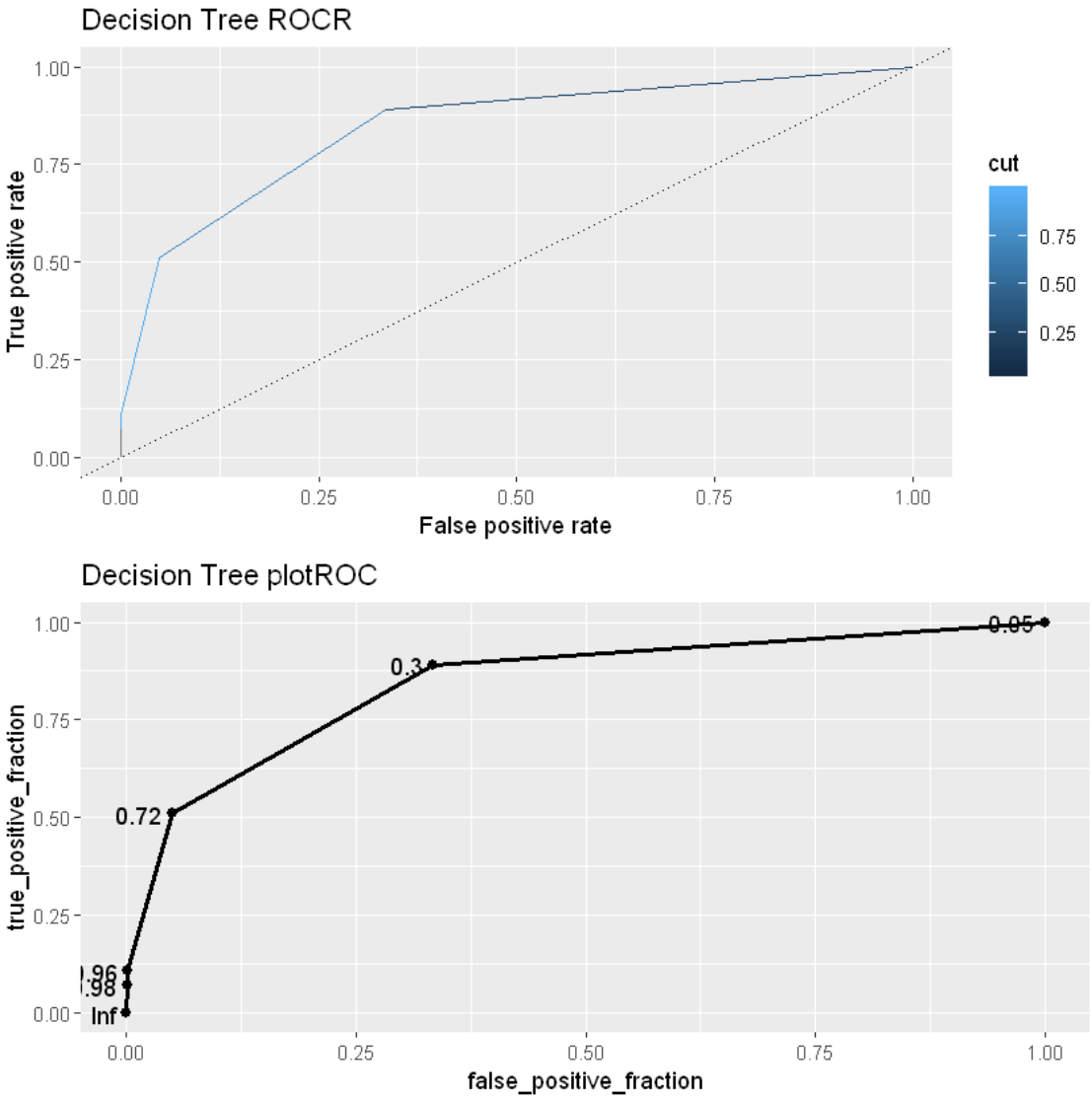




3.2 Decision Tree

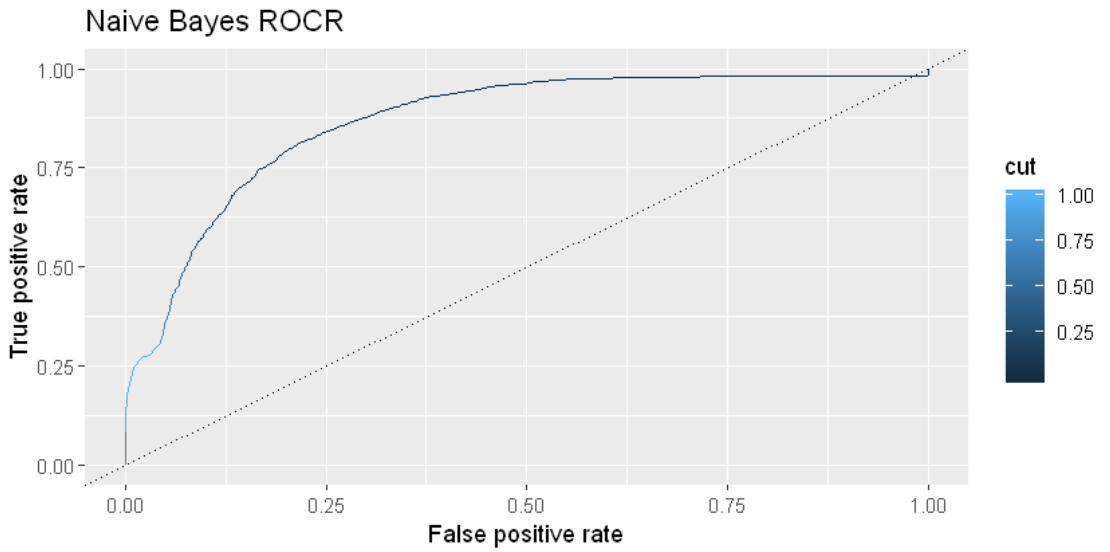
[15]:
```
mod_tree <- rpart(form, data = train)
income_tree_probs <- mod_tree %>%
  predict(newdata = test, type = "prob")
income_tree_probs <- income_tree_probs[, ' >50K']

pred_tree <- ROCR::prediction(income_tree_probs, test$income)
perf_tree <- ROCR::performance(pred_tree, 'tpr', 'fpr')
perf_tree_df <- data.frame(perf_tree@x.values, perf_tree@y.values,␣
 ↪perf_tree@alpha.values)
names(perf_tree_df) <- c("fpr", "tpr", "cut")
```

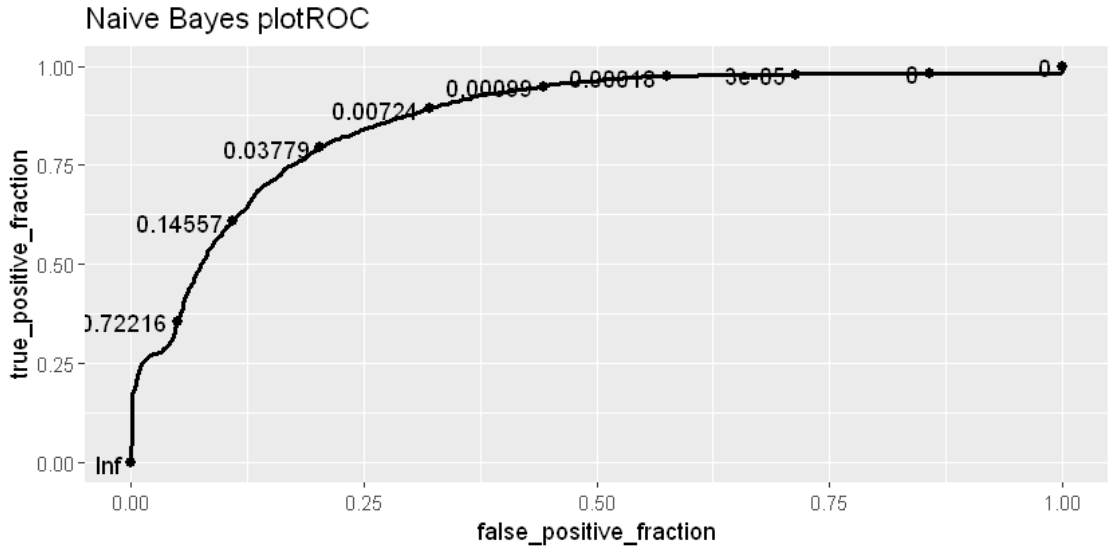
```
rocr_tree <- perf_tree_df %>% ggplot(aes(x = fpr, y = tpr, color = cut)) +
  geom_line() + geom_abline(intercept = 0, slope = 1, lty = 3) +
  ylab(perf_tree@y.name) + xlab(perf_tree@x.name) + ggtitle("Decision Tree␣
 ↪ROCR")

plotroc_tree_df <- data.frame(test_income_binary, income_tree_probs)
names(plotroc_tree_df) <- c('D', 'M')

plotroc_tree <- plotroc_tree_df %>% ggplot(aes(d = D, m = M)) +
  geom_roc(labelround = 2) + ggtitle("Decision Tree plotROC")

grid.arrange(rocr_tree, plotroc_tree)
```



3.3 Naive Bayes

[16]:
```
mod_nb <- naiveBayes(form, data = train)
income_nb_probs <- mod_nb %>%
  predict(newdata = test, type = "raw")
income_nb_probs <- income_nb_probs[, ' >50K']

pred_nb <- ROCR::prediction(income_nb_probs, test$income)
perf_nb <- ROCR::performance(pred_nb, 'tpr', 'fpr')
perf_nb_df <- data.frame(perf_nb@x.values, perf_nb@y.values, perf_nb@alpha.
 ↪values)
names(perf_nb_df) <- c("fpr", "tpr", "cut")

rocr_nb <- perf_nb_df %>% ggplot(aes(x = fpr, y = tpr, color = cut)) +
  geom_line() + geom_abline(intercept = 0, slope = 1, lty = 3) +
  ylab(perf_nb@y.name) + xlab(perf_nb@x.name) + ggtitle("Naive Bayes ROCR")

plotroc_nb_df <- data.frame(test_income_binary, income_nb_probs)
names(plotroc_nb_df) <- c('D', 'M')

plotroc_nb <- plotroc_nb_df %>% ggplot(aes(d = D, m = M)) +
  geom_roc(labelround = 5) + ggtitle("Naive Bayes plotROC")

grid.arrange(rocr_nb, plotroc_nb)
```



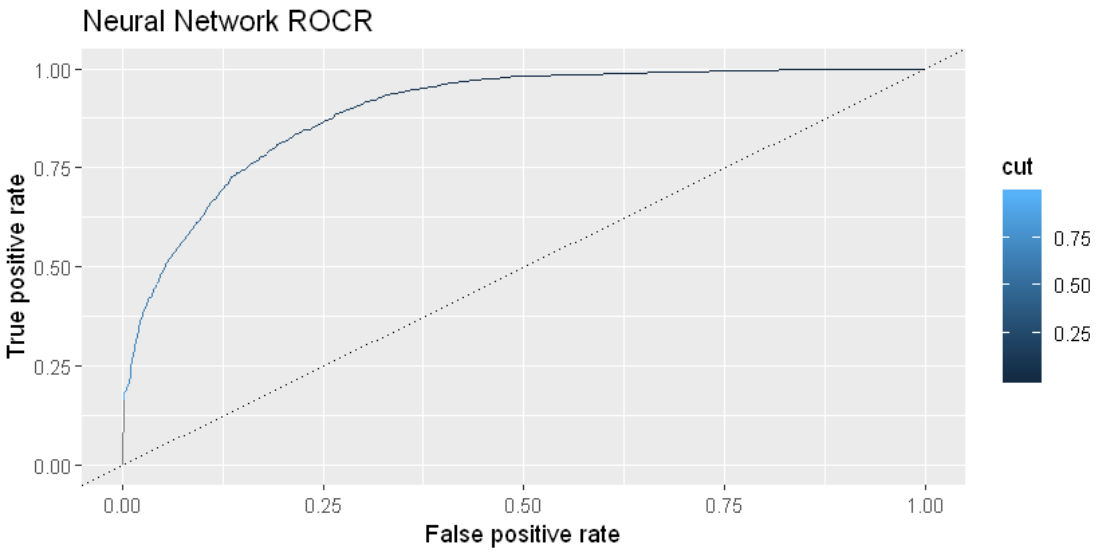


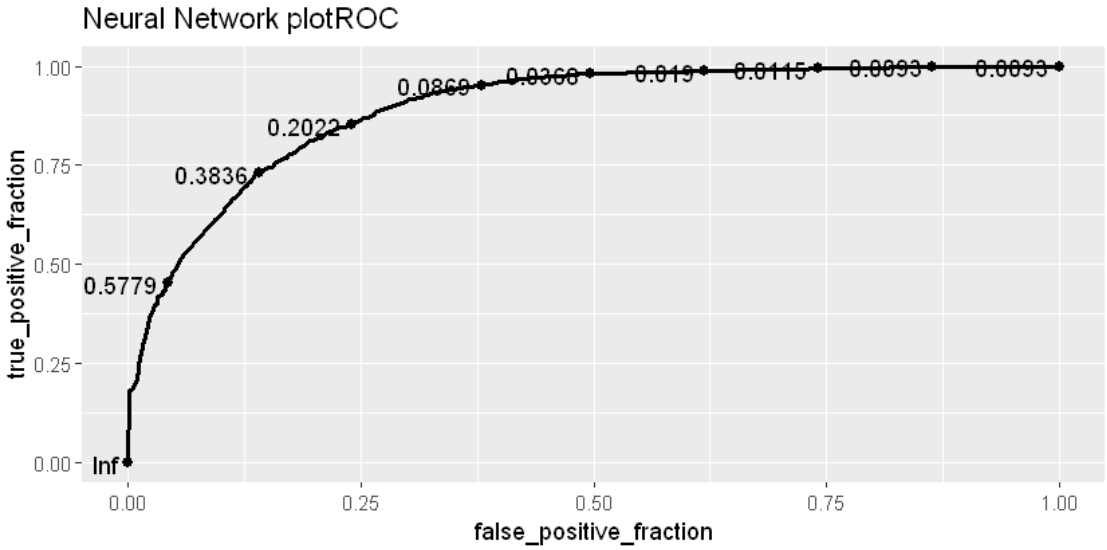
3.4 Neural Network

[17]:
```
mod_nn <- nnet(form, data = train, size = 5)
income_nn_probs <- mod_nn %>%
  predict(newdata = test, type = "raw")
income_nn_probs <- income_nn_probs[, 1]

pred_nn <- ROCR::prediction(income_nn_probs, test$income)
perf_nn <- ROCR::performance(pred_nn, 'tpr', 'fpr')
perf_nn_df <- data.frame(perf_nn@x.values, perf_nn@y.values, perf_nn@alpha.
 ↪values)
names(perf_nn_df) <- c("fpr", "tpr", "cut")
```

```r
rocr_nn <- perf_nn_df %>% ggplot(aes(x = fpr, y = tpr, color = cut)) +
  geom_line() + geom_abline(intercept = 0, slope = 1, lty = 3) +
  ylab(perf_nn@y.name) + xlab(perf_nn@x.name) + ggtitle("Neural Network ROCR")

plotroc_nn_df <- data.frame(test_income_binary, income_nn_probs)
names(plotroc_nn_df) <- c('D', 'M')

plotroc_nn <- plotroc_nn_df %>% ggplot(aes(d = D, m = M)) +
  geom_roc(labelround = 4) + ggtitle("Neural Network plotROC")

grid.arrange(rocr_nn, plotroc_nn)
```

```
# weights:  296
initial  value 18084.849266
iter  10 value 13245.725269
iter  20 value 12943.091588
iter  30 value 12441.781618
iter  40 value 11682.237535
iter  50 value 10105.808035
iter  60 value 9145.121687
iter  70 value 8924.892976
iter  80 value 8785.947300
iter  90 value 8641.198852
iter 100 value 8537.477736
final  value 8537.477736
stopped after 100 iterations
```

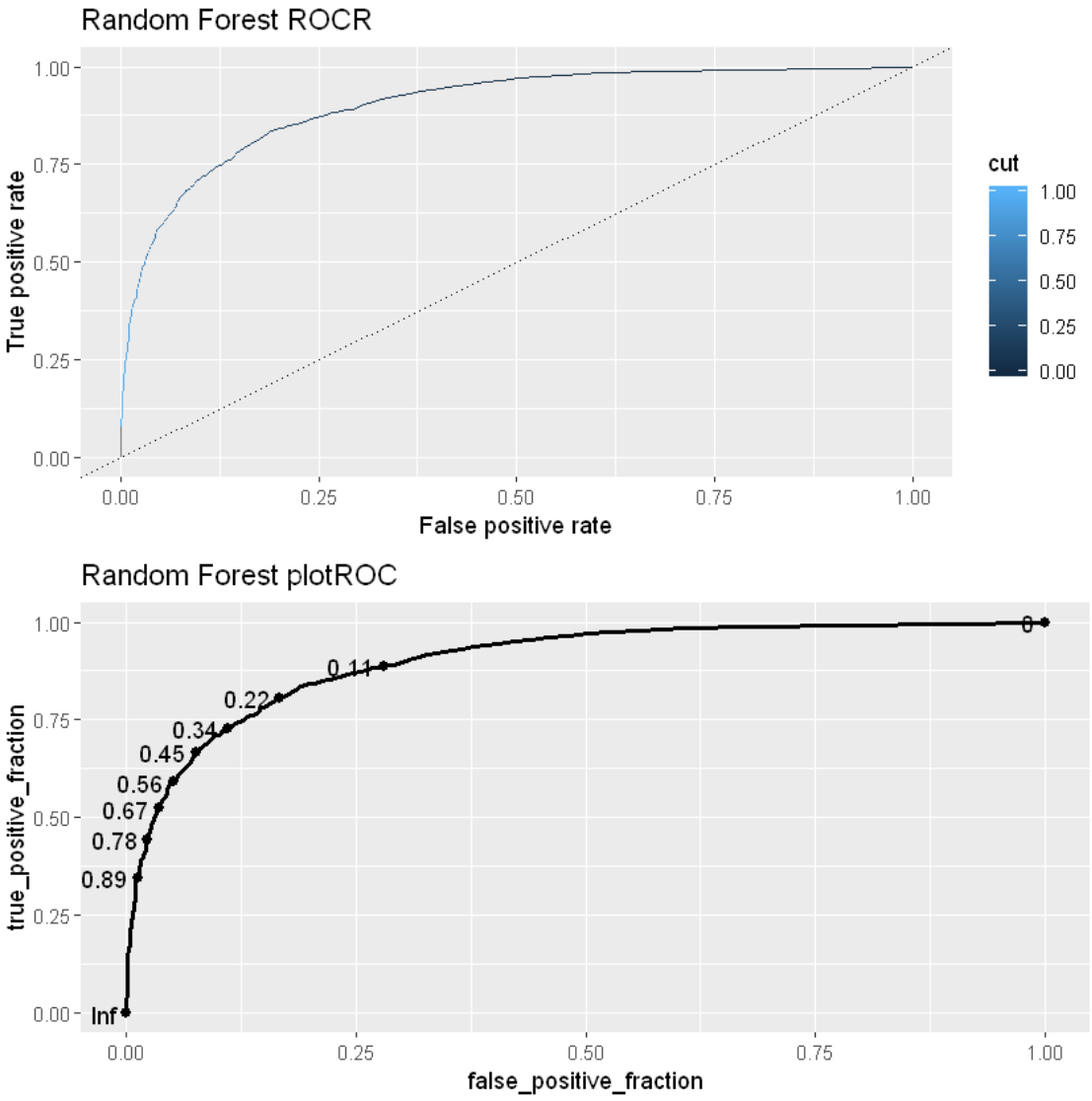




3.5 Random Forest

[18]:
```
mod_forest <- randomForest(form, data = train, ntree = 201, mtry = 3)
income_forest_probs <- mod_forest %>%
  predict(newdata = test, type = "prob")
income_forest_probs <- income_forest_probs[, ' >50K']

pred_forest <- ROCR::prediction(income_forest_probs, test$income)
perf_forest <- ROCR::performance(pred_forest, 'tpr', 'fpr')
perf_forest_df <- data.frame(perf_forest@x.values, perf_forest@y.values,␣
 ↪perf_forest@alpha.values)
names(perf_forest_df) <- c("fpr", "tpr", "cut")
```

```
rocr_forest <- perf_forest_df %>% ggplot(aes(x = fpr, y = tpr, color = cut)) +
  geom_line() + geom_abline(intercept = 0, slope = 1, lty = 3) +
  ylab(perf_forest@y.name) + xlab(perf_forest@x.name) + ggtitle("Random Forest␣
 ↪ROCR")

plotroc_forest_df <- data.frame(test_income_binary, income_forest_probs)
names(plotroc_forest_df) <- c('D', 'M')

plotroc_forest <- plotroc_forest_df %>% ggplot(aes(d = D, m = M)) +
  geom_roc(labelround = 2) + ggtitle("Random Forest plotROC")

grid.arrange(rocr_forest, plotroc_forest)
```



4 Comparing All Models

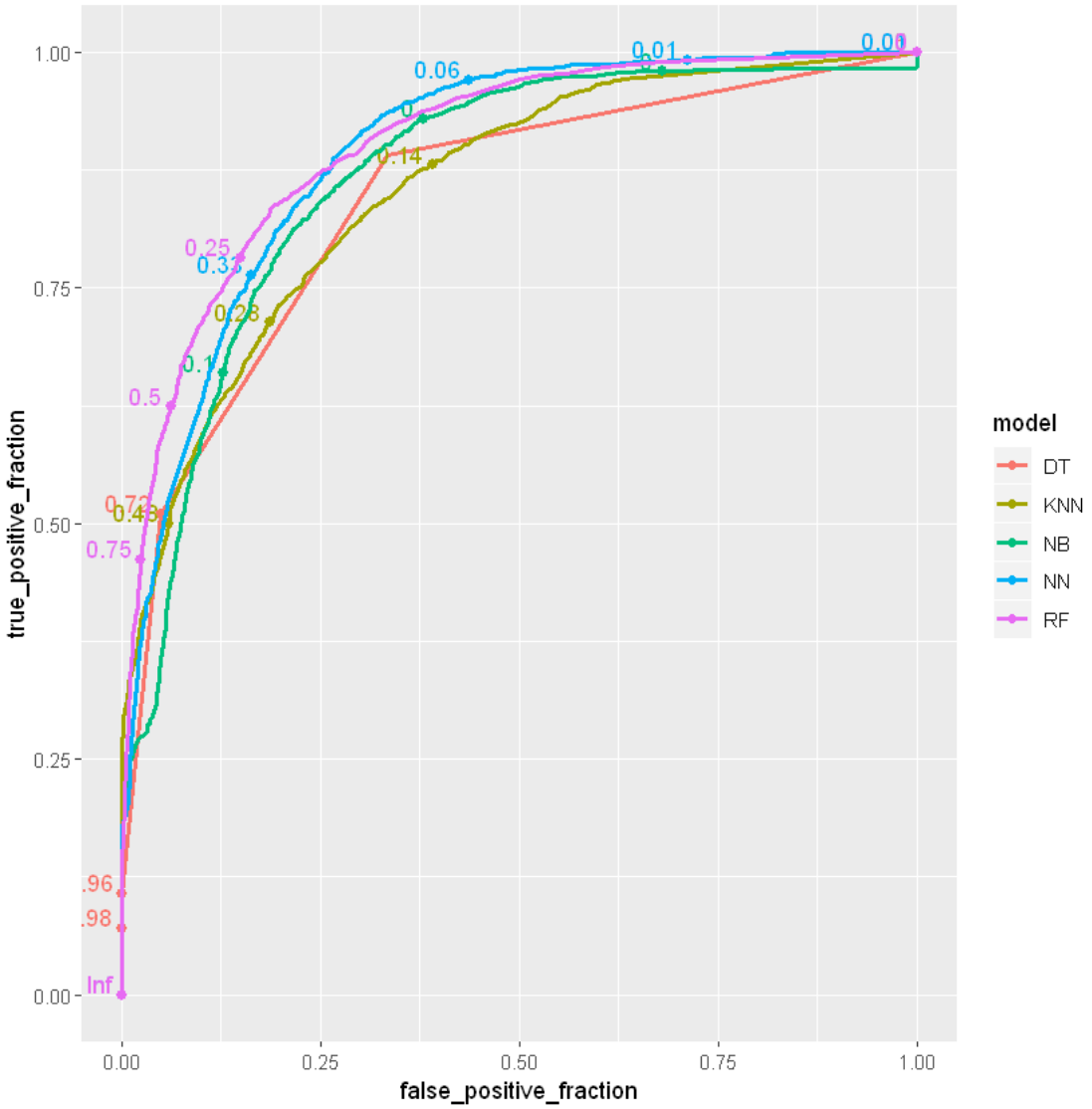
`plotROC` can easily calculate AUC of any `plotROC` ROC curve using `calc_auc`.

[19]:
```
mod_names <- c("RF", "KNN", "NB", "NN", "DT")
plotroc_all_list <- list(plotroc_forest, plotroc_knn, plotroc_nb, plotroc_nn,␣
↪plotroc_tree)
auc_df <- sapply(plotroc_all_list, calc_auc) %>% as.data.frame()
names(auc_df) <- mod_names
auc_df["AUC", ]
```

	RF	KNN	NB	NN	DT
AUC	0.9025387	0.8547584	0.8674455	0.8933798	0.844216

You can also plot several ROC curves on top of each other by combining them into a single data frame and labelling each case with its model. Then, when making the `plotROC ROC` curve, you specify color to equal the model in the `ggplot` aesthetic.

[20]:
```
plotroc_all_df <- data.frame(
  D = c(plotroc_forest_df$D, plotroc_knn_df$D, plotroc_nb_df$D,␣
↪plotroc_nn_df$D, plotroc_tree_df$D),
  M = c(plotroc_forest_df$M, plotroc_knn_df$M, plotroc_nb_df$M,␣
↪plotroc_nn_df$M, plotroc_tree_df$M),
  model = rep(mod_names, each = nrow(test))
)

plotroc_all <- plotroc_all_df %>% ggplot(aes(d = D, m = M, color = model)) +
  geom_roc(n.cuts = 5, labelround = 2)
plotroc_all
```

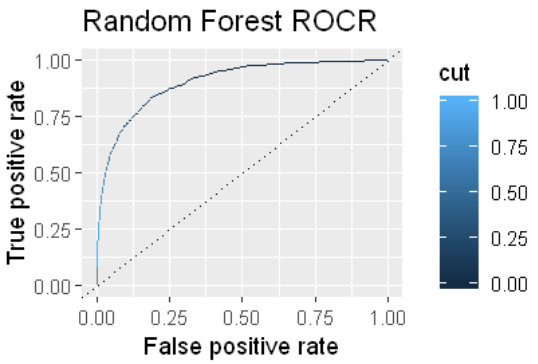


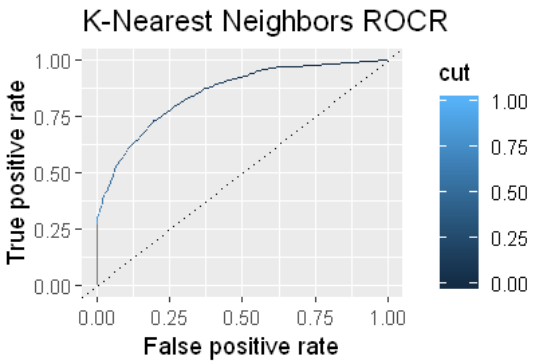
This combined form can also be turned into an interactive graph.

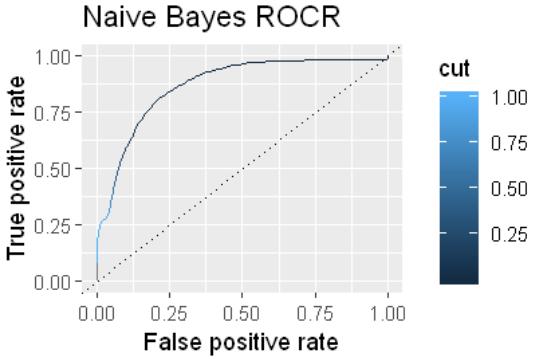
[21]:
```
plotroc_all %>% plot_interactive_roc(labelround = 2)
```

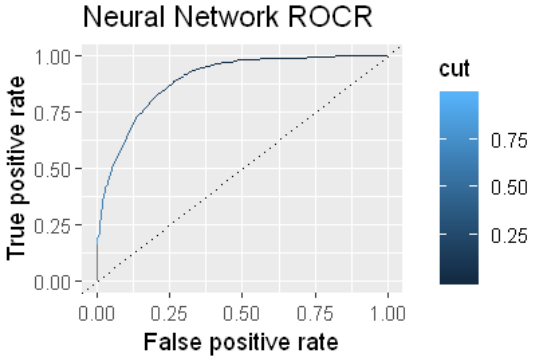
When arranging them on a grid, the labels can get quite crammed, but you can still discern where on the curve the points are concentrated. Compare this to a grid of `ROCR` ROC curves.

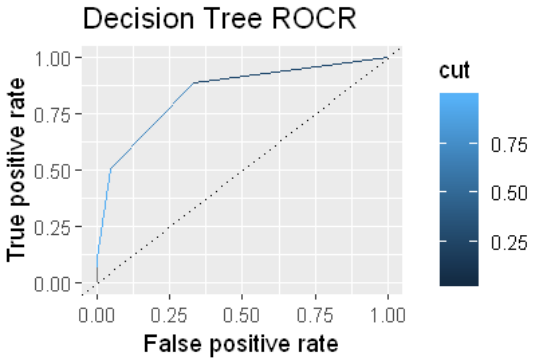
[22]:
```
grid.arrange(rocr_forest, rocr_knn, rocr_nb, rocr_nn, rocr_tree)
grid.arrange(plotroc_forest, plotroc_knn, plotroc_nb, plotroc_nn, plotroc_tree)
```

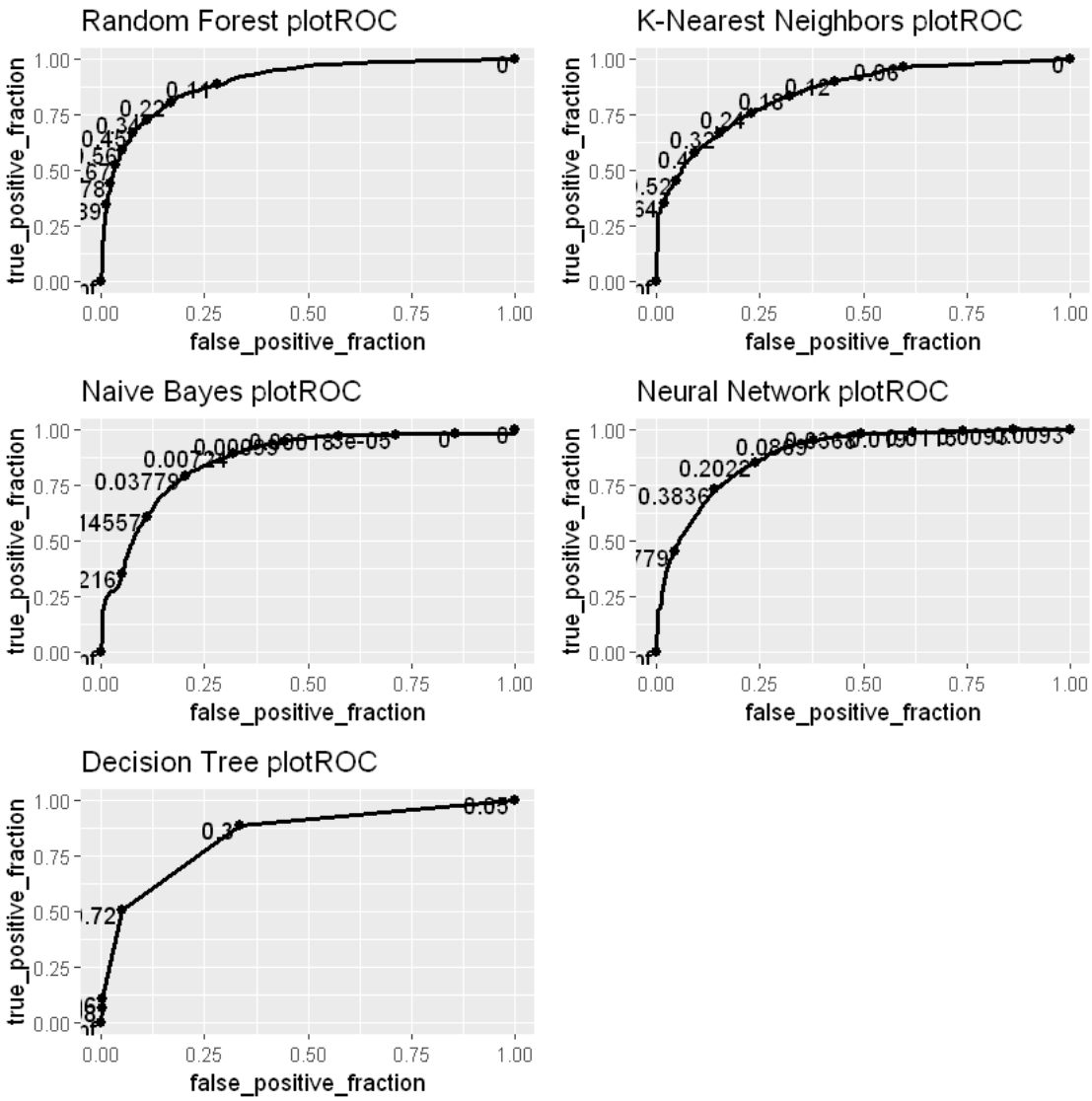

Random Forest ROCR
True positive rate
False positive rate
cut
1.00
0.75
0.50
0.25
0.00


K-Nearest Neighbors ROCR
True positive rate
False positive rate
cut
1.00
0.75
0.50
0.25
0.00


Naive Bayes ROCR
True positive rate
False positive rate
cut
1.00
0.75
0.50
0.25
0.00

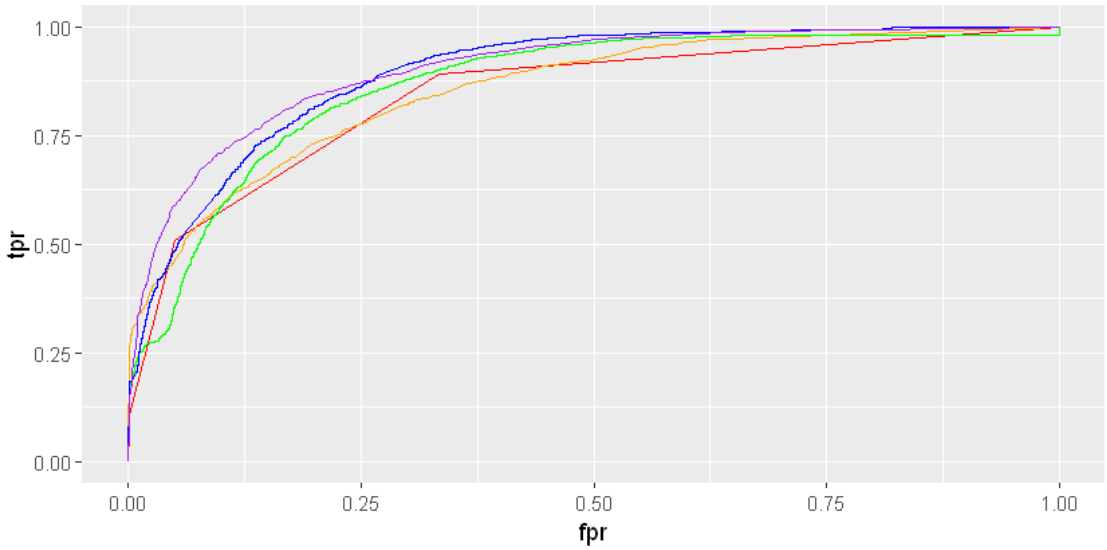

Neural Network ROCR
True positive rate
False positive rate
cut
1.00
0.75
0.50
0.25
0.00

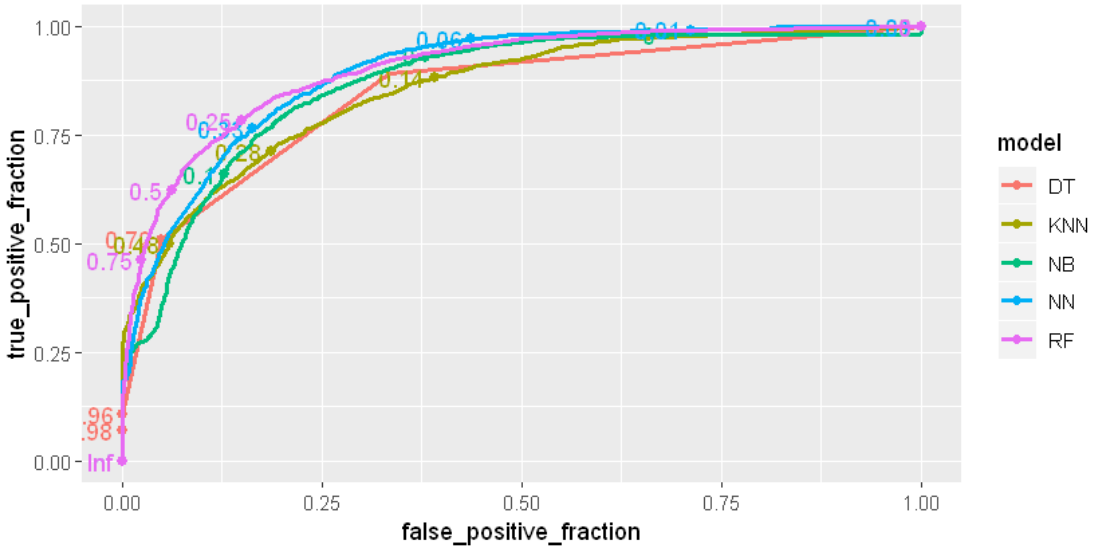

Decision Tree ROCR
True positive rate
False positive rate
cut
1.00
0.75
0.50
0.25
0.00



Finally, compare the `ROCR` ROC curve overlay from before with the `plotROC` ROC curve overlay.

[23]:
```
rocr_all <- ggplot() +
  geom_line(data = perf_tree_df, aes(x = fpr, y = tpr), color = "red") +
  geom_line(data = perf_knn_df, aes(x = fpr, y = tpr), color = "orange") +
  geom_line(data = perf_nb_df, aes(x = fpr, y = tpr), color = "green") +
  geom_line(data = perf_nn_df, aes(x = fpr, y = tpr), color = "blue") +
  geom_line(data = perf_forest_df, aes(x = fpr, y = tpr), color = "purple")

grid.arrange(rocr_all, plotroc_all)
```


1.00
0.75
0.50
0.25
0.00
tpr
0.00
0.25
0.50
0.75
1.00
fpr


1.00
0.75
0.50
0.25
0.00
true_positive_fraction
0.00
0.25
0.50
0.75
1.00
false_positive_fraction
model
DT
KNN
NB
NN
RF