PML Course Project

Pebblez

2025-12-01

Summary

This is the final report for Coursera’s Practical Machine Learning course, as part of the Data Science Specialization track offered by John Hopkins. The project uses data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants doing barbell lifts to predict the manner in which they did the exercise. This is the “classe” variable in the training set. We trained 3 models: Decision Tree, Random Forest, and Gradient Boosted Trees on the training set. We then tested the models using a validation set randomly selected from the training csv data, and obtained the accuracy and out of sample error rate. Based on those numbers, we concluded that **Random Forest** is the best model (accuracy = 0.997 and ose = 0.0029), and use it to predict 20 cases using the test csv set.

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. This analysis will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants who were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Data Sources

All data were downloaded on 1 Dec 2025 from the following links:

Training data (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

Test data (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The complete data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)

Setup

First we called the libraries necessary to complete the analysis, then we set the seed for reproducibility.

```
library(caret)
library(kernlab)
library(rattle)
set.seed(1234)
```

Loading and Preparing Data

Next we read in the data sets and went through a few steps to clean and partition the data.

```
#Load and clean data
train <- read.csv("pml-training.csv")
test <- read.csv("pml-testing.csv")
dim(train); dim(test)
```

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

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

```
    #remove columns with NA values
completedata <- (colSums(is.na(train)) == 0)
train <- train[, completedata]
dim(train)
```

```
## [1] 19622    93
```

```
    #remove Near Zero Variance variables
nvz <- nearZeroVar(train)
train <- train[,-nvz]
dim(train)
```

```
## [1] 19622    59
```

```
    #remove ID variables
train <- train[, -(1:5)]
dim(train)
```

```
## [1] 19622    54
```

```
#Partition cleaned train dataset into training and validation sets, leaving
#the test dataset unchanged for use later.

inTrain  <- createDataPartition(train$classe, p=0.7, list=FALSE)
trainset <- train[inTrain, ]
validset  <- train[-inTrain, ]
dim(trainset); dim(validset)
```

```
## [1] 13737    54
```

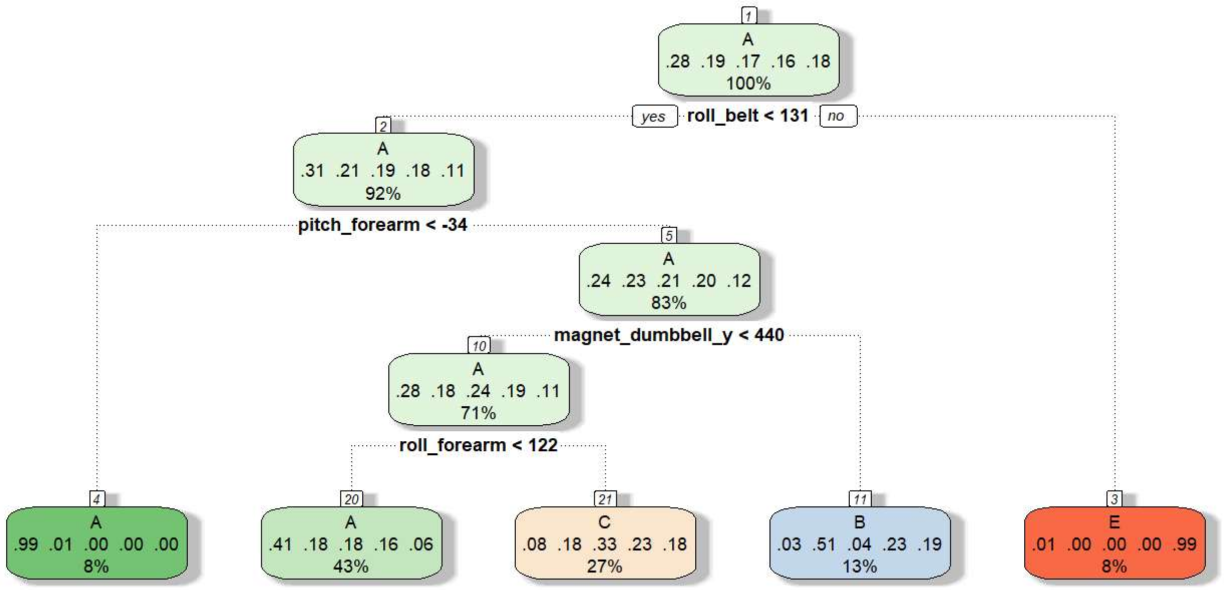
```
## [1] 5885   54
```

Creating and Testing Models

Next we applied three popular PML models (decision tree, random forests, and boosted trees) to identify the regression method with highest accuracy to be used later.

Decision Tree Method

```
Treefit<- train(classe~., method="rpart", data=trainset)
fancyRpartPlot(Treefit$finalModel, sub = "")
```

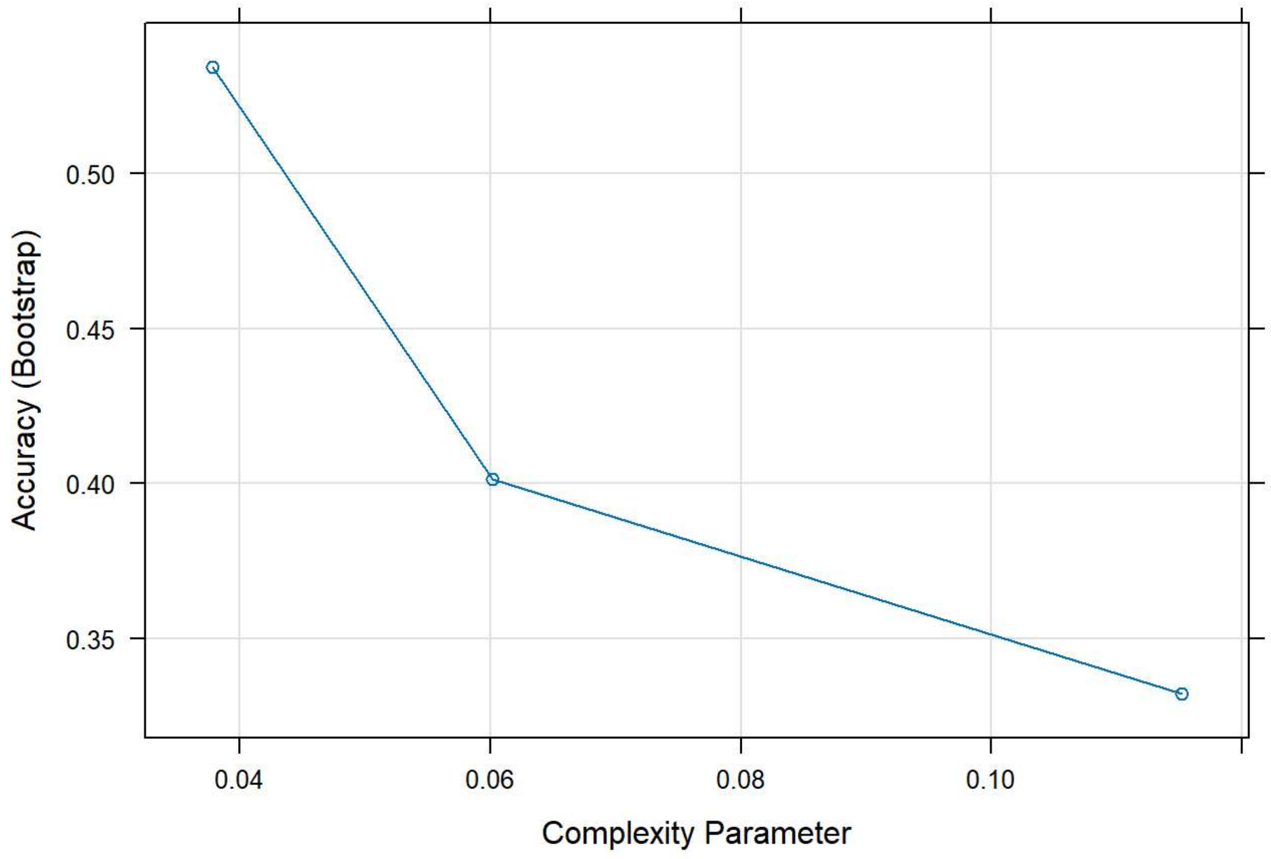


```
Treepred <- predict(Treefit, validset)
confusionMatrix(Treepred, factor(validset$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1519  473  484  451  156
##          B   28  401   45  167  148
##          C  123  265  497  346  289
##          D    0    0    0    0    0
##          E    4    0    0    0  489
##
## Overall Statistics
##
##                Accuracy : 0.4938
##                  95% CI : (0.4809, 0.5067)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.338
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9074  0.35206  0.48441   0.0000  0.45194
## Specificity            0.6286  0.91825  0.78946   1.0000  0.99917
## Pos Pred Value         0.4927  0.50824  0.32697      NaN  0.99189
## Neg Pred Value         0.9447  0.85518  0.87881   0.8362  0.89002
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2581  0.06814  0.08445   0.0000  0.08309
## Detection Prevalence   0.5239  0.13407  0.25828   0.0000  0.08377
## Balanced Accuracy      0.7680  0.63516  0.63693   0.5000  0.72555
```

```
Treeacc<- confusionMatrix(Treepred, factor(validset$classe))$overall["Accuracy"]
Treeose <- 1-as.numeric(confusionMatrix(factor(validset$classe),
                                        Treepred)$overall[1])
```

```
plot(Treefit)
```



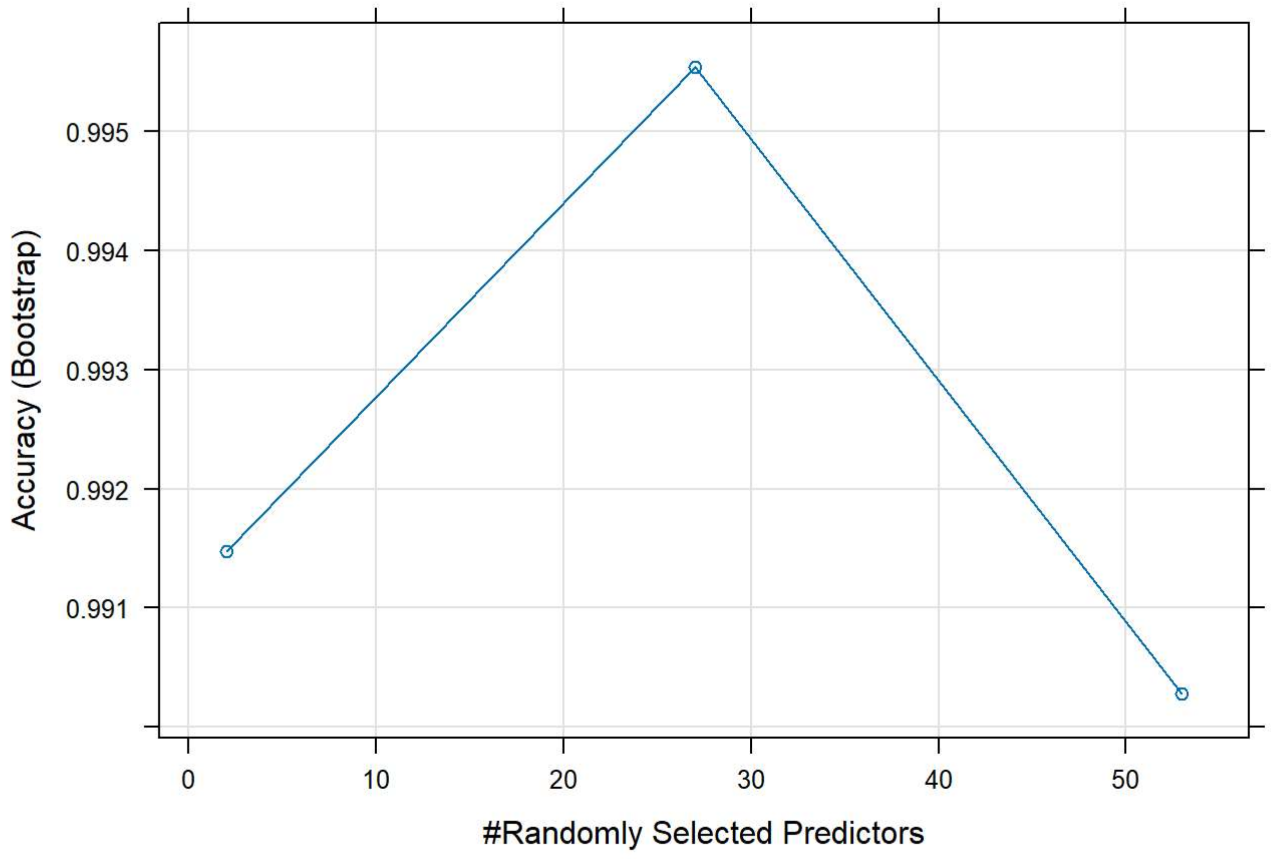
Random Forests Method

```
RFfit <- train(classe~., method="rf", data=trainset)
RFpred <- predict(RFfit, validset)
confusionMatrix(RFpred, factor(validset$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1135    6    0    0
##          C    0    1 1020    4    0
##          D    0    0    0  959    0
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9958, 0.9986)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9942   0.9948   1.0000
## Specificity            0.9993   0.9987   0.9990   1.0000   0.9998
## Pos Pred Value         0.9982   0.9947   0.9951   1.0000   0.9991
## Neg Pred Value         1.0000   0.9992   0.9988   0.9990   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1733   0.1630   0.1839
## Detection Prevalence   0.2850   0.1939   0.1742   0.1630   0.1840
## Balanced Accuracy      0.9996   0.9976   0.9966   0.9974   0.9999
```

```
RFacc<- confusionMatrix(RFpred, factor(validset$classe))$overall["Accuracy"]
RFose <- 1-as.numeric(confusionMatrix(factor(validset$classe),
                                        RFpred)$overall[1])
```

```
plot(RFfit)
```



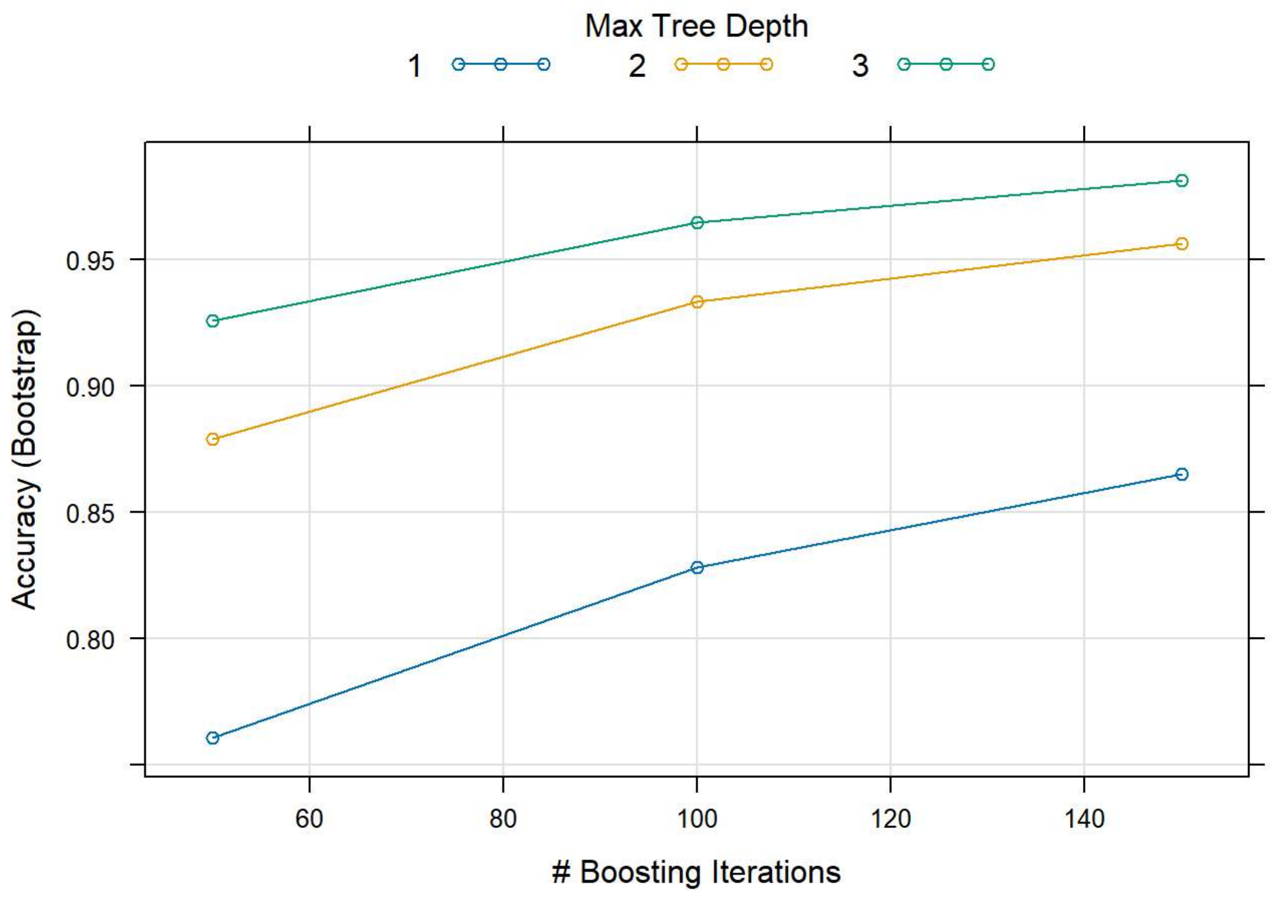
Boosted Trees Method

```
GBMfit <- train(classe~., method="gbm", data=trainset, verbose = F)
GBMpred <- predict(GBMfit, validset)
confusionMatrix(GBMpred, factor(validset$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    1    0
##          B    1 1126    7    3    2
##          C    0    9 1014    7    0
##          D    0    0    5  950    5
##          E    0    0    0    3 1075
##
## Overall Statistics
##
##                Accuracy : 0.992
##                  95% CI : (0.9894, 0.9941)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9899
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9886   0.9883   0.9855   0.9935
## Specificity            0.9988   0.9973   0.9967   0.9980   0.9994
## Pos Pred Value         0.9970   0.9886   0.9845   0.9896   0.9972
## Neg Pred Value         0.9998   0.9973   0.9975   0.9972   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1913   0.1723   0.1614   0.1827
## Detection Prevalence   0.2851   0.1935   0.1750   0.1631   0.1832
## Balanced Accuracy      0.9991   0.9929   0.9925   0.9917   0.9965
```

```
GBMacc<- confusionMatrix(GBMpred, factor(validset$classe))$overall["Accuracy"]
GBMose <- 1-as.numeric(confusionMatrix(factor(validset$classe),
                                      GBMpred)$overall[1])
```

```
plot(GBMfit)
```



Results

After running three different models, we compared the accuracy and out-of-sample error for each one. The results indicated that the **Random Forest** method is the best model, with 99.7% accuracy and 0.0029 estimated out-of-sample error. Therefore we used the Random Forest Model to predict our test cases.

```
acc<- rbind(Treeacc, RFacc, GBMacc)
ose<- rbind(Treeose, RFose, GBMose)
cbind(acc, ose)
```

```
##          Accuracy
## Treeacc 0.4937978 0.506202209
## RFacc   0.9974511 0.002548853
## GBMacc  0.9920136 0.007986406
```

Predicting Test Cases

Returning to the originally downloaded test dataset that we did not process or look at, we used the Random Forest model to predict the manner in which the exercise was done in each test case.

```
predictTEST <- predict(RFfit, newdata=test)
predictTEST
```

```
##  [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
```