Human Activity Recognition

We use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and our goal is to predict the manner in which the participants did barbell lifts, correctly and incorrectly in 5 different ways, denoted by the “classe” variable in the training set.

Loading Data

First we load the necessary packages and the training and testing datasets. There are 19,622 rows and 160 variables in the training set. A quick summary shows a lot of blank and NA values in several variables. The testing set contains just 20 rows and the same 160 variables as the training set.

```
library(caret)
library(randomForest)
training = read.csv('pml-training.csv',header=TRUE)
testing = read.csv('pml-testing.csv',header=TRUE)
```

Cleaning up

We focus on the 144 variables related to subject's sensors in the belt (36), arm (36), dumbbell (36) and forearm (36). For each group of variables corresponding to a body part we keep only those variables with no blank and no NA values, that is, 52 variables divided evenly among the 4 body parts. The outcome to be predicted is given by the variable 'classe'.

```
belt = training[,grep('_belt',names(training))]
belt = belt[,colSums(is.na(belt))==0]
belt = belt[,colSums(belt=='')==0]
arm = training[,grep('_arm',names(training))]
arm = arm[,colSums(is.na(arm))==0]
arm = arm[,colSums(arm=='')==0]
dumbbell = training[,grep('_dumbbell',names(training))]
dumbbell = dumbbell[,colSums(is.na(dumbbell))==0]
dumbbell = dumbbell[,colSums(dumbbell=='')==0]
forearm = training[,grep('_forearm',names(training))]
forearm = forearm[,colSums(is.na(forearm))==0]
forearm = forearm[,colSums(forearm=='')==0]
var.belt = colnames(belt)
var.arm = colnames(arm)
var.dumbbell = colnames(dumbbell)
var.forearm = colnames(forearm)
```

So we build our tidy training and testing sets as follows:

```
new.outcome = training['classe']
new.training = data.frame(belt,arm,dumbbell,forearm)
colnames(new.training) = c(var.belt,var.arm,var.dumbbell,var.forearm)
new.data = data.frame(new.training, classe=new.outcome)

new.testing = testing[,c(var.belt,var.arm,var.dumbbell,var.forearm)]
colnames(new.testing) = c(var.belt,var.arm,var.dumbbell,var.forearm)
```

Training

In order to do the training we choose the Random Forest method which is known to make highly accurate predictions. We set its number of trees to 200 and define a 3-fold cross-validation method in the train control.

```
set.seed(323)
(fit =
train(classe~.,data=new.data,method="rf",ntree=200,trControl=trainControl(method="cv",number=3,allowPara
```

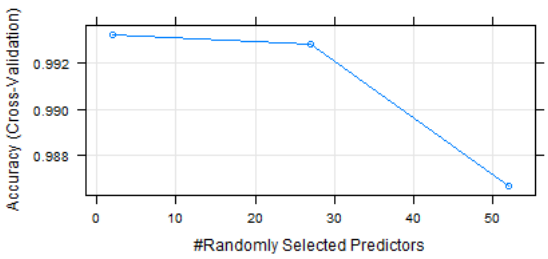
```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
##
## Summary of sample sizes: 13081, 13082, 13081
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9932219  0.9914255  0.001420210  0.001797697
##   27    0.9928652  0.9909748  0.001317996  0.001666936
##   52    0.9866984  0.9831728  0.005069296  0.006412598
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
(model = fit$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 200, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.48%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5578    2    0    0    0 0.0003584229
## B   14 3778    5    0    0 0.0050039505
## C    0   20 3400    2    0 0.0064289889
## D    0    0   41 3172    3 0.0136815920
## E    0    0    1    6 3600 0.0019406709
```

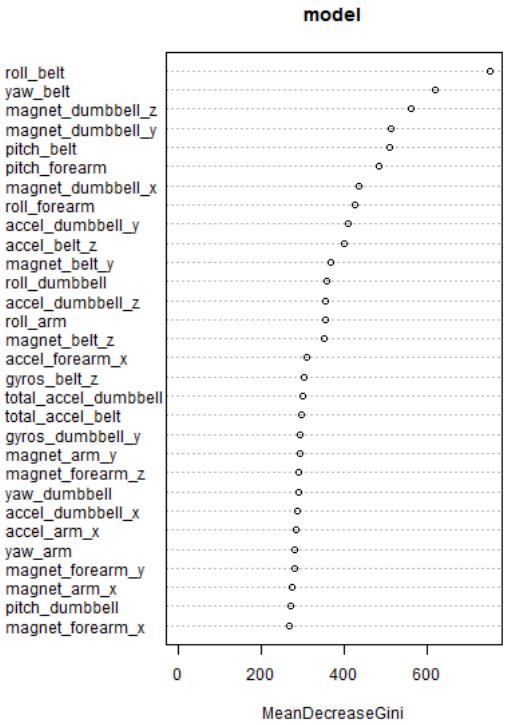
The selected model has mtry = 2 and an accuracy of 99.32219% as it can be seen in the following plot:

```
plot(fit)
```



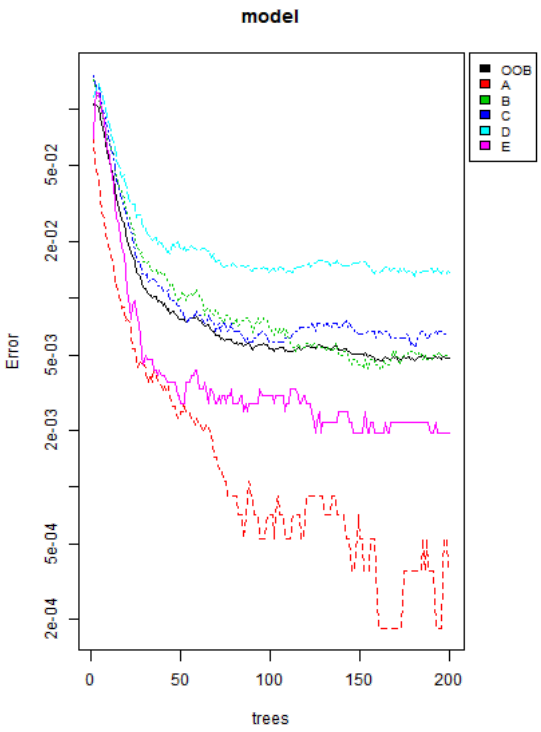
The figure below shows the importance of the variables:

```
varImpPlot(model)
```



The OOB error rate is 0.4790541% corresponding to 94 misclassifications. We should expect a higher error rate when doing real prediction with the testing set. The following plot shows the accuracy attained by the growing number of trees while predicting out of sample outcomes through cross-validation.

```
layout(matrix(c(1,2),nrow=1),
       width=c(4,1))
par(mar=c(5,4,4,0))
plot(model, log="y")
par(mar=c(5,0,4,2))
plot(c(0,1),type="n", axes=F, xlab="", ylab="")
legend("top", colnames(model$err.rate),col=1:6,cex=0.8,fill=1:6)
```



Prediction

Finally, we use our trained model in order to predict the outcomes for the testing set. Considering our OOB error rate, we should expect at least 1 misclassification every 208 cases.

```
(pred = predict(fit,new.testing))
```

```
##  [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
```

Our testing set has just 20 cases and, as it turns out, our model was able to predict correctly each and every one of them.