Machine Learning Project

Step 1: Upload training. Set seed.

set.seed(8675309)

library(caret)

library(AppliedPredictiveModeling)

library(randomForest) ## for randomForest()

library(rattle) ## for fancyRpartPlot()

library(rpart) ## for rpart()

library(rpart.plot) ## for rpart.plot()

## Upload training data

training = read.csv("training data for ML project",

header = TRUE,

stringsAsFactors=FALSE,

na.strings=c("NA", "N/A"))

dim(training)

|  |
| --- |
| dim(training)  [1] 19622 160 |
|  |
| |  | | --- | |  | |

Step 2: Clean and Partition training data

##clean predictor variables – training

> ##clean predictor variables for training data

> cldata <- training[,-c(1:7, 160)] ## drop the first 7 and the last columns

> cldata<- suppressWarnings(data.frame(apply(cldata, 2, as.numeric))) ## convert all columns to numeric

> cldata[is.na(cldata)] <- 0; sum(is.na(cldata)) ## replace <NA> with 0

[1] 0

> cldata <- cldata[vapply(cldata, function(x) length(unique(x)) > 1, logical(1L))] ## remove same value columns

dim(cldata)

[1] 19622 143

##select training with clean data

training <- data.frame(classe=training$classe, cldata); rm(cldata)

training$classe <- as.factor(training$classe)

Randomly divide training data in to two sets: 75% as “train” subset, and 25% as “test” subset.

## Data Partition

inTrain <- createDataPartition(y=training$classe, p=0.75, list=FALSE)

> myTraining <- training[inTrain, ];

> myTesting <- training[-inTrain, ]

> dim(myTraining); dim(myTesting)

[1] 14718 144

[1] 4904 144

Step 3: Run different models and test accuracy with the 4 “test” subset data

1. Random Forest with Principal Component Analysis

## TRAIN AND TEST MODELS: Principal Component Analysis for Feature Selection

prePCA <- preProcess(myTraining, method=c("center", "scale", "pca"), thresh = 0.8); prePCA

trainPCA <- predict(prePCA, myTraining)

summary(trainPCA)

## Random Forest with PCA

accRFPCA <- matrix(0, nrow=4, ncol=1)

for (i in 1:4) {

rfpca <- randomForest(classe~., data=trainPCA, method="class")

test = myTesting

testrfpca <- predict(prePCA, test)

predrfpca <- predict(rfpca, testrfpca)

accRFPCA[i,1] <- confusionMatrix(test$classe, predrfpca)$overall["Accuracy"]

}; rm(rfpca, test, testrfpca, predrfpca)

##variable importance for RF+PCA

importance(rfpca)

varImp(rfpca)

## Random Forest without PCA

accRF <- matrix(0, nrow=4, ncol=1)

RF <- randomForest(classe~., data=myTraining, method="class")

for (i in 1:4) {

test = myTesting

predRF <- predict(RF, test)

accRF[i,1] <- confusionMatrix(test$classe, predRF)$overall["Accuracy"]

}; rm(RF, test, predRF)

##variable importance for RF

importance(RF)

varImp(RF)

##Classication rpart+ PCA

cl\_rpt\_pca <- train(classe~., data=trainPCA, method="rpart")

fancyRpartPlot(cl\_rpt\_pca$finalModel, sub="")

acccl\_rpt\_pca <- matrix(0, nrow=4, ncol=1)

for (i in 1:4) {

test=myTesting

testcl\_rpt\_pca <- predict(prePCA, test)

predcl\_rpt\_pca <- predict(cl\_rpt\_pca, testcl\_rpt\_pca)

acccl\_rpt\_pca[i,1] <- confusionMatrix(test$classe, predcl\_rpt\_pca)$overall["Accuracy"]

}; rm(cl\_rpt\_pca, test, testcl\_rpt\_pca, predcl\_rpt\_pca)

##Classification (default rpart)

dfrpart <- train(classe~., data=myTraining, method="rpart")

fancyRpartPlot(dfrpart$finalModel, sub="")

accdfrpt <- matrix(0, nrow=4, ncol=1)

for (i in 1:4) {

test=myTesting

preddfrpart <- predict(dfrpart, test)

accdfrpt[i,1] <- confusionMatrix(test$classe, preddfrpart)$overall["Accuracy"]

}; rm(dfrpart, test, preddfrpart)

##Classification (rpart= "class")

clrpart <- rpart(classe~., data=myTraining, method="class")

fancyRpartPlot(clrpart, sub="")

accclrpart <- matrix(0, nrow=4, ncol=1)

for (i in 1:4) {

test=myTesting

predclrpart <- predict(clrpart, test, type ="class")

accclrpart[i,1] <- confusionMatrix(test$classe, predclrpart)$overall["Accuracy"]

}; rm(clrpart, test, predclrpart)

##Boosted Tree with PCA

BPCA <- train(classe~., method="gbm", data=trainPCA)

quiet(expr, all = TRUE)

accGBM <- matrix(0, nrow=4, ncol=1)

for (i in 1:4) {

test = myTesting

testGBM <- predict(prePCA, test)

predGBM <- predict(BPCA, testGBM)

accGBM[i,1] <- confusionMatrix(test$classe, predGBM)$overall["Accuracy"]

}; rm(BPCA, test, testGBM, predGBM, i)

quiet(expr, all = TRUE)

Step 5: Compare the accuracy and speed of each potential model, then select one to apply to testing data.

## Compare Accuracy

> acc\_all <- data.frame(accRFPCA, accRF, acccl\_rpt\_pca, accdfrpt, accclrpart, accGBM)

> names(acc\_all) <- c("RF+PCA", "RF", "rpart+PCA", "rpart(default)", "rpart(class)", "gbm+PCA"); acc\_all

RF+PCA RF rpart+PCA rpart(default) rpart(class) gbm+PCA

1 0.9600326 0.9946982 0.3882545 0.487969 0.7310359 0.7585644

2 0.9612561 0.9946982 0.3882545 0.487969 0.7310359 0.7585644

3 0.9616639 0.9946982 0.3882545 0.487969 0.7310359 0.7585644

4 0.9614600 0.9946982 0.3882545 0.487969 0.7310359 0.7585644

> colMeans(acc\_all)

RF+PCA RF rpart+PCA rpart(default) rpart(class) gbm+PCA

0.9611032 0.9946982 0.3882545 0.4879690 0.7310359 0.7585644

Step 6: Predict on testing data using selected model

> ##Upload and clean test data

> testing = read.csv("testing data for ML project.csv",

+ header = TRUE,

+ stringsAsFactors=FALSE,

+ na.strings=c("NA", "N/A"))

> cldata2 <- testing[,-c(1:7, 160)] ## drop the first 7 and the last columns

> cldata2<- suppressWarnings(data.frame(apply(cldata2, 2, as.numeric))) ## convert all columns to numeric

> cldata2[is.na(cldata2)] <- 0; sum(is.na(cldata2)) ## replace <NA> with 0

[1] 0

> dim(testing)

[1] 20 160

## Predict using test data

> testing <- data.frame(classe="", cldata2); rm(cldata2)

> testRF <- randomForest(classe~. , data=myTraining, method="class")

> pred <- predict(testRF, testing); pred

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

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

FIGURES

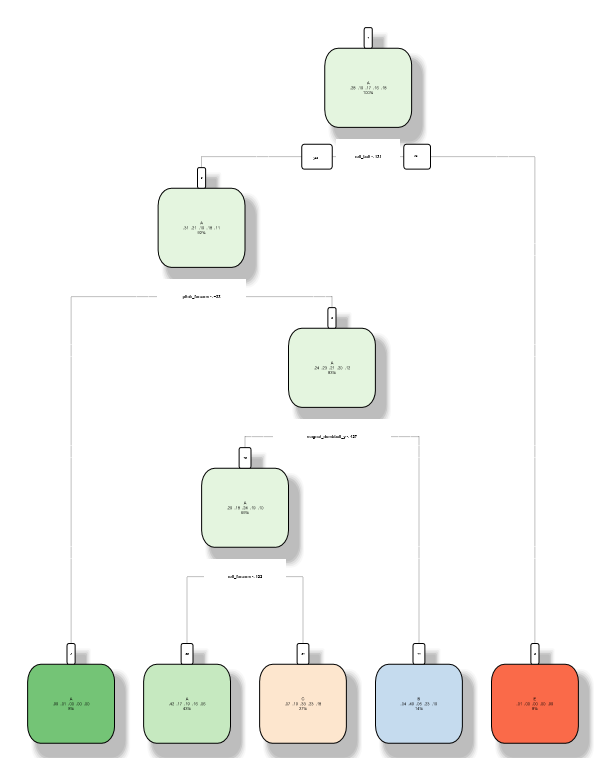
Figure 1: Classification – Default rpart()  
  


Figure 1: Random Forrest + PCA

