**Practical Machine Learning**

**1. Load all the necessary library.**

**> library(caret)**

Loading required package: lattice

Loading required package: ggplot2

**> install.packages("randomForest", dependencies = T)**

Installing package into ‘C:/Users/sharanyasrm/Documents/R/win-library/3.6’

(as ‘lib’ is unspecified)

--- Please select a CRAN mirror for use in this session ---

trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.6/randomForest\_4.6-14.zip'

Content type 'application/zip' length 250189 bytes (244 KB)

downloaded 244 KB

package ‘randomForest’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\sharanyasrm\AppData\Local\Temp\RtmpcvhWsW\downloaded\_packages

**> library(rpart)**

**> find.package('rpart')**

[1] "C:/Program Files/R/R-3.6.3/library/rpart"

**> library(RColorBrewer)**

**> install.packages("rattle")**

Installing package into ‘C:/Users/sharanyasrm/Documents/R/win-library/3.6’

(as ‘lib’ is unspecified)

also installing the dependencies ‘XML’, ‘rpart.plot’

trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.6/XML\_3.99-0.3.zip'

Content type 'application/zip' length 4258943 bytes (4.1 MB)

downloaded 4.1 MB

trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.6/rpart.plot\_3.0.8.zip'

Content type 'application/zip' length 1077637 bytes (1.0 MB)

downloaded 1.0 MB

trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.6/rattle\_5.3.0.zip'

Content type 'application/zip' length 5320173 bytes (5.1 MB)

downloaded 5.1 MB

package ‘XML’ successfully unpacked and MD5 sums checked

package ‘rpart.plot’ successfully unpacked and MD5 sums checked

package ‘rattle’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\sharanyasrm\AppData\Local\Temp\RtmpcvhWsW\downloaded\_packages

**2. Set the seed value and load the teat and training data.**

**> set.seed(1234)**

**> training <- read.csv("D:/PMLC/pml-training.csv", na.strings=c("NA","#DIV/0!",""))**

**> testing <- read.csv("D:/PMLC/pml-testing.csv", na.strings=c("NA","#DIV/0!",""))**

**3. Split the data as training 60% and test 20%**

**> inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)**

**> myTraining <- training[inTrain, ]; myTesting <- training[-inTrain, ]**

**> dim(myTraining); dim(myTesting)**

[1] 11776 160

[1] 7846 160

**4. Clean the data**

**> myDataNZV <- nearZeroVar(myTraining, saveMetrics=TRUE)**

**> myNZVvars <- names(myTraining) %in% c("new\_window", "kurtosis\_roll\_belt", "kurtosis\_picth\_belt",**

**+ "kurtosis\_yaw\_belt", "skewness\_roll\_belt", "skewness\_roll\_belt.1", "skewness\_yaw\_belt",**

**+ "max\_yaw\_belt", "min\_yaw\_belt", "amplitude\_yaw\_belt", "avg\_roll\_arm", "stddev\_roll\_arm",**

**+ "var\_roll\_arm", "avg\_pitch\_arm", "stddev\_pitch\_arm", "var\_pitch\_arm", "avg\_yaw\_arm",**

**+ "stddev\_yaw\_arm", "var\_yaw\_arm", "kurtosis\_roll\_arm", "kurtosis\_picth\_arm",**

**+ "kurtosis\_yaw\_arm", "skewness\_roll\_arm", "skewness\_pitch\_arm", "skewness\_yaw\_arm",**

**+ "max\_roll\_arm", "min\_roll\_arm", "min\_pitch\_arm", "amplitude\_roll\_arm", "amplitude\_pitch\_arm",**

**+ "kurtosis\_roll\_dumbbell", "kurtosis\_picth\_dumbbell", "kurtosis\_yaw\_dumbbell", "skewness\_roll\_dumbbell",**

**+ "skewness\_pitch\_dumbbell", "skewness\_yaw\_dumbbell", "max\_yaw\_dumbbell", "min\_yaw\_dumbbell",**

**+ "amplitude\_yaw\_dumbbell", "kurtosis\_roll\_forearm", "kurtosis\_picth\_forearm", "kurtosis\_yaw\_forearm",**

**+ "skewness\_roll\_forearm", "skewness\_pitch\_forearm", "skewness\_yaw\_forearm", "max\_roll\_forearm",**

**+ "max\_yaw\_forearm", "min\_roll\_forearm", "min\_yaw\_forearm", "amplitude\_roll\_forearm",**

**+ "amplitude\_yaw\_forearm", "avg\_roll\_forearm", "stddev\_roll\_forearm", "var\_roll\_forearm",**

**+ "avg\_pitch\_forearm", "stddev\_pitch\_forearm", "var\_pitch\_forearm", "avg\_yaw\_forearm",**

**+ "stddev\_yaw\_forearm", "var\_yaw\_forearm")**

**> myTraining <- myTraining[!myNZVvars]**

**> dim(myTraining)**

[1] 11776 100

**> myTraining <- myTraining[c(-1)]**

**> trainingV3 <- myTraining #creating another subset to iterate in loop**

**> for(i in 1:length(myTraining)) { #for every column in the training dataset**

**+ if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .6 ) { #if n?? NAs > 60% of total observations**

**+ for(j in 1:length(trainingV3)) {**

**+ if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) ==1) { #if the columns are the same:**

**+ trainingV3 <- trainingV3[ , -j] #Remove that column**

**+ }**

**+ }**

**+ }**

**+ }**

**> myTraining <- trainingV3**

**> rm(trainingV3)**

**> clean1 <- colnames(myTraining)**

**> clean2 <- colnames(myTraining[, -58]) #already with classe column removed**

**> myTesting <- myTesting[clean1]**

**> testing <- testing[clean2]**

#Check the dimensions

**> dim(myTesting)**

[1] 7846 58

**> dim(testing)**

[1] 20 57

**> for (i in 1:length(testing) ) {**

**+ for(j in 1:length(myTraining)) {**

**+ if( length( grep(names(myTraining[i]), names(testing)[j]) ) ==1) {**

**+ class(testing[j]) <- class(myTraining[i])**

**+ }**

**+ }**

**+ }**

**> testing <- rbind(myTraining[2, -58] , testing) #note row 2 does not mean anything, this will be removed right.. now:**

**> testing <- testing[-1,]**

**5. Fit the model and make predictions**

**> modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")**

**> predictionsA1 <- predict(modFitA1, myTesting, type = "class")**

**> confusionMatrix(predictionsA1, myTesting$classe)**

Confusion Matrix and Statistics

Reference

Prediction A B C D E

A 2157 68 10 1 0

B 60 1265 73 67 0

C 15 177 1261 141 70

D 0 8 15 962 111

E 0 0 9 115 1261

Overall Statistics

Accuracy : 0.8802

95% CI : (0.8728, 0.8873)

No Information Rate : 0.2845

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.8484

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: A Class: B Class: C Class: D Class: E

Sensitivity 0.9664 0.8333 0.9218 0.7481 0.8745

Specificity 0.9859 0.9684 0.9378 0.9796 0.9806

Pos Pred Value 0.9647 0.8635 0.7578 0.8777 0.9105

Neg Pred Value 0.9866 0.9604 0.9827 0.9520 0.9720

Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838

Detection Rate 0.2749 0.1612 0.1607 0.1226 0.1607

Detection Prevalence 0.2850 0.1867 0.2121 0.1397 0.1765

Balanced Accuracy 0.9762 0.9009 0.9298 0.8638 0.9276

**6. Predictions using random forest**

**> library(randomForest)**

randomForest 4.6-14

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

Attaching package: ‘randomForest’

The following object is masked from ‘package:ggplot2’:

margin

**> randomForest**

function (x, ...)

UseMethod("randomForest")

<bytecode: 0x17618768>

<environment: namespace:randomForest>

**> modFitB1 <- randomForest(classe ~. , data=myTraining)**

**> predictionsB1 <- predict(modFitB1, myTesting, type = "class")**

**> confusionMatrix(predictionsB1, myTesting$classe)**

Confusion Matrix and Statistics

Reference

Prediction A B C D E

A 2232 5 0 0 0

B 0 1513 2 0 0

C 0 0 1362 7 0

D 0 0 4 1278 1

E 0 0 0 1 1441

Overall Statistics

Accuracy : 0.9975

95% CI : (0.9961, 0.9984)

No Information Rate : 0.2845

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9968

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: A Class: B Class: C Class: D Class: E

Sensitivity 1.0000 0.9967 0.9956 0.9938 0.9993

Specificity 0.9991 0.9997 0.9989 0.9992 0.9998

Pos Pred Value 0.9978 0.9987 0.9949 0.9961 0.9993

Neg Pred Value 1.0000 0.9992 0.9991 0.9988 0.9998

Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838

Detection Rate 0.2845 0.1928 0.1736 0.1629 0.1837

Detection Prevalence 0.2851 0.1931 0.1745 0.1635 0.1838

Balanced Accuracy 0.9996 0.9982 0.9973 0.9965 0.9996

**7. Write the predictions to file B2.**

**> predictionsB2 <- predict(modFitB1, testing, type = "class")**

**> pml\_write\_files = function(x){**

**+ n = length(x)**

**+ for(i in 1:n){**

**+ filename = paste0("problem\_id\_",i,".txt")**

**+ write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)**

**+ }**

**+ }**

**> pml\_write\_files(predictionsB2)**

**8. Read the file B2**

**> predictionsB2**

1 2 3 41 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