FIT3152 Data analytics. Tutorial 08: Classification models

Pre-tutorial Activity

The "diamonds" data set comes packaged with ggplot2 and contains data about the price of diamonds as well as information on size and the 4 Cs affecting diamond price: carat (size), cut, colour and clarity.

1. Re-create the decision tree from week 07 pre-tutorial activity to predict the price level of diamonds using a 70%/30% split for training and testing datasets. Check prediction accuracy.

```
library(ggplot2)
D = diamonds
price_level = ifelse(D$price >= median(D$price), "High", "Low")
D = cbind(D, as.factor(price_level))
colnames(D)[11] = "price_level"

# partition on to 70% training and 30% test data
set.seed(9999)
train.row = sample(1:nrow(D), 0.7*nrow(D))
D.train = D[train.row,]
D.test = D[-train.row,]

# fit decision tree model to predict price_level
library(tree)
D.fit=tree(price_level ~ .-price, data=D.train)

summary(D.fit)
plot(D.fit)
text(D.fit, pretty = 0)


#Output:
Classification tree:
tree(formula = price_level ~ . - price, data = D.train)
Variables actually used in tree construction:
[1] "y"     "carat" "color"
Number of terminal nodes:  5
Residual mean deviance:  0.2438 = 9205 / 37750
Misclassification error rate: 0.04733 = 1787 / 37758


# create confusion matrix using prediction and check accuracy
D.predict = predict(D.fit, D.test, type = "class")
table(actual = D.test$price_level, predicted = D.predict)

      predicted
actual High  Low
  High 7635  497
  Low   283 7767
```

2. Perform a cross validation test and prune the decision tree based on the misclassification rate. Check the accuracy of prediction and comment on any improvements.

```
#cross validation test
cvtest = cv.tree(D.fit, FUN = prune.misclass)
cvtest

#cvtest output:

$size
[1] 5 4 2 1

$dev
[1]  1859  1859  1866 19190

$k
[1]  -Inf    0     29 17004

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"


#prune using size 4 considering lowest misclassification rate
pruned.Dfit = prune.misclass(D.fit, best = 4)
summary(pruned.Dfit)
plot(pruned.Dfit)
text(pruned.Dfit, pretty = 0)

#summary output
Classification tree:
snip.tree(tree = D.fit, nodes = 2L)
Variables actually used in tree construction:
[1] "y"     "carat" "color"
Number of terminal nodes:  4
Residual mean deviance:  0.2828 = 10680 / 37750
Misclassification error rate: 0.04733 = 1787 / 37758


# check accuracy using the pruned tree
PD.predict = predict(pruned.Dfit, D.test, type = "class")
table(actual = D.test$price_level, predicted = PD.predict

      predicted
actual High  Low
  High 7635  497
  Low   283 7767

#No improvement on the accuracy measure from the confusion matrix.
However, a simpler tree of size 4 is created instead of the previous tree
with size 5.
```

3. Using the pruned decision tree do prediction as probabilities and draw a ROC curve.

```
# do predictions as probabilities and draw ROC
library(ROCR)
PD.pred = predict(pruned.Dfit, D.test, type = "vector")
```

```
# computing a simple ROC curve (x-axis: fpr, y-axis: tpr)
# labels are actual values, predictors are probability of class low
pred <- prediction( PD.pred[,2], D.test$price_level)
perf <- performance(pred,"tpr","fpr")
plot(perf)
abline(0,1)
```

4. Create a lift chart using the pruned decision tree output.

```
#create a lift chart
dlift = performance(pred, "lift")
plot(dlift)
```

Tutorial Activities

Topics Covered:

- Improving the decision tree by Cross Validation and Pruning
- Naïve Bayes classification
- Classifier performance evaluation using ROC charts and Lift
- *Ensemble classifiers – this part will be done in Tutorial 9

References: Introduction to Data Mining, Tan, Steinbach and Kumar (available from Hargrave-Andrew library); An Introduction to Statistical Learning with applications in R (Springer Texts in Statistics), James, Witten, Hastie and Tibshirani, Chapter 8 (available on-line from Monash Library); Reference Manuals to each of the packages used (listed below), available from CRAN.

1 Work through the examples in the lecture slides. For the examples using R you will need to download and install the packages in the script below.

```
# set working directory to desktop
# setwd("~/Desktop")
# clean up the environment before starting
rm(list = ls())
install.packages("tree")
library(tree)
install.packages("e1071")
library(e1071)
install.packages(("ROCR"))
library(ROCR)
install.packages("rpart")
library(rpart)
```

2 Naïve Bayes Classification

ID	Colour	Size	Teeth	IsFriendly
1	red	medium	yes	no
2	blue	big	no	yes
3	green	medium	no	no
4	green	small	yes	no
5	blue	big	yes	yes
6	blue	small	yes	yes
7	red	small	no	yes
8	red	medium	no	yes
9	blue	medium	yes	yes
10	green	small	no	no

Given the Aliens data in the table above, use Naïve Bayes classification to predict whether or not the following aliens are friendly.

ID	Colour	Size	Teeth	IsFriendly
11	red	big	yes	?
12	green	big	yes	?
13	blue	small	no	?

Calculate the classification confidence (probability) for each of the 3 cases to be classified as friendly.

```
# To come.
```

Alien 11				**Alien 13**		
YES	6/10*2/6*2/6*3/6	0.033		YES	6/10*4/6*2/6*3/6	0.067
NO	4/10*1/4*0*2/4	0.000		NO	4/10*0	0.000
Alien 12						
YES	6/10*0	0.000				
NO	4/10*3/4*0	0.000				

3 Creating ROC Charts

The Aliens table below provides information about several aliens including a class attribute, IsFriendly. A decision tree has been used to classify the data and obtained confidence values for IsFriendly= “yes”.

a) Using this information, create a ROC chart for the Alien data.

Remember ROC charts graph TPR vs FPR for varying confidence thresholds. You can use the tables below the data table to assist you with the calculations.

b) What does this ROC chart tell you about the classifier?

c) What is the AUC value and what does this tell you about the classifier?

ID	Colour	Size	Teeth	IsFriendly	Confidence (IsFriendly 'yes')
1	red	medium	yes	no	0.7
2	blue	big	no	yes	0.9
3	green	medium	no	no	0.4
4	green	small	yes	no	0.1
5	blue	big	yes	yes	0.9
6	blue	small	yes	yes	0.8
7	red	small	no	yes	0.3
8	red	medium	no	yes	0.6
9	blue	medium	yes	yes	0.7
10	green	small	no	no	0.6

T = 0.1	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	6	0
Labels	Class = 0	4	0
TPR =	1	FPR =	1

T = 0.2	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	6	0
Labels	Class = 0	3	1
TPR =	1	FPR =	0.75

T = 0.3	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	6	0
Labels	Class = 0	3	1
TPR =	1	FPR =	0.75

T = 0.4	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	5	1
Labels	Class = 0	3	1
TPR =	0.83	FPR =	0.75

T = 0.5	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	5	1
Labels	Class = 0	2	2
TPR =	0.83	FPR =	0.5

T = 0.6	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	5	1
Labels	Class = 0	2	2
TPR =	0.83	FPR =	0.5

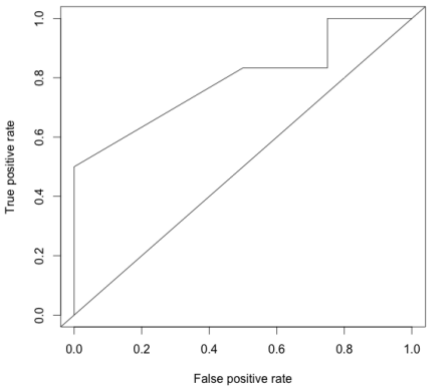
T = 0.7	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	4	2
Labels	Class = 0	1	3
TPR =	0.66	FPR =	0.25

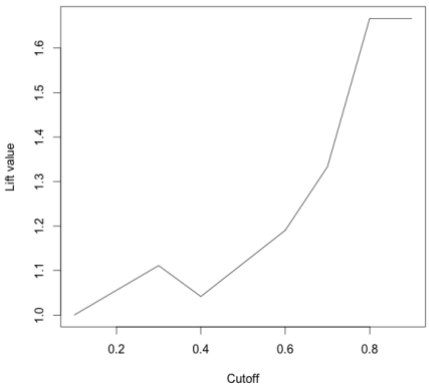
T = 0.8	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	3	3
Labels	Class = 0	0	4
TPR =	0.5	FPR =	0.0

T = 0.9	Predicted Class Labels		
Actual		Class = 1	Class = 0
Class	Class = 1	2	4
Labels	Class = 0	0	4
TPR =	0.33	FPR =	0

Is Friendly	Conf.	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	FPR	TPR
no	0.1	yes	no	no	no	no	no	no	no	no	no	1.00	1.00
yes	0.3	yes	yes	yes	no	no	no	no	no	no	no	0.75	1.00
no	0.4	yes	yes	yes	yes	no	no	no	no	no	no	0.75	1.00
yes	0.6	yes	yes	yes	yes	yes	yes	no	no	no	no	0.75	0.83
no	0.6	yes	yes	yes	yes	yes	yes	no	no	no	no	0.50	0.83
no	0.7	yes	yes	yes	yes	yes	yes	yes	no	no	no	0.50	0.83
yes	0.7	yes	yes	yes	yes	yes	yes	yes	no	no	no	0.25	0.66
yes	0.8	yes	yes	yes	yes	yes	yes	yes	yes	no	no	0.00	0.50
yes	0.9	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	0.00	0.33
yes	0.9	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	0.00	0.00

```
# adapting the code from the lecture notes
# computing a simple ROC curve (x-axis: fpr, y-axis: tpr)
pconfidence = c(0.7, 0.9, 0.4, 0.1, 0.9, 0.8, 0.3, 0.6, 0.7, 0.6)
plabels = c(0, 1, 0, 0, 1, 1, 1, 1, 1, 0)

# transform the inputs into a prediction object
cpred <- prediction(pconfidence, plabels)

# calculate the performance functions
cperf <- performance(cpred,"tpr","fpr")
 plot(cperf)
 abline(0,1)
# calc auc
cauc = performance(cpred, "auc")
print(as.numeric(cauc@y.values))

# calculate and plot lift
clift = performance(cpred, "lift")
 plot(clift)
 print(clift@y.values)
```





4 Creating a Lift chart using the Aliens data

If you select an alien at random, what is the chance it will be friendly?

a. Using the data provided, what is the value of the Lift if you choose an alien from the subset which the classifier is at least 80% confident of?

b. Now sketch a Lift chart, using steps of 20% (i.e. 80%, 60%, 40%, 20%)

See Above

5 Fitting a tree, cross validation and pruning in R

The Zoo data set "zoo.data.csv" contains data relating to seven classes of animals. Construct a decision tree using 70% training and 30% test data. Note that you will need to make sure the model knows that "type" is a factor. Make a table showing the actual vs predicted classifications using your test data. Calculate the accuracy of your model (by calculating correct classifications/all classifications). Are there any classes that are particularly difficult to predict?

Using cross validation and the "cv.tree" function see whether or not you are able to create a more accurate tree by pruning. Make a prediction with the new tree and compare its accuracy with your original tree.

```
#read in data
zoo=read.csv("zoo.data.csv")
zoo$type=factor(zoo$type)

#partition on to training and test data
set.seed(9999)
train.row = sample(1:nrow(zoo), 0.7*nrow(zoo))
z.train = zoo[train.row,]
z.test = zoo[-train.row,]

#fit tree model on training data
z.fit=tree(type~.-animal_name, data = z.train)
print(summary(z.fit))
plot(z.fit)
text(z.fit, pretty=0)
```

```
#test accuracy
z.predict = predict(z.fit, z.test, type = "class")
t1=table(predicted = z.predict, actual = z.test$type)
print(t1)

#cross valdiation and pruning
test.fit=cv.tree(z.fit, FUN=prune.misclass)
print(test.fit)

prune.zfit = prune.misclass(z.fit, best=5)
print(summary(prune.zfit))
plot(prune.zfit)
text(prune.zfit, pretty=0)

#test accuracy after pruning
zp.predict = predict(prune.zfit, z.test, type = "class")
t2=table(predicted = zp.predict, actual = z.test$type)
print(t2)
print(t1)
```

6 Implementing Naïve Bayes classification in R

Again, use the Zoo data set split into a 70% training set and 30% test set. Fit a Naïve Bayes classifier. Compare the difference in accuracy between using the decision tree classifier (in the previous question) and the Naïve Bayes classifier. Does Naïve Bayes perform better on some classes than others?

```
# #read in data
zoo=read.csv("zoo.data.csv")
zoo$type=factor(zoo$type)


#partition on to training and test data
set.seed(9999)
train.row = sample(1:nrow(zoo), 0.7*nrow(zoo))
z.train = zoo[train.row,]
z.test = zoo[-train.row,]

#fit model
z.model=naiveBayes(type~.-animal_name, data = z.train)


#test accuracy
zn.predict = predict(z.model, z.test)
tn=table(predicted = zn.predict, actual = z.test$type)
print(tn)
```

7 The Japanese credit data “JapaneseCredit.csv” has 690 records relating to credit card applications, and whether they were approved or not. All attribute names and values have been anonymised. Ref. http://archive.ics.uci.edu/ml/datasets/Japanese+Credit+Screening

There are a variety of attribute types: continuous, and nominal (some with a small number of different values, and some with large). There are also some missing values that have been indicated as NA. The class value indicates “+” or “-” indicating whether or not a credit applicant was approved.

(a) Split the data into a 70% training and a 30% test set and create a classification model using each of the following techniques in turn:

- Decision Tree
- Naïve Bayes

(b) Using the test data, classify each of the test cases into "+" or "-". Create a confusion matrix for each and report the accuracy of each model.

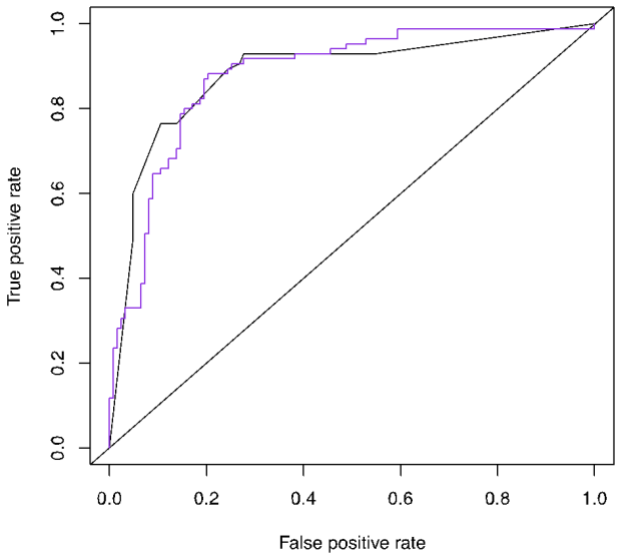
(c) Now, calculate the confidence of predicting a "+" outcome for each of the test cases and construct an ROC curve for each classifier. You should be able to plot all the curves on the same axis. Use a different colour for each classifier.

What does the ROC curve tell you about each of the classifiers. Is there a single "best" classifier?

(d) Examining each of the models, determine the most important variables in predicting whether or not an applicant would be granted a loan.

```
# JC <- read.csv("JapaneseCredit.csv"", stringsAsFactors = T)
set.seed(9999) #random seed
train.row = sample(1:nrow(JC), 0.7*nrow(JC))
JC.train = JC[train.row,]
JC.test = JC[-train.row,]

# Calculate a decision tree
JC.tree = tree(Class ~., data = JC.train)
#print(summary(JC.tree))
plot(JC.tree)
text(JC.tree, pretty = 0)

# do predictions as classes and draw a table
JC.predtree = predict(JC.tree, JC.test, type = "class")
t1=table(Predicted_Class = JC.predtree, Actual_Class = JC.test$Class)
cat("\n#Decision Tree Confusion\n")
print(t1)

# do predictions as probabilities and draw ROC
JC.pred.tree = predict(JC.tree, JC.test, type = "vector")
# computing a simple ROC curve (x-axis: fpr, y-axis: tpr)
# labels are actual values, predictors are probability of class
JCDpred <- prediction( JC.pred.tree[,2], JC.test$Class)
JCDperf <- performance(JCDpred,"tpr","fpr")
plot(JCDperf)
abline(0,1)



#########################################################################

# Calculate naive bayes
JC.bayes = naiveBayes(Class ~. , data = JC.train)
JC.predbayes = predict(JC.bayes, JC.test)
t2=table(Predicted_Class = JC.predbayes, Actual_Class = JC.test$Class)
cat("\n#NaiveBayes Confusion\n")
print(t2)
```

```
# outputs as confidence levels
JCpred.bayes = predict(JC.bayes, JC.test, type = 'raw')
JCBpred <- prediction( JCpred.bayes[,2], JC.test$Class)
JCBperf <- performance(JCBpred,"tpr","fpr")
plot(JCBperf, add=TRUE, col = "blueviolet")
```



8 Some of the performance measures mentioned or covered in the lecture and tutorial are: accuracy, precision, recall, true positive rate (TPR), false positive rate (FPR), sensitivity, specificity, AUC, lift.

(a) Write the formula or a simple explanation for how you calculate each.

(b) Write a simple explanation (or as simple as possible) of what each of these measures indicates about the classifier.

```
Accuracy: Number of correct predictions / Number of all predictions.
Probability of a prediction being correct.

Precision: Number of correct positive predictions / Number of all positive
      predictions.
Probability of a positive prediction being correct.  Basically, quantifies
      the accuracy of positive predictions of the classifier.

Recall: True Positives / True Positives + False Negatives.
Probability of a negative prediction being correct. Basically, quantifies
      the accuracy of negative predictions of the classifier.

Sensitivity (TPR):  Number of correctly predicted positive examples /
      Number of positive examples.
If we randomly select a positive example and make a prediction,
      probability that prediction will be correct. Basically, quantifies
      the ability of the classifier to identify positive examples
      correctly.

Specificity (TNR):Number of correctly predicted negative examples / Number
      of negative examples
```

If we randomly select a negative example and make a prediction, probability that prediction will be correct. Basically, quantifies the ability of the classifier to identify negative examples correctly.

True Positive Rate (TPR):

False Positive Rate (FPR): :Number of incorrectly predicted negative examples / Number of negative examples
If we randomly select a negative example and make a prediction, the probability that prediction will be incorrect. Basically, quantifies the inability of the classifier to identify negative examples correctly.

Area Under Curve (AUC): The area under ROC curve. Doesn't have a simple formula instead requires producing a ROC curve. Quantifies the ability of a classifier to differentiate between positive and negative examples over all confidence levels.