Practical Machine Learning Course Project

iago lopez

23/10/2015

1. Loading packages, the training and testing datasets

```
###
library(caret)
```

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

```
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(rattle)
```

```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Versión 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

```
library(klaR)
```

```
## Loading required package: MASS
```

```
### Training data (emty values as NA)
tr1 <- read.csv("pml-training.csv",na.strings=c("NA",""), header=TRUE)

### Test data (emty values as NA)
te1 <- read.csv("pml-testing.csv", na.strings=c("NA",""), header=TRUE)
```

2. Cleaning the training and testing data

```
### Create a vector with length 160 and count the NA in every column
v1 <- vector(length = 160)
for(i in 1:160){v1[i] <- sum(is.na(tr1[,i]))}

v2 <- vector(length = 160)
for(i in 1:160){v2[i] <- sum(is.na(te1[,i]))}
### New training data set only with columns with 0 NA (60)
tr1 <- tr1[,which(v1==0)]
te1 <- te1[,which(v2==0)]

### Remove names, dates and ID
tr1 <- tr1[,c(7:60)]
te1 <- te1[,c(7:60)]
```

3. Divide training data into training and testing

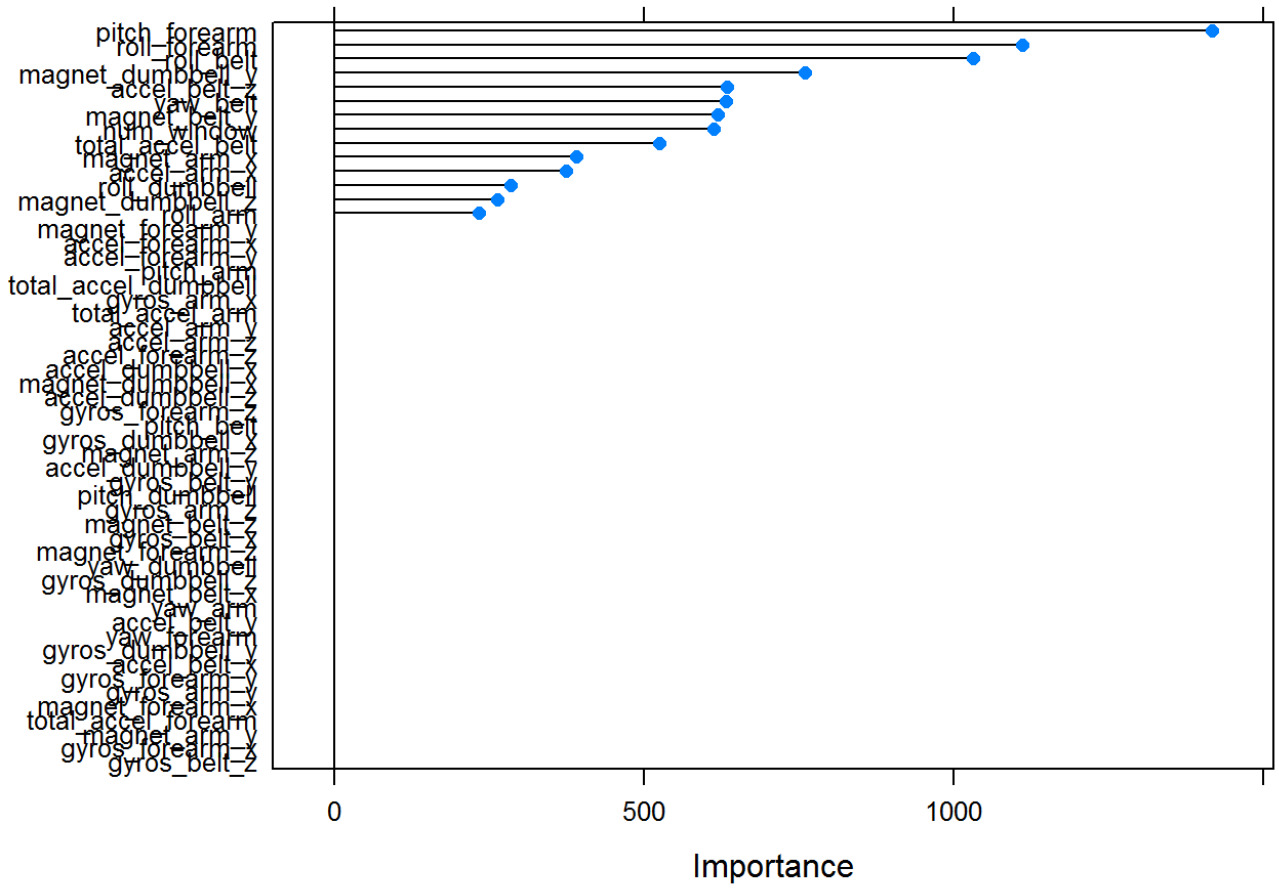
```
set.seed(99)
inTrain <- createDataPartition(y=tr1$classe, p=0.7, list = F)
training <- tr1[inTrain,]
testing <- tr1[-inTrain,]
```

4. Train the model

```
modFit1 <- train(classe ~ ., method = "rpart", data = training)
```

5. Estimate variable importance and see the list

```
importance <- varImp(modFit1, scale=FALSE)
plot(importance)
```



6. New train model with only 14 variables (highest importance)

```
training2 <- training[,c("pitch_forearm","roll_forearm","roll_belt","magnet_dumbbel
l_y","accel_belt_z","yaw_belt","num_window","magnet_belt_y","total_accel_belt","magne
t_arm_x","accel_arm_x","magnet_dumbbell_x","magnet_dumbbell_z","roll_arm","classe")]
modFit2 <- train(classe ~ ., method = "rpart", data = training2)
```

7. New test model in testing data

```
pre <- predict(modFit2,newdata = testing)
print(confusionMatrix(pre, testing$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1515  505  488  442  162
##          B   33  364   27  158  138
##          C  122  270  511  364  290
##          D    0    0    0    0    0
##          E    4    0    0    0  492
##
## Overall Statistics
##
##                Accuracy : 0.4897
##                  95% CI : (0.4769, 0.5026)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3323
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9050  0.31958  0.49805   0.0000  0.45471
## Specificity            0.6208  0.92499  0.78473   1.0000  0.99917
## Pos Pred Value         0.4868  0.50556  0.32820      NaN  0.99194
## Neg Pred Value         0.9427  0.84995  0.88101   0.8362  0.89052
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2574  0.06185  0.08683   0.0000  0.08360
## Detection Prevalence   0.5288  0.12234  0.26457   0.0000  0.08428
## Balanced Accuracy      0.7629  0.62228  0.64139   0.5000  0.72694
```

Accuracy of 48,97%. VERY POOR…

8. Use random forest with cross validation to create a new model

```
modFit3 <- train(classe ~ ., method="rf", trControl=trainControl(method = "cv", number
= 4), data=training2)
```

9. Try this new model in the test data set

```
pre2 <- predict(modFit3, newdata=testing)
print(modFit3$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 0.18%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    3 2649    6    0    0 0.003386005
## C    0    3 2393    0    0 0.001252087
## D    0    0    4 2248    0 0.001776199
## E    0    1    0    8 2516 0.003564356
```

```
print(confusionMatrix(pre2, testing$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1135    0    0    1
##          C    0    0 1026    1    0
##          D    0    0    0  963    5
##          E    0    0    0    0 1076
##
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   1.0000   0.9990   0.9945
## Specificity            0.9991   0.9998   0.9998   0.9990   1.0000
## Pos Pred Value         0.9976   0.9991   0.9990   0.9948   1.0000
## Neg Pred Value         1.0000   0.9992   1.0000   0.9998   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1743   0.1636   0.1828
## Detection Prevalence   0.2851   0.1930   0.1745   0.1645   0.1828
## Balanced Accuracy      0.9995   0.9981   0.9999   0.9990   0.9972
```

Accuracy of 99,81%. The out of sample error is the “error rate you get on new data set.”

Random Forest (preprocessing and cross validation) Testing Set:

1-0.9981 = 0.0019

10. Run the model against the 20 TEST set of the beggining.

```
pre3 <- predict(modFit3, newdata=te1)
print(pre3)
```

```
##  [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
```