# Practical Machine Learning Peer Assessment

## Summary

This analysis has been done to predict the manner in which persons performed weight lifting exercises. The data is collected from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The outcome variable has five classes and the total number of predictors are 159

## Loading various libraries, and preparing the data (using subdirectory data)

library(caret)

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

library(randomForest)

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

training <- read.csv("./data/pml-training.csv",row.names=1,na.strings = "")  
testing <- read.csv("./data/pml-testing.csv",row.names=1,na.strings = "NA")

## Preprocessing data

We need to exclude the variables that have almost to zero variance in both training and testing data. Then exclude the columns with missing values to avoid problems in training models.

# remove near zero covariates using caret package, 75 variables remaining in both sets (from 159)  
nsv = nearZeroVar(training,saveMetrics=TRUE)  
training = training[,!nsv$nzv]  
testing = testing[,!nsv$nzv]  
  
# Remove variables with missing values, 58 variables reamining in training set  
training\_na = training[,(colSums(is.na(training)) == 0)]  
testing\_na = testing[,(colSums(is.na(testing)) == 0)]  
  
# Remove unnecessary columns, (53 cariables remaining for training, 52 variables for testing)  
string\_train = c("user\_name","raw\_timestamp\_part\_1","raw\_timestamp\_part\_2","cvtd\_timestamp","num\_window")  
string\_test = c("user\_name","raw\_timestamp\_part\_1","raw\_timestamp\_part\_2","cvtd\_timestamp","num\_window","problem\_id")  
training\_col = training\_na[,!(names(training\_na) %in% string\_train)]  
testing\_col = testing\_na[,!(names(testing\_na) %in% string\_test)]  
dim(training\_col)

## [1] 19622 53

dim(testing\_col)

## [1] 20 52

Now we split the preprocessed training data into training set and validation set.

intrain = createDataPartition(y=training$classe, p=0.7, list=FALSE)  
training\_clean = training\_col[intrain,]  
validation\_clean = training\_col[-intrain,]

In the new training set and validation set we just created, there are 52 predictors and 1 response. We heck the correlations between the predictors and the outcome variable in the new training set. There doesn’t seem to be any predictors strongly correlated (all of them are close to 0) with the outcome variable, so linear regression model is not recommnded. Random forest model may be more suitable for this data.

cor = abs(sapply(colnames(training\_clean[, -ncol(training)]), function(x) cor(as.numeric(training\_clean[, x]), as.numeric(training\_clean$classe), method = "spearman")))

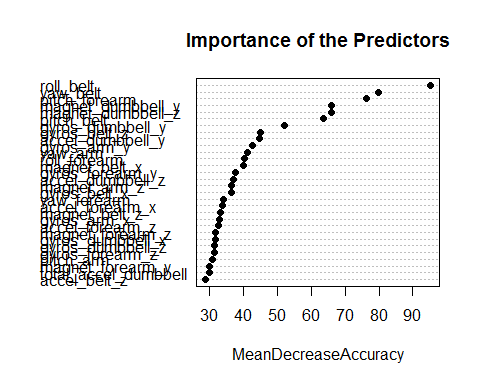
## Random Forest Model

We try to fit a random forest model and check the model performance on the validation set.

set.seed(4321)  
# Fit rf model  
rffit = train(classe ~ ., method = "rf", data = training\_clean, importance = T, trControl = trainControl(method = "cv", number = 4))  
validation\_pred = predict(rffit, newdata=validation\_clean)  
# Check model performance  
confusionMatrix(validation\_pred,validation\_clean$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1670 5 0 0 0  
## B 2 1131 8 0 0  
## C 1 2 1013 12 0  
## D 0 1 5 951 3  
## E 1 0 0 1 1079  
##   
## Overall Statistics  
##   
## Accuracy : 0.993   
## 95% CI : (0.9906, 0.995)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9912   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9976 0.9930 0.9873 0.9865 0.9972  
## Specificity 0.9988 0.9979 0.9969 0.9982 0.9996  
## Pos Pred Value 0.9970 0.9912 0.9854 0.9906 0.9981  
## Neg Pred Value 0.9990 0.9983 0.9973 0.9974 0.9994  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2838 0.1922 0.1721 0.1616 0.1833  
## Detection Prevalence 0.2846 0.1939 0.1747 0.1631 0.1837  
## Balanced Accuracy 0.9982 0.9954 0.9921 0.9923 0.9984

#check important variables  
imp = varImp(rffit)$importance  
varImpPlot(rffit$finalModel, sort = TRUE, type = 1, pch = 19, col = 1, cex = 1, main = "Importance of the Predictors")



The random forest algorithm generates a model with accuracy 0.9935. The top 4 most important variables are roll\_bell, yaw\_belt, pitch\_forearm and pitch\_belt.

## Prediction

The last step is to use the random forest model to predict on the testing set without the outcome variable and save the prediction output.

testing\_pred <- predict(rffit, newdata=testing\_col)  
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)  
 }  
}  
write\_files(testing\_pred)

## Results

We used 52 variables to build the random forest model with 4-fold cross validation.