Practical Machine Learning Course Project

SMK

11/26/2022

Synopsis

This document is created to summarize the results of Practical Machine Learning Course Project. Different machine learning models were created to identify how well the participants performed the barbell lifts. The goal of this project was to predict the manner in which they did the exercise. The analysis was done using the machine learning algorithms such as Decision Tree, Support Vector Machine, Random Forest, Gradient Boosted Trees using k-fold cross-validation on the training data set. It has been observed that the Random Forest Algorithm gave the highest accuracy based on the training and validations datasets. Hence, this model is being used to predict the performance of the test dataset.

Reading Dataset

The data for this exercise was collected from 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.

```
library(caret)
```

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

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

```
library(ggplot2)
library(lattice)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
set.seed(1234)

url_train <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_test <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

df_training <- read.csv(url(url_train))
df_testing <- read.csv(url(url_test))

# Let's check whether the datasets are properly loaded or not

dim(df_training)
```

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

```
dim(df_testing)
```

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

The given training and test data sets includes 160 variables. Also, there are 19,622 observations in the training set and 20 observations in the test set.

Data Pre-Processing

Next step after reading the data is for data pre-processing. This is an important step before adding data into the model in order to get the appropriate test results.

Different steps can be performed to cleanup the given data such as removing N/A variables, metadata columns, zero variance variables or constants.

```
# Removing columns with most N/As
df_training <- df_training[, colMeans(is.na(df_training)) < 0.9]

# Removing metadata columns
df_training <- df_training[, -c(1:7)]

# Removing near zero variance variables
nzv <- nearZeroVar(df_training)
df_training <- df_training[, -nzv]
dim(df_training)
```

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

Data Preperation for Data Modeling Exercise

After data pre-processing is done, next step is to divide the data into Training dataset and Validation dataset. This step prepares the dataset so that they can be passed to the Machine Learning Algorithms.

```
# Divide the training dataset into train and validation dataset

div_training <- createDataPartition(y=df_training$classe, p=0.7, list=F)
df_train <- df_training[div_training,]
df_valid <- df_training[-div_training,]
```

Let's setup the control for using 3-fold cross valdation.

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

Machine Learning Algorithms

Decision Tree

Let's start with the Decision Tree model and analyze it's performance.

```
dec_tree <- train(classe~., data=df_train, method="rpart", trControl=df_cntrl, tuneLength=5)

pred_dec_tree <- predict(dec_tree, df_valid)

eval_dec_tree <- confusionMatrix(pred_dec_tree, factor(df_valid$classe))

eval_dec_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1519  473  484  451  156
##          B   28  355   45   10  130
##          C   83  117  423  131  131
##          D   40  194   74  372  176
##          E    4    0    0    0  489
##
## Overall Statistics
##
##                Accuracy : 0.5366
##                  95% CI : (0.5238, 0.5494)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3957
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9074  0.31168  0.41228  0.38589  0.45194
## Specificity            0.6286  0.95512  0.90492  0.90165  0.99917
## Pos Pred Value         0.4927  0.62500  0.47797  0.43458  0.99189
## Neg Pred Value         0.9447  0.85255  0.87940  0.88228  0.89002
## Prevalence             0.2845  0.19354  0.17434  0.16381  0.18386
## Detection Rate         0.2581  0.06032  0.07188  0.06321  0.08309
## Detection Prevalence   0.5239  0.09652  0.15038  0.14545  0.08377
## Balanced Accuracy      0.7680  0.63340  0.65860  0.64377  0.72555
```

Random Forest

Let's build the Random Forest model next and analyze it's performance.

```
rnd_frst <- train(classe~., data=df_train, method="rf", trControl=df_cntrl, tuneLength=5)

pref_rnd_frst <- predict(rnd_frst, df_valid)

eval_rnd_frst <- confusionMatrix(pref_rnd_frst, factor(df_valid$classe))

eval_rnd_frst
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    1 1132    8    0    0
##          C    0    3 1016    5    1
##          D    0    0    2  958    0
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9958
##                  95% CI : (0.9937, 0.9972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9946
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9939   0.9903   0.9938   0.9991
## Specificity            0.9991   0.9981   0.9981   0.9996   0.9998
## Pos Pred Value         0.9976   0.9921   0.9912   0.9979   0.9991
## Neg Pred Value         0.9998   0.9985   0.9979   0.9988   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1924   0.1726   0.1628   0.1837
## Detection Prevalence   0.2850   0.1939   0.1742   0.1631   0.1839
## Balanced Accuracy      0.9992   0.9960   0.9942   0.9967   0.9994
```

Gradient Boosted Trees

Let's now build the Gradient Boosted Trees model and analyze it's performance.

```
grd_bst <- train(classe~., data=df_train, method="gbm", trControl=df_cntrl, tuneLength=5, verbose=F)

pred_grd_bst <- predict(grd_bst, df_valid)

eval_grd_bst <- confusionMatrix(pred_grd_bst, factor(df_valid$classe))

eval_grd_bst
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    6    0    0    0
##          B    2 1127   15    0    0
##          C    0    6 1007    8    3
##          D    0    0    4  954    1
##          E    0    0    0    2 1078
##
## 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            0.9988   0.9895   0.9815   0.9896   0.9963
## Specificity            0.9986   0.9964   0.9965   0.9990   0.9996
## Pos Pred Value         0.9964   0.9851   0.9834   0.9948   0.9981
## Neg Pred Value         0.9995   0.9975   0.9961   0.9980   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1915   0.1711   0.1621   0.1832
## Detection Prevalence   0.2851   0.1944   0.1740   0.1630   0.1835
## Balanced Accuracy      0.9987   0.9929   0.9890   0.9943   0.9979
```

SVM

```
svm <- train(classe~., data=df_train, method="svmLinear", trControl=df_cntrl, tuneLength=5, verbose=F)

pred_svm <- predict(svm, df_valid)

eval_svm <- confusionMatrix(pred_svm, factor(df_valid$classe))
```

```
eval_svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1537  154   79   69   50
##          B   29  806   90   46  152
##          C   40   81  797  114   69
##          D   61   22   32  697   50
##          E    7   76   28   38  761
##
## Overall Statistics
##
##                Accuracy : 0.7813
##                  95% CI : (0.7705, 0.7918)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.722
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9182   0.7076   0.7768   0.7230   0.7033
## Specificity            0.9164   0.9332   0.9374   0.9665   0.9690
## Pos Pred Value         0.8137   0.7177   0.7239   0.8086   0.8363
## Neg Pred Value         0.9657   0.9301   0.9521   0.9468   0.9355
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2612   0.1370   0.1354   0.1184   0.1293
## Detection Prevalence   0.3210   0.1908   0.1871   0.1465   0.1546
## Balanced Accuracy      0.9173   0.8204   0.8571   0.8447   0.8362
```

Model Selection

Let's check the accuracy and out of sample error rate for each of the above generated model.

```
mdls <- c("Decision Tree", "Random Forest", "GBM", "SVM")
acc <- round(c(eval_dec_tree$overall[1], eval_rnd_frst$overall[1], eval_grd_bst$overall[1], eval_svm$ov
err <- 1 - acc
data.frame(Accuracy=acc, Error=err, row.names=mdls)
```

```
##               Accuracy Error
## Decision Tree    0.537 0.463
## Random Forest    0.996 0.004
## GBM              0.992 0.008
## SVM              0.781 0.219
```

After running different machine learning models and analyzing performance of these in terms of accuracy, Random Forest model gives the highest accuracy 0.9954 and 0.0046 out of sample error rate

Conclusion

The Random Forest model resulted into highest accuracy and least out of sample error rate. Hence Random Forest algorithm is selected for modeling the given test set.

```
pred_test <- predict(rnd_frst, df_testing)

pred_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
```

Appendix

Figure 1: Correlation metrics of variables in the training dataset

```
plt_corr <- cor(df_train[, -length(names(df_train))])
corrplot(plt_corr, method = "color")
```

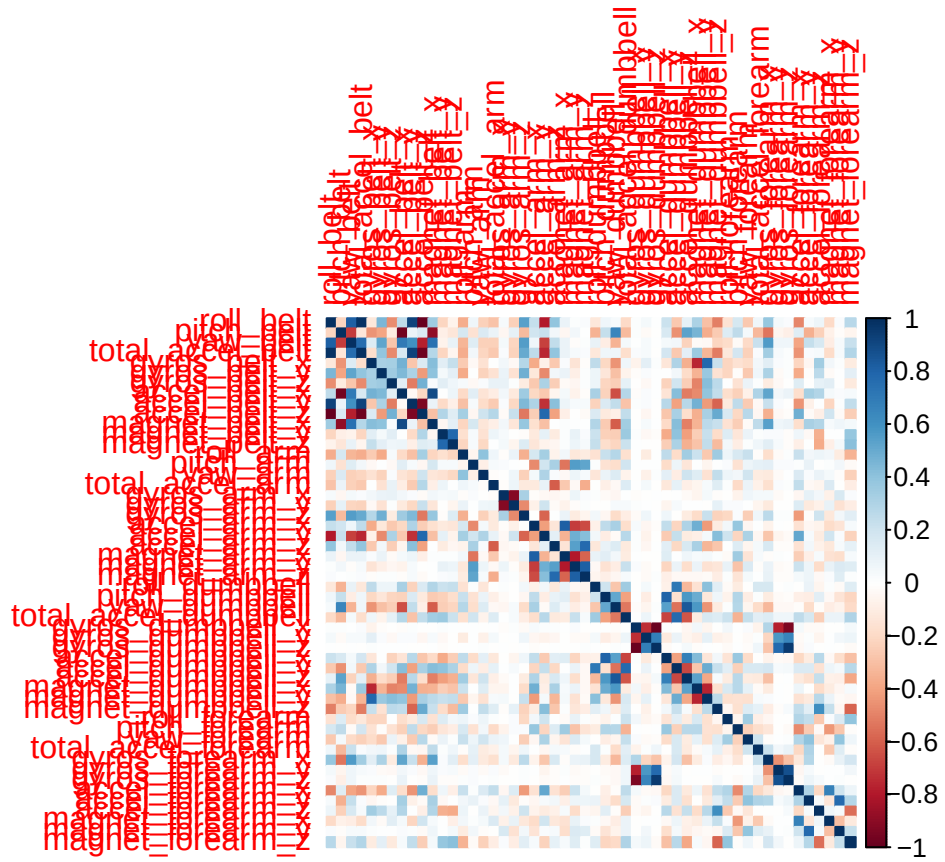


Figure 2: Decision Tree plot

```
fancyRpartPlot(dec_tree$finalModel)
```

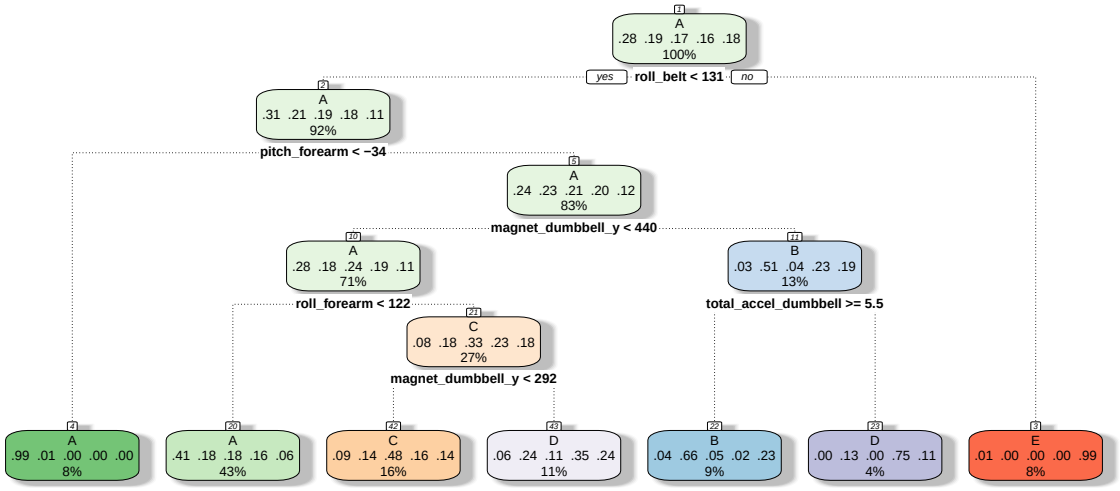


Figure 3: Random Forest plot

```
plot(rnd_frst)
```

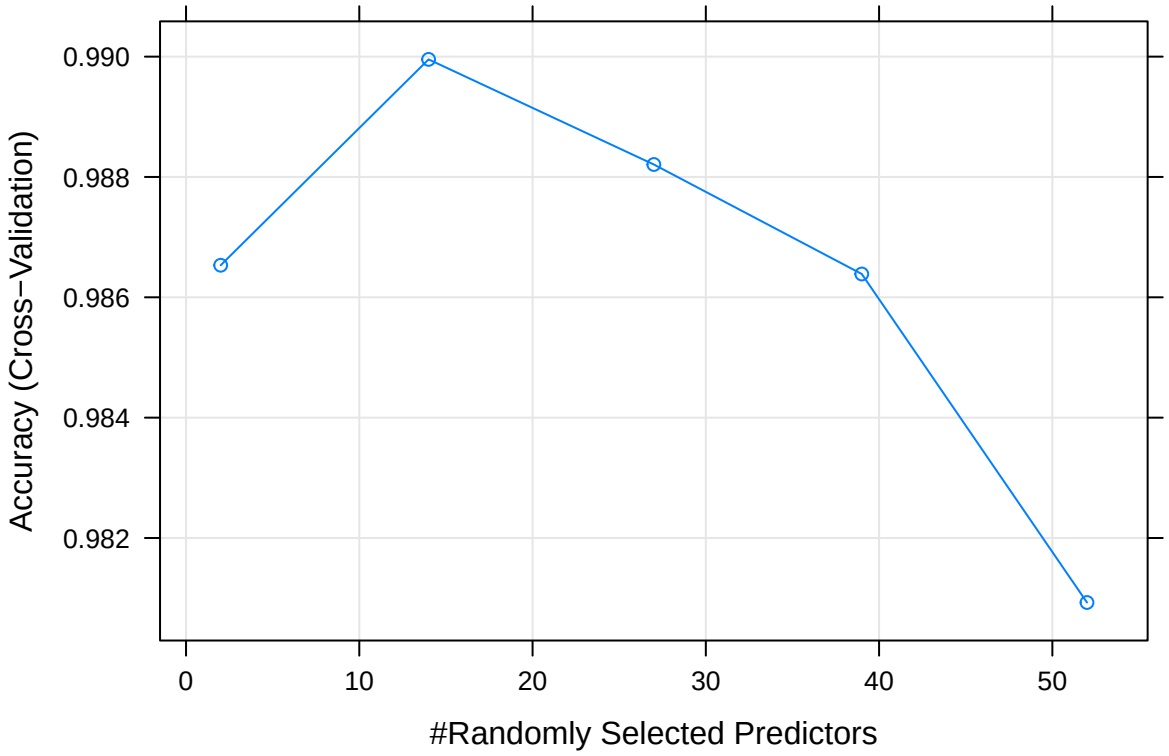
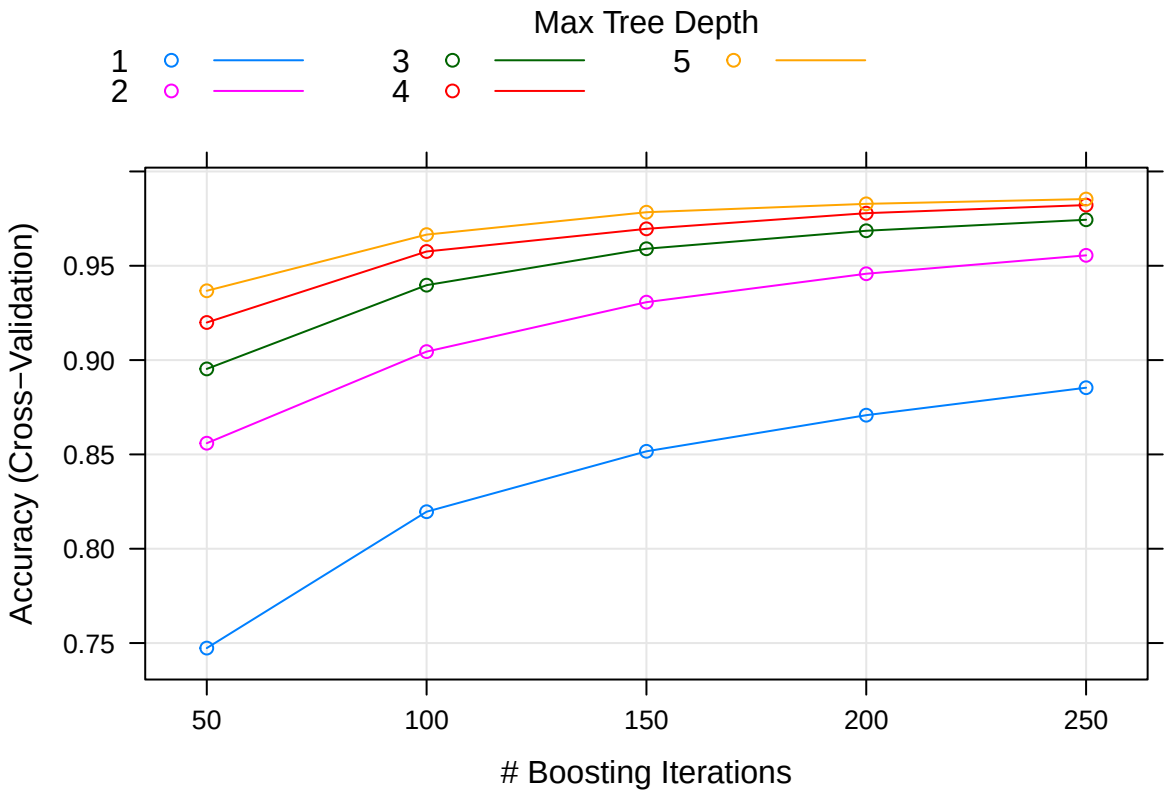


Figure 4: Gradient Boosted Trees plot

```
plot(grd_bst)
```


Max Tree Depth
1
2
3
4
5
Accuracy (Cross−Validation)
0.75
0.80
0.85
0.90
0.95
50
100
150
200
250
Boosting Iterations