Practical Machine Learning - Prediction Assignment Writeup

Marshall Too

Jan 5, 2021

Created with knitr

1. Executive Summary

This is a report of the Peer Assessment project from the Practical Machine Learning course. The goal of this analysis is to predict the manner in which the six participants performed their exercises. The machine learning algorithm, uses the "classe" variable in the training set, is applied to the 20 test cases available in the test data.

2. Libraries

```
library(caret)
library(rattle)
library(corrplot)
```

3. Load Data

Load the dataset.

```
TrainData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),heade
dim(TrainData)
TestData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),header='
dim(TestData)
```

4. Create a partition of the traning data set and clean data

```
 # The training dataset is  partitioned into 2 to create a Training set with 70% of the data for the mo


set.seed(32343)
inTrain <- createDataPartition(TrainData$classe, p = 0.7, list = FALSE)
trainData <- TrainData[inTrain, ]
testData <- TrainData[-inTrain, ]
dim(trainData)
```

```
## [1] 13737   160
```

```r
dim(testData)
```

```
## [1] 5885  160
```

```r
# trainData and testData have a large number of NA values and near-zero-variance (NZV) variables. Remov


NZV <- nearZeroVar(trainData)
trainData <- trainData[, -NZV]
testData  <- testData[, -NZV]
dim(trainData)
```

```
## [1] 13737   108
```

```r
dim(testData)
```

```
## [1] 5885  108
```

```r
# Remove variables that are mostly NA. A threshlod of 95 % is selected.
mostlyNA <- sapply(trainData, function(x) mean(is.na(x))) > 0.95
mostlyNATest <- sapply(testData, function(x) mean(is.na(x))) > 0.95
trainData <- trainData[, mostlyNA==F]
testData <- testData[, mostlyNATest==F]

dim(trainData)
```

```
## [1] 13737    59
```

```r
dim(testData)
```

```
## [1] 5885   59
```

```r
# Remove identification only variables (columns 1 to 5)The highly correlated variables are shown in dar
trainData <- trainData[, -(1:5)]
testData <- testData[, -(1:5)]

dim(trainData)
```

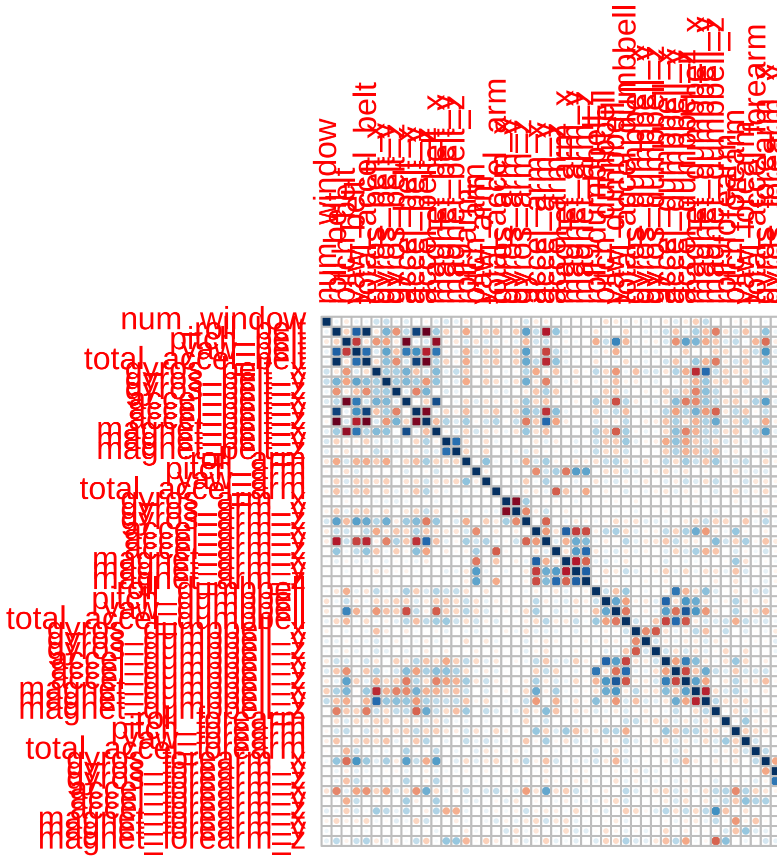
```
## [1] 13737    54
```

```r
dim(testData)
```

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

5. Data Analysis

```
correlation <- cor(trainData[, -54])
corrplot(correlation, method="circle")
```

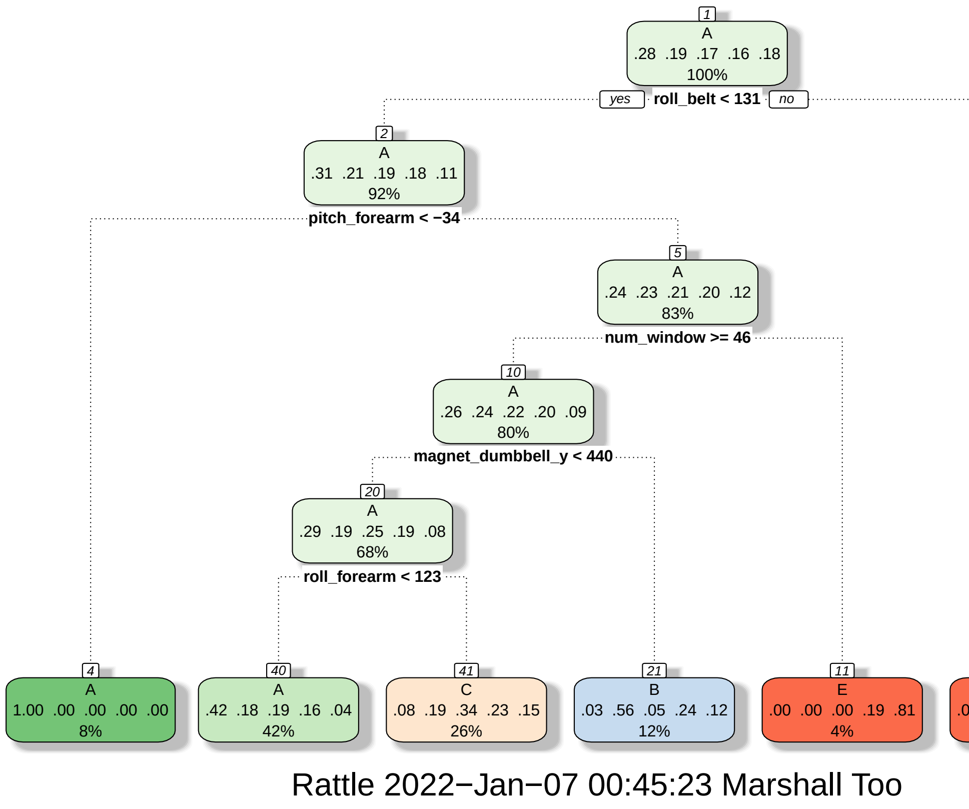


a) Check correlation among variables

```
# The circles with dark colors show highly correlated variables in the graph above. Correlations do not
```

```
trControl <- trainControl(method="cv", number=5)
model_CT <- train(classe~., , method="rpart", data=trainData, trControl=trControl)

fancyRpartPlot(model_CT$finalModel)
```



b) Classification tree method

```
predict_train <- predict(model_CT,newdata=testData)

confMatClassTree <- confusionMatrix(testData$classe,predict_train)
```

```
## Error: ‘data‘ and ‘reference‘ should be factors with the same levels.
```

```
#Display confusion matrix and model accuracy

confMatClassTree$table
```

```
## Error in eval(expr, envir, enclos): object 'confMatClassTree' not found
```

```
confMatClassTree$overall[1]
```

```
## Error in eval(expr, envir, enclos): object 'confMatClassTree' not found
```

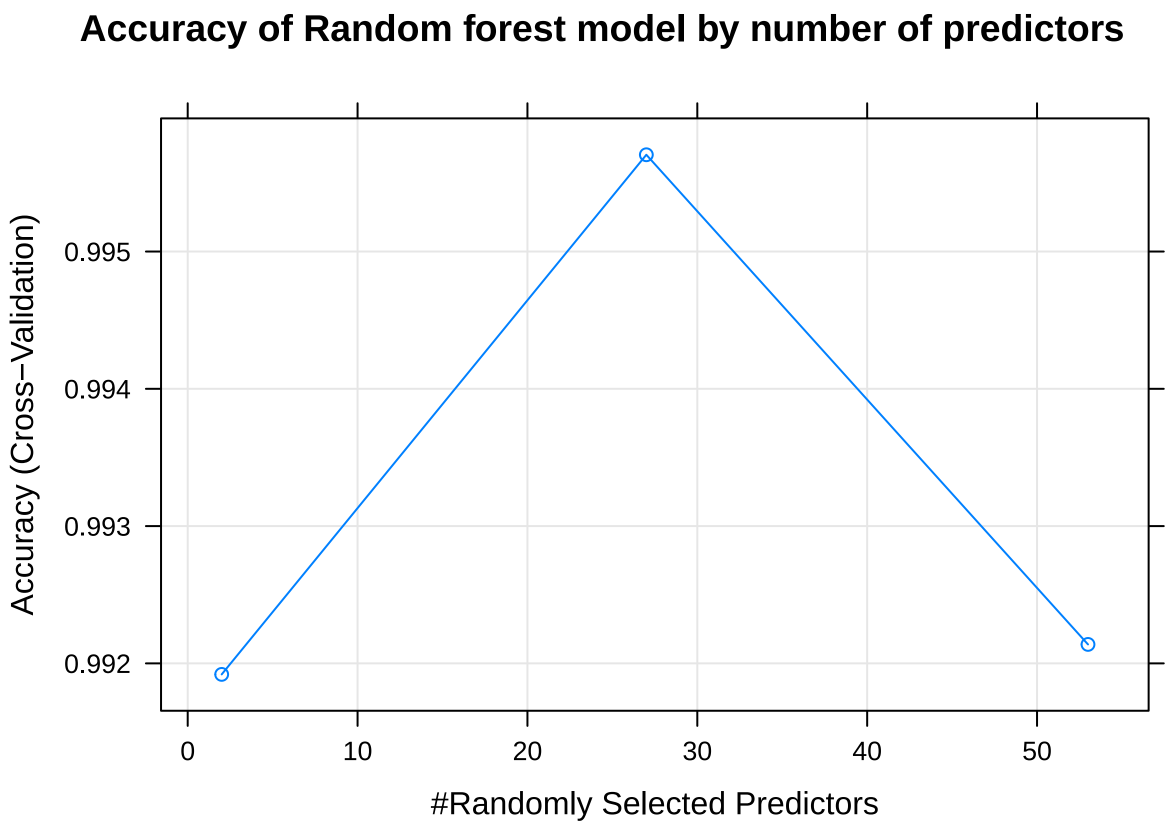
```


random_forest <- trainControl(method="cv", number=3, verboseIter=FALSE)
model_RF1 <- train(classe ~ ., data=trainData, method="rf", trControl=random_forest)
model_RF1$finalModel
```

c) Random forest method

```
##
## Call:
##  randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.17%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    6 2647    4    1    0 0.004138450
## C    0    3 2393    0    0 0.001252087
## D    0    0    7 2245    0 0.003108348
## E    0    0    0    3 2522 0.001188119
```

```
plot(model_RF1,main="Accuracy of Random forest model by number of predictors")
```



```
predict_train <- predict(model_RF1,newdata=testData)

confMatRF <- confusionMatrix(testData$classe,predict_train)
```

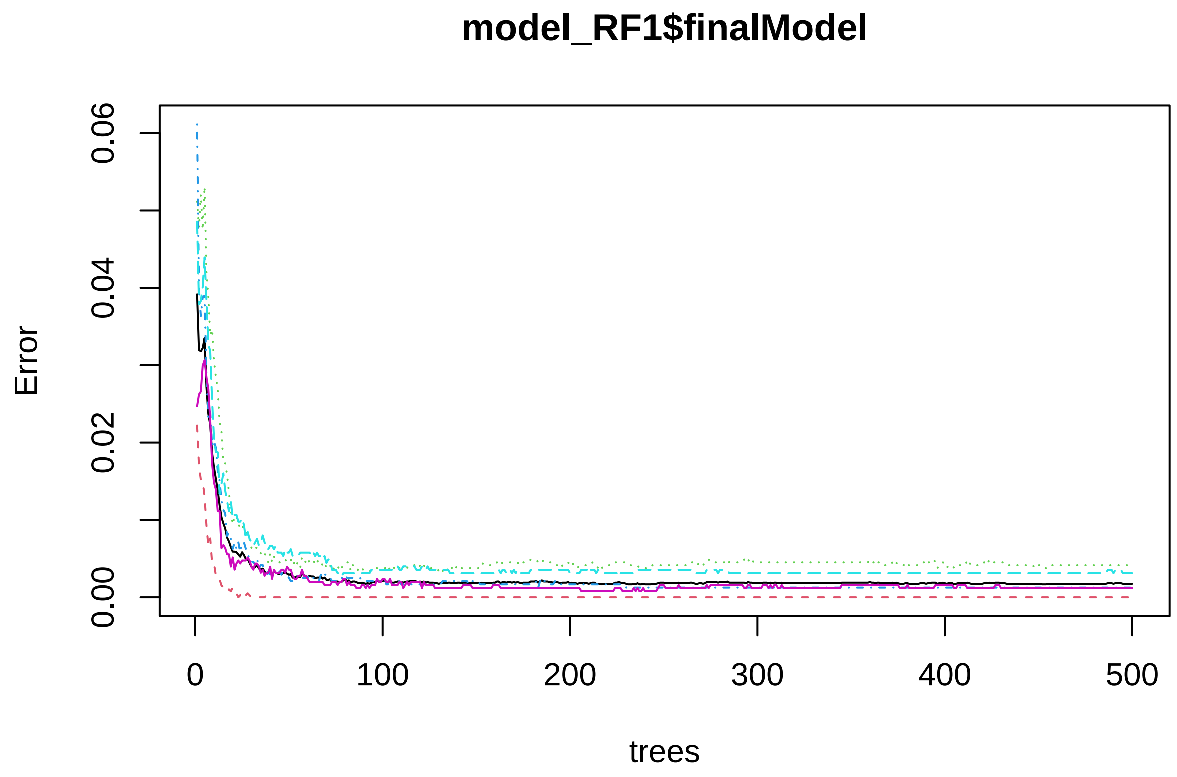
```
## Error: `data` and `reference` should be factors with the same levels.
```

```r
# Display confusion matrix and model accuracy

confMatRF
```

```
## Error in eval(expr, envir, enclos): object 'confMatRF' not found
```

```r
plot(model_RF1$finalModel)
```



```r
set.seed(12345)
GBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
model_GBM  <- train(classe ~ ., data=trainData, method = "gbm", trControl = GBM, verbose = FALSE)
model_GBM$finalModel
```

d) Generated Boosted Model (GBM)

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predictGBM <- predict(model_GBM, newdata=testData)
confMatGBM <- confusionMatrix(predictGBM, testData$classe)
```

```
## Error: ‘data‘ and ‘reference‘ should be factors with the same levels.
```

```
confMatGBM
```

```
## Error in eval(expr, envir, enclos): object 'confMatGBM' not found
```

6. Conclusion

```
# The predictive accuracies of the above methods are:

#Classification Tree Model: 49.62 %
#Generalized Boosted Model: 98.96 %
#Random Forest Model: 99.71 %
#
#The Random Forest model has the best accuracy and hence it is used for predictions on the 20 data poin


predict_test <- predict(model_RF1, newdata = TestData)
predict_test
```

```
##  [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
```