Human Activity Recognition

Shanmuga

20 July 2016

Human Activity Recognition

Load raw data

```
library(caret)
```

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

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

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(e1071)
library(ggplot2)
library(C50)

trainData <- read.csv("pml-training.csv",na.strings=c("NA","#DIV/0!",""))
testData <- read.csv("pml-testing.csv",na.strings=c("NA","#DIV/0!",""))
```

Clean the raw data

Remove first column that is not required - running number

```
cleanTrainData <- trainData[c(-1)]
```

Find NA values and exclude

```
notNA <- sapply(cleanTrainData,function(i){sum(is.na(i))/length(i)})<0.9
cleanTrainData <- cleanTrainData[, notNA]
```

Remove near zero variance columns/variables

```
nzv <- nearZeroVar(cleanTrainData, saveMetrics=TRUE)
cleanTrainData <- cleanTrainData[,nzv$nzv==FALSE]
```

Drop unnecessary columns from the testing dataset

```
columnNames <- colnames(cleanTrainData[,-58]) # remove class column.
cleanTestData <- testData[columnNames]
```

Split the training set into a training and validation set.

```
intrain<-createDataPartition(y=cleanTrainData$classe,p=0.7,list=FALSE)
training<-cleanTrainData[intrain,]
testing<-cleanTrainData[-intrain,]
```

Cross Validation to select the best model performance

```
fitControl <- trainControl(method = "cv", number = 10, repeats = 0,savePred=T, classProb=T)

modelHAR_1 <- train(classe~., data = training, method = "rpart", trControl= fitControl)
```

```
## Loading required package: rpart
```

```
modelHAR_2 <- train(classe~., data = training, method = "rpart1SE", trControl= fitControl)
modelHAR_3 <- train(classe~., data = training, method = "C5.0Tree", trControl= fitControl)
```

Confusion martix

```
activity_1 <- predict(modelHAR_1, newdata = testing[,-58])
conMatrix_1 <- confusionMatrix(activity_1, testing$classe)
conMatrix_1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1260  284   21   31   13
##          B  107  493   17    0    0
##          C   83  178  540   28   48
##          D  218  184  448  905  512
##          E    6    0    0    0  509
##
## Overall Statistics
##
##                Accuracy : 0.6299
##                  95% CI : (0.6174, 0.6423)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5355
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.7527  0.43284  0.52632   0.9388  0.47043
## Specificity            0.9171  0.97387  0.93064   0.7232  0.99875
## Pos Pred Value         0.7831  0.79903  0.61574   0.3992  0.98835
## Neg Pred Value         0.9032  0.87737  0.90296   0.9837  0.89330
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2141  0.08377  0.09176   0.1538  0.08649
## Detection Prevalence   0.2734  0.10484  0.14902   0.3852  0.08751
## Balanced Accuracy      0.8349  0.70335  0.72848   0.8310  0.73459
```

```
activity_2 <- predict(modelHAR_1, newdata = training[,-58])
conMatrix_2 <- confusionMatrix(activity_2, training$classe)
conMatrix_2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3021  665   68   79   31
##          B  208 1183   28    0    0
##          C  186  368 1270   59  150
##          D  483  442 1030 2114 1222
##          E    8    0    0    0 1122
##
## Overall Statistics
##
##                Accuracy : 0.6341
##                  95% CI : (0.6259, 0.6421)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5402
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7734  0.44507  0.53005   0.9387  0.44436
## Specificity            0.9143  0.97870  0.93272   0.7234  0.99929
## Pos Pred Value         0.7818  0.83369  0.62469   0.3995  0.99292
## Neg Pred Value         0.9104  0.88026  0.90379   0.9837  0.88871
## Prevalence             0.2843  0.19349  0.17442   0.1639  0.18381
## Detection Rate         0.2199  0.08612  0.09245   0.1539  0.08168
## Detection Prevalence   0.2813  0.10330  0.14799   0.3852  0.08226
## Balanced Accuracy      0.8438  0.71188  0.73139   0.8310  0.72182
```

```
activity_3 <- predict(modelHAR_3, newdata = training[,-58])
conMatrix_3 <- confusionMatrix(activity_3, training$classe)
conMatrix_3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 3900   12    0    0    0
##          B    5 2636    1    0    0
##          C    0    9 2395    2    2
##          D    1    1    0 2249    2
##          E    0    0    0    1 2521
##
## Overall Statistics
##
##                Accuracy : 0.9974
##                  95% CI : (0.9964, 0.9982)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9967
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9985   0.9917   0.9996   0.9987   0.9984
## Specificity            0.9988   0.9995   0.9989   0.9997   0.9999
## Pos Pred Value         0.9969   0.9977   0.9946   0.9982   0.9996
## Neg Pred Value         0.9994   0.9980   0.9999   0.9997   0.9996
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2839   0.1919   0.1743   0.1637   0.1835
## Detection Prevalence   0.2848   0.1923   0.1753   0.1640   0.1836
## Balanced Accuracy      0.9986   0.9956   0.9992   0.9992   0.9992
```

As the "C5.0Tree" has the highest accuracy (0.9972) when predicting the test data. we select this model to build our final model.

Build final model

Final model is build with the "C5.0Tree" algorithm using the entire training data set.

```
fitControl <- trainControl(method = "cv", number = 10, repeats = 0,savePred=T, classProb=T)
modelHAR_F <- train(classe~., data = cleanTrainData, method = "C5.0Tree", trControl= fitControl)
```

Perform prediction with final model

```
act_prediction <- predict(modelHAR_F, newdata = cleanTestData)
```

The predistions obtained are:

```
act_prediction
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```