Practical Machine Learning Course Project

Somtirtha Roy

January 31st, 2016

Introduction

This project is set up for us to experiment on the Weight Lifting Exercise Dataset. It has data from sensors like Jawbone Up, Nike FuelBand, and Fitbit. These are all fitness wear that collect data regarding the activity of the person wearing them. All this information can be harnessed to extract information regarding a person's health, behaviour and patterns in their daily activities. the data set which can be found at: http://groupware.les.inf.puc-rio.br/har contains data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They perform barbell lifts in five different ways each way classified according the level of precision.

We are trying to train a model to learn from the different features so that we can predict if a person's form is correct or not just by analysing the data sent from the sensors on the person.

Analysis

We import the two main packages we will need to perform our analysis.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3
```

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

```
library(corrplot)
```

Reading the training and test datasets.

```
# read the training and test sets
train_data = read.csv("data/pml-training.csv")
test_data = read.csv("data/pml-testing.csv")
names(train_data)
```

Preprocess the data

Cleaning and weeding out uneccessary information(noise) from the datasets. Getting rid of NA values.

```
# clean data
train_data = train_data[ , colSums(is.na(train_data)) == 0]
test_data = test_data[ , colSums(is.na(test_data)) == 0]
```

Getting rid of attributes that don't add any valuable information. Features with text values, starting with X, timestamp and window.

```
classe = train_data$classe
train_exclude = grepl("^X|timestamp|window", names(train_data))
train_data = train_data[, !train_exclude]
train_data = train_data[, sapply(train_data, is.numeric)]
train_data$classe = classe

test_exclude = grepl("^X|timestamp|window", names(test_data))
test_data = test_data[, !test_exclude]
if( length(sapply(test_data, is.numeric)) ) {
  test_data = test_data[, sapply(test_data, is.numeric)]
}
```

Splitting data

Splitting the training data into training and validation sets.

```
# split the training data into training and validation sets
set.seed(22519) # For reproducibile purpose
inTrain = createDataPartition(train_data$classe, p=0.70, list=F)
training_data = train_data[inTrain, ]
validation_data = train_data[-inTrain, ]
```

Training the classifier

We build a model using random forests and use cross validation in order to select the optimum number of and most relevant features for the target variable. We use a 5-fold cross validation and use a mximum of 250 trees for our purpose.

```
# training a classifier
# do cross validation
rf_model = train(classe ~ ., method="rf", trControl=trainControl(method="cv", 5), ntree=250, data=train
```

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

```
rf_model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 250, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.73%
```

```
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    3    3    0    0 0.001536098
## B   20 2628    9    1    0 0.011286682
## C    0   13 2375    8    0 0.008764608
## D    0    1   26 2222    3 0.013321492
## E    0    1    4    8 2512 0.005148515
```

Predicting and evaluation

We predict the data using the model on the validation set and calulate the accuracy of the model on the same data.

```
# predict data on test set
validation_pred = predict(rf_model, validation_data)

# make confusion matrix of the results
confusionMatrix(validation_data$classe, validation_pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    1
##          B    5 1129    5    0    0
##          C    0    1 1020    5    0
##          D    0    0   13  949    2
##          E    0    0    1    6 1075
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9908, 0.9951)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9914
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9982   0.9817   0.9885   0.9972
## Specificity            0.9995   0.9979   0.9988   0.9970   0.9985
## Pos Pred Value         0.9988   0.9912   0.9942   0.9844   0.9935
## Neg Pred Value         0.9988   0.9996   0.9961   0.9978   0.9994
## Prevalence             0.2850   0.1922   0.1766   0.1631   0.1832
## Detection Rate         0.2841   0.1918   0.1733   0.1613   0.1827
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9983   0.9981   0.9902   0.9927   0.9979
```

```r
accuracy = postResample(validation_pred, validation_data$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9932031 0.9914024
```

Results

As we can see the accuracy comes out to be 99.32 %

Predicting on the test data set

```r
# test on the test data set
test_pred = predict(rf_model, test_data)
test_pred
```

```
##  [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
```

Observation

We observe the results from the test data set and from the data set see that all the data gets predicted/classified correctly.