Santana's Practical Machine Learning Final Project

Marievee Santana

11/17/2021

Practical Machine Learning

Overview

This report is the final project for the Practical Machine Learning class offered by John's Hopkins University via Coursera. I

The objective of the project is to predict the manner in which 6 participants performed a variety of different exercises using an existing data set. The exercise type is captured in the "classe" variable in the training set. To complete the class requirements, ultimately the machine learning algorithm described here will be applied to the 20 test cases available in the test data and the predictions are submitted in appropriate format to the Course Project Prediction Quiz for automated grading.

Project Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use 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. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

The Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Packages Needed for Data Analysis

First, given the complexity of the analysis included here it is necessary to load several packages that will be needed to complete all the steps for this project.

```
library(knitr)
library(lattice)
library(ggplot2)
library(kernlab)
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)
library(corrplot)
library(Hmisc)
library(gmodels)
set.seed(1345)
```

Data Access & Clean Up

Loading the datasets.

```
setwd("~/Desktop/SantanaHardDisk/Courseracourses/courses/08_PracticalMachineLearning/FinalProject")
traincsv <- read.csv("pml-training.csv")
testcsv <- read.csv("pml-testing.csv")
print("dimensions of training and test sets"); c(dim(traincsv), dim(testcsv))
```

```
## [1] "dimensions of training and test sets"
```

```
## [1] 19622   160    20   160
```

Cleaning the data by removing near zero and missing observations from the training set.

```
traincsv <- traincsv[,colMeans(is.na(traincsv)) < .9] #removing na
traincsv <- traincsv[,-c(1:7)] #removing irrelevant metadata
nvz <- nearZeroVar(traincsv)
traincsv <- traincsv[,-nvz]
dim(traincsv)
```

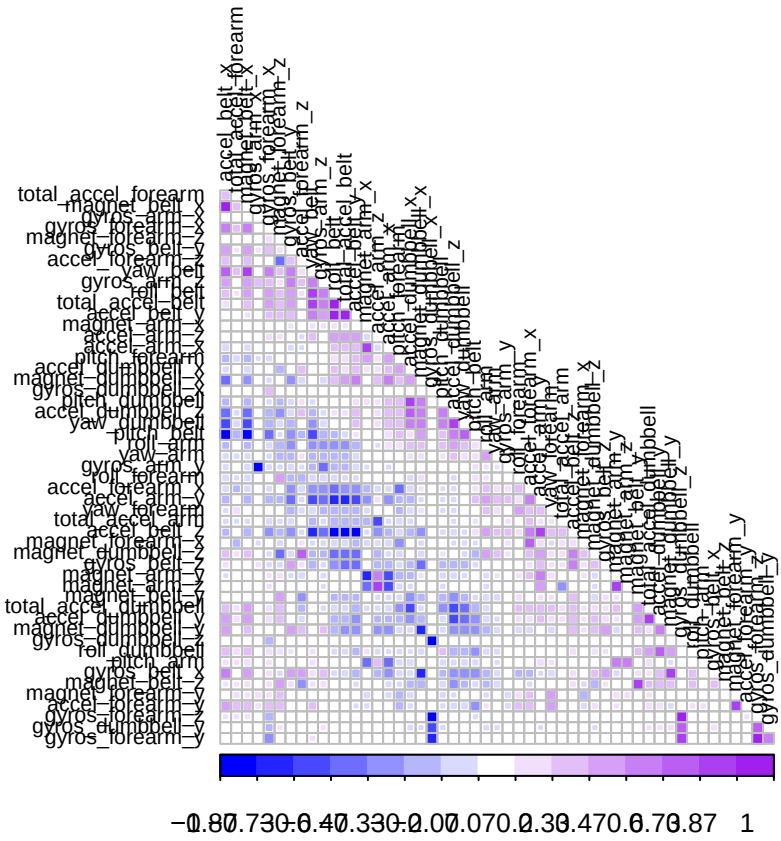
```
## [1] 19622    53
```

The next step is to split up the training set into a sub-training set and a validation set. The split used is 70:30. 70% training set & 30% validation set.

```
inTrain <- createDataPartition(y=traincsv$classe, p=0.7, list=F)
train <- traincsv[inTrain,]
valid <- traincsv[-inTrain,]
```

Before starting the modeling work it is good practice to run correlation analysis to get a better feel for the available data in the training set.

```
corMatrix <- cor(train[, -53])
corrplot(corMatrix, order = "AOE", method = "square", type = "lower", diag = FALSE, col= colorRampPalet
```



Higher correlation values are denoted by larger darker squares, where negative values are red and positive values are blue. Because it looks like there are several variables that are highly correlated with one another, I will proceed to remove variables with a high correlation (i.e., higher than +/- 0.90).

```
c <- findCorrelation(corMatrix, cutoff = .90)
train <- train[,-c]
valid <- valid[,-c]
```

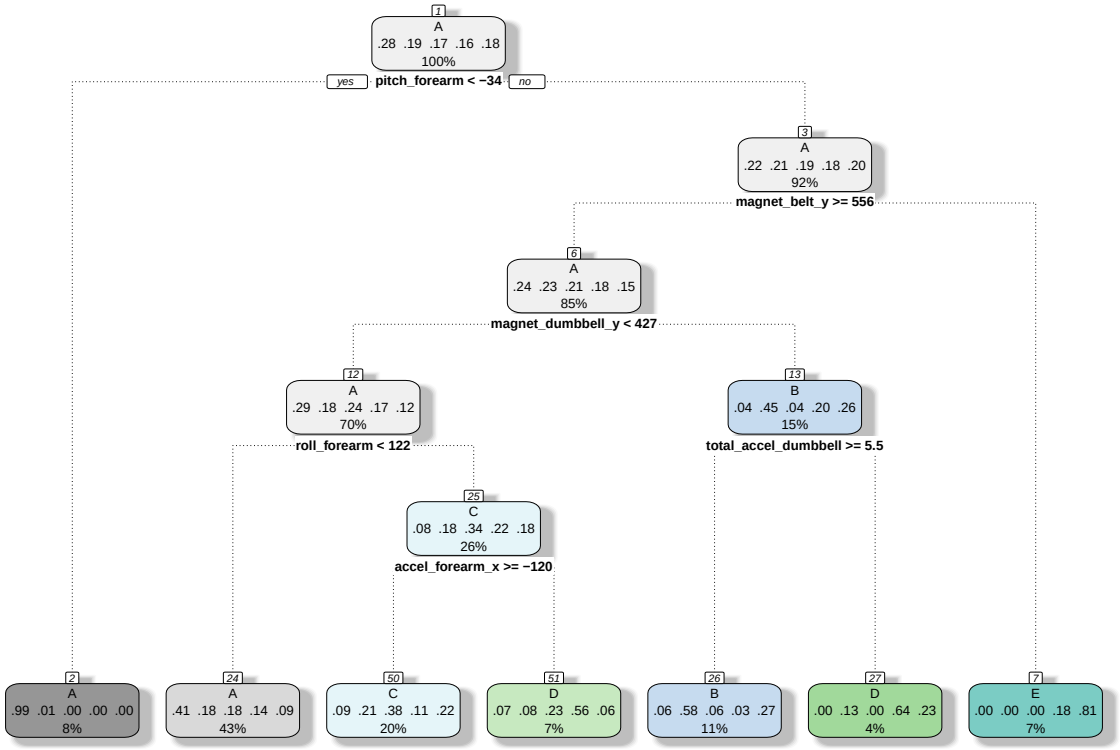
Creating the Models

I will explore 3 popular machine learning algorithms: Decision Trees, Random Forest, and Support-Vector Machines (SVM for short). I will use the same control for training to use 3-fold cross validation

```
control <- trainControl(method="cv", number=3, verboseIter=F)
```

Decision Trees

```
modTrees <- train(classe~., data=train, method="rpart", trControl = control, tuneLength = 5)
fancyRpartPlot(modTrees$finalModel, palettes=c("Greys", "Blues", "BuGn", "Greens", "GnBu"))
```



Rattle 2021−Nov−18 10:11:37 pg

```
predTrees <- predict(modTrees, valid)
cmTrees <- confusionMatrix(predTrees, factor(valid$classe))
cmTrees
```

Decision Trees Prediction

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1525  448  482  376  234
##          B   35  358   38   19  141
##          C   96  261  424  125  252
##          D   17   72   82  360   83
##          E    1    0    0   84  372
##
## Overall Statistics
##
##                Accuracy : 0.5164
##                  95% CI : (0.5035, 0.5292)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.3693
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9110  0.31431  0.41326  0.37344  0.34381
## Specificity            0.6343  0.95091  0.84894  0.94838  0.98230
## Pos Pred Value         0.4976  0.60575  0.36615  0.58632  0.81400
## Neg Pred Value         0.9472  0.85247  0.87265  0.88541  0.86920
## Prevalence             0.2845  0.19354  0.17434  0.16381  0.18386
## Detection Rate         0.2591  0.06083  0.07205  0.06117  0.06321
## Detection Prevalence   0.5208  0.10042  0.19677  0.10433  0.07766
## Balanced Accuracy      0.7726  0.63261  0.63110  0.66091  0.66306
```

Random Forest

```
modRF <- train(classe~., data=train, method="rf", trControl = control, tuneLength = 5)

predRF <- predict(modRF, valid)
cmRF <- confusionMatrix(predRF, factor(valid$classe))
cmRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    4    0    0    0
##          B    1 1131    5    0    0
##          C    1    4 1018   12    0
##          D    0    0    3  951    0
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9925, 0.9964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9930   0.9922   0.9865   1.0000
## Specificity            0.9991   0.9987   0.9965   0.9994   0.9998
## Pos Pred Value         0.9976   0.9947   0.9836   0.9969   0.9991
## Neg Pred Value         0.9995   0.9983   0.9984   0.9974   1.0000
```

```
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1922   0.1730   0.1616   0.1839
## Detection Prevalence   0.2848   0.1932   0.1759   0.1621   0.1840
## Balanced Accuracy      0.9989   0.9959   0.9944   0.9930   0.9999
```

Support-Vector Machines (SVM)

```
modSVM <- train(classe~., data=train, method="svmLinear", trControl = control, tuneLength = 5, verbose =

predSVM <- predict(modSVM, valid)
cmSVM <- confusionMatrix(predSVM, factor(valid$classe))
cmSVM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1545  168  115  102   57
##          B   26  781   90   37  154
##          C   49   68  747  103   66
##          D   49   33   56  665   98
##          E    5   89   18   57  707
##
## Overall Statistics
##
##                Accuracy : 0.7553
##                  95% CI : (0.7441, 0.7663)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6883
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9229   0.6857   0.7281   0.6898   0.6534
## Specificity            0.8950   0.9353   0.9411   0.9520   0.9648
## Pos Pred Value         0.7776   0.7178   0.7231   0.7381   0.8071
## Neg Pred Value         0.9669   0.9254   0.9425   0.9400   0.9251
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2625   0.1327   0.1269   0.1130   0.1201
## Detection Prevalence   0.3376   0.1849   0.1755   0.1531   0.1489
## Balanced Accuracy      0.9090   0.8105   0.8346   0.8209   0.8091
```

Comparing all 3 Methods–Accuracy

```
Accuracy_summary <- c("Decision Trees" = cmTrees$overall[1], "Random Forests" = cmRF$overall[1], "SVM" =
Accuracy_summary
```

```
## Decision Trees.Accuracy Random Forests.Accuracy             SVM.Accuracy
##               0.5163976               0.9947324               0.7553101
```

The best model to use is Random Forests with an accuracy rate of 0.994. This will be the model used for the test set.

Predictions on Test Set

```
pred <- predict(modRF, testcsv)
print(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
```