Prediction Assignment Writeup, Course: Practical Machine Learning by John Hopkins university

Sören Nonnengart

2022-08-24

## Introduction

Here you see the results of my final Coursera’s Practical Machine Learning course, as part of the Data Science Specialization track offered by John Hopkins with coursera.

In this project, I will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise. This is the “classe” variable in the training set.

I will train 4 models overall:

* Decision Trees
* Random Forest
* Gradient Boosted Trees
* Support Vector Machine using k-folds cross validation on the training set.

I then predict using a validation set randomly selected from the training csv data to obtain the accuracy and out of sample error rate. Based on those numbers, I decide on the best model, and use it to predict 20 cases using the test csv set.

## Loading Data and Libraries

#### Libraries that will to be used for the analysis

library(caret)  
library(kernlab)  
library(rattle)  
library(lattice)  
library(ggplot2)  
library(corrplot)

### set seed

set.seed(318925)

### load the training and testdatasets

## [1] "/Users/sorennonnengart/Coursera/Data\_science/tasks/course\_8\_Machine learning"

traincsv <- read.csv("pml-training.csv")  
testcsv <- read.csv("pml-testing.csv")

dim(traincsv)

## [1] 19622 160

dim(testcsv)

## [1] 20 160

#### You can see that there are 160 variables and 19622 observations in the training set and 20 for the test dataset.

## Cleaning the Data

#### Removing NA’s in the variables

traincsv <- traincsv[,colMeans(is.na(traincsv)) < .9]   
traincsv <- traincsv[,-c(1:7)]

#### Removing near zero variance variables.

nvz <- nearZeroVar(traincsv)  
traincsv <- traincsv[,-nvz]  
dim(traincsv)

## [1] 19622 53

#### The removing of the unnecessary variables are finished, so now the training dataset can be split into a validation and sub training set. The testing set “testcsv” will be left alone, and used for the final quiz test cases.

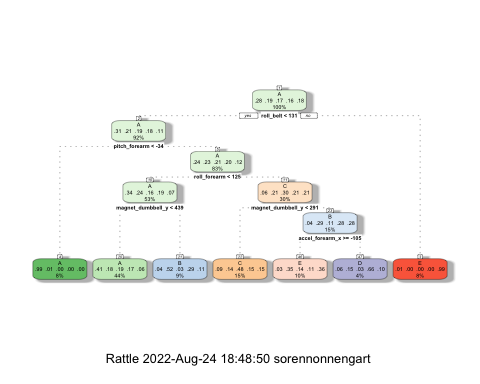
inTrain <- createDataPartition(y=traincsv$classe, p=0.7, list=F)  
train <- traincsv[inTrain,]  
valid <- traincsv[-inTrain,]

## Now create and test the models

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

## Decision Tree

mod\_trees <- train(classe~., data=train, method="rpart", trControl = control, tuneLength = 5)  
fancyRpartPlot(mod\_trees$finalModel)



pred\_trees <- predict(mod\_trees, valid)  
cmtrees <- confusionMatrix(pred\_trees, factor(valid$classe))  
cmtrees

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1500 468 461 447 161  
## B 29 303 16 178 73  
## C 95 103 471 121 124  
## D 17 42 10 158 33  
## E 33 223 68 60 691  
##   
## Overall Statistics  
##   
## Accuracy : 0.5307   
## 95% CI : (0.5178, 0.5435)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.387   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.8961 0.26602 0.45906 0.16390 0.6386  
## Specificity 0.6350 0.93763 0.90883 0.97927 0.9200  
## Pos Pred Value 0.4939 0.50584 0.51532 0.60769 0.6428  
## Neg Pred Value 0.9389 0.84185 0.88835 0.85671 0.9187  
## Prevalence 0.2845 0.19354 0.17434 0.16381 0.1839  
## Detection Rate 0.2549 0.05149 0.08003 0.02685 0.1174  
## Detection Prevalence 0.5161 0.10178 0.15531 0.04418 0.1827  
## Balanced Accuracy 0.7655 0.60183 0.68395 0.57159 0.7793

## Random Forest

mod\_rf <- train(classe~., data=train, method="rf", trControl = control, tuneLength = 5)  
pred\_rf <- predict(mod\_rf, valid)  
cmrf <- confusionMatrix(pred\_rf, factor(valid$classe))  
cmrf

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1667 8 0 0 0  
## B 4 1131 3 0 0  
## C 3 0 1022 8 0  
## D 0 0 1 956 1  
## E 0 0 0 0 1081  
##   
## Overall Statistics  
##   
## Accuracy : 0.9952   
## 95% CI : (0.9931, 0.9968)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.994   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9958 0.9930 0.9961 0.9917 0.9991  
## Specificity 0.9981 0.9985 0.9977 0.9996 1.0000  
## Pos Pred Value 0.9952 0.9938 0.9894 0.9979 1.0000  
## Neg Pred Value 0.9983 0.9983 0.9992 0.9984 0.9998  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2833 0.1922 0.1737 0.1624 0.1837  
## Detection Prevalence 0.2846 0.1934 0.1755 0.1628 0.1837  
## Balanced Accuracy 0.9970 0.9958 0.9969 0.9956 0.9995

## Gradient Boosted Trees

mod\_gbm <- train(classe~., data=train, method="gbm", trControl = control, tuneLength = 5, verbose = F)  
pred\_gbm <- predict(mod\_gbm, valid)  
cmgbm <- confusionMatrix(pred\_gbm, factor(valid$classe))  
cmgbm

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1665 11 0 0 2  
## B 5 1122 9 0 1  
## C 4 6 1009 13 3  
## D 0 0 6 950 1  
## E 0 0 2 1 1075  
##   
## Overall Statistics  
##   
## Accuracy : 0.9891   
## 95% CI : (0.9861, 0.9916)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9862   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9946 0.9851 0.9834 0.9855 0.9935  
## Specificity 0.9969 0.9968 0.9946 0.9986 0.9994  
## Pos Pred Value 0.9923 0.9868 0.9749 0.9927 0.9972  
## Neg Pred Value 0.9979 0.9964 0.9965 0.9972 0.9985  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2829 0.1907 0.1715 0.1614 0.1827  
## Detection Prevalence 0.2851 0.1932 0.1759 0.1626 0.1832  
## Balanced Accuracy 0.9958 0.9910 0.9890 0.9920 0.9965

## Support Vector Machines

mod\_svm <- train(classe~., data=train, method="svmLinear", trControl = control, tuneLength = 5, verbose = F)  
pred\_svm <- predict(mod\_svm, valid)  
cmsvm <- confusionMatrix(pred\_svm, factor(valid$classe))  
cmsvm

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1543 152 76 59 72  
## B 35 823 98 28 141  
## C 48 60 796 118 61  
## D 39 24 31 729 60  
## E 9 80 25 30 748  
##   
## Overall Statistics  
##   
## Accuracy : 0.7883   
## 95% CI : (0.7776, 0.7987)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7308   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9217 0.7226 0.7758 0.7562 0.6913  
## Specificity 0.9147 0.9364 0.9409 0.9687 0.9700  
## Pos Pred Value 0.8113 0.7316 0.7350 0.8256 0.8386  
## Neg Pred Value 0.9671 0.9336 0.9521 0.9530 0.9331  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2622 0.1398 0.1353 0.1239 0.1271  
## Detection Prevalence 0.3232 0.1912 0.1840 0.1500 0.1516  
## Balanced Accuracy 0.9182 0.8295 0.8584 0.8625 0.8307

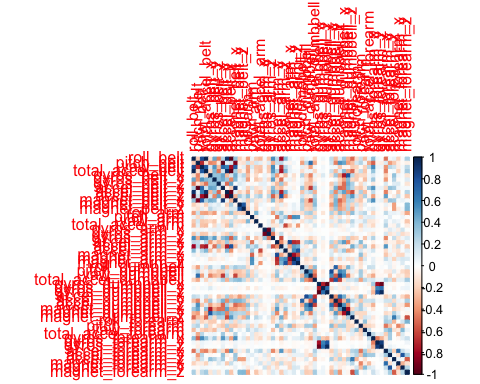
## Predictions on Test Set

pred <- predict(mod\_rf, testcsv)  
print(pred)

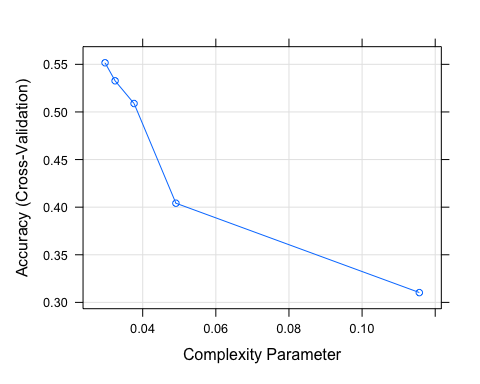
## [1] B A B A A E D B A A B C B A E E A B B B  
## Levels: A B C D E

## Plots

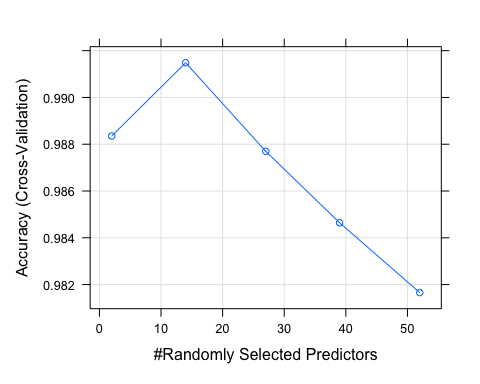
corrPlot <- cor(train[, -length(names(train))])  
corrplot(corrPlot, method="color")



plot(mod\_trees)



plot(mod\_rf)



plot(mod\_gbm)

