Practical_Machine_Learning_Project

Sihwan Kim

2020 3 29

Practical Machine Learning Project

Synopsis

The main section has three parts 1. Pre-analysis: Select meaningful variables to make a prediction. 2. Machine learning: Using a subset of training data (10% of the training set), we will compare accuracies of learning models using three methods - “CART”, “GBM”, and “Random Forest(RF)”.
3. Final model: The RF showed best accuracy, so with a subset of traning data (70% of the training set), we will make a final learning model and we will get the answer of testing set.

Pre-analysis (Exploring data)

First, we read csv files:

```
training0<-read.csv('pml-training.csv')
testing0<-read.csv('pml-testing.csv')
```

Next, columns with acceptable names are selected, and predictors with many NAs or near-zero variances are excluded.

```
library(caret)
```

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

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

```
dcol <- colnames(training0)
training <- training0[,grep("belt|forearm|arm|dumbbell",dcol)] # column names
training <- training [,colSums(is.na(training)) < 19000] # columns with NAs are excluded
dnzv <- nearZeroVar(training) # variables with near-zero variance
training <- training[,-dnzv]
training <- cbind(training, classe = training0$classe); rm(training0) # Training set
```

Machine learning

CART method

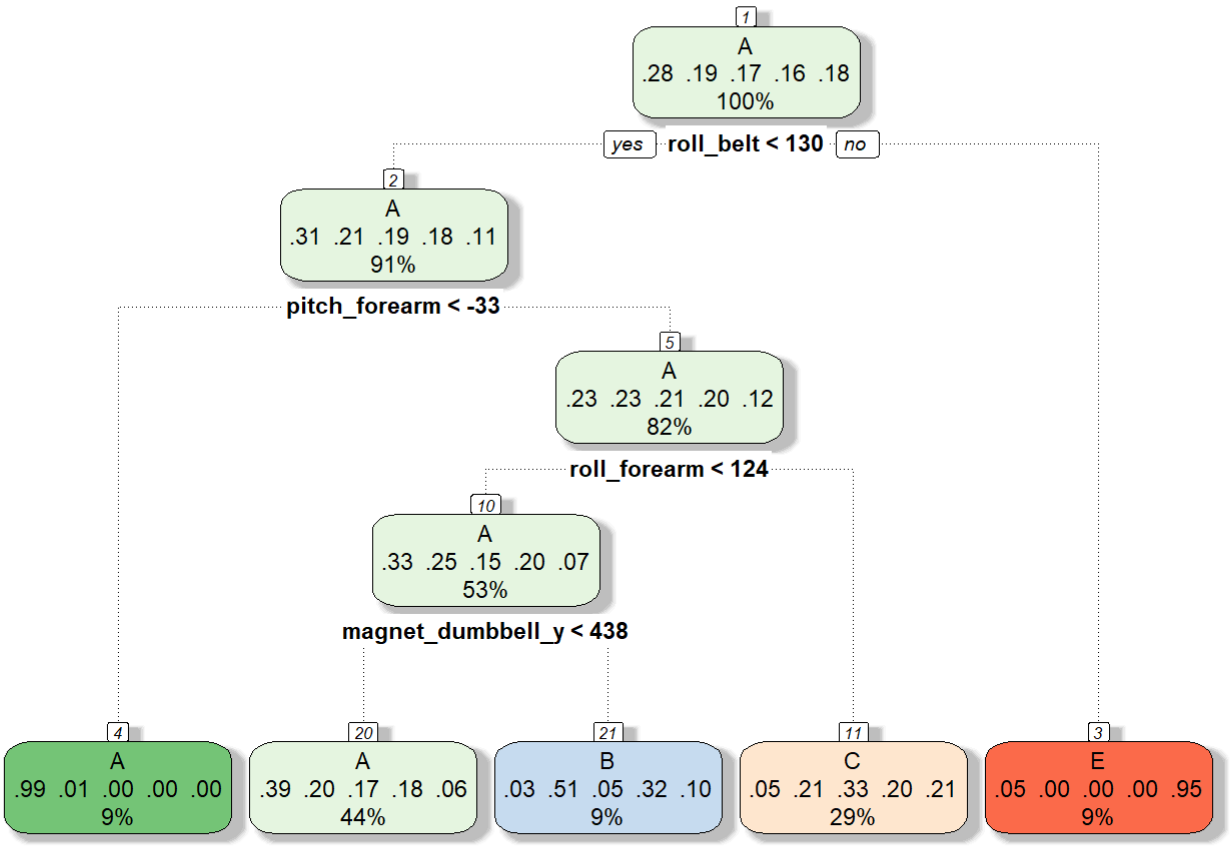
First, we try “CART” method on the training data set.
We have two sets: mytrain to make a prediction model, and mytesting to test models.

```
set.seed(8484)
inTrain<- createDataPartition(training$classe, p=0.1, list=FALSE)
inTesting<-createDataPartition(training$classe, p=0.1, list=FALSE)
mytrain <- training[inTrain,]
  mytesting <-training[inTesting,]
  fit_rp<- train(classe ~., data = mytrain, method = "rpart")
         # predict
  mypd <- predict(fit_rp, newdata = mytesting)
  rpaccuracy <- confusionMatrix(mypd,mytesting$classe)$overall['Accuracy']
```

```
  print(rpaccuracy)
```

```
##  Accuracy
## 0.4602851
```

```
    suppressMessages(library(rattle))
  fancyRpartPlot(fit_rp$finalModel)
```



Rattle 2020-3-30 11:32:22 김시환

As you see, the accuracy was not acceptable, try another model using “Gradient boost” method.

GBM method

```
fit_gb <-  train(classe ~., data = mytrain, method="gbm",verbose=FALSE)
mypd <- predict(fit_gb, newdata = mytesting)
gbaccuracy <- confusionMatrix(mypd,mytesting$classe)$overall['Accuracy']
```

And we see the acceptable accuracy of GBM model :

```
print(gbaccuracy)
```

```
##  Accuracy 
## 0.9424644
```

Random forset method

Next, we test the “Random Forest” method.

```
fitControl <- trainControl( method = "repeatedcv", number = 2)
fit_rf <- train(classe ~., data = mytrain, method = "rf", fitControl = trainControl )
mypd <- predict(fit_rf, newdata = mytesting)
rfaccuracy <- confusionMatrix(mypd,mytesting$classe)$overall['Accuracy']
```

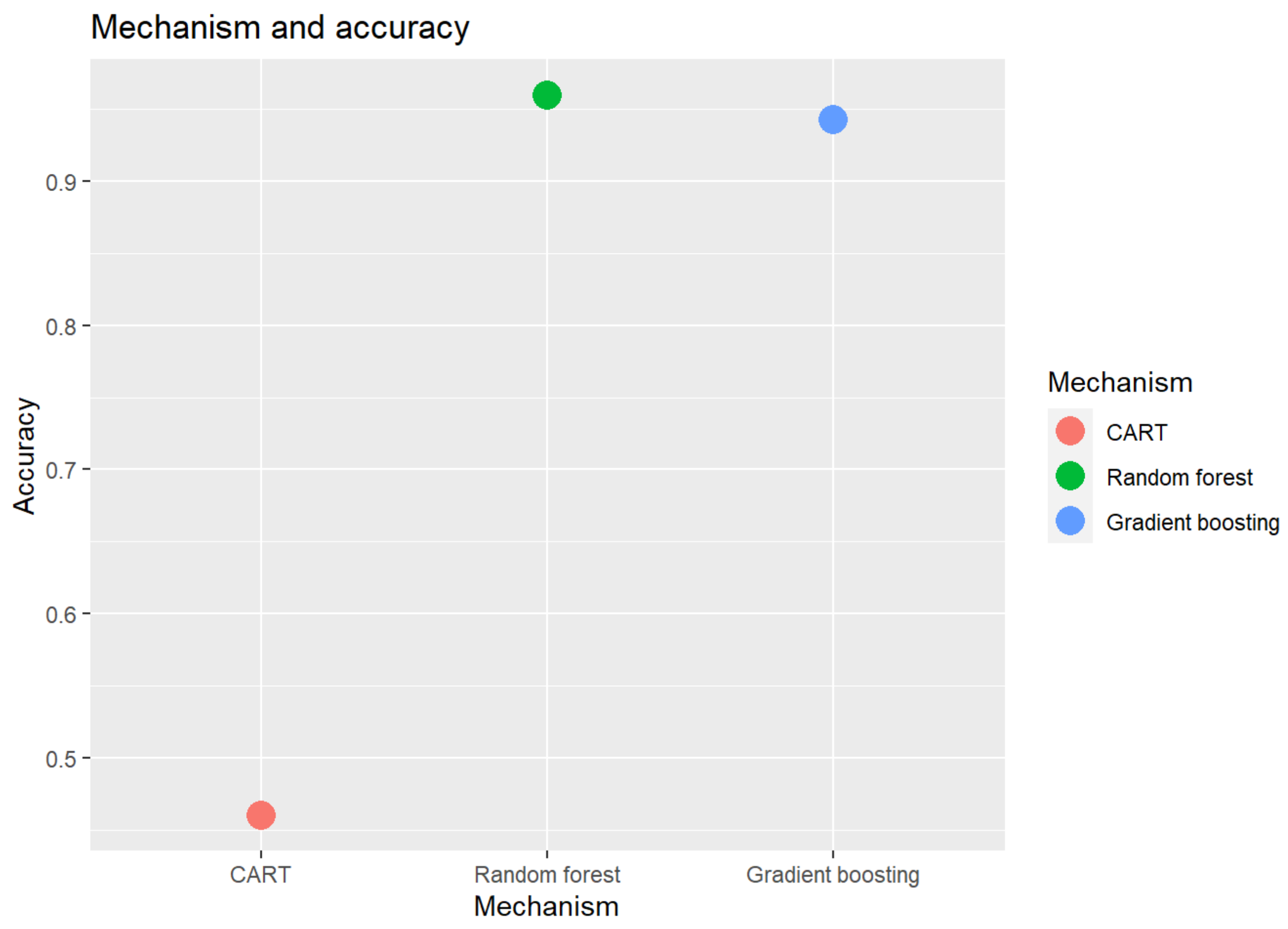
The accuracy of this model is:

```
print(rfaccuracy)
```

```
## Accuracy 
## 0.959776
```

We will simply compare accuracies of the three models.

```
miniaccu <- rbind( data.frame(Mechanism="CART", Accuracy = rpaccuracy),
                   data.frame(Mechanism="Random forest",Accuracy=rfaccuracy),
                  data.frame(Mechanism="Gradient boosting", Accuracy=gbaccuracy))
library(ggplot2)
ggplot(data=miniaccu, aes(x=Mechanism,y=Accuracy,colour=Mechanism)) +geom_point(size=5)+
  ggtitle("Mechanism and accuracy")+xlab("Mechanism")+ylab("Accuracy")
```



So “Random forest” generates the best prediction model.

Final model

```
set.seed(8484)
inTrain<- createDataPartition(training$classe, p=0.7, list=FALSE)
mytrain <- training[inTrain,]
mylearn <- train(classe ~., data = mytrain, method = "rf", fitControl = trainControl )
```

‘mylearn’ is the last prediction model with the training set.

```
library(randomForest)
varImpPlot(mylearn$finalModel, main = "Importance of Features in The Final Model")
```

So we have the final answer:

```
testing<-testing0[,colnames(training)[1:52]]  # To select appropriate columns
pdlearn <- predict(mylearn,testing) # Prediction of training samples
names(pdlearn)<-1:20
print(pdlearn)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```