Block 6: Classification

Huizhi Lin (88112551813)

November 30, 2017

The aim of this project is to build classifiers to discriminate normal from tumor tissue, based on metabolic gene expression patterns. I have used the following tissue (sub)sets: skin, breast and all tissues. First I load all the libraries I need to use in this project.

```
library(rpart)
library(chemometrics)
library(class)
library(tree)
library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.
```

Then I created three data frames for data of skin, breast and all tissues, and inspect the data sets.

```
skin <- read.table("get_normal_vs_tumor2_RAW_Skin.out",sep=" ",header=TRUE)
breast <- read.table("get_normal_vs_tumor2_RAW_Breast.out",sep=" ",header=TRU
E)
all <- read.table("get_normal_vs_tumor_RAW.out",sep=" ",header=TRUE)
dim(skin)

## [1]   72 2562

tail(colnames(skin))

## [1] "SLC23A2" "SLC23A1" "SLC12A6" "KCNE2"   "NAT1"    "tissue"

dim(breast)

## [1]  503 2562

tail(colnames(breast))

## [1] "SLC23A2" "SLC23A1" "SLC12A6" "KCNE2"   "NAT1"    "tissue"

dim(all)

## [1] 2132 2562

tail(colnames(all))

## [1] "SLC23A2" "SLC23A1" "SLC12A6" "KCNE2"   "NAT1"    "tissue"
```

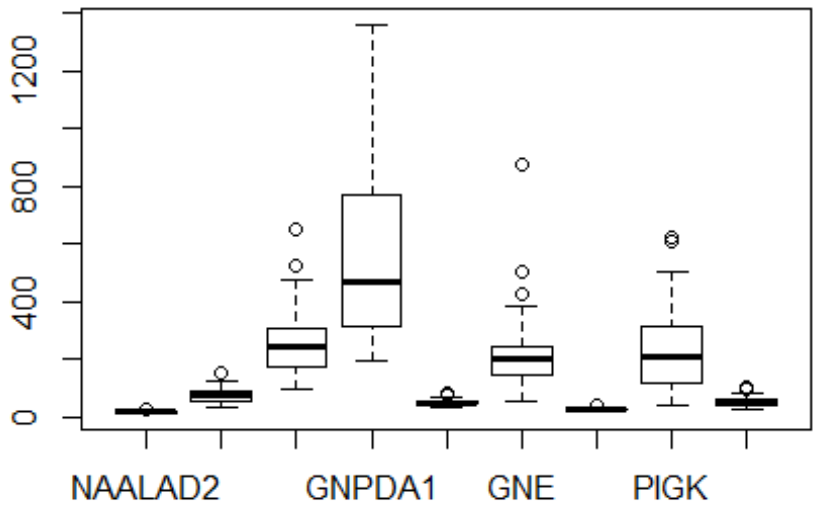
All data frames contains 2562 columns which repersent 2561 genes and the 2562nd column (tissue) contains the indicator of sample labels (normal or tumor). The data frame of skin tiusses has 72 rows (samples: 29 healthy and 43 tumor). The data frame of breast tiusses has 503 rows (samples: 142 healthy and 361 tumor). The data frame of breast tiusses has 2130 rows (samples: 688 healthy and 1444 tumor).

Skin Tissues

k-nearest neighbour

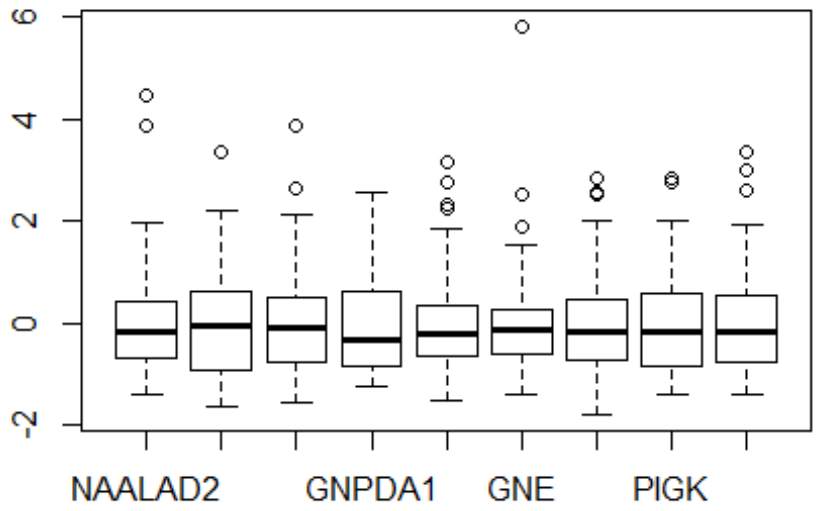
Firstly, I made boxplot of several genes to check their ranges.

```
boxplot(skin[,1:9])
```



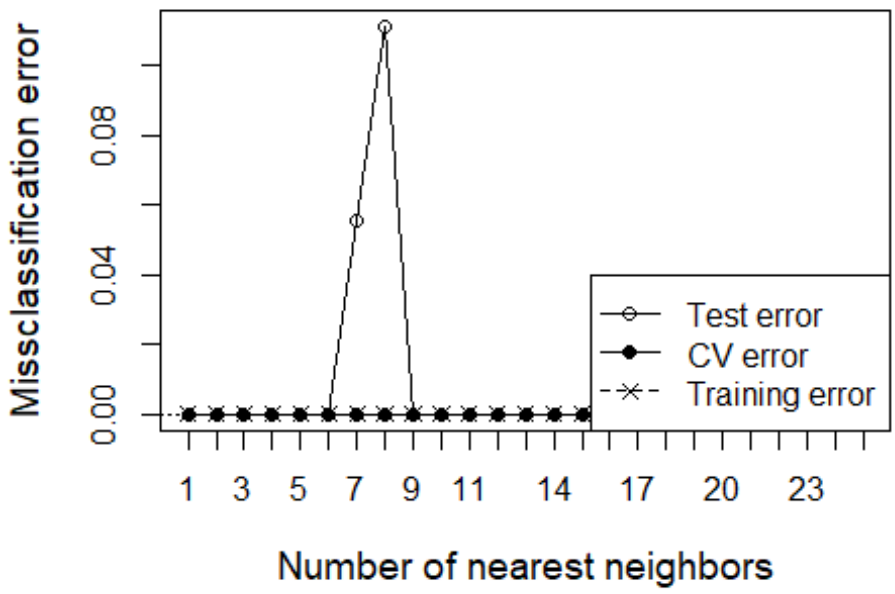
The boxplot shows that there are huge differences in the ranges between different genes. Since the K-nearest neighbour uses Euclidean distance, and it is sensitive to scale of the individual features, I chose to create a new scaled data for the futher analyses.

```
nskin <- skin
nskin[,-2562] <- scale(skin[,-2562])
boxplot(nskin[,1:9])
```



Then I tried cross-validation to find the optimal k

```
ntrain.skin <- round(nrow(skin)*3/4)
set.seed(30)
train.skin <- sample(1:nrow(skin),ntrain.skin)
res.skin <- knnEval(nskin[,-2562],nskin[,2562],train.skin,kfold=10,knnvec=seq
(1,25,1),legpos="bottomright")
```



The plot shows that k=1 is good enough. Therefore I crated the model with k =1, and calculated the accuracy of the results and inspect the confusion matrices on both the trainning set and the test set.

```
# trainning set
preds <- knn(nskin[train.skin,-2562],nskin[train.skin,-2562],nskin[train.skin
,2562],k=1)
length(which(preds==nskin[train.skin,2562]))/length(preds)

## [1] 1

table(preds,nskin[train.skin,2562])

##
## preds    normal tumor
##   normal     23     0
##   tumor       0    31

# test set
preds <- knn(nskin[train.skin,-2562],nskin[-train.skin,-2562],nskin[train.ski
n,2562],k=1)
length(which(preds==nskin[-train.skin,2562]))/length(preds)

## [1] 1

table(preds,nskin[-train.skin,2562])
```

```
##
## preds    normal tumor
##   normal      6     0
##   tumor       0    12
```

Because k=1, the accuracy on the trainning set is 100%. Moreover the result also yields 100% accuracy on the test set.

Logistic regression

First I created the model and summary it to see which probesets are assigned large weight in the model.

```
lr.skin <- glm(tissue~.,data=skin[train.skin,],family="binomial")
summary(lr.skin)

##
## Call:
## glm(formula = tissue ~ ., family = "binomial", data = skin[train.skin,
##     ])
##
## Deviance Residuals:
##  [1]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [24]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [47]  0  0  0  0  0  0  0  0
##
## Coefficients: (2508 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.256e+02  7.691e+06       0        1
## NAALAD2      6.083e+00  3.187e+05       0        1
## NAALADL1    -1.155e+00  3.818e+04       0        1
## ACOT8       -4.404e-02  3.610e+03       0        1
## GNPDA1      -4.335e-02  2.592e+03       0        1
## KCNE3       -5.061e-02  2.937e+04       0        1
## GNE         -7.159e-02  2.629e+03       0        1
## HCN4        -6.908e-01  1.101e+05       0        1
## PIGK        -2.685e-03  7.176e+03       0        1
## SLC17A4     -5.358e-01  6.098e+04       0        1
## ABCC5       -9.257e-02  4.567e+03       0        1
## ABCB6        2.942e-02  4.940e+03       0        1
## ABCC9       -2.383e-01  3.872e+04       0        1
## ABCF2       -6.675e-03  1.219e+04       0        1
## ATP9A       -2.196e-02  2.873e+03       0        1
## KCNK7       -1.016e-02  7.072e+03       0        1
## UST          5.227e-01  1.287e+04       0        1
## ADA          4.129e-01  1.103e+04       0        1
## AASS         6.043e-02  1.351e+04       0        1
## ATP6AP2      1.606e-02  1.387e+03       0        1
## LPCAT3      -6.493e-02  9.013e+03       0        1
## CHST4       -2.119e+00  5.943e+04       0        1
```

```
## SLC25A13     3.145e-03  2.345e+03       0        1
## SLC25A15     1.025e-01  3.157e+04       0        1
## DHRS9       -2.139e-01  1.331e+04       0        1
## ALG3        -2.297e-02  4.004e+03       0        1
## NME6         2.749e-01  2.749e+04       0        1
## DHRS2       -4.531e-02  8.971e+02       0        1
## MFSD10      -7.771e-02  7.215e+03       0        1
## COQ7        -1.109e-01  2.158e+04       0        1
## SLC35B1      2.901e-02  4.226e+03       0        1
## KCNMB2       1.673e+00  1.309e+05       0        1
## GPHN         2.300e-01  1.908e+04       0        1
## SLC17A2      2.161e+00  9.076e+04       0        1
## GLYAT        1.920e+00  1.076e+05       0        1
## ABCC4        7.105e-03  1.357e+04       0        1
## TCIRG1       1.651e-02  6.665e+02       0        1
## B3GALT5     -1.457e+00  8.236e+04       0        1
## RRAGB       -8.685e-01  6.362e+04       0        1
## AKR1A1      -1.356e-02  8.943e+02       0        1
## B3GNT3       3.302e-01  1.275e+04       0        1
## ABCA7        7.536e-01  2.410e+04       0        1
## ABCA9        3.449e-01  1.246e+04       0        1
## ABCA8       -9.683e-03  1.161e+03       0        1
## CACNG3       5.535e-01  2.146e+04       0        1
## CACNG2      -9.094e-02  5.615e+04       0        1
## CDO1        -1.301e-01  7.255e+03       0        1
## BPNT1       -3.512e-01  1.761e+04       0        1
## CEPT1       -9.216e-02  6.296e+03       0        1
## ATP8A1       1.821e-01  1.294e+04       0        1
## PEMT         1.101e-01  8.018e+03       0        1
## ST3GAL6      6.773e-03  1.317e+03       0        1
## CDS1        -4.208e-02  1.970e+03       0        1
## CDIPT        2.493e-02  1.643e+03       0        1
## LYPLA1              NA         NA      NA       NA
.....................
## NAT1                NA         NA      NA       NA
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7.3670e+01  on 53  degrees of freedom
## Residual deviance: 3.1329e-10  on  0  degrees of freedom
## AIC: 108
##
## Number of Fisher Scoring iterations: 25
```

Based on the summary, all the genes until gene “CDIPT” are informative for this model. While a lot genes are not used in this model (with ‘NA’ as Estimate). Then I created a list of all the names of these genes (53 in total, sorted by Estimate).

```
genes.skin <- names(sort(lr.skin$coefficients[-1]))
print(length(genes.skin))
```

```
## [1] 53
```

```
print(genes.skin)
```

```
##  [1] "CHST4"    "B3GALT5"  "NAALADL1" "RRAGB"    "HCN4"     "SLC17A4"
##  [7] "BPNT1"    "ABCC9"    "DHRS9"    "CDO1"     "COQ7"     "ABCC5"
## [13] "CEPT1"    "CACNG2"   "MFSD10"   "GNE"      "LPCAT3"   "KCNE3"
## [19] "DHRS2"    "ACOT8"    "GNPDA1"   "CDS1"     "ALG3"     "ATP9A"
## [25] "AKR1A1"   "KCNK7"    "ABCA8"    "ABCF2"    "PIGK"     "SLC25A13"
## [31] "ST3GAL6"  "ABCC4"    "ATP6AP2"  "TCIRG1"   "CDIPT"    "SLC35B1"
## [37] "ABCB6"    "AASS"     "SLC25A15" "PEMT"     "ATP8A1"   "GPHN"
## [43] "NME6"     "B3GNT3"   "ABCA9"    "ADA"      "UST"      "CACNG3"
## [49] "ABCA7"    "KCNMB2"   "GLYAT"    "SLC17A2"  "NAALAD2"
```

I also calculate the accuracy of the results and inspect the confusion matrices on both the trainning set and the test set.

```
# training set
preds <- predict(lr.skin,newdata=skin[train.skin,],type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
preds <- ifelse(preds>0.5,"tumor","normal")
length(which(preds==skin[train.skin,2562]))/length(preds)
```

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

```
table(preds,skin[train.skin,2562])
```

```
##
## preds    normal tumor
##   normal     23     0
##   tumor       0    31
```

```
# test set
preds <- predict(lr.skin,newdata=skin[-train.skin,],type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
preds <- ifelse(preds>0.5,"tumor","normal")
length(which(preds==skin[-train.skin,2562]))/length(preds)
```

```
## [1] 0.6666667
```

```
table(preds,skin[-train.skin,2562])
```

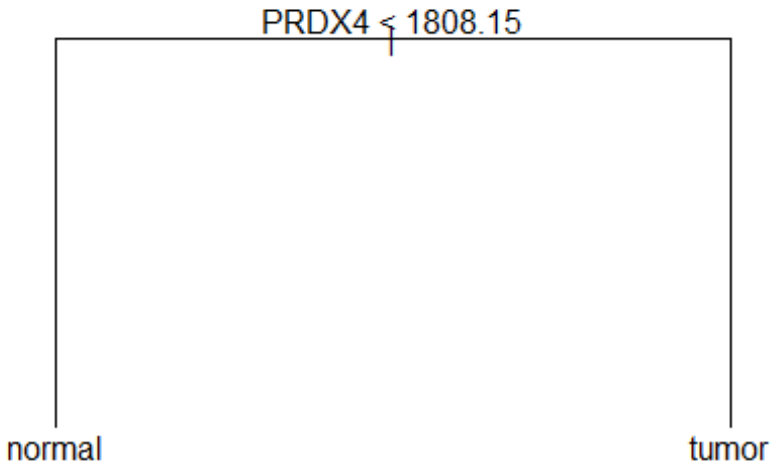
```
##
## preds    normal tumor
##   normal      4     4
##   tumor       2     8
```

The accuracy on the trainning set is 100%. But the accuracy on the test set is only 0.67, and the confusion matrix shows that there are both false positive and false negetive. This means the model is overtrained.

Dicision tree

First I created the decision tree and visualized the tree.

```
tree.skin <- tree(tissue~.,data=skin[train.skin,])
plot(tree.skin)
text(tree.skin)
```



This is a very simple tree and it only has two leaves and one split. If The expression of gene *PRDX4* is large than 1808.15, than the sample is predicted as normal sample, while the sample with the expression of gene *PRDX4* below 1808.15 is predicted as tumor sample.

```
# training set
preds <- predict(tree.skin,newdata=skin[train.skin,],type="class")
length(which(preds==skin[train.skin,2562]))/length(preds)

## [1] 1

table(preds,skin[train.skin,2562])

##
## preds    normal tumor
##   normal     23     0
##   tumor       0    31
```

```r
# test set
preds <- predict(tree.skin,newdata=skin[-train.skin,],type="class")
length(which(preds==skin[-train.skin,2562]))/length(preds)
```

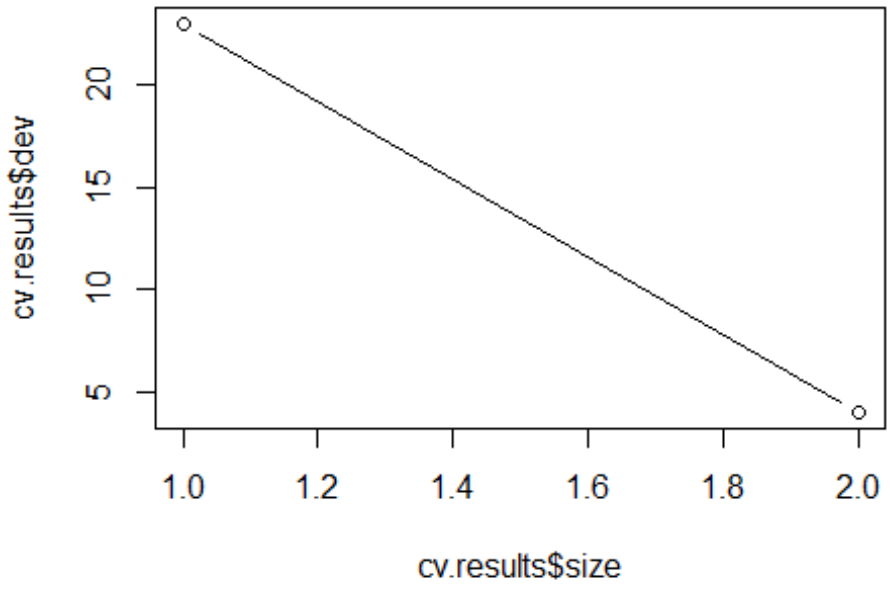
```
## [1] 1
```

```r
table(preds,skin[-train.skin,2562])
```

```
##
## preds    normal tumor
##   normal      6     0
##   tumor       0    12
```

Although the tree is very simple, the accuracy is 100% on both the training set and test set. Although the tree is too simple to be pruned, I still made the plot and it proved that the tree does not need pruning

```r
cv.results <- cv.tree(tree.skin, FUN=prune.misclass)
plot(cv.results$size, cv.results$dev, type="b")
```



Random forest

Lastly, I also created a random forest classifier, and calculated the accuracy.

```r
forest.skin <- randomForest(tissue~.,data=skin[train.skin,])
# accuracy on training data
preds <- predict(forest.skin,newdata=skin[train.skin,],type="class")
length(which(preds==skin[train.skin,2562]))/length(preds)
```

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

```
table(preds,skin[train.skin,2562])
```

```
##
## preds    normal tumor
##   normal     23     0
##   tumor       0    31
```

```
# accuracy on test data
preds <- predict(forest.skin,newdata=skin[-train.skin,],type="class")
length(which(preds==skin[-train.skin,2562]))/length(preds)
```

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

```
table(preds,skin[-train.skin,2562])
```

```
##
## preds    normal tumor
##   normal      6     0
##   tumor       0    12
```

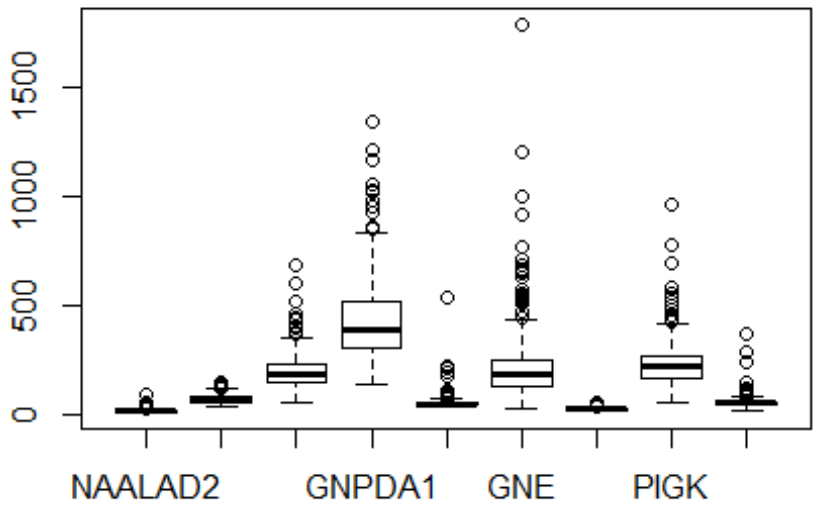
It yields 100% accuracy on both the trainning data and test data. Therefore it is a rather good classifier. For the skin tissue dataset, KNN, dicision tree and random forest clsssifiers are equally good while the logistic regression classifier performs badly.

Breast Tissues

k-nearest neighbour

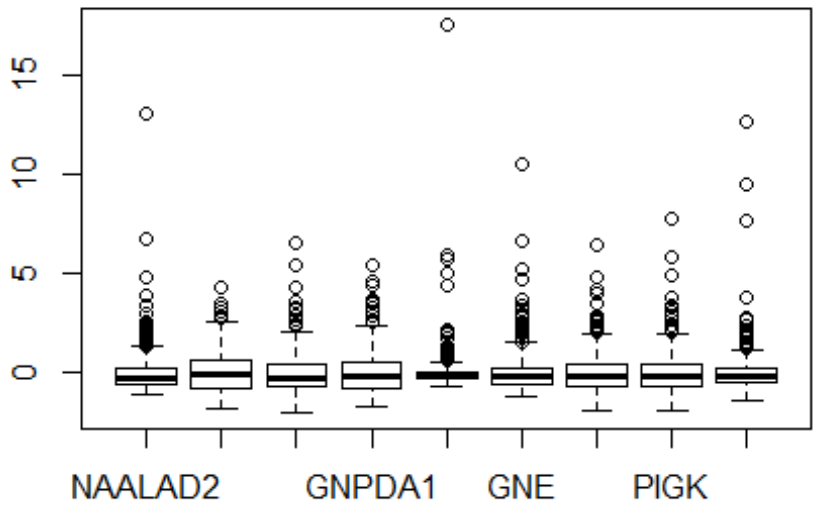
Firstly, I made boxplot of several genes to check their ranges.

```
boxplot(breast[,1:9])
```



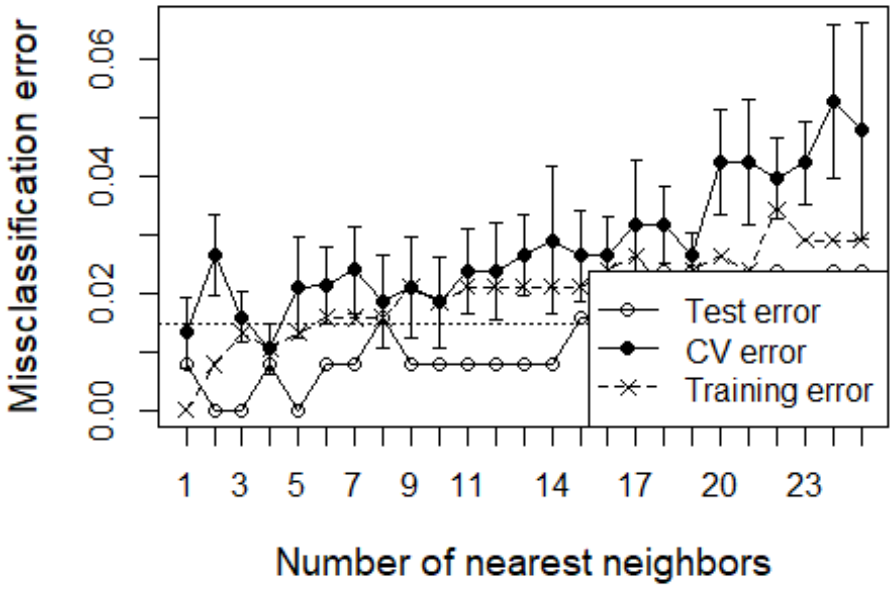
The boxplot shows that there are huge differences in the ranges between different genes. Then I created a scaled data frame.

```
nbreast <- breast
nbreast[,-2562] <- scale(breast[,-2562])
boxplot(nbreast[,1:9])
```



I used cross-validation to find the optimal k.

```
ntrain.breast <- round(nrow(breast)*3/4)
set.seed(30)
train.breast <- sample(1:nrow(breast),ntrain.breast)
res.breast <- knnEval(nbreast[,-2562],nbreast[,2562],train.breast,kfold=10,kn
nvec=seq(1,25,1),legpos="bottomright")
```



The plot shows that the optimal k is 4.

Therefore I crated the model with k =4, and calculated the accuracy of the results and inspect the confusion matrices on both the trainning set and the test set.

```
# trainning set
preds <- knn(nbreast[train.breast,-2562],nbreast[train.breast,-2562],nbreast[
train.breast,2562],k=4)
length(which(preds==nbreast[train.breast,2562]))/length(preds)

## [1] 0.9893899

table(preds,nbreast[train.breast,2562])

##
## preds    normal tumor
##   normal    100     4
##   tumor       0   273

# test set
preds <- knn(nbreast[train.breast,-2562],nbreast[-train.breast,-2562],nbreast
[train.breast,2562],k=4)
length(which(preds==nbreast[-train.breast,2562]))/length(preds)

## [1] 1

table(preds,nbreast[-train.breast,2562])
```

```
##
## preds    normal tumor
##   normal     42     0
##   tumor       0    84
```

The accuracy on the trainning set is 98.9%, there are 4 tumor samples have been predicted as normal. However, althouth I expect the accuracy of the test set is lower, the result also yields 100% accuracy on the test set.

Logistic regression

First I created the model and summary it to see which probesets are assigned large weight in the model.

```
lr.breast <- glm(tissue~.,data=breast[train.breast,],family="binomial")

## Warning: glm.fit: algorithm did not converge

summary(lr.breast)

##
## Call:
## glm(formula = tissue ~ ., family = "binomial", data = breast[train.breast,

##     ])
##
## Deviance Residuals:
##   [1]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##  [24]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##  [47]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##  [70]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##  [93]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [116]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [139]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [162]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [185]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [208]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [231]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [254]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [277]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [300]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [323]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [346]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [369]  0  0  0  0  0  0  0  0  0
##
## Coefficients: (2185 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.290e+02  1.677e+07       0        1
## NAALAD2     -1.117e+00  7.109e+04       0        1
## NAALADL1     4.893e-01  2.328e+04       0        1
## ACOT8       -2.487e-01  1.542e+04       0        1
```

```
## GNPDA1       1.357e-01  8.982e+03       0        1
## KCNE3        8.294e-01  5.894e+04       0        1

.....
## NAT1                NA         NA      NA       NA
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4.3617e+02  on 376  degrees of freedom
## Residual deviance: 2.1872e-09  on   0  degrees of freedom
## AIC: 754
##
## Number of Fisher Scoring iterations: 25
```

Then I created a list of all the names of these genes (53 in total, sorted by Estimate).

```
genes.breast <- names(sort(lr.breast$coefficients[-1]))
print(length(genes.breast))

## [1] 376
```

I also calculate the accuracy of the results and inspect the confusion matrices on both the trainning set and the test set.

```
# training set
preds <- predict(lr.breast,newdata=breast[train.breast,],type="response")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

preds <- ifelse(preds>0.5,"tumor","normal")
length(which(preds==breast[train.breast,2562]))/length(preds)

## [1] 1

table(preds,breast[train.breast,2562])

##
## preds    normal tumor
##   normal    100     0
##   tumor       0   277

# test set
preds <- predict(lr.breast,newdata=breast[-train.breast,],type="response")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

preds <- ifelse(preds>0.5,"tumor","normal")
length(which(preds==breast[-train.breast,2562]))/length(preds)

## [1] 0.4761905
```

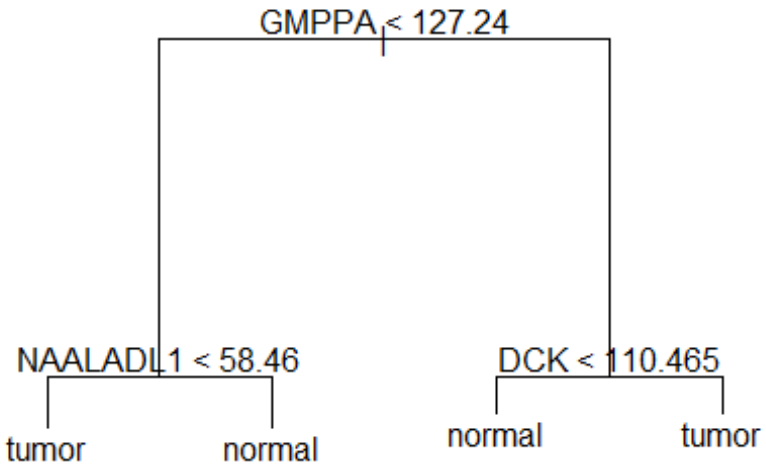
```
table(preds,breast[-train.breast,2562])

##
## preds    normal tumor
##   normal     22    46
##   tumor      20    38
```

The accuracy on the trainning set is 100%. But the accuracy on the test set is only 0.48, and the confusion matrix shows that there are both a lot of false positive and false negetive cases. This means the model is overtrained.

Dicision tree

First I created the decision tree and visualized the tree.

```
tree.breast <- tree(tissue~.,data=breast[train.breast,])
plot(tree.breast)
text(tree.breast)
```



This is a tree with 4 leaves and 3 splits. It splits first based on the expression level of gene *GMPPA*, and later splits based on the expression level of gene *NAALADL1* and *DCK*.

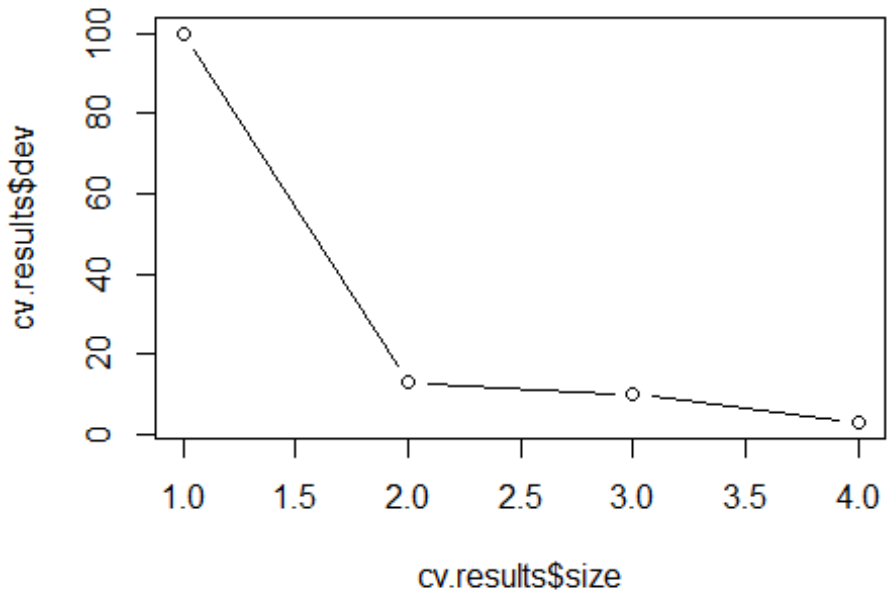
```
# training set
preds <- predict(tree.breast,newdata=breast[train.breast,],type="class")
length(which(preds==breast[train.breast,2562]))/length(preds)

## [1] 0.9973475
```

```
table(preds,breast[train.breast,2562])

##
## preds    normal tumor
##   normal    100     1
##   tumor       0   276

# test set
preds <- predict(tree.breast,newdata=breast[-train.breast,],type="class")
length(which(preds==breast[-train.breast,2562]))/length(preds)

## [1] 0.952381

table(preds,breast[-train.breast,2562])

##
## preds    normal tumor
##   normal     41     5
##   tumor       1    79
```

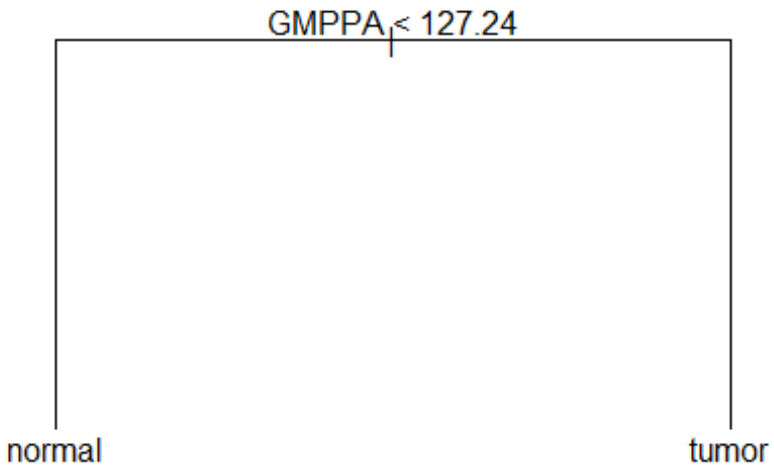
The accuracy on the trainning set is 99.7%. And as expected, the accuracy is slightly lower on the test set (95.2%). The confusion matrix shows that there is one normal sample has been predicted as tumor while 5 tumor samples have been predicted as normal.

Then I used cross-validation to find the optimal level of pruning.

```
cv.results <- cv.tree(tree.breast, FUN=prune.misclass)
plot(cv.results$size, cv.results$dev, type="b")
```



```
smalltree.breast <- prune.misclass(tree.breast,best=2)
plot(smalltree.breast); text(smalltree.breast)
```



Based on the plot form the cross-validaton, the optimal level is 2. Then I also plot the pruned tree, which is slightly simpler than the orignal tree.

Random forest

Lastly, I also created a random forest classifier, and calculated the accuracy.

```
forest.breast <- randomForest(tissue~.,data=breast[train.breast,])
# accuracy on training data
preds <- predict(forest.breast,newdata=breast[train.breast,],type="class")
length(which(preds==breast[train.breast,2562]))/length(preds)

## [1] 1

table(preds,breast[train.breast,2562])

##
## preds    normal tumor
##   normal    100     0
##   tumor       0   277

# accuracy on test data
preds <- predict(forest.breast,newdata=breast[-train.breast,],type="class")
length(which(preds==breast[-train.breast,2562]))/length(preds)

## [1] 0.9920635
```

```
table(preds,breast[-train.breast,2562])
```

```
##
## preds    normal tumor
##   normal     41     0
##   tumor       1    84
```

It yields 100% accuracy on the trainning data and 99.2% accuracy on test data.There is only one normal sample has been predicted as tumor. Therefore it is a rather good classifier. Same as the skin tissue dataset, KNN, dicision tree and random forest clsssifiers are equally good while the logistic regression classifier performs badly.

All Tissues

pre-select a subset

I created a subset from the *all* dataframe with 200 genes. I chose the genes that have large weight in the logistic regression models from the skin and breast dataset.

```
### find genes
intersect(genes.skin,genes.breast)
```

```
##  [1] "CHST4"    "B3GALT5"  "NAALADL1" "RRAGB"    "HCN4"     "SLC17A4"
##  [7] "BPNT1"    "ABCC9"    "DHRS9"    "CDO1"     "COQ7"     "ABCC5"
## [13] "CEPT1"    "CACNG2"   "MFSD10"   "GNE"      "LPCAT3"   "KCNE3"
## [19] "DHRS2"    "ACOT8"    "GNPDA1"   "CDS1"     "ALG3"     "ATP9A"
## [25] "AKR1A1"   "KCNK7"    "ABCA8"    "ABCF2"    "PIGK"     "SLC25A13"
## [31] "ST3GAL6"  "ABCC4"    "ATP6AP2"  "TCIRG1"   "CDIPT"    "SLC35B1"
## [37] "ABCB6"    "AASS"     "SLC25A15" "PEMT"     "ATP8A1"   "GPHN"
## [43] "NME6"     "B3GNT3"   "ABCA9"    "ADA"      "UST"      "CACNG3"
## [49] "ABCA7"    "KCNMB2"   "GLYAT"    "SLC17A2"  "NAALAD2"
```

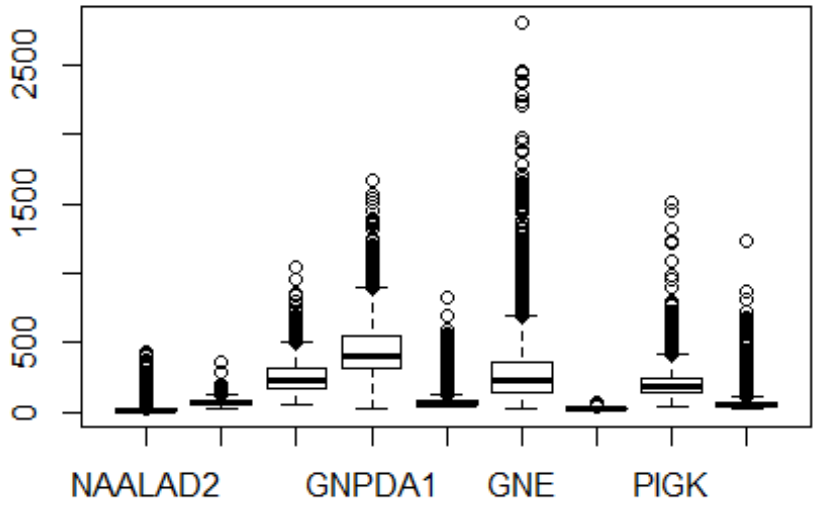
```
idx <- which(genes.skin %in% genes.breast)
genes.extra <- genes.breast[-idx]
genes <- c(genes.skin,genes.extra[1:147])
# create new datafram
idx.col <- which(genes %in% colnames(all))
all.small <- all[,c(idx.col,2562)]
```

As shown above, first I check the genes in both models. I found that all the genes have large weight in the model from the skin dataset are also in the model from the breast dataset. Therefore, I kept those the genes (53) and added anothor 147 genes from breast sample models to the reduced dataframe. I also add the column *tissue* as the last column into the dataframe.

k-nearest neighbour

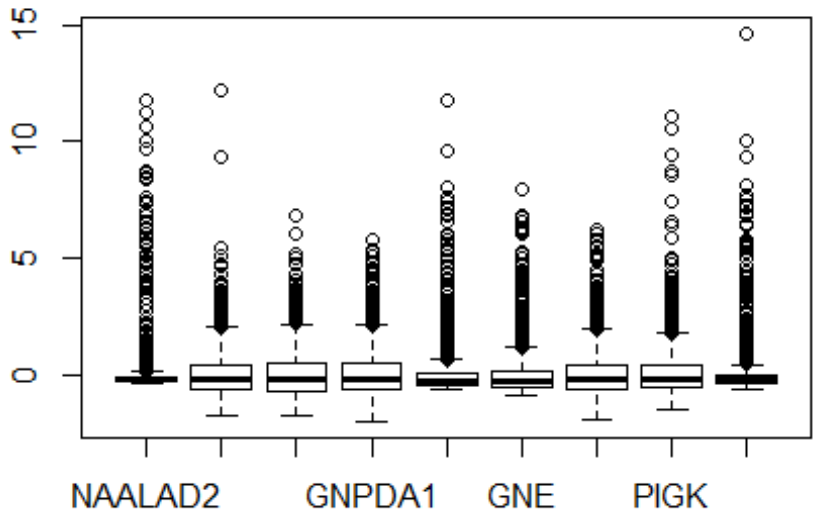
Firstly, I made boxplot of several genes to check their ranges.

```
boxplot(all.small[,1:9])
```



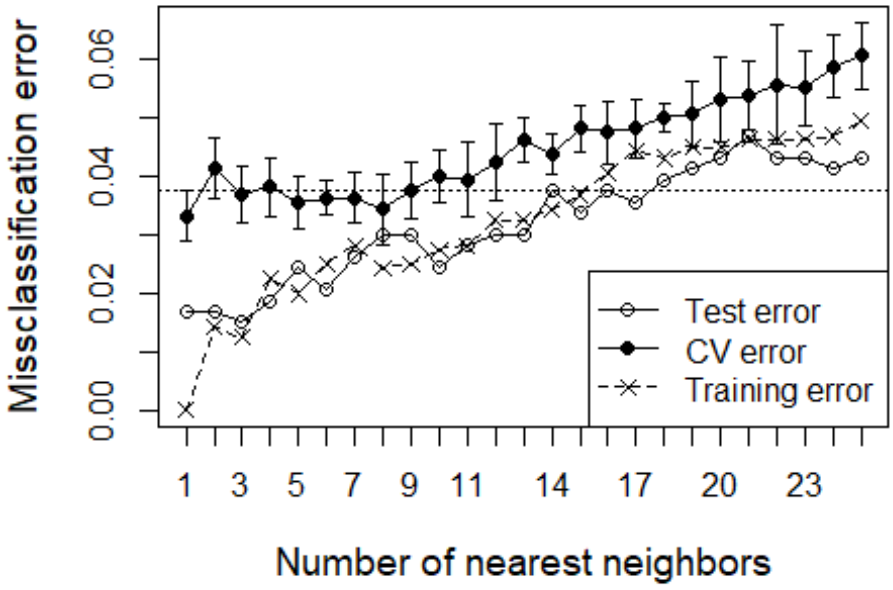
Because there are huge differences in the ranges between genes I created scaled dataset for the futher analyses.

```
nall.small <- all.small
nall.small[,-201] <- scale(all.small[,-201])
boxplot(nall.small[,1:9])
```



Then I used cross-validation to find the optimal k.

```
ntrain.all.small <- round(nrow(all.small)*3/4)
set.seed(30)
train.all.small <- sample(1:nrow(all.small),ntrain.all.small)
res.all.small <- knnEval(nall.small[,-201],nall.small[,201],train.all.small,k
fold=10,knnvec=seq(1,25,1),legpos="bottomright")
```



The plot shows that k=1 is the best chose. Therefore I crated the model with k =1, and calculated the accuracy of the results and inspect the confusion matrices on both the trainning set and the test set.

```
# trainning set
preds <- knn(nall.small[train.all.small,-201],nall.small[train.all.small,-201
],nall.small[train.all.small,201],k=1)
length(which(preds==nall.small[train.all.small,201]))/length(preds)

## [1] 1

table(preds,nall.small[train.all.small,201])

##
## preds    normal tumor
##   normal    502     0
##   tumor       0  1097

# test set
preds <- knn(nall.small[train.all.small,-201],nall.small[-train.all.small,-20
1],nall.small[train.all.small,201],k=1)
length(which(preds==nall.small[-train.all.small,201]))/length(preds)

## [1] 0.9831144

table(preds,nall.small[-train.all.small,201])

##
## preds    normal tumor
```

```
##   normal    180     3
##   tumor       6   344
```

Because k=1, the accuracy on the trainning set is 100%. While the result yields 98.3% accuracy on the test set. There are both normal and tumor samples have been predict wrong.

Logistic regression

First I created the model and summary it to see which probesets are assigned large weight in the model.

```
lr.all.small <- glm(tissue~.,data=all.small[train.all.small,],family="binomia
l")

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(lr.all.small)

##
## Call:
## glm(formula = tissue ~ ., family = "binomial", data = all.small[train.all.
small,
##     ])
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -5.746e-05  -2.100e-08   2.100e-08   2.100e-08   5.926e-05
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.273e+01  3.252e+05   0.000        1
## NAALAD2     -7.997e-02  7.528e+02   0.000        1
## NAALADL1    -1.653e-01  9.921e+02   0.000        1
## ACOT8       -5.048e-02  3.217e+02   0.000        1
## GNPDA1       2.138e-02  1.460e+02   0.000        1
## KCNE3        3.329e-02  5.464e+02   0.000        1
## GNE         -2.136e-02  7.585e+01   0.000        1
## HCN4         3.277e-01  3.029e+03   0.000        1
## PIGK         5.476e-02  2.011e+02   0.000        1
## SLC17A4      4.545e-02  4.077e+02   0.000        1
## ABCC5        5.338e-03  1.106e+02   0.000        1
## ABCB6       -2.508e-02  2.415e+02   0.000        1
## ABCC9       -3.858e-01  1.104e+03   0.000        1
## ABCF2        1.837e-01  6.206e+02   0.000        1
## ATP9A        7.885e-03  5.653e+01   0.000        1
## KCNK7        1.696e-01  4.962e+02   0.000        1
## UST         -5.793e-02  5.859e+02   0.000        1
## ADA         -3.039e-03  6.356e+01   0.000        1
```

```
## AASS          1.210e-01  6.057e+02   0.000        1
## ATP6AP2       1.502e-04  3.509e+01   0.000        1
## LPCAT3       -3.800e-02  3.648e+02   0.000        1
## CHST4        -8.909e-03  9.506e+01   0.000        1

.....
## SLC26A8      -2.284e-02  4.077e+02   0.000        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.9899e+03  on 1598  degrees of freedom
## Residual deviance: 8.3989e-08  on 1398  degrees of freedom
## AIC: 402
##
## Number of Fisher Scoring iterations: 25
```

```
# training set
preds <- predict(lr.all.small,newdata=all.small[train.all.small,],type="respo
nse")
preds <- ifelse(preds>0.5,"tumor","normal")
length(which(preds==all.small[train.all.small,201]))/length(preds)
```

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

```
table(preds,all.small[train.all.small,201])
```

```
##
## preds    normal tumor
##   normal    502     0
##   tumor       0  1097
```

```
# test set
preds <- predict(lr.all.small,newdata=all.small[-train.all.small,],type="resp
onse")
preds <- ifelse(preds>0.5,"tumor","normal")
length(which(preds==all.small[-train.all.small,201]))/length(preds)
```

```
## [1] 0.9606004
```

```
table(preds,all.small[-train.all.small,201])
```

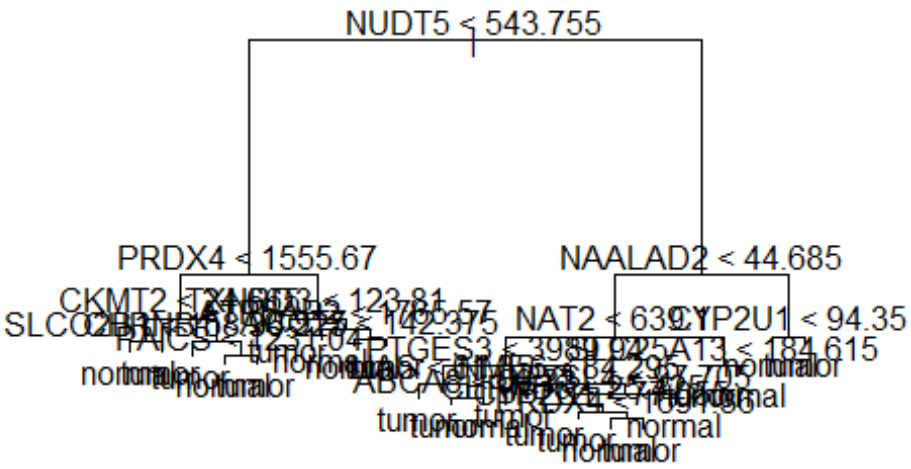
```
##
## preds    normal tumor
##   normal    175    10
##   tumor      11   337
```

The accuracy on the trainning set is 100%. But the accuracy on the test set is only 96.1%, and the confusion matrix shows that there are both false positive and false negetive cases. Not as the skin and breast datasets before, this model is not overtrained. This may becasue I did not use all the genes, I only used the genes that might be informative (can discriminate between normal and tumor tissues).

Dicision tree

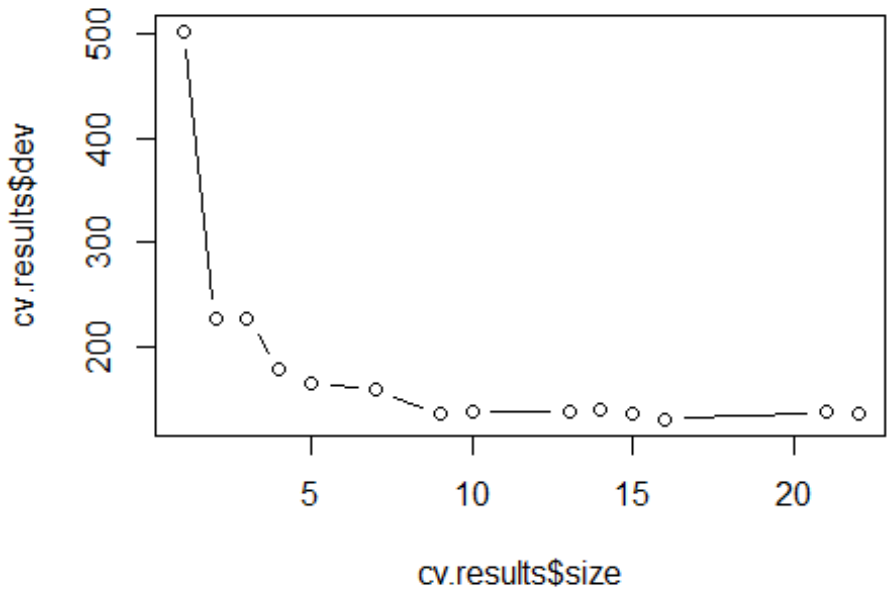
First I created the decision tree and visualized the tree.

```
tree.all.small <- tree(tissue~.,data=all.small[train.all.small,])
plot(tree.all.small)
text(tree.all.small)
```

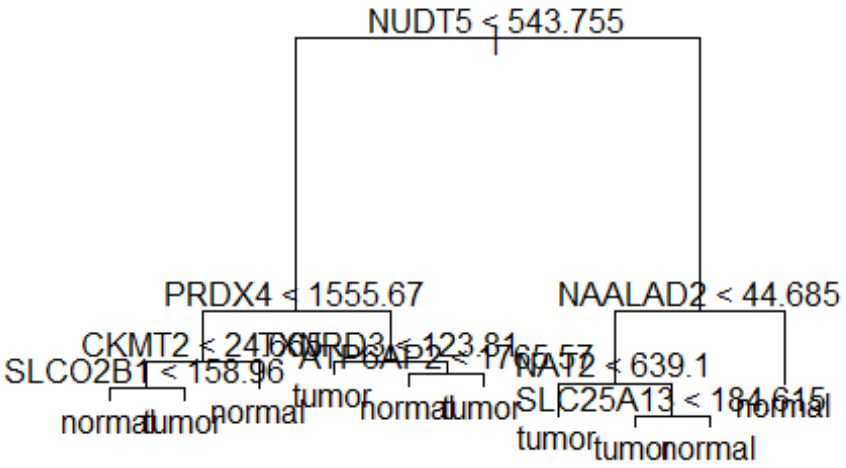


This is a relatively complex tree and need pruning.

```
cv.results <- cv.tree(tree.all.small, FUN=prune.misclass)
plot(cv.results$size, cv.results$dev, type="b")
```




```
smalltree.all.small <- prune.misclass(tree.all.small,best=10)
plot(smalltree.all.small); text(smalltree.all.small)
```



Based on the plot form the cross-validaton, the optimal level is around 10. Then I also plot the pruned tree, which is much simpler than the orignal tree.

```
# training set
preds <- predict(smalltree.all.small,newdata=all.small[train.all.small,],type
="class")
length(which(preds==all.small[train.all.small,201]))/length(preds)

## [1] 0.9449656

table(preds,all.small[train.all.small,201])

##
## preds    normal tumor
##   normal    446    32
##   tumor      56  1065

# test set
preds <- predict(smalltree.all.small,newdata=all.small[-train.all.small,],typ
e="class")
length(which(preds==all.small[-train.all.small,201]))/length(preds)

## [1] 0.945591

table(preds,all.small[-train.all.small,201])

##
## preds    normal tumor
##   normal    166     9
##   tumor      20   338
```

After pruning, the accuracy is 94.5% on training set and 94.6% on test set. Moreover there are both normal and tumor samples have been predicted wrong in both trainning set and test set. The dicision tree classifier of all tissue dataset is much complex but less accurat than the skin and breast dataset. This might due to the different gene experision in different tissues.

Random forest

Lastly, I also created a random forest classifier, and calculated the accuracy.

```
forest.all.small <- randomForest(tissue~.,data=all.small[train.all.small,])
# accuracy on training data
preds <- predict(forest.all.small,newdata=all.small[train.all.small,],type="c
lass")
length(which(preds==all.small[train.all.small,201]))/length(preds)

## [1] 1

table(preds,all.small[train.all.small,201])
```

```
##
## preds    normal tumor
##   normal    502     0
##   tumor       0  1097
```

```
# accuracy on test data
preds <- predict(forest.all.small,newdata=all.small[-train.all.small,],type="
class")
length(which(preds==all.small[-train.all.small,201]))/length(preds)
```

```
## [1] 0.9924953
```

```
table(preds,all.small[-train.all.small,201])
```

```
##
## preds    normal tumor
##   normal    182     0
##   tumor       4   347
```

It yields 100% accuracy on the trainning data and the accuracy on test data is 99.2%. There are 4 normal samples have been predicted as tumoe. Therefore it is a rather good classifier. The resluts shows that it is more difficult to bulit a accurate clssifier with the dataset of all tissue than the skin and breast dataset. This because same genes may have different experision level in different tissues, and their differences bwteen tumor and normal samples may also differ in tissues.

Discussion

Since there may be differenc features between skin, breast and all other tissues, I expect low accuracy when use a model on other dataset. Therefore, I tried to use the random forest models from these three dataset on each other and calculated the accuracy.

```
# skin and breast
preds <- predict(forest.skin,newdata=breast,type="class")
length(which(preds==breast[,2562]))/length(preds)
```

```
## [1] 0.6918489
```

```
table(preds,breast[,2562])
```

```
##
## preds    normal tumor
##   normal    142   155
##   tumor       0   206
```

```
preds <- predict(forest.breast,newdata=skin,type="class")
length(which(preds==skin[,2562]))/length(preds)
```

```
## [1] 0.6388889
```

```
table(preds,skin[,2562])
```

```
##
## preds    normal tumor
##   normal      3     0
##   tumor      26    43
```

```
# skin and all
preds <- predict(forest.all.small,newdata=skin,type="class")
length(which(preds==skin[,2562]))/length(preds)
```

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

```
table(preds,skin[,2562])
```

```
##
## preds    normal tumor
##   normal     29     0
##   tumor       0    43
```

```
preds <- predict(forest.skin,newdata=all,type="class")
length(which(preds==all[,2562]))/length(preds)
```

```
## [1] 0.674015
```

```
table(preds,all[,2562])
```

```
##
## preds    normal tumor
##   normal    610   617
##   tumor      78   827
```

```
# breast and all
preds <- predict(forest.all.small,newdata=breast,type="class")
length(which(preds==breast[,2562]))/length(preds)
```

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

```
table(preds,breast[,2562])
```

```
##
## preds    normal tumor
##   normal    142     0
##   tumor       0   361
```

```
preds <- predict(forest.breast,newdata=all,type="class")
length(which(preds==all[,2562]))/length(preds)
```

```
## [1] 0.8100375
```

```
table(preds,all[,2562])
```

```
##
## preds    normal tumor
```

```
##   normal    295    12
##   tumor     393  1432
```

The result shows that besides use the classifier developed from skin or breast on the all tissue dataset, other preditions have relatively low accuracy. It proves the idea that it is not suitable to use a model trained on one of these datasets to predict for any of the others.