Practical machine learning course project

Leonardo Windlin Cesar

02/22/2016

Setting things up

The first step is really basic. We'll just set things up like loading libraries, setting the work directory and dowloading the data sets. **Important:** when reading the data, we'll assign the non available (NA) to be equal to blank and *NA*, so that when we clean the data we'll remove both *NA* and empty variables.

```
#Load the caret package and set the directory
library(caret)
```

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

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

```
setwd("C:/Users/Leonardo/Documents/GitHub")

#Download files and read the data sets
fileUrltrain<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
fileUrltest<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url=fileUrltrain,destfile = "./pml-training.csv", mode='wb')
download.file(url=fileUrltest,destfile = "./pml-testing.csv", mode='wb')
traindata<-read.csv("pml-training.csv",na.strings=c("", "NA"))
testdata<-read.csv("pml-testing.csv",na.strings=c("", "NA"))
```

Cleaning the data sets

Now it's a really important step: we'll clean the data set, i.e., we'll remove variables with *NA*, highly correlated variables and variables with zero or near zero variance. We also remove variables that aren't useful for prediction, like the user names.

```
# Remove variables with NAs
traindata<-traindata[,colSums(is.na(traindata)) == 0]

# Remove variables that can't be used as predictors
traindata<-subset(traindata,select=-c(X, user_name,raw_timestamp_part_1,
                                      raw_timestamp_part_2,cvtd_timestamp,
                                      new_window, num_window))

# Remove variables with zero or near zero variance
zerovar<-nearZeroVar(traindata[sapply(traindata, is.numeric)],
                     saveMetrics=TRUE)
traindata<-traindata[,zerovar[,'nzv']==0]

# Remove highly correlated variables, I'll use 0.8 as cutoff
corre<-cor(traindata[sapply(traindata,is.numeric)])
```

```
highcorre<-findCorrelation(corre,cutoff = 0.8)
highcorre<-sort(highcorre)
traindata<-traindata[,-c(highcorre)]
```

Training the model

Now for the actual modeling. We'll use a random forest model due to the nature of the problem. Note that this step may take a while (about ten minutes). We also use four fold croos validation in order to have a accurate model.

Make training set and test set

We make training and test sets out of the clean data set. The training set has 60% of the original observations and test set with the remaining.

```
i <- createDataPartition(traindata$classe, p=0.6, list=FALSE )
training <- traindata[i,]
testing <- traindata[-i,]
```

Random forest

```
rfcontrol<-trainControl(method="cv", 4)
modfit<-train(classe ~ .,data=training,method='rf',trControl=rfcontrol)
print(modfit)
```

```
## Random Forest
##
## 11776 samples
##    39 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 8832, 8832, 8831, 8833
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9844601  0.9803382  0.001946319  0.002463313
##   20    0.9866683  0.9831329  0.002733071  0.003460933
##   39    0.9732514  0.9661601  0.004025276  0.005086466
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 20.
```

As we can see, we come to a model with 20 predictor (mtr) resulting in 99.3% accuracy.

Testing the model

Finnaly we test the model in the training set.

```
predict <- predict(modfit, testing)
confusionMatrix(testing$classe, predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    0    0    0    1
##          B   14 1496    7    0    1
##          C    0   19 1344    5    0
##          D    3    0   12 1271    0
##          E    0    0    6    1 1435
##
## Overall Statistics
##
##                Accuracy : 0.9912
##                  95% CI : (0.9889, 0.9932)
##     No Information Rate : 0.2865
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9889
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9924   0.9875   0.9817   0.9953   0.9986
## Specificity            0.9998   0.9965   0.9963   0.9977   0.9989
## Pos Pred Value         0.9996   0.9855   0.9825   0.9883   0.9951
## Neg Pred Value         0.9970   0.9970   0.9961   0.9991   0.9997
## Prevalence             0.2865   0.1931   0.1745   0.1628   0.1832
## Detection Rate         0.2843   0.1907   0.1713   0.1620   0.1829
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9961   0.9920   0.9890   0.9965   0.9988
```

```
accuracy <- postResample(predict, testing$classe)
print(accuracy)
```

```
##  Accuracy     Kappa
## 0.9912057 0.9888733
```

On the training set, we get 98.9% accuracy.