

PracticalMachineLearning

KamolcT

2022-10-22

Practical Machine Learning - Course Project

Introduction

For this project, we are given data from accelerometers on the belt, forearm, arm, and dumbbell of 6 research study participants. Our training data consists of accelerometer data and a label identifying the quality of the activity the participant was doing. Our testing data consists of accelerometer data without the identifying label. Our goal is to predict the labels for the test set observations.

Below is the code I used when creating the model, estimating the out-of-sample error, and making predictions. I also include a description of each step of the process. `##` Data Preparation I load the caret package, and read in the training and testing data:

```
library(knitr)
library(rpart)
library(rpart.plot)
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(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
```

```
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
```

```
##
```

```
##      importance
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(caret)
```

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

```
##
```

```
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

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

```
set.seed(12345)
```

The next step is loading the dataset from the URL provided above. The training dataset is then partitioned in 2 to create a Training set (70% of the data) for the modeling process and a Test set (with the remaining 30%) for the validations. The testing dataset is not changed and will only be used for the quiz results generation.

```
# set the URL for the download
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

```
# download the datasets
training <- read.csv(url(UrlTrain))
testing  <- read.csv(url(UrlTest))
```

```
# create a partition with the training dataset
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]
dim(TrainSet)
```

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

```
dim(TestSet)
```

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

Both created datasets have 160 variables. Those variables have plenty of NA, that can be removed with the cleaning procedures below. The Near Zero variance (NZV) variables are also removed and the ID variables as well.

```
# remove variables with Nearly Zero Variance
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]
dim(TrainSet)
```

```
## [1] 13737 104
```

```
# remove variables that are mostly NA
AllNA <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet  <- TestSet[, AllNA==FALSE]
dim(TrainSet)
```

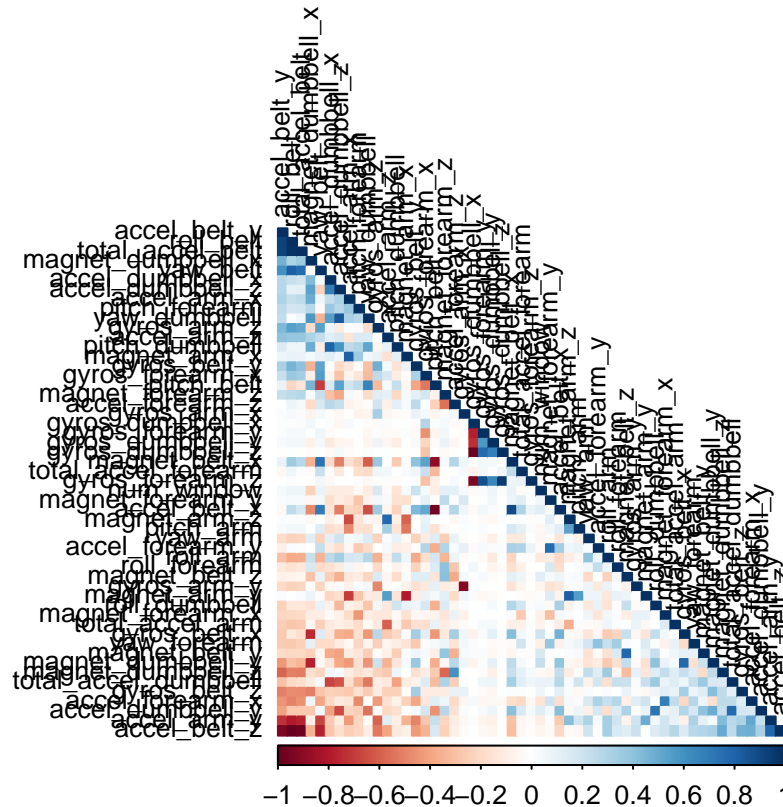
```
## [1] 13737 59
```

```
# remove identification only variables (columns 1 to 5)
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)
```

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

Correlation Analysis

```
corMatrix <- cor(TrainSet[, -54])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
          tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



The highly correlated variables are shown in dark colors in the graph above. To make an even more compact analysis, a PCA (Principal Components Analysis) could be performed as pre-processing step to the datasets. Nevertheless, as the correlations are quite few, this step will not be applied for this assignment.

Prediction Model Building

Three methods will be applied to model the regressions (in the Train dataset) and the best one (with higher accuracy when applied to the Test dataset) will be used for the quiz predictions. The methods are: Random Forests, Decision Tree and Generalized Boosted Model, as described below. A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models.

1 Method: Random Forest

```
# model fit
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf",
                           trControl=controlRF)
modFitRandForest$finalModel
```

```
##
## Call:
## randomForest(x = x, y = y, mtry = param$mtry)
```

```

##                Type of random forest: classification
##                Number of trees: 500
## No. of variables tried at each split: 27
##
##                OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A      B      C      D      E  class.error
## A 3904      2      0      0      0 0.0005120328
## B      6 2647      4      1      0 0.0041384500
## C      0      5 2391      0      0 0.0020868114
## D      0      0      9 2243      0 0.0039964476
## E      0      0      0      5 2520 0.0019801980

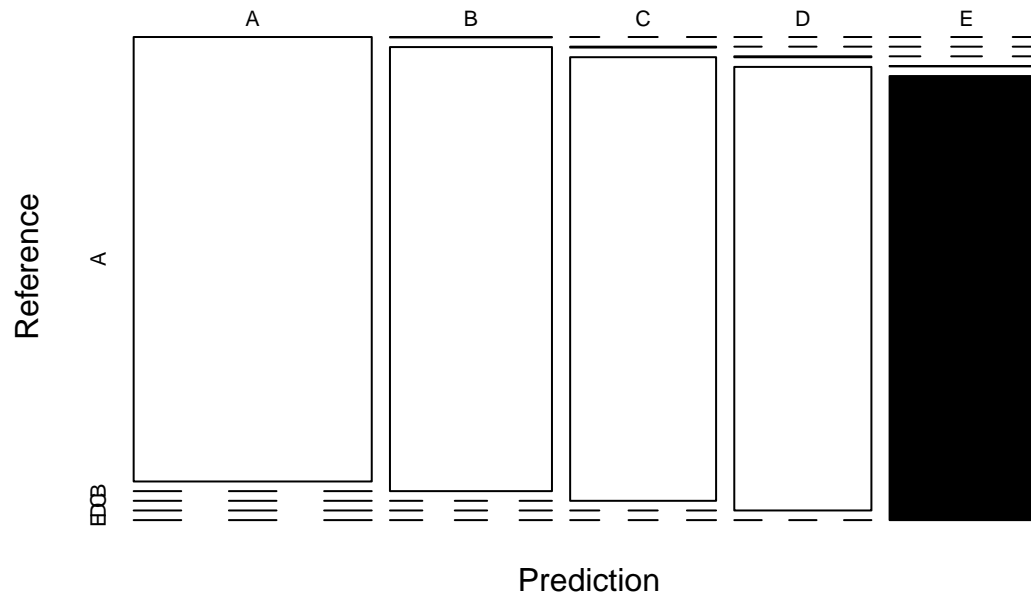
# prediction on Test dataset
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(as.factor(TestSet$classe),predictRandForest)
confMatRandForest

## Confusion Matrix and Statistics
##
##                Reference
## Prediction      A      B      C      D      E
##      A 1674      0      0      0      0
##      B      1 1138      0      0      0
##      C      0      2 1024      0      0
##      D      0      0      2  962      0
##      E      0      0      0      1 1081
##
## Overall Statistics
##
##                Accuracy : 0.999
##                95% CI : (0.9978, 0.9996)
##      No Information Rate : 0.2846
##      P-Value [Acc > NIR] : < 2.2e-16
##
##                Kappa : 0.9987
##
##      McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##                Class: A Class: B Class: C Class: D Class: E
## Sensitivity          0.9994   0.9982   0.9981   0.9990   1.0000
## Specificity          1.0000   0.9998   0.9996   0.9996   0.9998
## Pos Pred Value       1.0000   0.9991   0.9981   0.9979   0.9991
## Neg Pred Value       0.9998   0.9996   0.9996   0.9998   1.0000
## Prevalence           0.2846   0.1937   0.1743   0.1636   0.1837
## Detection Rate       0.2845   0.1934   0.1740   0.1635   0.1837
## Detection Prevalence 0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy     0.9997   0.9990   0.9988   0.9993   0.9999

# plot matrix results
plot(confMatRandForest$table, col = confMatRandForest$byClass,
     main = paste("Random Forest - Accuracy =",
                  round(confMatRandForest$overall['Accuracy'], 4)))

```

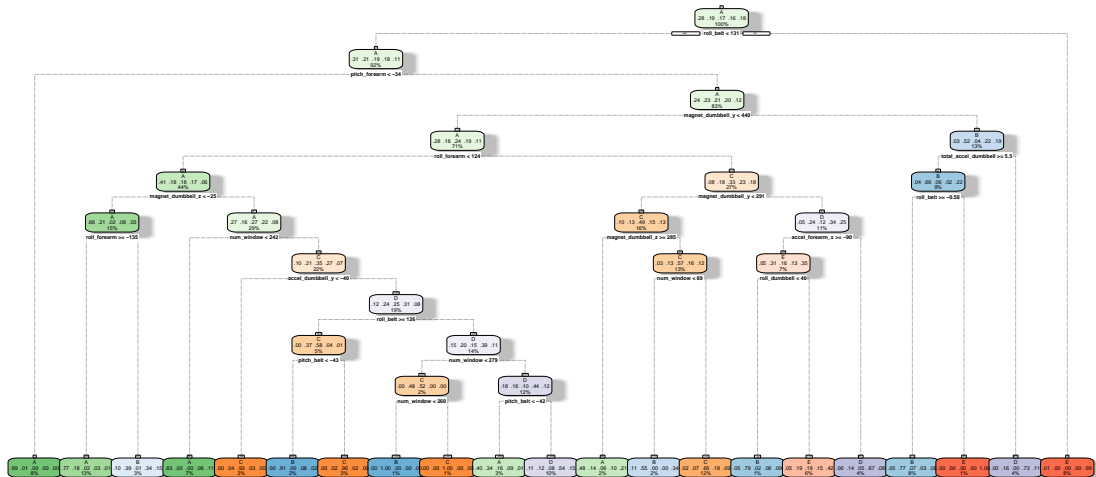
Random Forest – Accuracy = 0.999



2 Method: Decision Trees

```
# model fit
set.seed(12345)
modFitDecTree <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(modFitDecTree)
```

Warning: labs do not fit even at cex 0.15, there may be some overplotting



Rattle 2022–Oct–25 20:51:43 kamolchanoktangrsri

```
# prediction on Test dataset
predictDecTree <- predict(modFitDecTree, newdata=TestSet, type="class")
```

```
confMatDecTree <- confusionMatrix(predictDecTree, as.factor(TestSet$classe))
confMatDecTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##           A 1502  201   59   66   74
##           B   58  660   37   64  114
##           C    4   66  815  129   72
##           D   90  148   54  648  126
##           E   20   64   61   57  696
##
## Overall Statistics
##
##           Accuracy : 0.7342
##           95% CI : (0.7228, 0.7455)
##           No Information Rate : 0.2845
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.6625
##
## Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.8973   0.5795   0.7943   0.6722   0.6433
## Specificity      0.9050   0.9425   0.9442   0.9151   0.9579
## Pos Pred Value   0.7897   0.7074   0.7505   0.6079   0.7751
## Neg Pred Value   0.9568   0.9033   0.9560   0.9344   0.9226
## Prevalence       0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate   0.2552   0.1121   0.1385   0.1101   0.1183
## Detection Prevalence 0.3232   0.1585   0.1845   0.1811   0.1526
## Balanced Accuracy 0.9011   0.7610   0.8693   0.7936   0.8006
```

3 Method: Generalized Boosted Model

```
# model fit
set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM <- train(classe ~ ., data=TrainSet, method = "gbm",
                   trControl = controlGBM, verbose = FALSE)
modFitGBM$finalModel
```

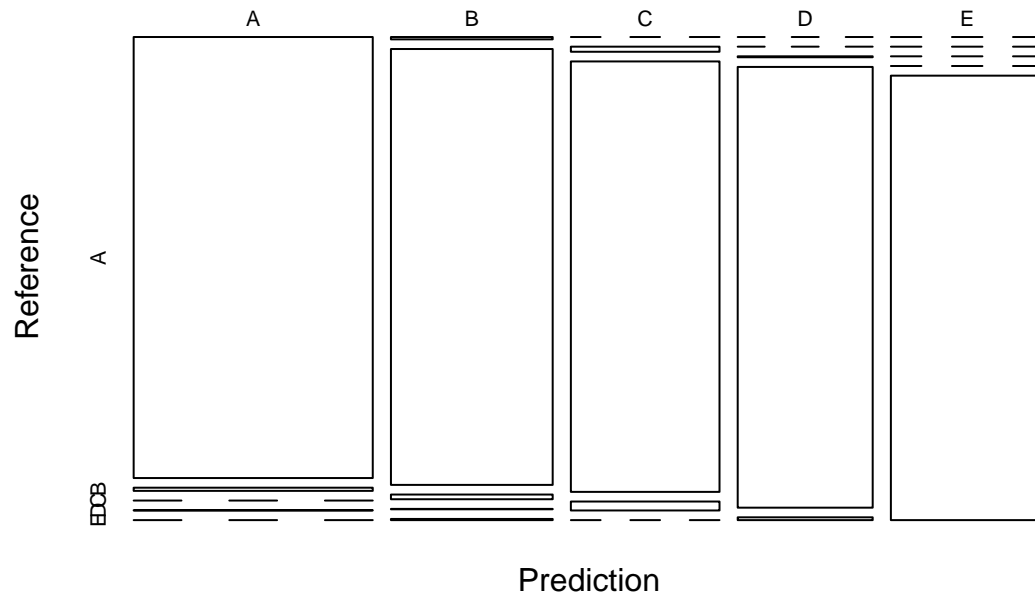
```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
# prediction on Test dataset
predictGBM <- predict(modFitGBM, newdata=TestSet)
confMatGBM <- confusionMatrix(predictGBM, as.factor(TestSet$classe))
confMatGBM
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##           A 1668   12    0    1    0
##           B    6 1115   12    1    3
##           C    0   12 1012   21    0
##           D    0    0    2  941    6
##           E    0    0    0    0 1073
##
## Overall Statistics
##
##           Accuracy : 0.9871
##           95% CI : (0.9839, 0.9898)
##           No Information Rate : 0.2845
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.9837
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.9964  0.9789  0.9864  0.9761  0.9917
## Specificity      0.9969  0.9954  0.9932  0.9984  1.0000
## Pos Pred Value   0.9923  0.9807  0.9684  0.9916  1.0000
## Neg Pred Value   0.9986  0.9949  0.9971  0.9953  0.9981
## Prevalence       0.2845  0.1935  0.1743  0.1638  0.1839
## Detection Rate   0.2834  0.1895  0.1720  0.1599  0.1823
## Detection Prevalence 0.2856  0.1932  0.1776  0.1613  0.1823
## Balanced Accuracy 0.9967  0.9871  0.9898  0.9873  0.9958
##
## plot matrix results
plot(confMatGBM$stable, col = confMatGBM$byClass,
     main = paste("GBM - Accuracy =", round(confMatGBM$overall['Accuracy'], 4)))
```

GBM – Accuracy = 0.9871



Applying the Selected Model to the Test Data

The accuracy of the 3 regression modeling methods above are:

- Random Forest : 0.999
- Decision Tree : 0.7342
- GBM : 0.9871 In that case, the Random Forest model will be applied to predict the 20 quiz results (testing dataset) as shown below.

```
predictTEST <- predict(modFitRandForest, newdata=testing)
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
```