Practical Machine Learning Course Project — Find the classe of Exercise

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

```
  The goal of this project is to predict the manner in which they did the exercise. This is the
 "classe" variable in the training set. 20 different test cases are predicted using the model t
rained by learning from the train set.
```

Load and cleanse the data

Read the files

```
trainset<-read.csv("pml-training.csv",sep=",",header=TRUE)
testset<-read.csv("pml-testing.csv",sep=",",header=TRUE)
```

Clean the data: remove all the columns where any of the data are N.A.

```
library(data.table)
DT <- as.data.table(trainset)
trainset1<-data.frame(DT[,which(unlist(lapply(DT, function(x)!any(is.na(x))))),with=F])

DT1 <- as.data.table(testset)
testset1<-data.frame(DT1[,which(unlist(lapply(DT1, function(x)!any(is.na(x))))),with=F])
```

refine the train and test set to make sure all column names exist in both data frames. The first 7 columns should not be used for the training and prediction.

```
trainnames<-colnames(trainset1)[colnames(trainset1) %in% colnames(testset1)]
trainset2<-trainset1[c(trainnames)]
trainset2<-cbind(trainset2,trainset1["classe"])[,8:60]
testset2<-testset1[c(trainnames)][8:59]
```

Train the set and predict

train the trainset with classification tree

```
library("caret")
```

```
## Warning: package 'caret' was built under R version 3.1.1
```

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

```
set.seed(384)
modFit <- train(classe ~ ., data = trainset2, method="rpart")
```

```
## Loading required package: rpart
```

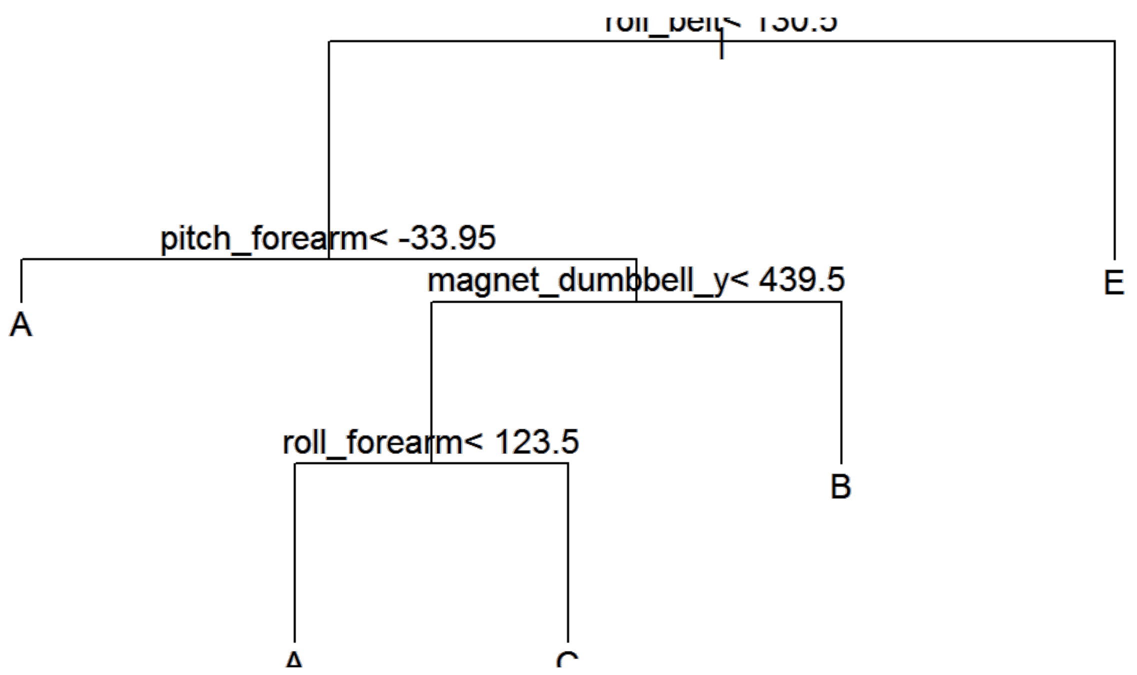
```
## Warning: package 'e1071' was built under R version 3.1.1
```

```
print(modFit, digits=3)
```

```
## CART
##
## 19622 samples
##    52 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 19622, 19622, 19622, 19622, 19622, 19622, ...
##
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0357  0.505     0.356   0.035        0.0585
##   0.06    0.412     0.201   0.063        0.105
##   0.115   0.34      0.0842  0.039        0.059
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0357.
```

plot the dicision tree

```
plot(modFit$finalModel)
text(modFit$finalModel)
```



cross validation of the fit model

```
ctrl <- trainControl(method = "cv", savePred=T, classProb=T,number = 5)
mod <- train(classe~., data=trainset2, method = "rpart", trControl = ctrl)
print(mod, digits=3)
```

```
## CART
##
## 19622 samples
##    52 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 15699, 15699, 15695, 15698, 15697
##
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa   Accuracy SD  Kappa SD
##   0.0357  0.5       0.347   0.0151       0.0208
##   0.06    0.468     0.295   0.0593       0.0988
##   0.115   0.332     0.0732  0.0439       0.0668
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0357.
```

predict the test set

```
print(predict(modFit, newdata=testset2))
```

```
##  [1] C A C A A C C A A A C C C A C A A A A C
## Levels: A B C D E
```

The best accuracy is not good enough. Try random forest It is too time consuming to train the whole data set. Subset the data

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.1
```

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

```
d_small <- createDataPartition(y=trainset2$classe, p=0.25, list=FALSE)
trainset2_small <- trainset2[d_small,]
set.seed(384)
modFit1 <- train(classe ~ ., data = trainset2_small, method="rf")
print(modFit1, digits=3)
```

```
## Random Forest
##
## 4907 samples
##   52 predictors
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 4907, 4907, 4907, 4907, 4907, 4907, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     0.96      0.95   0.00632      0.00802
##   27    0.964     0.954  0.0048       0.00609
##   52    0.95      0.936  0.00793      0.01
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
print(predict(modFit1, newdata=testset2))
```

```
##  [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
```

cross validation of the fit model

```
ctrl <- trainControl(method = "cv", savePred=T, classProb=T,number = 5)
mod1 <- train(classe~., data=trainset2_small, method = "rf", trControl = ctrl)
print(mod1, digits=3)
```

```
## Random Forest
##
## 4907 samples
##   52 predictors
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 3926, 3925, 3926, 3927, 3924
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     0.966     0.957  0.0063       0.00798
##   27    0.968     0.96   0.0101       0.0128
##   52    0.957     0.946  0.0136       0.0172
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Conclusions

predicted classe of activity

```
print(predict(modFit1, newdata=testset2))
```

```
##  [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
```

The error rate of the final model (random forest) used for the prediction is 1-0.974=0.026 via corss-validated resampling. The model seemed to be quite accurate in predicting the activity.