Naïve Bayes and Random Forest

Rischan Mafrur

Tuesday, December 23, 2014

In this document, I tried to use another methods Naive Bayes and Random Forest for our dataset.

The best accuracy that we achieved from svm is 82,67 % using 6 best features: "MeanX", "AbsX", "MeanY", "MinY", "MeanZ", "SdZ".

Naive Bayes. To use Naive Bayes, we have to load `caret` library first.

```
library("caret")
```

```
## Warning: package 'caret' was built under R version 3.1.2
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.1.2
```

Loading and Spliting data to train and test set.

```
setwd("D:/Dropbox/LAB/COURSE/3/ubi/data")

features <- read.csv("all_features.csv",header = TRUE)

#Naive Bayes

splitdf <- function(dataframe, seed=NULL) {
  if (!is.null(seed)) set.seed(seed)
  index <- 1:nrow(dataframe)
  trainindex <- sample(index, trunc(length(index)*(90/100)))
  trainset <- dataframe[trainindex, ]
  testset <- dataframe[-trainindex, ]
  list(trainset=trainset,testset=testset)
}


splits <- splitdf(features, seed=808)
str(splits)
```

```
## List of 2
##  $ trainset:'data.frame':    1809 obs. of  185 variables:
##   ..$ MeanX : num [1:1809] 0.388 0.677 1.975 -0.295 -0.628 ...
##   ..$ SdX   : num [1:1809] 0.241 0.662 0.614 2.543 1.112 ...
```

```
##   ..$ FFT70 : num [1:202] -3.032 -1.926 0.199 3.022 -2.054 ...
##   ..$ FFT71 : num [1:202] -0.0871 -2.9484 0.2034 3.0401 -1.9826 ...
##   ..$ FFT72 : num [1:202] -2.177 -2.176 0.194 3.091 -1.13 ...
##   ..$ FFT73 : num [1:202] -2.293 -1.975 0.196 3.2 -1.476 ...
##   ..$ FFT74 : num [1:202] 0.223 -2.031 0.207 3.108 -1.357 ...
##   ..$ FFT75 : num [1:202] -1.788 -3.28 0.205 2.974 -1.78 ...
##   .. [list output truncated]
```

```
lapply(splits,nrow)
```

```
## $trainset
## [1] 1809
##
## $testset
## [1] 202
```

```
lapply(splits,head)
```

```
## $trainset
##           MeanX       SdX      MaxX        MinX       AbsX      RmsX
## 1830  0.3881605 0.2408002 0.6750037 -0.09745778 0.13839924 0.4496783
## 390   0.6772714 0.6618920 1.3973221 -0.74342764 0.06637231 0.9257269
## 730   1.9745577 0.6136734 3.6204522  1.16065601 0.45150434 2.0645792
## 1552 -0.2953630 2.5432578 3.5277322 -8.93233448 1.29653156 2.5293555
## 1976 -0.6275057 1.1123193 1.2046492 -4.71997104 1.04180666 1.2635859
## 811  -0.1500456 0.6254347 1.0858497 -0.74796217 0.54080810 0.6149158
##       FFT157     FFT158    FFT159    FFT160 label
## 5   3.334803  0.8925399  2.977633  2.611604 agung
## 7   6.760834  4.3028679  5.591767  6.273775 agung
## 17 -1.159119 -1.1739646 -1.158037 -1.159235 agung
## 30 -5.032368 -5.1147427 -5.274684 -5.433591 agung
## 34  1.151835  1.4673167  1.109478  1.195168 agung
## 42  1.321143  1.4609143  1.213389  1.564682 agung
```

```
training <- splits$trainset
testing <- splits$testset
```

Applying Naive Bayes

```
x <- subset(training, select=-label)
y <- training$label

#Generating Naive Bayes Model
model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))

#Show confusion Matrix for Naive Bayes Model with cross validation k = 10.
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##          y
```

```
##           agung alvin gde rischan
##   agung     309    21  68      24
##   alvin     129   383 184     119
##   gde         5     1 200       0
##   rischan    22    45  52     247
##
## Overall Statistics
##
##                Accuracy : 0.6296
##                  95% CI : (0.6069, 0.6519)
##     No Information Rate : 0.2786
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5079
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.6645       0.8511     0.3968         0.6333
## Specificity                0.9159       0.6821     0.9954         0.9161
## Pos Pred Value             0.7322       0.4699     0.9709         0.6749
## Neg Pred Value             0.8875       0.9326     0.8104         0.9009
## Prevalence                 0.2570       0.2488     0.2786         0.2156
## Detection Rate             0.1708       0.2117     0.1106         0.1365
## Detection Prevalence       0.2333       0.4505     0.1139         0.2023
## Balanced Accuracy          0.7902       0.7666     0.6961         0.7747
```

```
#Testing using real test data
x1 <- subset(testing, select=-label)
y1 <- testing$label

xtab2 <- table(predict(model$finalModel,x1)$class,y1)

confusionMatrix(xtab2)
```

```
## Confusion Matrix and Statistics
##
##          y1
##           agung alvin gde rischan
##   agung      31     7   8       6
##   alvin      17    27  16      18
##   gde         9     2  26       0
##   rischan     3     6   3      23
##
## Overall Statistics
##
##                Accuracy : 0.5297
##                  95% CI : (0.4584, 0.6001)
##     No Information Rate : 0.297
```

```
##     P-Value [Acc > NIR] : 4.469e-12
##
##                   Kappa : 0.377
##  Mcnemar's Test P-Value : 0.0003252
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.5167       0.6429     0.4906          0.4894
## Specificity                0.8521       0.6813     0.9262          0.9226
## Pos Pred Value             0.5962       0.3462     0.7027          0.6571
## Neg Pred Value             0.8067       0.8790     0.8364          0.8563
## Prevalence                 0.2970       0.2079     0.2624          0.2327
## Detection Rate             0.1535       0.1337     0.1287          0.1139
## Detection Prevalence       0.2574       0.3861     0.1832          0.1733
## Balanced Accuracy          0.6844       0.6621     0.7084          0.7060
```

Naive Bayes with only use 6 best features from SFFS.

```
setwd("D:/Dropbox/LAB/COURSE/3/ubi/data")
features <- read.csv("all_features.csv",header = TRUE)
new_f <- subset(features, select=c("MeanX", "AbsX", "MeanY", "MinY", "MeanZ",
"SdZ","label"))

splitdf <- function(dataframe, seed=NULL) {
  if (!is.null(seed)) set.seed(seed)
  index <- 1:nrow(dataframe)
  trainindex <- sample(index, trunc(length(index)*(90/100)))
  trainset <- dataframe[trainindex, ]
  testset <- dataframe[-trainindex, ]
  list(trainset=trainset,testset=testset)
}


splits <- splitdf(new_f, seed=808)
str(splits)
```

```
## List of 2
##  $ trainset:'data.frame':    1809 obs. of  7 variables:
##   ..$ MeanX: num [1:1809] 0.388 0.677 1.975 -0.295 -0.628 ...
##   ..$ AbsX : num [1:1809] 0.1384 0.0664 0.4515 1.2965 1.0418 ...
##   ..$ MeanY: num [1:1809] -9.71 -8.12 -9.84 -10.33 -9.98 ...
##   ..$ MinY : num [1:1809] -10.15 -9.01 -12.96 -14.07 -12.38 ...
##   ..$ MeanZ: num [1:1809] 1.92 3.927 -0.356 -1.271 -1.235 ...
##   ..$ SdZ  : num [1:1809] 0.65 1.436 3.012 3.276 0.893 ...
##   ..$ label: Factor w/ 4 levels "agung","alvin",..: 4 1 2 3 4 2 1 4 2 2
...
##  $ testset :'data.frame':    202 obs. of  7 variables:
##   ..$ MeanX: num [1:202] 0.5974 0.0769 -0.2685 -0.3494 1.4408 ...
```

```
##   ..$ AbsX : num [1:202] 1.587 2.64 0.178 1.86 3.158 ...
##   ..$ MeanY: num [1:202] -10.95 -9.52 -9.49 -10.71 -12.04 ...
##   ..$ MinY : num [1:202] -17.86 -14.8 -9.91 -18.79 -17.55 ...
##   ..$ MeanZ: num [1:202] 1.035 0.952 3.859 1.169 2.61 ...
##   ..$ SdZ  : num [1:202] 4.45 7.22 1.19 1.69 7.22 ...
##   ..$ label: Factor w/ 4 levels "agung","alvin",..: 1 1 1 1 1 1 1 1 1 1
...
```

```
lapply(splits,nrow)
```

```
## $trainset
## [1] 1809
##
## $testset
## [1] 202
```

```
lapply(splits,head)
```

```
## $trainset
##           MeanX       AbsX      MeanY       MinY      MeanZ       SdZ
## 1830  0.3881605 0.13839924  -9.713926 -10.146953  1.9198548 0.6497238
## 390   0.6772714 0.06637231  -8.122215  -9.013465  3.9272541 1.4361473
## 730   1.9745577 0.45150434  -9.838797 -12.957370 -0.3558127 3.0116910
## 1552 -0.2953630 1.29653156 -10.325177 -14.068901 -1.2712022 3.2755954
## 1976 -0.6275057 1.04180666  -9.979438 -12.380973 -1.2346527 0.8928233
## 811  -0.1500456 0.54080810 -10.087610 -11.291659  0.7369852 0.2269612
##        label
## 1830 rischan
## 390    agung
## 730    alvin
## 1552     gde
## 1976 rischan
## 811    alvin
##
## $testset
##          MeanX      AbsX      MeanY      MinY      MeanZ      SdZ label
## 5   0.59742959 1.5867704 -10.954408 -17.85851  1.0352185 4.448004 agung
## 7   0.07686185 2.6395196  -9.519404 -14.80353  0.9521123 7.219119 agung
## 17 -0.26854840 0.1779955  -9.490250  -9.90878  3.8594697 1.186131 agung
## 30 -0.34944954 1.8598252 -10.710136 -18.79459  1.1687857 1.686225 agung
## 34  1.44083419 3.1577843 -12.042180 -17.55239  2.6103309 7.215945 agung
## 42  1.27533278 0.6075398  -9.865321 -17.50970 -0.1746398 6.212753 agung
```

```
newtraining <- splits$trainset
newtesting <- splits$testset

x <- subset(newtraining, select=-label)
y <- newtraining$label
```

```r
#Generating Naive Bayes Model
model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))

#predict(model$finalModel,x)

xtab <- table(predict(model$finalModel,x)$class,y)

#Show confusion Matrix for Naive Bayes Model with cross validation k = 10.
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##          y
##           agung alvin gde rischan
##   agung     352    26  60      51
##   alvin      13   356 150      70
##   gde        24    10 203      14
##   rischan    76    58  91     255
##
## Overall Statistics
##
##                Accuracy : 0.6446
##                  95% CI : (0.622, 0.6666)
##     No Information Rate : 0.2786
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5284
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.7570       0.7911     0.4028          0.6538
## Specificity                0.8981       0.8286     0.9632          0.8414
## Pos Pred Value             0.7198       0.6044     0.8088          0.5312
## Neg Pred Value             0.9144       0.9230     0.8068          0.8984
## Prevalence                 0.2570       0.2488     0.2786          0.2156
## Detection Rate             0.1946       0.1968     0.1122          0.1410
## Detection Prevalence       0.2703       0.3256     0.1388          0.2653
## Balanced Accuracy          0.8275       0.8098     0.6830          0.7476
```

```r
#Testing using real test data
x1 <- subset(newtesting, select=-label)
y1 <- newtesting$label

xtab2 <- table(predict(model$finalModel,x1)$class,y1)
```

```
## Warning in FUN(1:202[[202L]], ...): Numerical 0 probability for all
## classes with observation 36
```

```
## Warning in FUN(1:202[[202L]], ...): Numerical 0 probability for all
## classes with observation 103

## Warning in FUN(1:202[[202L]], ...): Numerical 0 probability for all
## classes with observation 109

## Warning in FUN(1:202[[202L]], ...): Numerical 0 probability for all
## classes with observation 129

## Warning in FUN(1:202[[202L]], ...): Numerical 0 probability for all
## classes with observation 132

## Warning in FUN(1:202[[202L]], ...): Numerical 0 probability for all
## classes with observation 137

## Warning in FUN(1:202[[202L]], ...): Numerical 0 probability for all
## classes with observation 141

## Warning in FUN(1:202[[202L]], ...): Numerical 0 probability for all
## classes with observation 142

## Warning in FUN(1:202[[202L]], ...): Numerical 0 probability for all
## classes with observation 147
```

```
confusionMatrix(xtab2)
```

```
## Confusion Matrix and Statistics
##
##          y1
##           agung alvin gde rischan
##   agung      41     1  12       6
##   alvin       1    37  11      11
##   gde         6     1  20       1
##   rischan    12     3  10      29
##
## Overall Statistics
##
##                Accuracy : 0.6287
##                  95% CI : (0.5581, 0.6955)
##     No Information Rate : 0.297
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5059
##  Mcnemar's Test P-Value : 0.0004661
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.6833       0.8810    0.37736         0.6170
## Specificity                0.8662       0.8562    0.94631         0.8387
## Pos Pred Value             0.6833       0.6167    0.71429         0.5370
## Neg Pred Value             0.8662       0.9648    0.81034         0.8784
```

```
## Prevalence                0.2970       0.2079    0.26238          0.2327
## Detection Rate            0.2030       0.1832    0.09901          0.1436
## Detection Prevalence      0.2970       0.2970    0.13861          0.2673
## Balanced Accuracy         0.7748       0.8686    0.66183          0.7279
```

Random Forest To use random forest, we have to load `randomForest` library first.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.2
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Apply Random Forest to our dataset

```
trainData <- training
testData <- testing



gait_rf <- randomForest(label~.,data=trainData,ntree=100,proximity=TRUE)
xtab <- table(predict(gait_rf),trainData$label)
#Show the confusion matrix and accuracy using real data test
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##
##           agung alvin gde rischan
##   agung     427     3  23      27
##   alvin       5   420  29      30
##   gde        13    14 406      52
##   rischan    20    13  46     281
##
## Overall Statistics
##
##                Accuracy : 0.848
##                  95% CI : (0.8306, 0.8642)
##     No Information Rate : 0.2786
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7967
##  Mcnemar's Test P-Value : 0.0107
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.9183       0.9333     0.8056          0.7205
## Specificity                0.9606       0.9529     0.9395          0.9443
## Pos Pred Value             0.8896       0.8678     0.8371          0.7806
## Neg Pred Value             0.9714       0.9774     0.9260          0.9248
```

```
## Prevalence                 0.2570       0.2488     0.2786          0.2156
## Detection Rate             0.2360       0.2322     0.2244          0.1553
## Detection Prevalence       0.2653       0.2676     0.2681          0.1990
## Balanced Accuracy          0.9394       0.9431     0.8725          0.8324
```

Tried to using 6 best features:

```
setwd("D:/Dropbox/LAB/COURSE/3/ubi/data")
features <- read.csv("all_features.csv",header = TRUE)
new_f <- subset(features, select=c("MeanX", "AbsX", "MeanY", "MinY", "MeanZ",
"SdZ","label"))

splitdf <- function(dataframe, seed=NULL) {
  if (!is.null(seed)) set.seed(seed)
  index <- 1:nrow(dataframe)
  trainindex <- sample(index, trunc(length(index)*(90/100)))
  trainset <- dataframe[trainindex, ]
  testset <- dataframe[-trainindex, ]
  list(trainset=trainset,testset=testset)
}


splits <- splitdf(new_f, seed=808)
str(splits)
```

```
## List of 2
##  $ trainset:'data.frame':    1809 obs. of  7 variables:
##   ..$ MeanX: num [1:1809] 0.388 0.677 1.975 -0.295 -0.628 ...
##   ..$ AbsX : num [1:1809] 0.1384 0.0664 0.4515 1.2965 1.0418 ...
##   ..$ MeanY: num [1:1809] -9.71 -8.12 -9.84 -10.33 -9.98 ...
##   ..$ MinY : num [1:1809] -10.15 -9.01 -12.96 -14.07 -12.38 ...
##   ..$ MeanZ: num [1:1809] 1.92 3.927 -0.356 -1.271 -1.235 ...
##   ..$ SdZ  : num [1:1809] 0.65 1.436 3.012 3.276 0.893 ...
##   ..$ label: Factor w/ 4 levels "agung","alvin",..: 4 1 2 3 4 2 1 4 2 2
...
##  $ testset :'data.frame':    202 obs. of  7 variables:
##   ..$ MeanX: num [1:202] 0.5974 0.0769 -0.2685 -0.3494 1.4408 ...
##   ..$ AbsX : num [1:202] 1.587 2.64 0.178 1.86 3.158 ...
##   ..$ MeanY: num [1:202] -10.95 -9.52 -9.49 -10.71 -12.04 ...
##   ..$ MinY : num [1:202] -17.86 -14.8 -9.91 -18.79 -17.55 ...
##   ..$ MeanZ: num [1:202] 1.035 0.952 3.859 1.169 2.61 ...
##   ..$ SdZ  : num [1:202] 4.45 7.22 1.19 1.69 7.22 ...
##   ..$ label: Factor w/ 4 levels "agung","alvin",..: 1 1 1 1 1 1 1 1 1 1
...
```

```
lapply(splits,nrow)
```

```
## $trainset
## [1] 1809
##
```

```
## $testset
## [1] 202
```

```
lapply(splits,head)
```

```
## $trainset
##           MeanX       AbsX      MeanY       MinY      MeanZ       SdZ
## 1830  0.3881605 0.13839924  -9.713926 -10.146953  1.9198548 0.6497238
## 390   0.6772714 0.06637231  -8.122215  -9.013465  3.9272541 1.4361473
## 730   1.9745577 0.45150434  -9.838797 -12.957370 -0.3558127 3.0116910
## 1552 -0.2953630 1.29653156 -10.325177 -14.068901 -1.2712022 3.2755954
## 1976 -0.6275057 1.04180666  -9.979438 -12.380973 -1.2346527 0.8928233
## 811  -0.1500456 0.54080810 -10.087610 -11.291659  0.7369852 0.2269612
##        label
## 1830 rischan
## 390    agung
## 730    alvin
## 1552     gde
## 1976 rischan
## 811    alvin
##
## $testset
##          MeanX      AbsX      MeanY      MinY      MeanZ      SdZ label
## 5   0.59742959 1.5867704 -10.954408 -17.85851  1.0352185 4.448004 agung
## 7   0.07686185 2.6395196  -9.519404 -14.80353  0.9521123 7.219119 agung
## 17 -0.26854840 0.1779955  -9.490250  -9.90878  3.8594697 1.186131 agung
## 30 -0.34944954 1.8598252 -10.710136 -18.79459  1.1687857 1.686225 agung
## 34  1.44083419 3.1577843 -12.042180 -17.55239  2.6103309 7.215945 agung
## 42  1.27533278 0.6075398  -9.865321 -17.50970 -0.1746398 6.212753 agung
```

```
newtraining <- splits$trainset
newtesting <- splits$testset



trainData <- newtraining
testData <- newtesting

library(randomForest)

gait_newrf <- randomForest(label~.,data=trainData,ntree=100,proximity=TRUE)
xtab2 <- table(predict(gait_newrf),trainData$label)
#show the result with only using 6 best features
confusionMatrix(xtab2)
```

```
## Confusion Matrix and Statistics
##
##
##           agung alvin gde rischan
##   agung     419    14  30      31
```

```
##   alvin      12   376  40      40
##   gde        16    26 385      58
##   rischan    18    34  49     261
##
## Overall Statistics
##
##                Accuracy : 0.7966
##                  95% CI : (0.7773, 0.8149)
##     No Information Rate : 0.2786
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7279
##  Mcnemar's Test P-Value : 0.06028
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.9011       0.8356     0.7639         0.6692
## Specificity                0.9442       0.9323     0.9234         0.9288
## Pos Pred Value             0.8482       0.8034     0.7938         0.7210
## Neg Pred Value             0.9650       0.9448     0.9101         0.9109
## Prevalence                 0.2570       0.2488     0.2786         0.2156
## Detection Rate             0.2316       0.2078     0.2128         0.1443
## Detection Prevalence       0.2731       0.2587     0.2681         0.2001
## Balanced Accuracy          0.9226       0.8839     0.8436         0.7990
```