Project

PRACTICAL MACHINE LEARNING PROJECT

data: http://groupware.les.inf.puc-rio.br/har Classe A- good performance (as specified), classe B-C-D-E: Common mistakes: B=throwing the elbows to the front # C=lifting the dumbbell only halfway D=lowering the dumbbell only halfway # E=throwing the hips to the front (Class E)

GOAL= to predict the manner in which they did the exercise (Classe A-B-C-D-E). This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

```
library(caret); library(lattice); library(ggplot2);library(kernlab);
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(randomForest);library(rpart); library(rpart.plot) ;
```

```
## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
cat("\f") #clear screen
```

```
md<-"C:/Users/Aner/Desktop/DATA_SCIENCE_SPECIALIZATION/08_Practical_Machine_Learning/Project/";
setwd(md)
f_train<-'pml-training.csv';
f_test<-'pml-testing.csv';
d_trainval<-read.csv(f_train); #View(d_trainval) ## data of training set+cross validation set
d_test<-read.csv(f_test); #View(d_test) ## data of test set
```

PARTITION AND PREPROCESS THE DATA

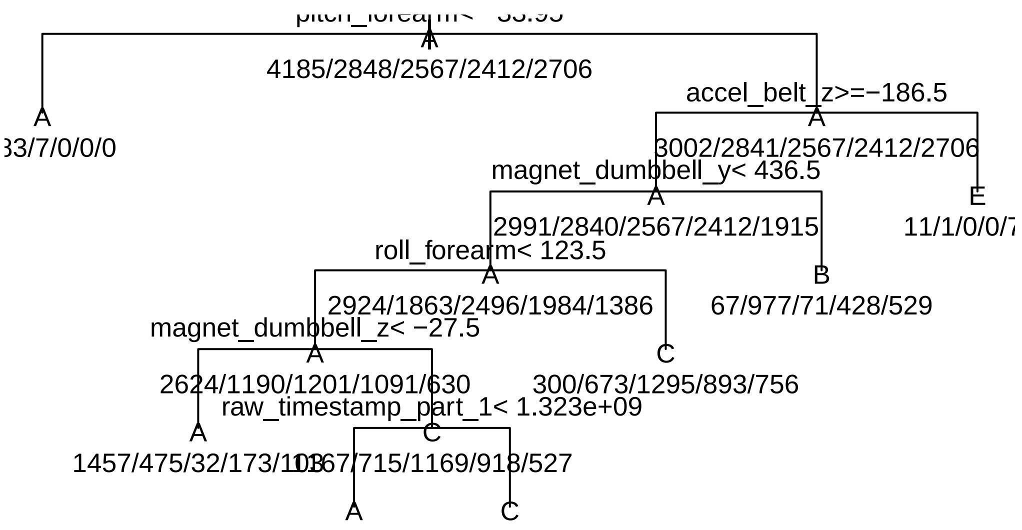
```
# take only the numeric variables and the class
ixxx<-sapply(d_trainval,is.numeric)==TRUE
d_train_numeric<-d_trainval[,ixxx]
ixxx2<-which(colMeans(is.na(d_train_numeric)) > 0.5)
d_train_numeric<-d_train_numeric[, -ixxx2] ## Remove columns with more than 50% NA
#d_test<-d_test[,ixxx]
#d_test<-d_test[,-ixxx2]
```

```
#d_train_numeric<-preProcess(d_train_numeric, method=c("center","scale"))### preProcess
classe<-as.factor(d_trainval$classe)
d_train_numeric<-data.frame(classe,d_train_numeric) #View(d_train_numeric)
ix <- createDataPartition(d_train_numeric[,1], p = 3/4)[[1]]


d_train <- d_train_numeric[ ix,]#trainig
d_val <- d_train_numeric[-ix,] #validation

###==============================================================================
### Preprocess- remove correlated variables
M<-abs(cor(d_train[,-1]))
M[upper.tri(M)] <- 0
diag(M)<-0 #which(M>0.8,arr.ind=T)
ix_not_correlated<-(!apply(M,2,function(x) any(x > 0.65)) )##index of columns of NONE-correlated variabl
d_train <- d_train[,ix_not_correlated]
d_val <- d_val[,ix_not_correlated]
#ix_not_correlated_TEST<-which(ix_not_correlated, arr.ind = FALSE, useNames = TRUE)-1 ## -1 since 1st i
#d_test<-d_test[,ix_not_correlated_TEST]
```

TRAINING AND PREDICTION

```
###==============================================================================
### PREDICTION WITH TREES
fit<-train(classe~.,data=d_train, method="rpart")
#fancyRpartPlot(fit$finalModel)
plot(fit$finalModel,uniform=TRUE,main="Classification Trees")
text(fit$finalModel, use.n=TRUE, all=TRUE, cex=.8)
```

Classification Trees



```
pred<-predict(fit,newdata=d_val) ##predicting on the validation data
cm<-confusionMatrix(pred,d_val$classe)
cm$overall['Accuracy']
```

```
##  Accuracy
## 0.5163132
```

```
### Accuracy is not high, only 0.52.  Lets try to improve performance

#### RANDOM FOREST
fit2<-randomForest(classe~.,data=d_train, method="class",prox=TRUE)
pred2<-predict(fit2,newdata=d_val) ##predicting on the validation data
cm2<-confusionMatrix(pred2,d_val$classe)
cm2$overall['Accuracy']
```

```
##  Accuracy
## 0.9959217
```

PREDICTION ON THE TEST SET

```
TEST_PRED <- predict(fit2, newdata=d_test)
TEST_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
```

===