Practical Machine Learning

safuan

December 25, 2015

Prediction Assignment Writeup

This project is to take 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.

Data Set up

Set up the environment using the following chunk

```
pml_testing <- read.csv(file="pml-testing.csv",head=TRUE,sep=",")
pml_training <- read.csv(file="pml-training.csv",head=TRUE,sep=",")
```

Let's have a look on the 'pml_training' dataset

```
str(pml_training)
```

We have a total number of 19,622 observations. Let's slice them onto training and testing data sets.

```
library(caret)
```

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

```
inTrain <- createDataPartition(y=pml_training$classe,
                               p=0.6,
                               list = FALSE)
training <- pml_training[inTrain,]
testing <- pml_training[-inTrain,]
summary(training)
```

Preprocessing

The barbell exercises as mentioned above, are classed to 5 different ways. Data is gathered through the data sets, where the column names are suffixes with _belt, _arm, _dumbbell and _forearm. We have shown using 'summary(training)' that column 1 is basically just the row numbers and column 2 contains the user names, which both are actually not required in training the predictor. So are column 3 to column 7.

1. Let's just get the column with the data from accelerometers only. However, column 160 is required since it is the **classe** column.

```
accelerometers <- grep(pattern = "_belt|_arm|_dumbbell|_forearm", names(training))
training <- training[,c(accelerometers,160)]
```

2. Now, let's run nearZeroVar the eradicate the variables which have little variabilities and hence should not be used as predictors.

```
nsv <- nearZeroVar(training,saveMetrics = TRUE)
training <- training[,!nsv$nzv]
```

3. We can see from the summary command that there are a lot of NAs in the data set. Here, we will omit any column with ~85% of NAs (>10000 NAs).

```
training <- training[,colSums(is.na(training))<10000]
```

Prediction

Let's use Random Forest classifier, with 5-fold cross validations. (Random Forest - out sample error should be very minimal and accuracy is exceptional). A better machine can even use a larger cross valiations to increase accuracy.

```
require(randomForest)
set.seed(2016)
RFmodFit <- train(training$classe~.,data = training, method="rf",
                  trControl=trainControl(method = "cv",number = 5))
RFmodFit
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9422, 9421, 9420, 9421, 9420
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9869225  0.9834555  0.003225740  0.004081619
##   27    0.9880264  0.9848533  0.001955844  0.002474838
##   52    0.9809777  0.9759385  0.003899040  0.004933205
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
RFmodFit$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: 27
##
##         OOB estimate of  error rate: 0.98%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3340    4    3    0    1 0.002389486
## B   19 2246   13    1    0 0.014480035
## C    0   13 2031   10    0 0.011197663
## D    0    2   32 1894    2 0.018652850
## E    0    2    4    9 2150 0.006928406
```

We now have the predictor RFmodFit. Apply it to the testing data set.

```
RFpredict <- predict(RFmodFit,newdata=testing)
confusionMatrix(RFpredict,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    6    0    0    0
##          B    1 1509   12    0    0
##          C    0    3 1354    7    2
##          D    0    0    2 1279    1
##          E    0    0    0    0 1439
##
## Overall Statistics
##
##                Accuracy : 0.9957
##                  95% CI : (0.9939, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9945
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9941   0.9898   0.9946   0.9979
## Specificity            0.9989   0.9979   0.9981   0.9995   1.0000
## Pos Pred Value         0.9973   0.9915   0.9912   0.9977   1.0000
## Neg Pred Value         0.9998   0.9986   0.9978   0.9989   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1923   0.1726   0.1630   0.1834
## Detection Prevalence   0.2851   0.1940   0.1741   0.1634   0.1834
## Balanced Accuracy      0.9992   0.9960   0.9940   0.9970   0.9990
```

Using random forest we achieve a model with accuracy 99%

Use the prediction on the pml_testing data set

We now have the predictor RFmodFit. Apply it to the pml_testing data set.

```
RFpredict_pml <- predict(RFmodFit,newdata=pml_testing)
RFpredict_pml
```

```
##  [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
```

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
```

Results

We have used the random forest model 5-fold cross validation. The out-of-sample error is very small.

we can now submit the project result.

```
pml_write_files(RFpredict_pml)
```