Chapter 8. Data Analytic Thinking: What Is a Good Model?

Fundamental concepts: What is desired from data analysis from business consideration; Expected value as a key evaluation framework; Comparative baseline models/methods.

Exemplary techniques: Evaluation metrics; Estimation costs and benefits; Calculating expected profit; Creating baseline methods for comparison.

Further Reading:

- Provost and Fawcett (2013): Chapters 7 & 8.

Evaluation Classifiers

As discussed before, evaluation is a key step in data mining, which can be carried out using a 'holdout' data set or cross-validation, to detect/avoid overfitting.

However the accuracy, defined below, is simplistic and has some well-known problem

$$\text{accuracy} = \frac{\text{No. of correct decision made}}{\text{Total No. of decision made}}$$

Consider binary classification: two classes only labelled as '*Positive*' and '*Negative*'

Confusion Matrix

	Positive	Negative
Yes	No. true positive	No. false positive
No	No. false negative	No. true negative

True classes: Positive and Negative

Predicted Classes: Yes, it's positive, or, No, it's negative.

Note. *Negative* denotes the normal and often uninteresting status, and *Positive* denotes unusual ones.

The goal of classification is to sift through a large number of normal and uninteresting (i.e. negative) individuals in order to single out a relative small number of unusual (i.e. positive) ones.

Examples: looking for defrauded customers, checking for defective parts, targeting consumers who actually would respond to an ad.

When the classes are skewed with the ratio 999:1, a simple rule – always choose the majority class gives 99.9% accuracy. This is unlikely to be satisfactory!

Consider two classifiers for customer churn:

Classifier A: 80% accuracy

Classifier B: 64% accuracy

Is A better than B? — Not necessary!

They may use different test data.

In reality, the churn rate is only about 10%, now both A and B look bad, or?

We need to look into the data sets used by A and B carefully, as they may be quite different from each other, and also different from real data with different churn rates.

There are two types of errors: false positive, and false negative

Now test the two classifiers on 1000 customers with 500 positive (churn) and 500 negative (not churn). The results are as follows:

Classifier A

	churn	not churn
Y	500	200
N	0	300

Classifier B

	churn	not churn
Y	300	0
N	200	500

Classifier A: 200 false positive

Classifier B: 200 false negative

For application to real data with 10% churn rate, B achieves overall 96% accuracy while A achieves merely 64%.

Unequal costs should be associated with the two types of errors

Unequal Costs and Benefits

The costs for false positive and false negative are often different from each other.

The benefits for predicting true positive and true negative correctly are also different.

For customer churn example, the cost of giving a customer a retention incentive *unnecessarily* (a false positive) and that of losing a customer because no incentive was offered (a false negative) are different!

Note. This also applies to other data analysis problems such as regression: positive residuals and negative residuals may lead to very different business costs.

For example, a movie recommendation model predicts how many stars each customer would give to an unseen movie. For such a problem, the MSE or regression R^2 are no longer appropriate.

Another example is to use variance or STD as a measure for risk.

Questions to ask:

What is important for business?

What is the goal of analysis?

Are we assessing the results of data analysis appropriately given the actual goal?

Is the metric used meaningful? Is there a better one?

Expected Value: A Key Analytical Framework

Expected value computation: provides a framework in organizing the thinking about data-analytic problems into three stages:

1. the structure of the problem

2. the elements of the analysis that can be extracted from the data

3. the elements of the analysis that need to be acquired from other sources (e.g. business knowledge of subject matter experts)

An expected value may represent expected profits (to be maximized), expected losses (to be minimized).

General form of an expected value: Let $O_1, O_2, O_3, \cdots$ denote all the possible outcomes, $p_i = P(O_i)$ denote the probability of the occurrence of O_i, and V_i be the value when O_i occurs. Then the expected value is

$$EV = p_1 V_1 + p_2 V_2 + p_3 V_3 + \cdots .$$

1. Possible outcomes $O_1, O_2, O_3, \cdots$ are identified from a proper understanding the structure of the problem

2. Probabilities $p_1, p_2, p_3, \cdots$ are evaluated from data analysis

3. Values $V_1, V_2, V_3, \cdots$ are obtained from other sources.

Using Expected Value to Frame Classifier Use

Consider a targeted marketing problem, we assign each consumer a class of *likely responder* versus a class of *not likely responder*. (Then the resource should be spent on the individuals in the likely responder class.)

This is typically an unbalanced classification problem, as, for example, the response rate for an advertisement is small, or very small, say 1%.

Hence the predicting every one to NP yields accuracy 99%, suggesting no advertising at all!

Let $\mathbf{x}$ denote the feature variable of an individual, $p_r(\mathbf{x})$ be the probability to response. Suppose the profit from buying the product is $£100$, and the ad cost is $£1$. Then the expected value for this individual is

$$EV(\mathbf{x}) = (100 - 1)p_r(\mathbf{x}) - 1 \cdot (1 - p_r(\mathbf{x})) = 100p_r(\mathbf{x}) - 1.$$

Hence we should send the ad to this individual if $EV(\mathbf{x}) > 0$, which is

$$p_r(\mathbf{x}) > 0.01.$$

Using Expected Value to Evaluate Classifiers

Since each model will make some decisions better than the others, we need compare them collectively.

Models to be compared for a classification problems may include:

- Some baseline models (such as completely random classifiers)
- Bayes or Naive Bayes classifiers
- K-NN classifiers
- Decision trees
- Logistic regression
- Hand-crafted model suggested by subject matter experts

Aggregated expected value over the whole population

For a two-class classification problem with the total 110 individual, suppose the confusion matrix is

	p	n
Y	56	7
N	5	42

The rates for one individual falling in each cell are estimated by

$$p(Y,p) = 56/110 = 0.51, \quad p(Y,n) = 7/110 = 0.06,$$

$$p(N,p) = 5/110 = 0.05, \quad p(N,n) = 42/110 = 0.38.$$

To calculated the expected value or expected benefit, we need to know the benefit (or cost) for each of the above 4 cells.

Let us continue with the targeted AD example,

- A *false positive* occurs when we classify a consumer as a likely responder and therefore target her, but she does not respond. So the benefit is $b(Y, n) = -1$, or the cost is $c(Y, n) = 1$.
- A *false negative* is a consumer who was predicted not to be a likely responder (so was not offered the product), but would have bought it if offered. In this case, no money was spent and nothing was gained, so $b(N, p) = 0$.
- A *true positive* is a consumer who is offered the product and buys it. The benefit in $b(Y, p) = 100 - 1 = 99$
- A *true negative* is a consumer who was not offered a deal and who would not have bought it even if it had been offered. The benefit in this case is zero (no profit but no cost), so $b(N, n) = 0$.

The cost-benefit matrix is

	p	n
Y	99	-1
N	0	0

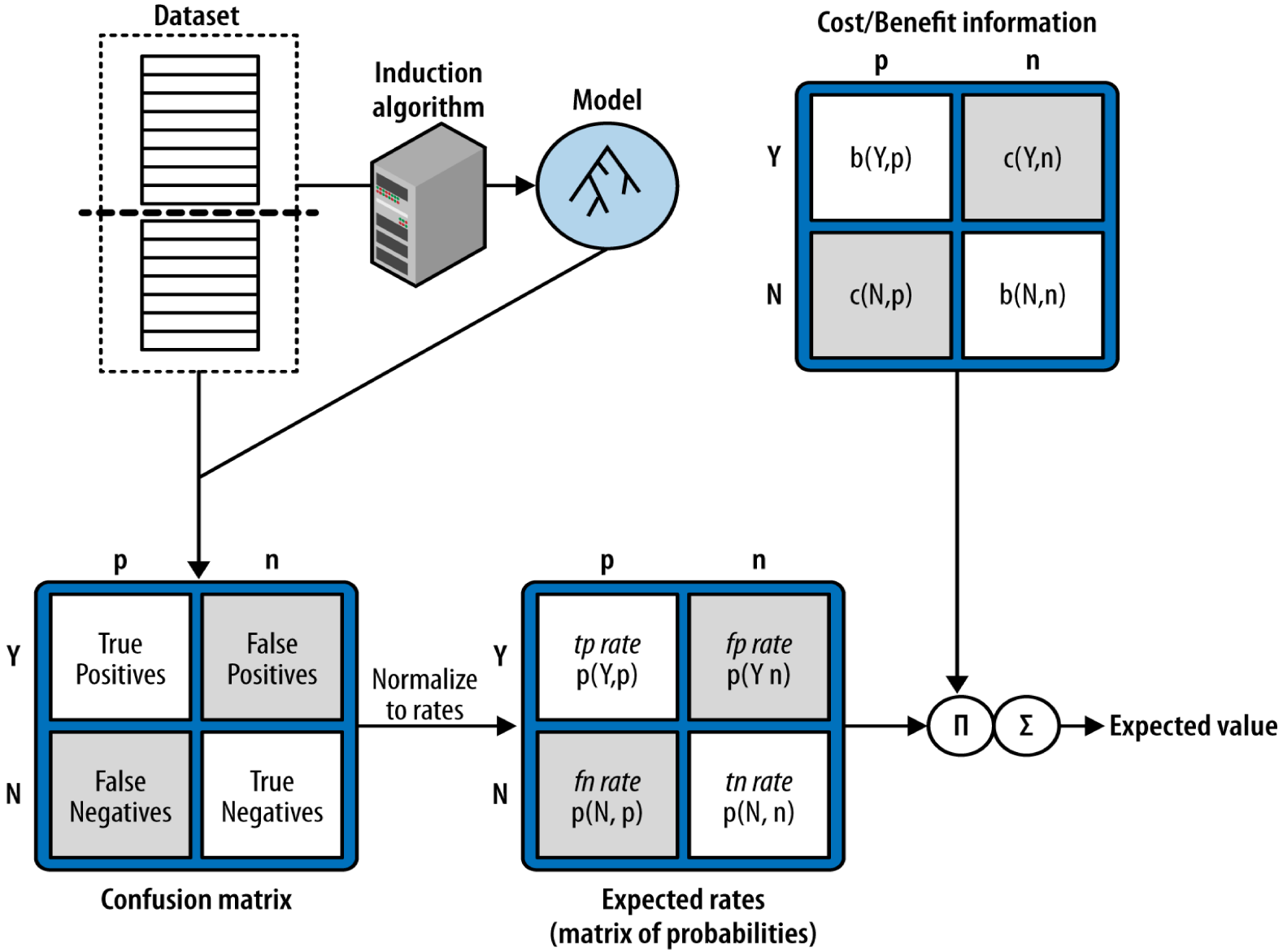
Thus the expected benefit is

$$p(Y,p)\cdot b(Y,p)+p(Y,n)\cdot b(Y,n)+p(N,p)\cdot b(N,p)+p(N,n)\cdot b(N,n)$$
$$=0.51\times 99-0.06\times 1=50.43.$$

Remark. (i) Avoid 'double count' by, for example, using $b(N,p)=-99$. (This can be detected by calculating the increased benefit from moving one individual from (N,p) to (Y,p).)

(ii) *R computing*: Let `B` be a cost-benefit matrix, `P` be a probability matrix. The expected benefit is `sum(B*P)`.

(iii) Extension to the cases with more than two classes is obvious.


Dataset
Induction algorithm
Model
Cost/Benefit information
p
n
Y
N
b(Y,p)
c(Y,n)
c(N,p)
b(N,n)
True Positives
False Positives
False Negatives
True Negatives
Normalize to rates
tp rate p(Y,p)
fp rate p(Y n)
fn rate p(N, p)
tn rate p(N, n)
Π
Σ
Expected value
Confusion matrix
Expected rates (matrix of probabilities)

Other often used metrics for 2-class classification problems

Let *TP, FP, TN and FN* denote, respectively, the number of true positive, false positive, true negative and false negative. Then $n = TP + FP + TN + FN$, $TP + FN$ is the number of true positive individuals, and $TN + FP$ is the number of true negative individuals, and

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

- True positive rate: $TP/(TP + FN)$ – rate being correct when the truth is positive
- False negative rate: $FN/(TP + FN)$ – rate being incorrect when the truth is positive

True negative rate and false positive rate are defined in the similar manner

In text classification,

- Precision: $TP/(TP + FP)$
- Recall: $TP/(TP + FN)$ i.e. true positive rate
- F-measure: $F_1 = 2\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$, i.e. the harmonic mean of Precision and Recall

Ideally choose a model to maximize both Precision and Recall, which is often impossible. A compromise is to maximize F_1.

ROC Graphs (Receiver Operating Characteristics Graphs)

For each classifier, plot *True Positive Rate* (also called hit rate)

$$TPR = TP/(TP + FN) \qquad \text{(the bigger the better)}$$

against *False Positive Rate* (also called false alarm rate)

$$FPR = FP/(FP + TN) \qquad \text{(the smaller the better)}$$

Some characteristics of an ROC graph (see the graph on next page):

- Main diagonal corresponds to random classifiers. For example, E at (0.6, 0.6) classifies each individual to *Positive* with probability 0.6.

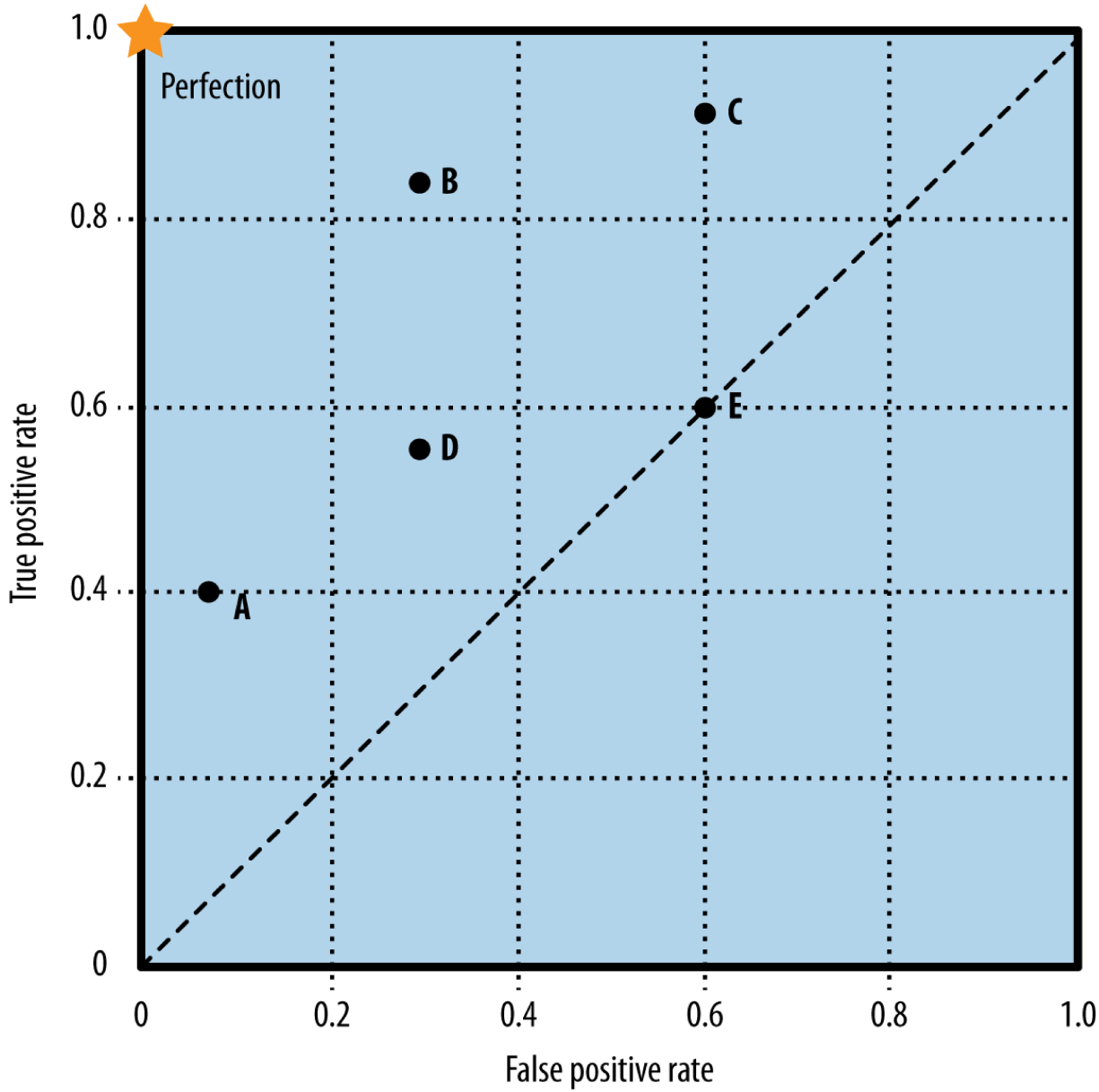
- Any admissible classifier should be above the main diagonal.

- The (0, 1) indicates the perfect classifier which identify every individuals correctly.

- In the graph on next page, A is more conservative than B, which in turn is more conservative than C.

- B is a better classifier than D, as it has a higher TPR and the equal FNR.

Note. ROC graphs do not take into account of costs/benefits. Nevertheless it is a more appropriate measure that the (overall) misclassification rate. For example, it rules out the decision of taking no action in an advertising campaign with a population with responsive rate 1%, as such a decision entails point (0, 0) on an ROC graph.


1.0
Perfection
C
B
0.8
0.6
E
D
0.4
A
0.2
0
True positive rate
0
0.2
0.4
0.6
0.8
1.0
False positive rate

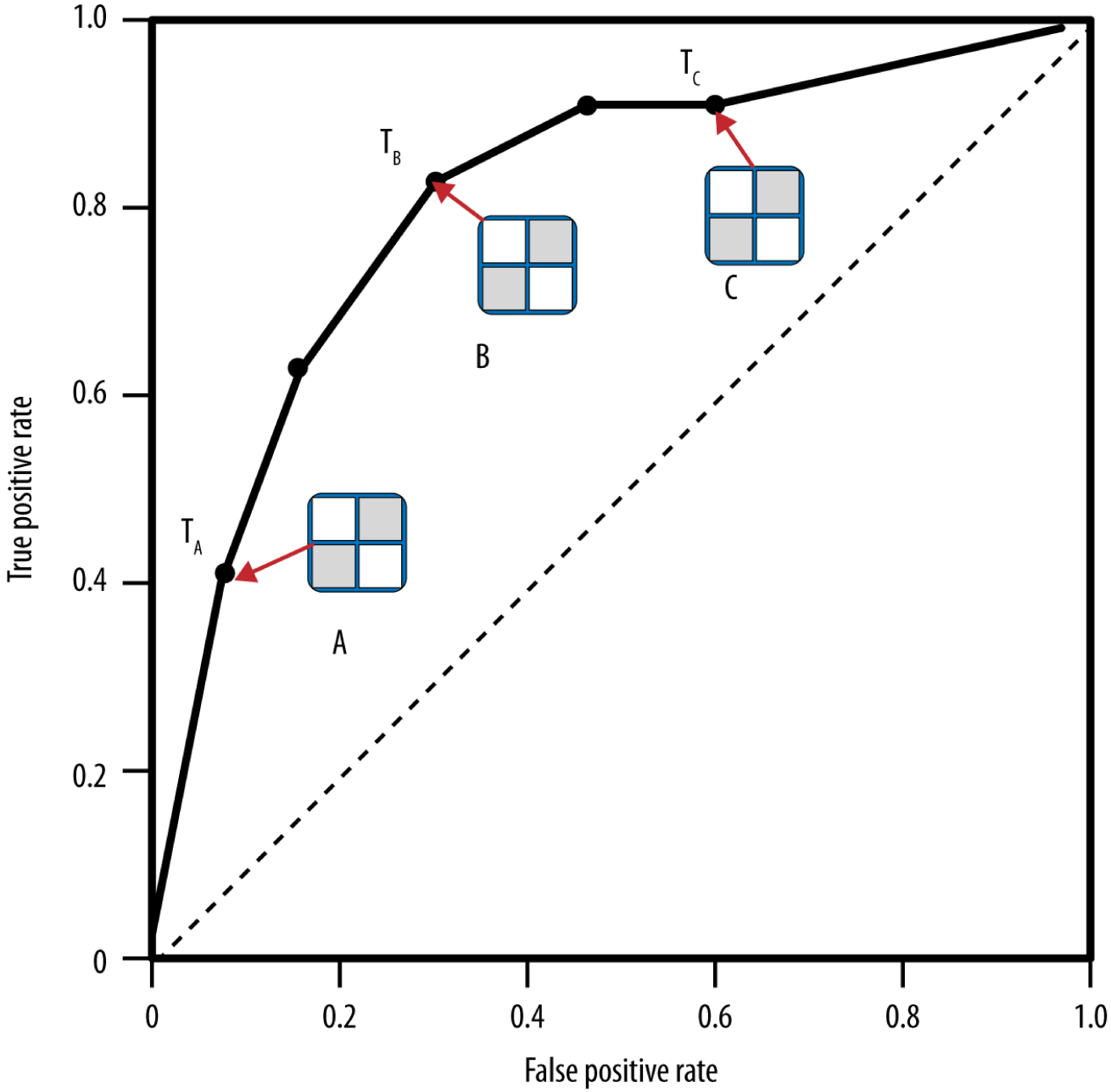
ROC curves

For any classifier for two-class problem, let $P(Y|\mathbf{X})$ be the predictive probability for *positive* given feature $\mathbf{X}$, and $P(N|\mathbf{X}) = 1 - P(Y|\mathbf{X})$ be the probability to predict *negative*.

Classification decision: predict *positive* if $P(Y|\mathbf{X}) > t$, where $t \in [0, 1)$ is a threshold.

The threshold value t is determined by, e.g. by using the expected value. The conventional practice: $t = 0.5$

For each $t \in [0, 1)$, the decision rule $P(Y|\mathbf{X}) > t$ leads to a different confusion matrix, therefore a point in the ROC graph. Let t vary from 0 to 1, it generates an **ROC curve**.



ROC curve for one classifier: each point on the curve corresponds to a different threshold value t and a different confusion matrix.

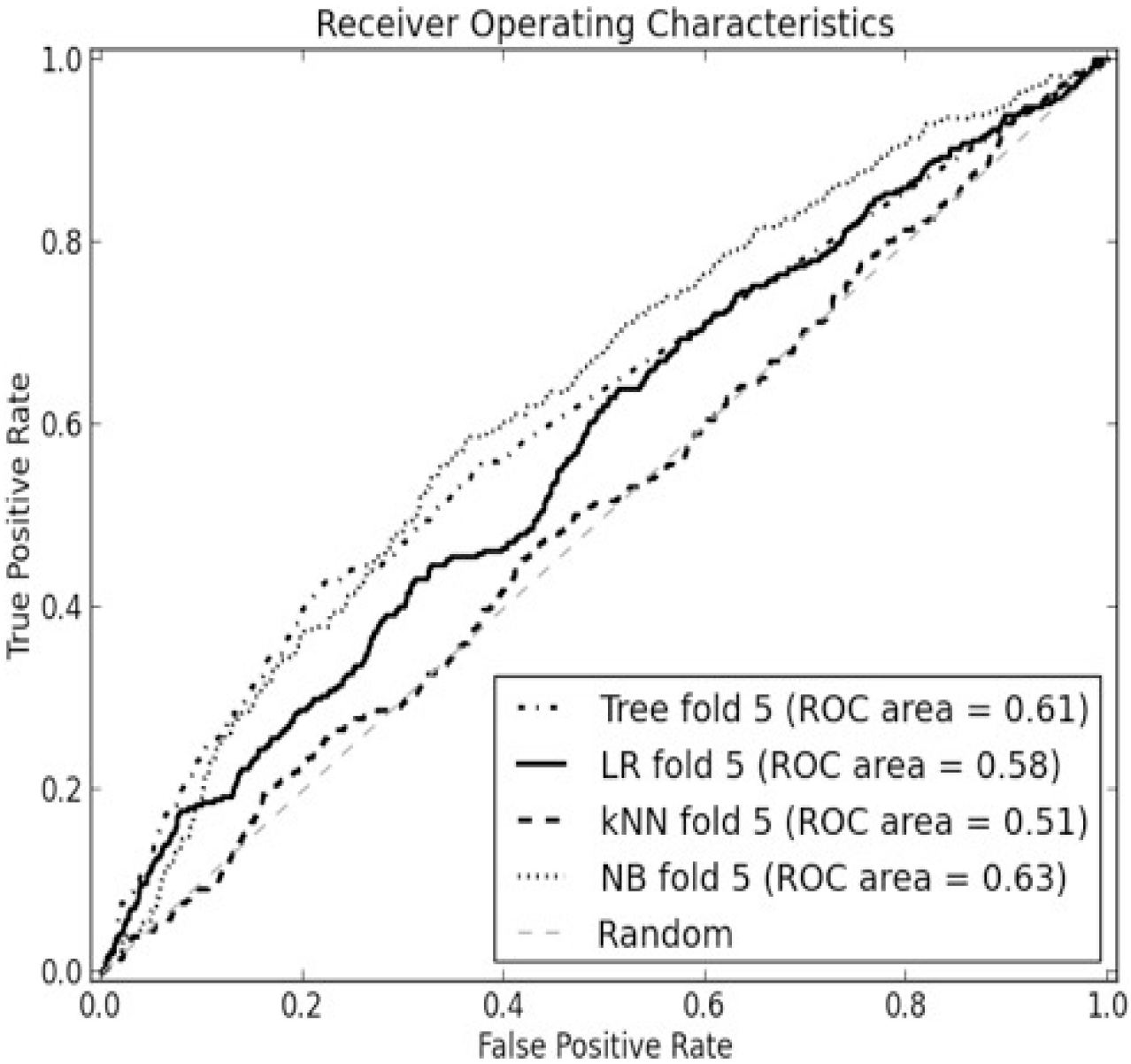
We can compare different classifiers by comparing their ROC curves.

Drawing a vertical line (all the points on the line have the same FPR), the classifier at the top achieves the maximum TPR

Drawing a horizontal line (all the points on the line have the same TPR), the classifier at the most left achieves the minimum FPR

If we prefer one index to compare different classifier, use the **Area Under the ROC curve (AUC)**.

For random classifier, AUC=0.5. So any reasonable classifiers should have AUC>0.5 at least.



ROC curves of the classifiers on one fold of cross-validation for the churn problem

LR: logistic regression

kNN: K-nearest neighbours

NB: Naive Bayesian

Compare Performance with Some Baseline Models

It is important to consider carefully what would be a reasonable baseline for a given problem. The selected model should perform better than the baseline model according to a appropriate performance metric.

Below are some possible baseline models.

For classification problems,

- completely random model (i.e. assign to all classes with the equal probability)
- majority classifier

For regression problems,

mean $\bar{y}$

the nearest neighbour

For time series,

the previous value

average of today over many years

For predicting how many stars a customer would give to a particular film: a convex combination of the average ranking from other customer and the average ranking from this customer over different films.

The (naive) Bayesian methods are other candidates which should be included in the comparison

For simplification, consider a classification problem for which Y is the label of classes and $\mathbf{X} = (X_1, \cdots, X_m)$ are predictors/features taking discrete values.

It follows the Bayesian formula that

$$p(Y = j|\mathbf{X}) = \frac{p(Y = j, \mathbf{X})}{p(\mathbf{X})}.$$

Thus for given $\mathbf{X}$, and the Bayes classifier sets $Y = \ell$, where ℓ satisfies the inequality $p(Y = \ell|\mathbf{X}) \geq \max_j p(Y = j|\mathbf{X})$, or equivalently

$$p(Y = \ell, \mathbf{X}) \geq \max_j p(Y = j, \mathbf{X}).$$

Lift: $p(Y = j|\mathbf{X}) \Big/ p(Y = j)$

Note both $p(Y = j, \mathbf{X})$ (also $p(Y = j)$ and $p(\mathbf{X})$) can be estimated using the relatively frequencies from the training data.

The naive Bayes assumes the conditional independence:

$$p(\mathbf{X}|Y) = \prod_{i=1}^{m} p(X_i|Y).$$

Then

$$p(Y = j|\mathbf{X}) = \frac{p(Y = j, \mathbf{X})}{p(\mathbf{X})} = \frac{p(Y = j)}{p(\mathbf{X})} p(\mathbf{X}|Y = j) = \frac{p(Y = j)}{p(\mathbf{X})} \prod_{i=1}^{m} p(X_i|Y = j)$$

Thus for given $\mathbf{X} = (X_1, \cdots, X_m)$, the naive Bayes classifier sets $Y = \ell$ with ℓ satisfying the following inequality

$$p(Y = \ell) \prod_{i=1}^{m} p(X_i|Y = \ell) \geq \max_j p(Y = j) \prod_{i=1}^{m} p(X_i|Y = j)$$

- R packages `ROCR` and `pROC` produce ROC curves

- How to draw ROC curves for the classification problems with more than two classes?

  1. Multiple two-class classification: one class versus all other classes.

  2. For more general approaches, see

     Krzanowski, W.K. and Hand, D.J. (2009). *ROC Curves for Continuous Data*. CRC Press.

R package: ROCR

Two important functions: `prediction` and `performance`

- First apply `prediction` to data:
  ```
  prediction.output = prediction(pred, labels, label.ordering = NULL)
  ```
  `pred` consists of predicted measures (eg. estimated probabilities, odds), can be the outputs of a decision tree, a logistic regression and etc. `labels` are true binary (i.e. 0 or 1) labels of classes. Both `pred` and `labels` can be a vector, matrix or data.frame, but they must be of the same size.

- To produce ROC graph,
  ```
  roc1=performance(prediction.output, measure="tpr", x.measure="fpr")
  plot(roc1, col=as.list(1:10))
  abline(a=0,b=1) # adding a straight line y=a+bx
  ```

- To calculate AUC (area under the curve):
  ```
  auc1=performance(prediction.output, measure="auc")
  ```

```
auc1@y.values
```

- To produce the overall accuracy curve against cut-off probability:

```
acc1=performance(prediction.output, measure="acc")
plot(acc1, col=as.list(1:10)) # p is No. of ACC curves
```

More information on `ROCR`:

`https://www.r-bloggers.com/a-small-introduction-to-the-rocr-package/`

Example: re-visit the spam email example in Chapter 3.

4601 emails: 2788 true mails, and 1812 spam mails

57 quantitative predictors

```
> spamData=read.table("spam.txt")
> dim(spamData)
[1] 4601   58    # last colunm is labels: 1 for spam, 0 for true mail
> spamNames=read.table("spamNames.txt")  # names of 58 variables
> dim(spamNames)
[1] 58  1
> names(spamData)=spamNames[,1]
> names(spamData)
 [1] "make"       "address" "all"       "3d"        "our"       "over"       "remove"
 [8] "internet"   "order"   "mail"      "receive"   "will"      "people"     "report"
[15] "addresses"  "free"    "business"  "email"     "you"       "credit"     "your"
[22] "font"       "000"     "money"     "hp"        "hpl"       "george"     "650"
[29] "lab"        "labs"    "telnet"    "857"       "data"      "415"        "85"
[36] "technology" "1999"    "parts"     "pm"        "direct"    "cs"         "meeting"
[43] "original"   "project" "re"        "edu"       "table"     "conference" ";"
[50] "("          "["       "!"         "$"         "#"         "CAPAVE"     "CAPMAN"
[57] "CAPTOT"     "LABEL"
```

In the 54-th line of file "spamNames.txt", # is written as "#" !!!

```
> attach(spamData)
> spam=rep("No", 4601);  spam[LABEL==1]="Yes"
> spamData1=data.frame(spamData, spam)
> train=sample(1:4601, 3065, replace=F) # 3065 emails as training sample
> spamTest=spamData1[-train,]
> dim(spamTest)
[1] 1536   59     # 1536 emails for testing
> library(tree)
> tree1=tree(spam~.-LABEL, data=spamData1, subset=train)
Error: unexpected ',' in "tree1=tree(label~,"
```

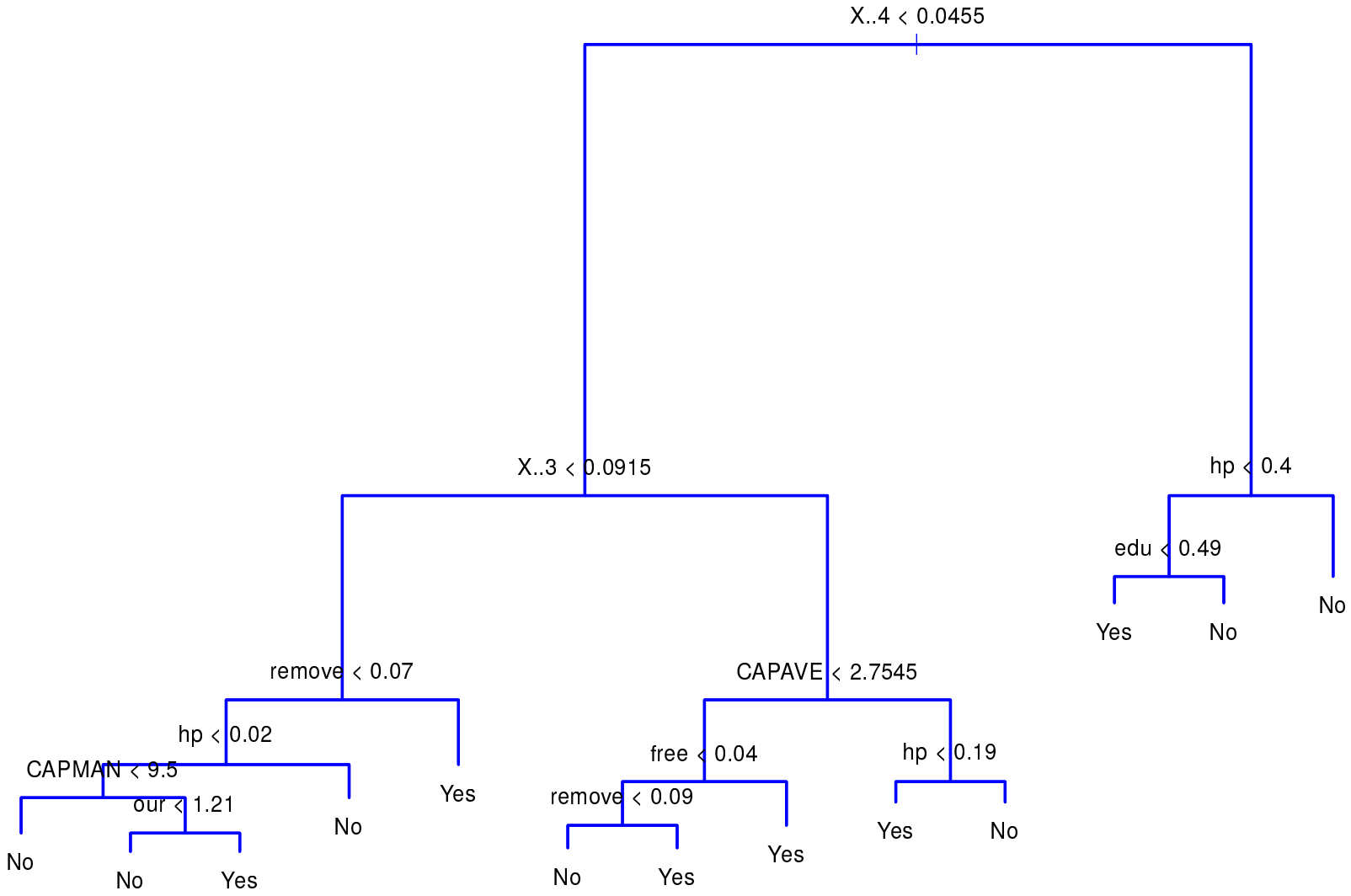
The error message is not clear. Checking it with google, we find out that we have used some illegal names such as '415', '[' in R!

```
> make.names(names(spamData1), unique=T, allow_=T)
 [1] "make"       "address"  "all"      "X3d"      "our"      "over"       "remove"
 [8] "internet"   "order"    "mail"     "receive"  "will"     "people"     "report"
[15] "addresses"  "free"     "business" "email"    "you"      "credit"     "your"
[22] "font"       "X000"     "money"    "hp"       "hpl"      "george"     "X650"
[29] "lab"        "labs"     "telnet"   "X857"     "data"     "X415"       "X85"
[36] "technology" "X1999"    "parts"    "pm"       "direct"   "cs"         "meeting"
[43] "original"   "project"  "re"       "edu"      "table"    "conference" "X."
[50] "X..1"       "X..2"     "X..3"     "X..4"     "X..5"     "CAPAVE"     "CAPMAN"
```

```
[57] "CAPTOT"     "LABEL"   "spam"
> names(spamData1)=make.names(names(spamData1), unique=T, allow_=T)
> detach(spamData)  # NECESSARY, otherwise R will be confused with same names
                    # from the two data sets
> attach(spamData1)
> tree1=tree(spam~.-LABEL, data=spamData1, subset=train) # exclude LABEL!
> summary(tree1)

Classification tree:
tree(formula = spam ~ . - LABEL, data = spamData1, subset = train)
Variables actually used in tree construction:
 [1] "X..4" "remove" "X..3" "hp"  "CAPMAN" "our" "CAPAVE" "free" "george" "edu"
Number of terminal nodes:  13
Residual mean deviance:  0.4804 = 1466 / 3052
Misclassification error rate: 0.08418 = 258 / 3065
> tree2=cv.tree(tree1, FUN=prune.misclass)
> tree2$size
 [1] 13 11  8  7  6  5  3  2  1
> tree2$dev
 [1]  311  323  323  325  337  368  430  624 1178
> plot(tree1, col="blue", lwd=2)
> text(tree1, pretty=0)
```

Note `tree1` with 13 terminal nodes is the CV-selected model, no need to prune further. **Note `X..4`=$, `X..3`=!**

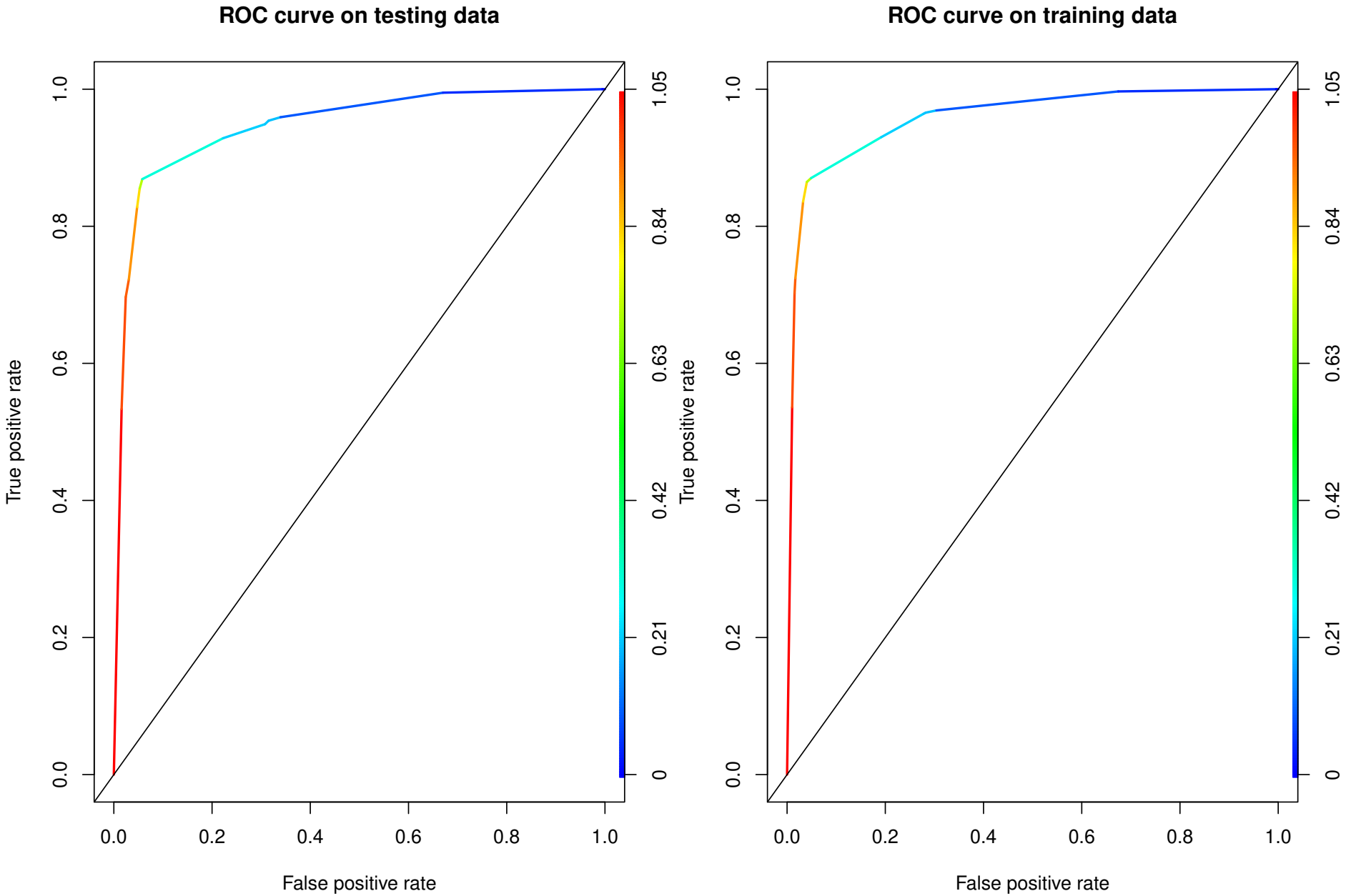

X..4 < 0.0455
X..3 < 0.0915
hp < 0.4
edu < 0.49
Yes
No
No
remove < 0.07
CAPAVE < 2.7545
hp < 0.02
free < 0.04
hp < 0.19
CAPMAN < 9.5
Yes
remove < 0.09
Yes
No
our < 1.21
No
No
Yes
No
No
Yes
No
Yes

Now we calculate its predicted probabilities for 'Yes' (spam) on both training and testing data.

```
> predT=predict(tree1, spamTest, type="vector") # check ?predict.tree
> predTest.tree=predT[,2] # predicted probabilities for 'Yes' (spam)
> length(predTest.tree)
[1] 1536
> summary(predTest.tree)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.04288 0.14237 0.40017 0.88931 0.94986
> predT=predict(tree1, spamData1[train,], type="vector")
> predTrain.tree=predT[,2]
> length(predTrain.tree)
[1] 3065
```

To produce ROC curves,

```
> library(ROCR)
> prediction.treeTest=prediction(predTest.tree, LABEL[-train])
> prediction.treeTrain=prediction(predTrain.tree, LABEL[train])
> rocTest=performance(prediction.treeTest, measure="tpr", x.measure="fpr")
> rocTrain=performance(prediction.treeTrain, measure="tpr", x.measure="fpr")
> par(mfrow=c(1,2))
> plot(rocTest, lwd=2, colorkey=T, colorize=T, main="ROC curve on testing data")
> abline(0,1)    # You may like to check ?plot.performance
> plot(rocTrain, lwd=2, colorkey=T, colorize=T,, main="ROC curve on training data")
> abline(0,1)
```


ROC curve on testing data
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0
0.21
0.42
0.63
0.84
1.05
ROC curve on training data
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0
0.21
0.42
0.63
0.84
1.05

Option `colorize=T` adds the color code to the ROC curve according to the cut-off probability for 'Yes' (spam), `colorkey=T` adds the color key for the cut-off probability vertically on the right. Note that when the cut-off probability increases, both TPR and FPR decrease. This makes an ROC graph much more imformative.

In practice we need to choose the cut-off probability. A convenient choice is 0.5. However a more meaningful approach is to consider costs/benefits for different scenarios and choose the one which maximize (or minimize) the expected benefit (or cost).

To count for the importance of not filtering out genuine emails, we define the cost/benefit matrix as follows:

	Email	Spam
No	0	-1
Yes	-4	0

```
> predLab=ifelse((predTrain.tree>=0.5), "Yes","No")
> confusion=table(predLab, spam[train], deparse.level=2)
> confusion
```

```
       spam[train]
predLab    0    1
    No  1799  170
    Yes   88 1008
> CB=matrix(c(0,-1,-4,0), nrow=2, byrow=T)
> CB
     [,1] [,2]
[1,]    0   -1
[2,]   -4    0
> sum(CB*confusion)/sum(confusion) # compute expected benifit
[1] -0.17031
```

The expected benefit for using this filter is -0.170. To reduce the number of false positives, we should increase the cut-off probability. Below we find the value of the cut-off probability, which maximizes the expected benefit.

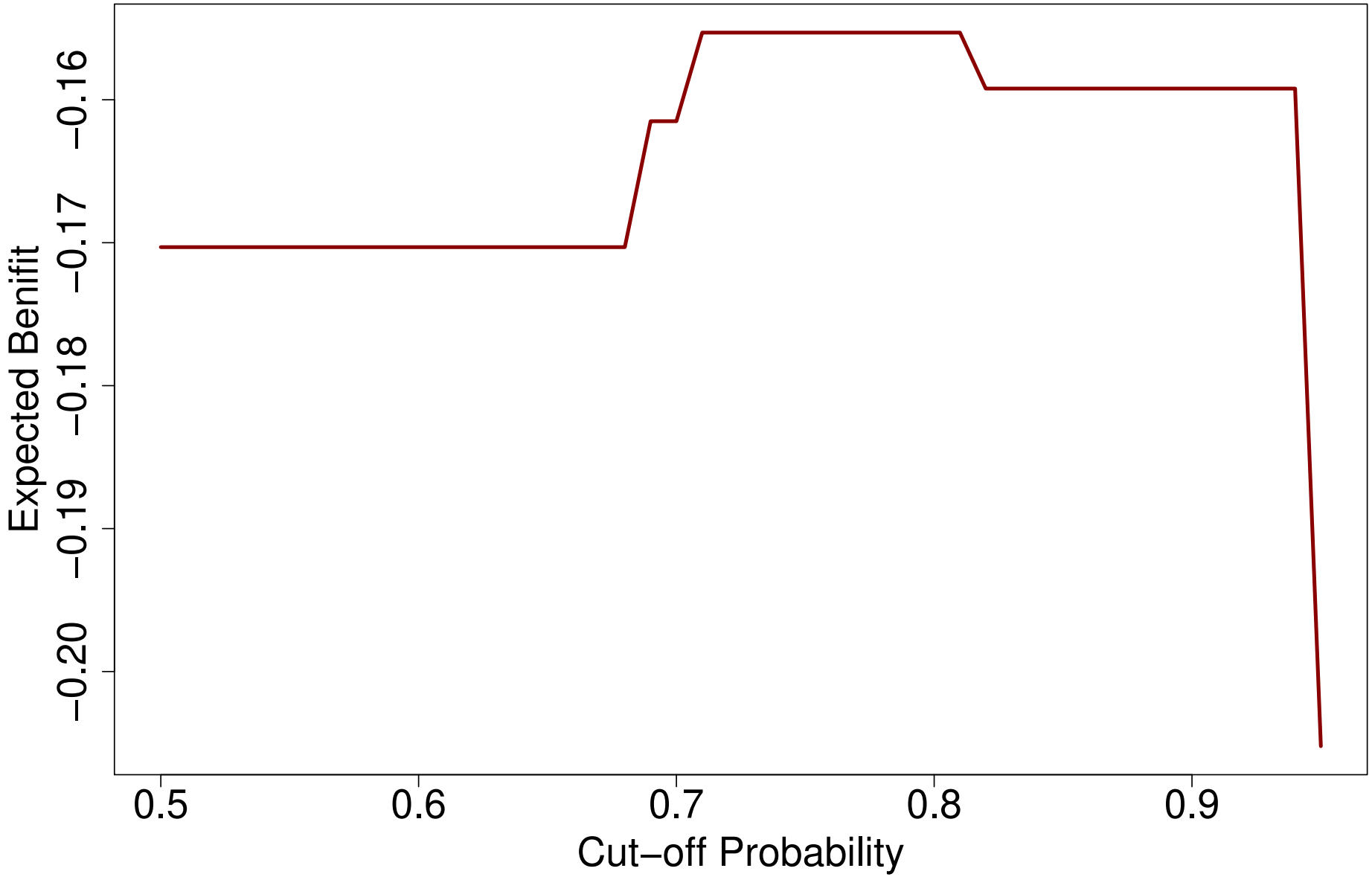
```
> alpha=seq(0.5, 0.95, 0.01)
> eBenifit=vector(length=length(alpha))
> for(i in 1:length(alpha)) {
        predLab=ifelse((predTrain.tree>=alpha[i]), "Yes","No")
        confusion=table(predLab, LABEL[train],deparse.level=2)
        eBenifit[i]=sum(CB*confusion)/sum(confusion)
 }
> plot(alpha, eBenifit, type="l", lwd=3, col="darkred", xlab="Cut-off Probability",
```

```
     ylab="Expected benifit")
> alpha[eBenifit==max(eBenifit)]
    # find value(s) of alpha which maximize expected benifit
 [1] 0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.80 0.81
```

Based on this, we can use the cut-off probability between 0.71 and 0.81. Now we test the performace with 0.5 and 0.75:

```
> predLab=ifelse((predTest.tree>=0.5), "Yes", "No")
> confusion=table(predLab, spam[-train])
> sum(CB*confusion)/sum(confusion)
[1] -0.2363281
> predLab=ifelse((predTest.tree>=0.75), "Yes", "No")
> confusion=table(predLab, spam[-train])
> sum(CB*confusion)/sum(confusion)
[1] -0.2063802
```

With the cut-off probability at 0.75, the expected cost is lower than that at 0.5.


Expected Benifit
−0.16
−0.17
−0.18
−0.19
−0.20
0.5
0.6
0.7
0.8
0.9
Cut−off Probability

To calculate the areas under curves,

```
> performance(prediction.treeTrain, measure="auc")@y.values
[[1]]
[1] 0.9523803  # area under curve for traning sample
> performance(prediction.treeTest, measure="auc")@y.values
[[1]]
[1] 0.9374082 # area under curve for testing sample
```

Now we construct the K-NN classifiers, with $K = 3$ or 5, based on the training data, and then check the performance on the testing data. We also compare them with the tree classifier obtained above.

We only use the variables selected in the tree model to define the distances.

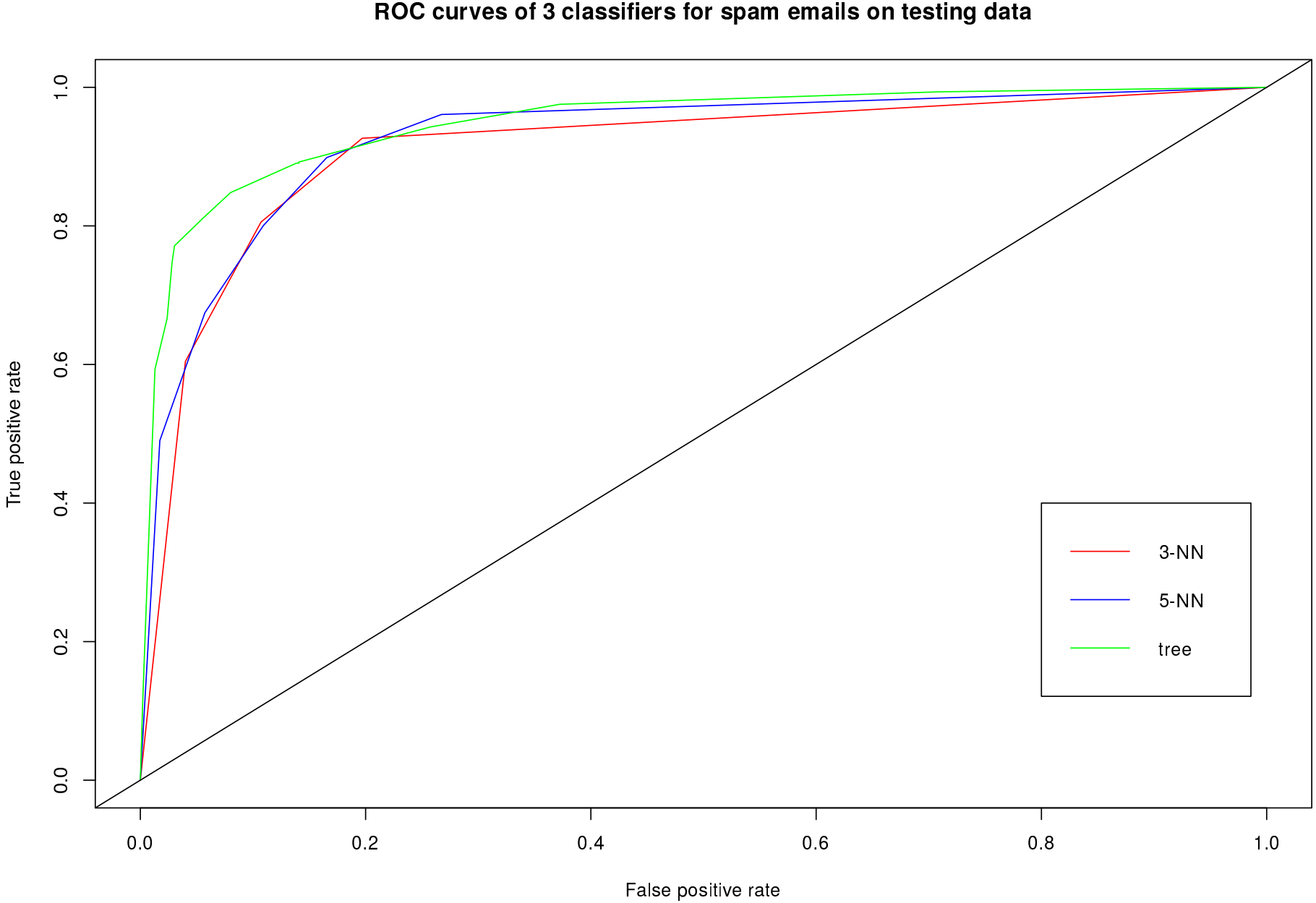
```
> library(dplyr)
> X=select(spamData1, X..4, X..3, remove, hp, CAPMAN, our, CAPAVE, free, edu)
> dim(X)
[1] 4601    9
> Xtrain=X[train,]
> Xtest=X[-train,]
> dim(Xtrain)
```

```
[1] 3065    9
> dim(Xtest)
[1] 1536    9
> summary(Xtrain)
      X..4              X..3              remove              hp          
 Min.   :0.00000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.0000  
 1st Qu.:0.00000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.0000  
 Median :0.00000   Median : 0.0000   Median :0.0000   Median : 0.0000  
 Mean   :0.07362   Mean   : 0.2652   Mean   :0.1099   Mean   : 0.5281  
 3rd Qu.:0.05000   3rd Qu.: 0.3060   3rd Qu.:0.0000   3rd Qu.: 0.0000  
 Max.   :5.30000   Max.   :19.1310   Max.   :7.2700   Max.   :20.8300  
     CAPMAN              our              CAPAVE              free          
 Min.   :   1.00   Min.   :0.0000   Min.   :   1.000   Min.   : 0.0000  
 1st Qu.:   6.00   1st Qu.:0.0000   1st Qu.:   1.571   1st Qu.: 0.0000  
 Median :  15.00   Median :0.0000   Median :   2.250   Median : 0.0000  
 Mean   :  53.02   Mean   :0.3156   Mean   :   5.081   Mean   : 0.2504  
 3rd Qu.:  43.00   3rd Qu.:0.3800   3rd Qu.:   3.657   3rd Qu.: 0.0900  
 Max.   :9989.00   Max.   :9.0900   Max.   :1021.500   Max.   :20.0000  
      edu         
 Min.   : 0.0000  
 1st Qu.: 0.0000  
 Median : 0.0000  
 Mean   : 0.1737  
 3rd Qu.: 0.0000  
 Max.   :10.0000
```

Since data are sparse (i.e. many zeros), we use the correlation based distance measure $1 - \text{Corr}$.

```
> D=-cor(t(Xtest), t(Xtrain))+1 # correlation based distances between rows of Xtest
         # and rows of Xtrain, cor(X1, X2) returns correlations between
         # columns of two matrices X1, X2. Hence transpose
> dim(D)
> [1] 1536 3065
> inDex=matrix(nrow=1536, ncol=5)
> for(i in 1:1536) inDex[i,]=sort.int(D[i,], index.return = T)$ix[1:5]
 # inDex[i, ] contains the row indices of the 5 NN of Xtest[i,] among
 # all rows in Xtrain.   Check ?sort.int
> predKNN=matrix(nrow=1536, ncol=2)
> Y=LABEL[train]
> for(i in 1:1536) predKNN[i,]=c(mean(Y[inDex[i,1:3]]), mean(Y[inDex[i,]]))
> summary(predKNN)
       V1                V2
 Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.3333   1st Qu.:0.2000
 Median :0.3333   Median :0.2000
 Mean   :0.2582   Mean   :0.2939
 3rd Qu.:0.3333   3rd Qu.:0.4000
 Max.   :1.0000   Max.   :1.0000
> pred3=prediction(data.frame(predKNN, predTest.tree), data.frame(LABEL[-train],
    LABEL[-train],LABEL[-train]))
```

```
> roc3=performance(pred3, measure ="tpr", x.measure ="fpr")
> dev.off()
> plot(roc3, col=as.list(c("red","blue","green")), main="ROC curves of 3 classifiers
         for spam emails on testing data")
> legend(0.8, 0.4, c("3-NN","5-NN", "tree"), col=c("red","blue","green"), lty=c(1,1,1))
> abline(0,1)
```


ROC curves of 3 classifiers for spam emails on testing data
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
3-NN
5-NN
tree

Based on the ROC graph, the tree model seems to be the best though three classifiers do not differ that much.

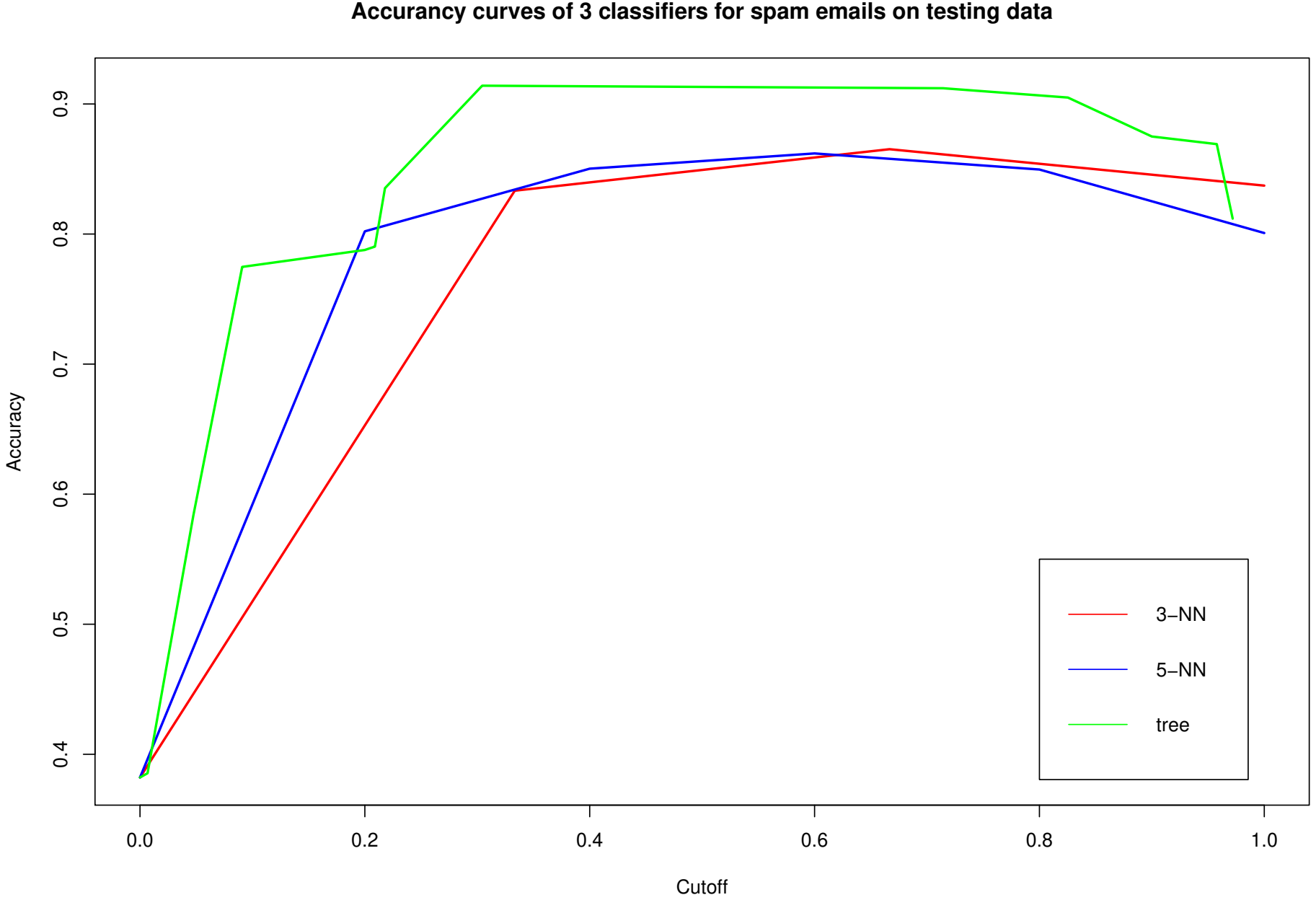
```
> performance(pred3, measure ="auc")@y.values
[[1]]
[1] 0.9107125
[[2]]
[1] 0.9266129
[[3]]
[1] 0.9473464
```

According to AUC, the tree model is the best classifier among the three, while the K-NN classifier with $K = 5$ performs better than that with $K = 3$.

To produce over all accurancy rate (i.e. 1 - misclassification rate) curves

```
> acc3=performance(pred3, measure="acc")
> plot(acc3, lwd=2, col=as.list(c("red","blue","green")), main="Accurancy curves
      of 3 classifiers for spam emails on testing data")
> legend(0.8, 0.55, c("3-NN","5-NN", "tree"), col=c("red","blue","green"),
    lty=c(1,1,1))
> detach(spamData1) # do this upon the completion of a project: a good practice
```

The clean R scripts for this example are collected in the file 'spamEmail.r' available in Moodle.


Accurancy curves of 3 classifiers for spam emails on testing data
Accuracy
0.4
0.5
0.6
0.7
0.8
0.9
0.0
0.2
0.4
0.6
0.8
1.0
Cutoff
3−NN
5−NN
tree