Predicting Weight Lifting Exercise manner

Sarwar Alam

12/7/2020

Introduction

Here we have used the Weight Lifting Exercise dataset for our experiment on predicting the manner of how the exercise was done.Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes

In our result we got a stunning accuracy of 99.34% on the unseen data and error rate: 0.71%

loading requred package

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.3
```

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

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

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.3
```

```
## 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
```

```
library(rpart)
library(partykit)
```

```
## Warning: package 'partykit' was built under R version 4.0.3

## Loading required package: grid

## Loading required package: libcoin

## Warning: package 'libcoin' was built under R version 4.0.3

## Loading required package: mvtnorm

## Warning: package 'mvtnorm' was built under R version 4.0.3
```

Data Preprocessing

load the Data

```
dat_train <- read.csv("~/pml-training.csv")
dat_test <- read.csv("~/pml-testing.csv")
```

Data Cleaning for the better fitting of ML models

```
table(complete.cases(dat_train))
```

```
##
## FALSE  TRUE
## 19216   406
```

Removing columns which are containing NA values entirely beacuse there are 19216 rows which contain NA values if we remove all the rows then we will have very less data and will not have a good fitted model Also there are many columns containing NA values heavily so it's better to remove them

```
dat_train <- dat_train[, colSums(is.na(dat_train))== 0]
dat_test <- dat_test[, colSums(is.na(dat_test))==0]
```

Getting rid of columns whcich are not much importance like "X" ,"raw_timestamp_part_1","raw_timestamp_part_2","cvtd
"new_window","num_window"

```
dat_train <- dat_train[, !grepl("^X|timestamp|window", names(dat_train))]
dat_test <- dat_test[, !grepl("^X|timestamp|window", names(dat_test))]
#keeping only numeric columns
classe<-dat_train$classe
dat_train <- dat_train[, sapply(dat_train, is.numeric)]
dat_train$classe <- classe
dat_test <- dat_test[, sapply(dat_test, is.numeric)]
```

creating a hold out data for measuring the performance of the ML models

```r
set.seed(125)
inTrain <- createDataPartition(dat_train$classe, p=0.72, list=F)
train_data <- dat_train[inTrain, ]
hold_data <- dat_train[-inTrain, ]
```

Fitting model

```r
cvCtrl <- trainControl(method ="cv",5)

c5Tune <- train(classe~.,data=train_data,method = "rf",ntree=200,
trControl = cvCtrl)
```

performance of the model on the unseen data

```r
pred <- predict(c5Tune, hold_data)
confusionMatrix(factor(hold_data$classe), pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1562    0    0    0    0
##          B    5 1057    1    0    0
##          C    0    8  944    6    0
##          D    0    0   11  888    1
##          E    0    0    1    4 1004
##
## Overall Statistics
##
##                Accuracy : 0.9933
##                  95% CI : (0.9907, 0.9953)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9915
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9968   0.9925   0.9864   0.9889   0.9990
## Specificity            1.0000   0.9986   0.9969   0.9974   0.9989
## Pos Pred Value         1.0000   0.9944   0.9854   0.9867   0.9950
## Neg Pred Value         0.9987   0.9982   0.9971   0.9978   0.9998
## Prevalence             0.2853   0.1939   0.1743   0.1635   0.1830
## Detection Rate         0.2844   0.1925   0.1719   0.1617   0.1828
## Detection Prevalence   0.2844   0.1936   0.1744   0.1639   0.1837
## Balanced Accuracy      0.9984   0.9956   0.9917   0.9931   0.9989
```

The overall accuracy of the model

```
accuracy<-confusionMatrix(factor(hold_data$classe), pred)$overall[[1]]*100
paste(as.character(round(accuracy,2)),'%',sep="")
```

```
## [1] "99.33%"
```

Predicting on the test data provided for the project

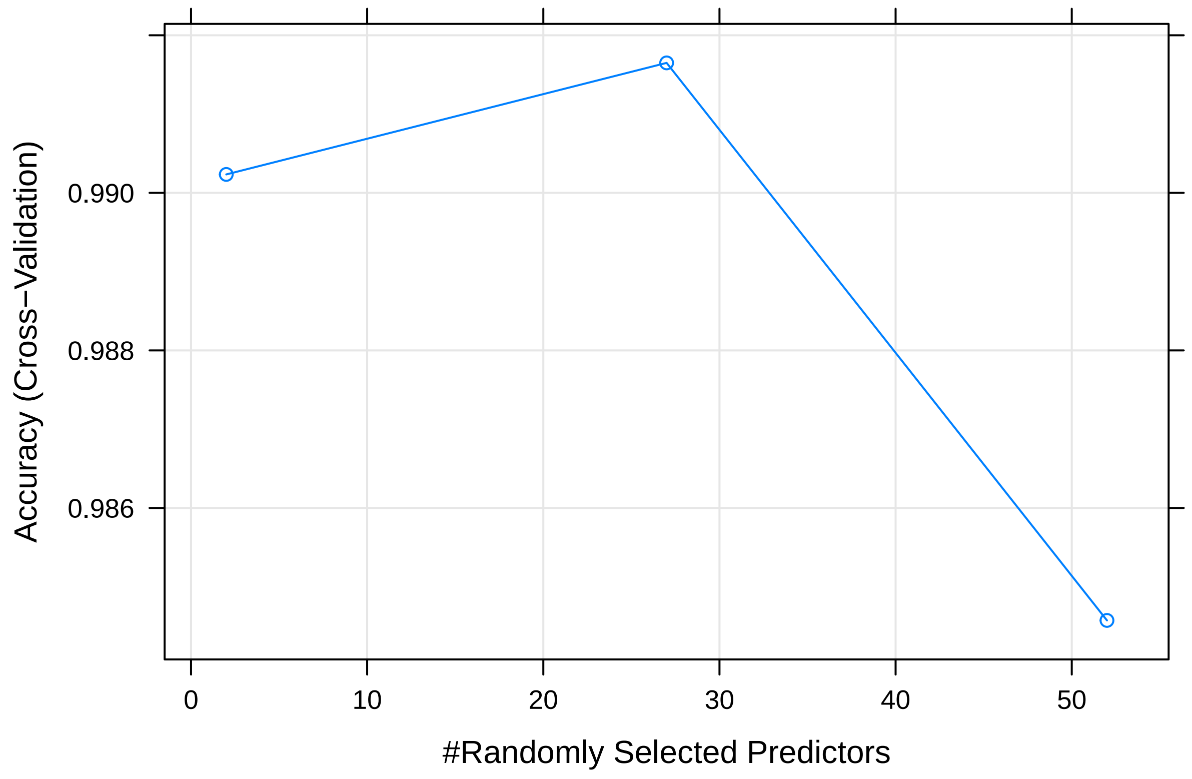
```
# there is no use of problem_id so get rid of that column

test_output<-predict(c5Tune,dat_test[,-length(dat_test)])
test_output
```

```
##  [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
```

Some figures on the founding of the results

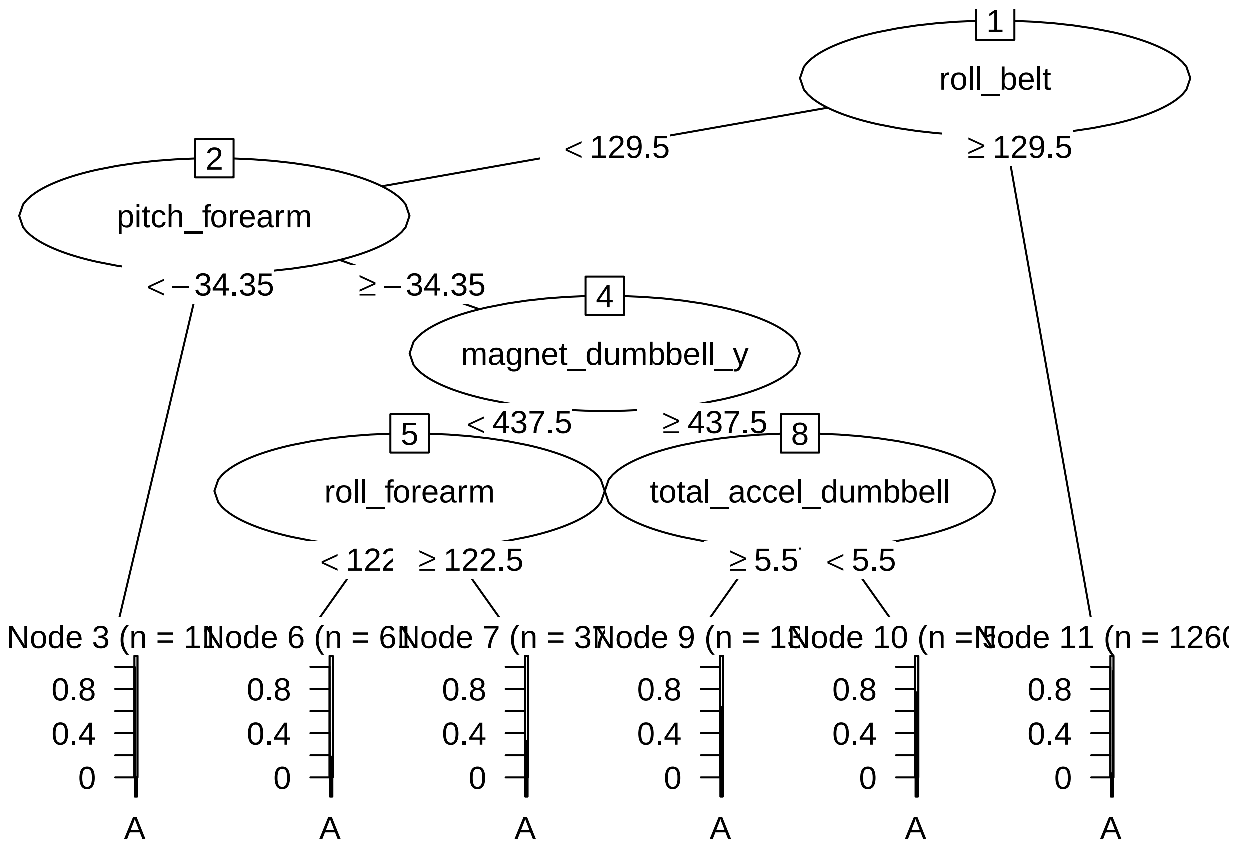
Figure pertaining to the cross validation accuracy on randomly selected predictors

```
plot(c5Tune)
```



We can convert the rpart object to a new class called party and plot it to see more in the terminal nodes

```r
tree <- rpart(classe ~ ., data = train_data,control = rpart.control(maxdepth = 4))
plot(as.party(tree))
```


1
roll_belt
< 129.5
≥ 129.5
2
pitch_forearm
< −34.35
≥ −34.35
4
magnet_dumbbell_y
< 437.5
≥ 437.5
5
roll_forearm
< 122
≥ 122.5
8
total_accel_dumbbell
≥ 5.5
< 5.5
Node 3 (n = 1
Node 6 (n = 6
Node 7 (n = 3
Node 9 (n = 1
Node 10 (n =
Node 11 (n = 126
0.8
0.4
0
A