Colorectal Cancer Prediction, Using Gene Expression Data

Santhosh Subramanian

1 Introduction

This project attempts to create a prediction model to detect colorectal cancer using gene expression.

As part of the Cancer research conducted at Princeton University, colorectal cancer cells are analyzed with genechip microarrays for differentially expressed genes and this data is publically available at http://genomics-pubs.princeton.edu/oncology/ (http://genomics-pubs.princeton.edu/oncology/).

The 'Notterman Carcinoma Data' data set from princeton research is used in this project for prediction modelling of colorectal cancer from gene expression.

2 About the Data

The notterman data set at http://genomics-pubs.princeton.edu/oncology/Data/CarcinomaNormalDatasetCancerResearch.txt (http://genomics-pubs.princeton.edu/oncology/Data/CarcinomaNormalDatasetCancerResearch.txt) is a data.frame containing 18 paired samples (18 tumor and 18 normal samples) of 7457 gene expression values from Notterman et al. (2001).

1. Variable "Accession Number" is the gene expression accession number.
2. Variable "Description" describes the gene expression.
3. The remaining variables called "Tumor XX" or "Normal XX" are samples with the associated gene expression information. The title itself indicates if the sample was a Tumor or a Normal sample.
4. Variable "T-Test Tumor vs Normal" provides the t-test p-value for each gene expression for tumor vs normal sample.

In the upcoming sections, data is acquired, processed and the prediction model built.

```
#Load all the necessary libraries for this project and set the working directory in hidden mode.

library (plyr)
library (caret)
library (klaR)
library (MASS)
library (pROC)
```

3 Download and Read the Data

Data is directly downloaded from the source URL http://genomics-pubs.princeton.edu/oncology/Data/CarcinomaNormalDatasetCancerResearch.txt (http://genomics-pubs.princeton.edu/oncology/Data/CarcinomaNormalDatasetCancerResearch.txt) and stored as CarcinomaNormalDatasetCancerResearch.txt.

```
fileName = "CarcinomaNormalDatasetCancerResearch.txt"
if (!file.exists (fileName))
    download.file (url = "http://genomics-pubs.princeton.edu/oncology/Data/CarcinomaNormalDatasetCancerRe
search.txt", destfile = fileName)
```

The gene expression data is read. While reading, the following are performed:

1. The first 8 lines do not contain the gene expression data and are hence skipped.
2. The name of the columns are explicitely specified, for ease of use and readability.
3. The data values are separated by tabs and this is specified while reading. However, there are few empty double tabs and to overcome these, variables Dummy1-5 are used in the column names.
4. As the gene expression description contains '#', which are typically used as a comment character, this can lead to data being incorrectly read. This overcome by overriding the comment character.

Data is read into a data frame called **data**.

```
colNames = c ("AccessionNumber", "DescriptionSample", "Dummy1",
              "Tumor27", "Tumor29", "Tumor34", "Tumor28", "Tumor35",
              "Tumor8", "Tumor3", "Tumor9", "Tumor4", "Tumor32",
              "Tumor39", "Tumor10", "Tumor33", "Tumor5", "Tumor11",
              "Tumor6", "Tumor12", "Tumor40", "Normal27", "Normal29",
              "Normal34", "Normal28", "Normal35", "Normal8", "Normal3",
              "Normal9", "Normal4", "Normal32", "Normal39", "Normal10",
              "Normal33", "Normal5", "Normal11", "Normal6", "Normal12",
              "Normal40", "TTestTumorVSNormal",
              "Dummy2", "Dummy3", "Dummy4", "Dummy5")

data = read.table (fileName, header = FALSE, stringsAsFactors = FALSE,
                   quote = "", skip = 8, sep = "\t", comment.char = "",
                   col.names = colNames)
```

Below is a glimpse of the data and the class of each variable. All the variables are read in the proper class type.

```
str (data)
```

```
## 'data.frame':    7466 obs. of  44 variables:
##  $ AccessionNumber   : chr  "X53416" "M83670" "X90908" "M97496" ...
##  $ DescriptionSample : chr  "Human mRNA for actin-binding protein (filamin) (ABP-280)" "\"\"\"Human ca
rbonic anhydrase IV mRNA, complete cds\"\"\"" "H.sapiens mRNA for I-15P (I-BABP) protein" "\"\"\"Homo sap
iens guanylin mRNA, complete cds\"\"\"" ...
##  $ Dummy1            : logi  NA NA NA NA NA NA ...
##  $ Tumor27           : int  70 -81 25 10 22 113 36 163 9 25 ...
##  $ Tumor29           : int  108 -30 -7 60 0 24 8 113 -3 9 ...
##  $ Tumor34           : int  75 -1 5 48 6 34 27 35 -1 4 ...
##  $ Tumor28           : int  871 4 14 78 -6 85 65 227 19 22 ...
##  $ Tumor35           : int  -92 -34 14 19 11 -6 27 -8 9 -5 ...
##  $ Tumor8            : int  21 -13 5 11 -18 78 4 143 14 -1 ...
##  $ Tumor3            : int  225 118 -5 175 -40 108 54 272 32 3 ...
##  $ Tumor9            : int  -346 -35 37 42 39 18 14 28 34 -7 ...
##  $ Tumor4            : int  -378 31 -29 105 -54 19 -4 11 16 -9 ...
##  $ Tumor32           : int  475 -79 20 41 10 65 -13 225 17 10 ...
##  $ Tumor39           : int  133 45 -11 72 5 45 34 175 -2 14 ...
##  $ Tumor10           : int  -359 -41 -2 67 20 18 25 8 -8 -3 ...
##  $ Tumor33           : int  465 -104 5 18 10 25 22 14 6 1 ...
##  $ Tumor5            : int  -694 -45 -34 66 -11 28 19 97 12 0 ...
##  $ Tumor11           : int  -357 6 20 68 17 4 -18 4 1 -14 ...
##  $ Tumor6            : int  -343 -38 28 45 36 -1 37 17 2 -6 ...
##  $ Tumor12           : int  -166 -23 20 33 -8 4 -10 -6 -1 -1 ...
##  $ Tumor40           : int  71 -117 -1 8 4 59 5 22 6 0 ...
##  $ Normal27          : int  1314 320 2 1231 3 902 311 1249 259 125 ...
##  $ Normal29          : int  1815 172 -9 700 5 1354 718 1327 161 134 ...
##  $ Normal34          : int  624 275 -15 446 0 461 226 622 119 78 ...
##  $ Normal28          : int  462 379 -21 1128 3 229 180 684 104 108 ...
##  $ Normal35          : int  672 549 -11 1070 0 454 323 793 197 105 ...
##  $ Normal8           : int  -107 127 14 491 25 76 -5 460 75 -10 ...
##  $ Normal3           : int  1079 463 -14 1296 -7 656 282 1536 148 180 ...
##  $ Normal9           : int  -503 641 -8 2261 -27 130 25 395 63 37 ...
##  $ Normal4           : int  632 185 4723 1107 3344 1114 166 2100 72 108 ...
##  $ Normal32          : int  1412 288 -21 731 8 673 298 1033 179 217 ...
##  $ Normal39          : int  1122 416 -7 1559 5 836 376 1164 190 278 ...
##  $ Normal10          : int  644 625 -28 1762 -7 799 273 1599 98 113 ...
##  $ Normal33          : int  1760 320 -8 917 -2 1060 473 1472 226 357 ...
##  $ Normal5           : int  488 564 -29 1303 -9 1236 265 2203 41 83 ...
##  $ Normal11          : int  164 330 -13 721 12 504 154 1381 78 106 ...
##  $ Normal6           : int  1282 116 -4 542 0 1070 344 1903 42 161 ...
##  $ Normal12          : int  68 718 -50 1816 -28 251 38 598 73 41 ...
##  $ Normal40          : int  928 332 4 412 0 681 267 1132 357 187 ...
##  $ TTestTumorVSNormal: num  0 0 0.369 0 0.349 ...
##  $ Dummy2            : logi  NA NA NA NA NA NA ...
##  $ Dummy3            : logi  NA NA NA NA NA NA ...
##  $ Dummy4            : logi  NA NA NA NA NA NA ...
##  $ Dummy5            : logi  NA NA NA NA NA NA ...
```

4 Clean the Data

Data is cleaned by removing the variables Dummy1-5. NAs from the last two rows are also removed.

A dictionary of translation between AccessionNumber and DescriptionSample is created. The variable DescriptionSample is then removed.

```
#Verify that the Dummy1-5 variables are only NAs. This yields all zeros.
#   Commenting out to keep the report concise.
#sum (!is.na (data$Dummy1)); sum (!is.na (data$Dummy2))
#sum (!is.na (data$Dummy3)); sum (!is.na (data$Dummy4))
#sum (!is.na (data$Dummy5))

data$Dummy1 = NULL; data$Dummy2 = NULL; data$Dummy3 = NULL
data$Dummy4 = NULL; data$Dummy5 = NULL

dictionary = subset (data, select = c (AccessionNumber, DescriptionSample))

data$DescriptionSample = NULL

#summary (data)
data = data [-7466, ]
data = data [-7465, ]
```

The data is now clean and ready for transformations.

5 Explore and Transform the Data

Data is explored and transformed as follows:

1. The variable TTestTumorVSNormal consists of the t-test result (p-value) for each gene expression. The t-test tests the null hypothesis that the mean of a gene expression is the same for both sample types - Tumor and Normal.

When the p-value is significant 0.05 or lesser, then the alternate hypothesis that the means are different is true. Which in turn means that the particular gene expression is a good predictor of the sample type - Tumor or Normal.

The below shows that there are 2096 gene expressions that have a significant p-value and would be good predictors of sample type. The non-significant predictors are removed from the dataset.

```
indexOfSigPredictors = which (data$TTestTumorVSNormal <= 0.05, arr.ind = T)
length (indexOfSigPredictors)
```

```
## [1] 2096
```

```
data = data [indexOfSigPredictors, ]
data$TTestTumorVSNormal = NULL
```

2. The outcome (predicted value) is not available as a variable for the modelling. Hence, a new variable called ‘tumor’ specifying if the sample is a tumor sample or not is created based on the samples’ names.

```
#Tumor or Normal
#"Tumor27", "Tumor29", "Tumor34", "Tumor28", "Tumor35",
# "Tumor8", "Tumor3", "Tumor9", "Tumor4", "Tumor32",
# "Tumor39", "Tumor10", "Tumor33", "Tumor5", "Tumor11",
# "Tumor6", "Tumor12", "Tumor40", "Normal27", "Normal29",
# "Normal34", "Normal28", "Normal35", "Normal8", "Normal3",
# "Normal9", "Normal4", "Normal32", "Normal39", "Normal10",
# "Normal33", "Normal5", "Normal11", "Normal6", "Normal12",
# "Normal40"
tumor = c ("Yes", "Yes", "Yes", "Yes", "Yes",
           "Yes", "Yes", "Yes", "Yes", "Yes",
           "Yes", "Yes", "Yes", "Yes", "Yes",
           "Yes", "Yes", "Yes", "No", "No",
           "No", "No", "No", "No", "No",
           "No", "No", "No", "No", "No",
           "No", "No", "No", "No", "No",
           "No")
```

3. The next intended tranformation is to make the samples as rows of data and the gene accession numbers as columns (or predictor variables).
4. Before we do this, we need to check if the AccessionNumbers are indeed unique as they are expected to be. The below shows that they are not and nearly 800 duplicates exist.

```
dim (data)
```

```
## [1] 2096   37
```

```
length (unique (data$AccessionNumber))
```

```
## [1] 1971
```

5. The best way to handle this would be to communicate with the research group to understand and rectify in the best suitable manner. Considering that we cannot do it now, below are the two options available (both yield a similar variable importance upon modelling - not shown in this project for the sake of simplicity)

First choice, calculate the mean for each repeated AccessionNumber and then use the means and eliminate the duplicates. We will call this new data frame **dupRemData**

```
dupRemData =  ddply (data, .(AccessionNumber), summarize,
                     Tumor27 = mean (Tumor27), Tumor29 = mean (Tumor29),
                     Tumor34 = mean (Tumor34), Tumor28 = mean (Tumor28),
                     Tumor35 = mean (Tumor35), Tumor8 = mean (Tumor8),
                     Tumor3 = mean (Tumor3), Tumor9 = mean (Tumor9),
                     Tumor4 = mean (Tumor4), Tumor32 = mean (Tumor32),
                     Tumor39 = mean (Tumor39), Tumor10 = mean (Tumor10),
                     Tumor33 = mean (Tumor33), Tumor5 = mean (Tumor5),
                     Tumor11 = mean (Tumor11), Tumor6 = mean (Tumor6),
                     Tumor12 = mean (Tumor12), Tumor40 = mean (Tumor40),
                    Normal27 = mean (Normal27), Normal29 = mean (Normal29),
                    Normal34 = mean (Normal34), Normal28 = mean (Normal28),
                    Normal35 = mean (Normal35), Normal8 = mean (Normal8),
                    Normal3 = mean (Normal3), Normal9 = mean (Normal9),
                    Normal4 = mean (Normal4), Normal32 = mean (Normal32),
                    Normal39 = mean (Normal39), Normal10 = mean (Normal10),
                    Normal33 = mean (Normal33), Normal5 = mean (Normal5),
                    Normal11 = mean (Normal11), Normal6 = mean (Normal6),
                    Normal12 = mean (Normal12), Normal40 = mean (Normal40))
```

A sanity check is also performed on the duplicate removal by printing the values of AccessionNumber U22055 which did not have a duplicate before and after the transformation. And the values of AccessionNumber X53416 which repeated thrice.

```
dupRemData[dupRemData$AccessionNumber == "U22055", ]
```

```
##      AccessionNumber Tumor27 Tumor29 Tumor34 Tumor28 Tumor35 Tumor8 Tumor3
## 1623          U22055     138      93      62      30      81     38     55
##      Tumor9 Tumor4 Tumor32 Tumor39 Tumor10 Tumor33 Tumor5 Tumor11 Tumor6
## 1623    102     25      47      75      59      62    116      54    130
##      Tumor12 Tumor40 Normal27 Normal29 Normal34 Normal28 Normal35 Normal8
## 1623      70      75       45       16       28       -4       -9      57
##      Normal3 Normal9 Normal4 Normal32 Normal39 Normal10 Normal33 Normal5
## 1623     -28     -13     -51      -73       25      -21       34      21
##      Normal11 Normal6 Normal12 Normal40
## 1623       -4      25        3       19
```

```
data[data$AccessionNumber == "U22055", ]
```

```
##      AccessionNumber Tumor27 Tumor29 Tumor34 Tumor28 Tumor35 Tumor8 Tumor3
## 7459          U22055     138      93      62      30      81     38     55
##      Tumor9 Tumor4 Tumor32 Tumor39 Tumor10 Tumor33 Tumor5 Tumor11 Tumor6
## 7459    102     25      47      75      59      62    116      54    130
##      Tumor12 Tumor40 Normal27 Normal29 Normal34 Normal28 Normal35 Normal8
## 7459      70      75       45       16       28       -4       -9      57
##      Normal3 Normal9 Normal4 Normal32 Normal39 Normal10 Normal33 Normal5
## 7459     -28     -13     -51      -73       25      -21       34      21
##      Normal11 Normal6 Normal12 Normal40
## 7459       -4      25        3       19
```

```
dupRemData[dupRemData$AccessionNumber == "X53416", ]
```

```
##      AccessionNumber Tumor27 Tumor29 Tumor34 Tumor28 Tumor35 Tumor8 Tumor3
## 1757          X53416    77.5   157.5      81   749.5     -44  109.5  243.5
##      Tumor9 Tumor4 Tumor32 Tumor39 Tumor10 Tumor33 Tumor5 Tumor11 Tumor6
## 1757   -194 -199.5     410   168.5  -205.5   416.5   -306    -261 -181.5
##      Tumor12 Tumor40 Normal27 Normal29 Normal34 Normal28 Normal35 Normal8
## 1757   -95.5      94     1212     1757    575.5    428.5      711    77.5
##      Normal3 Normal9 Normal4 Normal32 Normal39 Normal10 Normal33 Normal5
## 1757  1183.5  -244.5   829.5     1253   1093.5      747   1570.5     753
##      Normal11 Normal6 Normal12 Normal40
## 1757      317    1405     93.5      890
```

```
data[data$AccessionNumber == "X53416", ]
```

```
##    AccessionNumber Tumor27 Tumor29 Tumor34 Tumor28 Tumor35 Tumor8 Tumor3
## 1           X53416      70     108      75     871     -92     21    225
## 30          X53416      85     207      87     628       4    198    262
##    Tumor9 Tumor4 Tumor32 Tumor39 Tumor10 Tumor33 Tumor5 Tumor11 Tumor6
## 1    -346   -378     475     133    -359     465   -694    -357   -343
## 30    -42    -21     345     204     -52     368     82    -165    -20
##    Tumor12 Tumor40 Normal27 Normal29 Normal34 Normal28 Normal35 Normal8
## 1     -166      71     1314     1815      624      462      672    -107
## 30     -25     117     1110     1699      527      395      750     262
##    Normal3 Normal9 Normal4 Normal32 Normal39 Normal10 Normal33 Normal5
## 1     1079    -503     632     1412     1122      644     1760     488
## 30    1288      14    1027     1094     1065      850     1381    1018
##    Normal11 Normal6 Normal12 Normal40
## 1       164    1282       68      928
## 30      470    1528      119      852
```

Now the data frame is transposed to make the AccessionNumber the predictor variables. The outcome variable is also attached to the data frame as dependent variable **tumor**.

This new data frame is **geneExpDupAvgData**.

```
row.names (dupRemData) = dupRemData$AccessionNumber
dupRemData$AccessionNumber = NULL

geneExpDupAvgData = data.frame (t(dupRemData))
geneExpDupAvgData$tumor = factor (tumor)
dim (geneExpDupAvgData)
```

```
## [1]   36 1972
```

Second choice, would be to rename the duplicated by appending a _1, _2, and so on. This may be a good choice if a gene expression was incorrectly named. This new AccessionNumber is called AccessionNumber.new.

This new data frame with renamed AccessionNumber is dupRenameData.

Similar to the previous choice, the data is transposed and the independent variable **tumor** added.

This new data frame is **transposedData**.

```
dupRenameData = transform (data,
                    AccessionNumber.new = ifelse (
                        duplicated(AccessionNumber) |
                            duplicated(AccessionNumber, fromLast=TRUE),

                        paste (AccessionNumber,
                               ave (AccessionNumber, AccessionNumber,
                                    FUN=seq_along),
                               sep='_'),
                        AccessionNumber)
                    )
dupRenameData$AccessionNumber = NULL

row.names (dupRenameData) = dupRenameData$AccessionNumber.new
dupRenameData$AccessionNumber.new = NULL

geneExpDupRenamedData = data.frame (t(dupRenameData))
geneExpDupRenamedData$tumor = factor (tumor)
dim (geneExpDupRenamedData)
```

```
## [1]   36 2097
```

Now the data sets geneExpDupAvgData (with duplicates averaged) and transposedData (with duplicated renamed) are ready for modelling.

6 Prediction Model Building

For a dataset such as this where the number of predictor variable are enourmous (here 7000+) compared to the number of observations or samples (here 36), one of the best classifiers would be the **naive bayes' classifier.**

We will use the naive bayes' through the **caret package**, with a cross validation option of **Leave One Out Cross Validation** to overcome overfitting.

For ease of modelling for this project we will use only the **geneExpDupRenamedData** dataset.

6.1 Partition Data into Train and Test

Partitioning the data into training and test should have been the first operation right after downloading the data. Due to the amount of cleaning and transformation needed, it is done here (Highly recommended to be moved up before cleaning and transformation to avoid overfitting).

Data is partitioned at a 70-30 of training-test.

```
set.seed (32523)
inTrain = createDataPartition (y = geneExpDupRenamedData$tumor, p = 0.7,
                               list = FALSE)
myTrain = geneExpDupRenamedData [inTrain, ]
myTest = geneExpDupRenamedData [-inTrain, ]
dim(myTrain); dim(myTest)
```

```
## [1]   26 2097
```

```
## [1]   10 2097
```

The training data is used for modelling and the test data for model verification.

6.2 Preprocessing for Model Building

Scaling and centering is a preprocessing that will be performed, so that not any one gene expression influences the outcome more.

Also, based on the below, several gene expressions are higly correlated. To overcome this, principal component analysis is also performed as a preprocessing step.

(Output is not printed due to it's enormity).

```
corMat = cor (myTrain[, !colnames(myTrain) %in% c("tumor")])
diag (corMat) = 0 #Remove self correlations
#which (abs(corMat) > 0.8, arr.ind = T)
```

```
preProc = c("center", "scale", "pca")
```

6.3 Cross Validation

As mentioned earlier, cross validation is performed to avoid overfitting. Leave One Out is chosen due to the low sample size.

```
trCtrl = trainControl (method = "LOOCV")
```

6.4 Build the Prediction Model

Finally, the model is build using the naive bayes' classifier on the training data with preprocessing options of centering, scaling, PCA and the cross validation option if LOOCV.

```
set.seed (32523)
modelFit = train (tumor ~ ., data = myTrain, trControl = trCtrl,
             preProcess = preProc, method = "nb")

confusionMatrix (myTrain$tumor, predict (modelFit, myTrain))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  13   0
##        Yes  0  13
##
##               Accuracy : 1
##                 95% CI : (0.8677, 1)
##    No Information Rate : 0.5
##    P-Value [Acc > NIR] : 1.49e-08
##
##                  Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##            Sensitivity : 1.0
##            Specificity : 1.0
##         Pos Pred Value : 1.0
##         Neg Pred Value : 1.0
##             Prevalence : 0.5
##         Detection Rate : 0.5
##    Detection Prevalence : 0.5
##       Balanced Accuracy : 1.0
##
##        'Positive' Class : No
##
```

The confusion matrix above shows that the **in-sample prediction error is 0%**, implying that **all the samples in the training dataset have been correctly identified to be tumor or not by the build naive bayes' classifier model**.

6.5 Measuring the Performance of the model

Next we apply the built model to the test dataset.

The below confusion matrix shows that the **out-of-sample prediction error is 10%, with a specificity of 83.3% and sensitivity of 100%. This means that our model has been able to identify all tumor samples correctly, and 83% of the normal samples correctly.**

```
confusionMatrix (myTest$tumor, predict (modelFit, myTest))
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction No Yes
##        No   4   1
##        Yes  0   5
##
##               Accuracy : 0.9
##                 95% CI : (0.555, 0.9975)
##    No Information Rate : 0.6
##    P-Value [Acc > NIR] : 0.04636
##
##                  Kappa : 0.8
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 1.0000
##             Specificity : 0.8333
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4000
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9167
##
##        'Positive' Class : No
##
```

6.6 Variable Importance in Model

Below are the top 20 gene expressions that have been crucial in the classification of tumor in the build model.

```
varImp (modelFit)
```

```
## ROC curve variable importance
##
##   only 20 most important variables shown (out of 2096)
##
##          Importance
## H20426          100
## R61502          100
## X64559          100
## T55741          100
## M36981          100
## U37019          100
## U34994          100
## H57136_1        100
## X52679          100
## M77836          100
## L29254          100
## R36977          100
## Z49269_3        100
## T96548          100
## T48804          100
## T51529          100
## Z50753          100
## T64297          100
## T49732          100
## R08183          100
```

7 Summary

The usage of data science for healthcare and IOT is a wave that will disrupt and make human life better.

This project is one such that will help identify colorectal cancer from microarray gene expression.

The model has been able to perform with an accuracy of 90% on out-of-sample test data with a training performed on a small dataset of 30 samples.

Increasing the sample size is likely to increase the prediction accuracy and there by aid better diagnosis, research and understanding of colorectal cancer.