Practical Machine Learning

bjpotrat

5/3/2019

Background Info

In this project, your goal will be to use data from accelerators on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Data-set). Create a model that predicts if the barbell lifts were done correctly.

Load Libraries and Data

```
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)
library(corrplot)
library(gbm)
```

Get Training Data

```
TrainData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-tra
ining.csv"),header=TRUE, na.strings=c("NA",""))
dim(TrainData)
```

```
## [1] 19622   160
```

Get Test Data

```
TestData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-test
ing.csv"),header=TRUE, na.strings=c("NA",""))
dim(TestData)
```

```
## [1]  20 160
```

Clean Data

Remove Columns where 97.5% or more is blank or NA. The first 7 columns can be remove as they do not provide useful data for modeling.

```
indColToRemove <- which(colSums(is.na(TrainData) |TrainData=="")>0.975*dim(TrainData)
[1])
TrainDataClean <- TrainData[,-indColToRemove]
TrainDataClean <- TrainDataClean[,-c(1:7)]
dim(TrainDataClean)
```

```
## [1] 19622    53
```

```
indColToRemove <- which(colSums(is.na(TestData) |TestData=="")>0.975*dim(TestData)[1])
TestDataClean <- TestData[,-indColToRemove]
TestDataClean <- TestDataClean[,-c(1:7)]
dim(TestDataClean)
```

```
## [1] 20 53
```

Create Training Sets and Test sets with a 0.6 and 0.75 ratio.

```
set.seed(12345)
InTrain1 <- createDataPartition(TrainDataClean$classe, p=0.75, list=FALSE)
Train1 <- TrainDataClean[InTrain1,]
Test1 <- TrainDataClean[-InTrain1,]
InTrain2 <- createDataPartition(TrainDataClean$classe, p=0.6, list=FALSE)
Train2 <- TrainDataClean[InTrain2,]
Test2 <- TrainDataClean[-InTrain2,]
```

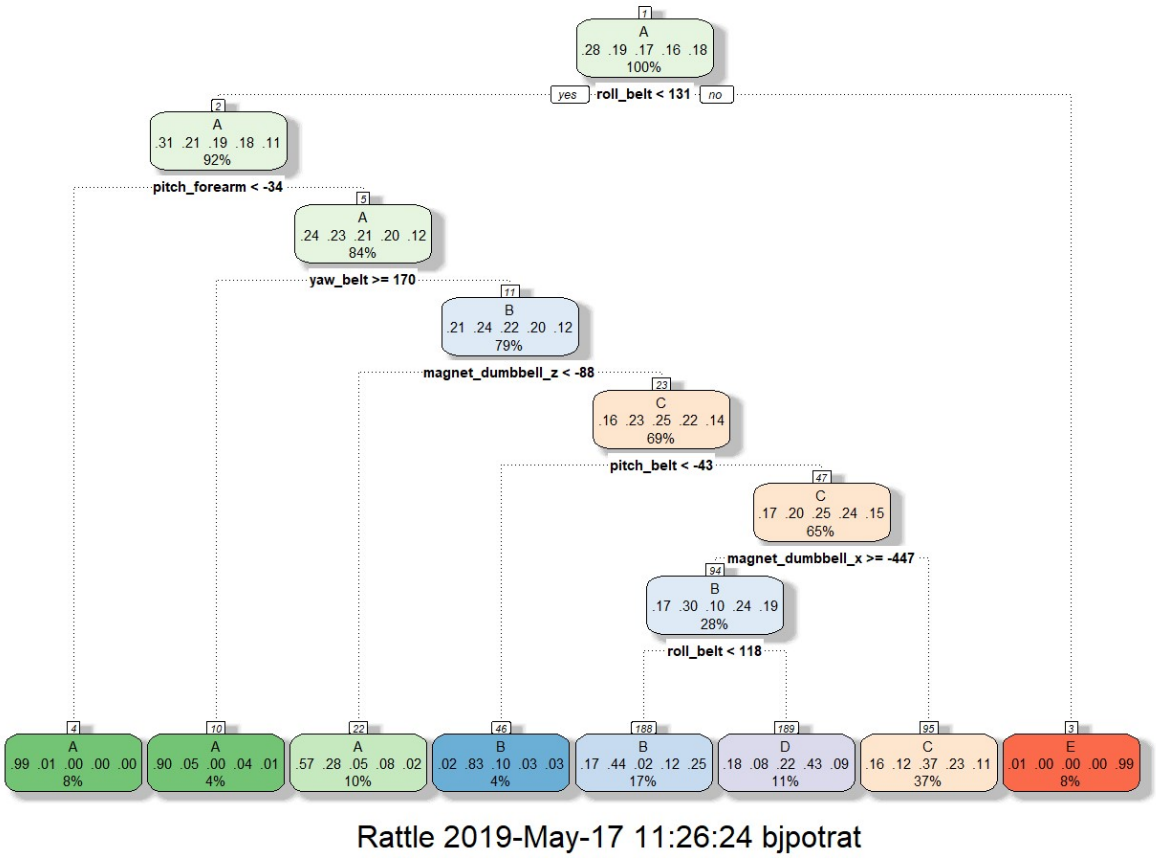
Model Creation and testing

I will compare the accuracy of the following three different models: Classification tree Random forest Gradient boosting

Classification Tree

Train the model with test set 1

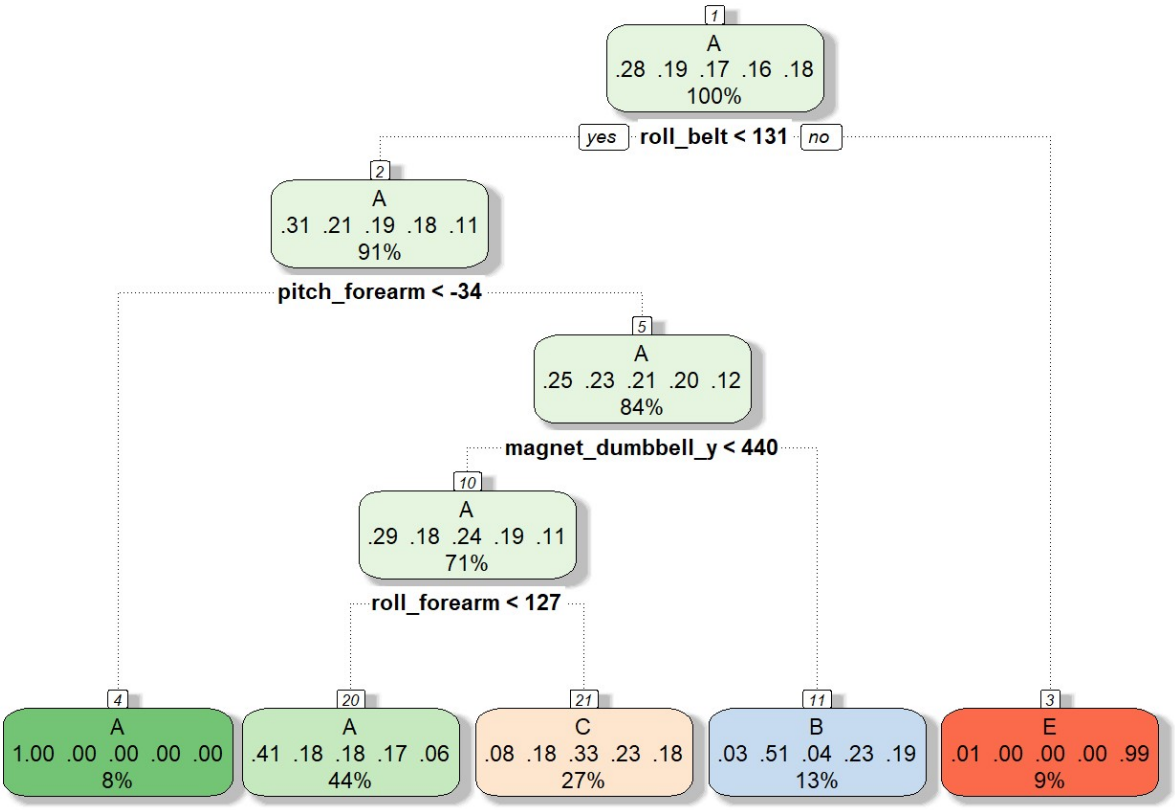
```
trControl <- trainControl(method="cv", number=5)
model_CT1 <- train(classe~., data=Train1, method="rpart", trControl=trControl)
fancyRpartPlot(model_CT1$finalModel)
```



Rattle 2019-May-17 11:26:24 bjpotrat

Train a model with test set 2 for sanity.

```
model_CT2 <- train(classe~., data=Train2, method="rpart", trControl=trControl)
fancyRpartPlot(model_CT2$finalModel)
```



There is a significant difference in the models generated with a small variation in the test-set size. Therefore this is probably not a great choice for prediction.

```
trainpred <- predict(model_CT1, newdata = Test1)
confusionMatrix(trainpred, Test1$classe)$overall[1]
```

```
##  Accuracy
## 0.5415987
```

The accuracy of our model is: 0.5416 which is not great.

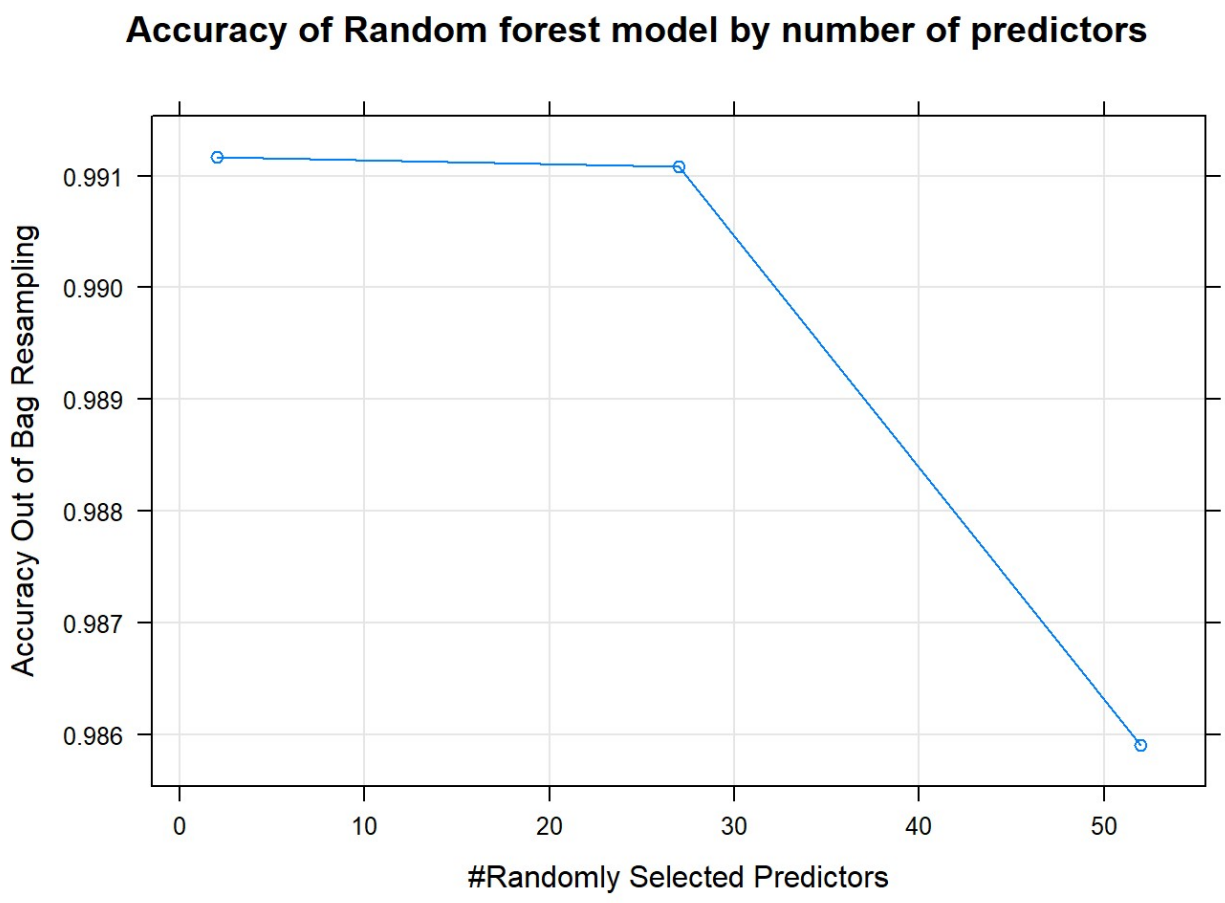
Random Forest

The Train2 and Test2 will be used for model evaluation as it is the preferred ratio.

```
trControl <- trainControl(method="oob", number=3)
model_RF <- train(classe~., data=Train2, method="rf", trControl=trControl, verbose=FAL
SE)
print(model_RF)
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9911685  0.9888280
##   27    0.9910836  0.9887204
##   52    0.9859035  0.9821660
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
plot(model_RF,main="Accuracy of Random forest model by number of predictors")
```



The optimal number of predictors is 2.

```
trainpred <- predict(model_RF,newdata=Test2)
confusionMatrix(trainpred, Test2$classe)$overall[1]
```

```
##  Accuracy
## 0.9900586
```

The random forest seems to be pretty accurate at 99%.

Gradient boosting method

```
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
model_GBM <- train(classe~., data=Train2, method="gbm", trControl=controlGBM, verbose=
FALSE)
model_GBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

This method does not look like it is working well based on the 52 variables and 52 are contributing to the model.

```
predictGBM <- predict(model_GBM, newdata=Test2)
confusionMatrix(predictGBM, Test2$classe)$overall[1]
```

```
##  Accuracy
## 0.9588325
```

Just checking the accuracy 95%.

Conclusion and Quiz Results

Therefore the final model I will use for prediction is Random Forest with a 0.99 accuracy.

In that case, the Random Forest model will be applied to predict the 20 quiz results (testing data-set) as shown below.

```
predictQuiz <- predict(model_RF, newdata=TestDataClean)
predictQuiz
```

```
##  [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
```