PredictionAssignment

Sandeep Krishna R

02/02/2021

Introduction

This analysis involves the classification and an attempt at accurate prediction of the nature of a personal activity(exercise) performed by different individuals based on a plethora of continuously monitored variables pertinent to these activities.
Data Loading the data

```
training = read.csv("pml-training.csv")
testing = read.csv("pml-testing.csv")
dim(training)
```

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

```
table(training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

The dimensions indicate that there are around 160 measurements(predictor variables) and around 19622 observations. The classe is the response variable(nature of exercise), which is a factor with 5 levels.

Preprocessing

Partitioning the training set

The training set is split into train and validation sets.

```
library(caret)
set.seed(123)
inTrain = createDataPartition(training$classe, p = 0.7, list = FALSE)
train = training[inTrain, ]
validation = training[-inTrain, ]
```

Feature selection

The variables with near zero variance, more than 40% missing values and descriptive fields are not considered for the analysis.

```
nzvcol = nearZeroVar(train)
train = train[, -nzvcol]
cntlength <- sapply(train, function(x) {
  sum(!(is.na(x) | x == ""))
})
nullcol <- names(cntlength[cntlength < 0.6 * length(train$classe)])
descriptcol <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
                 "cvtd_timestamp", "new_window", "num_window")
excludecols <- c(descriptcol, nullcol)
train <- train[, !names(train) %in% excludecols]
```

Training the model

We implement the random forest method as it is a suitable tree-based method for classification type problems.

```
mod = train(classe~., data = train, method = "rf", ntrees = 10)
ptrain = predict(mod, train)
train$classe = as.factor(train$classe)
confusionMatrix(ptrain, train$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
```

```
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

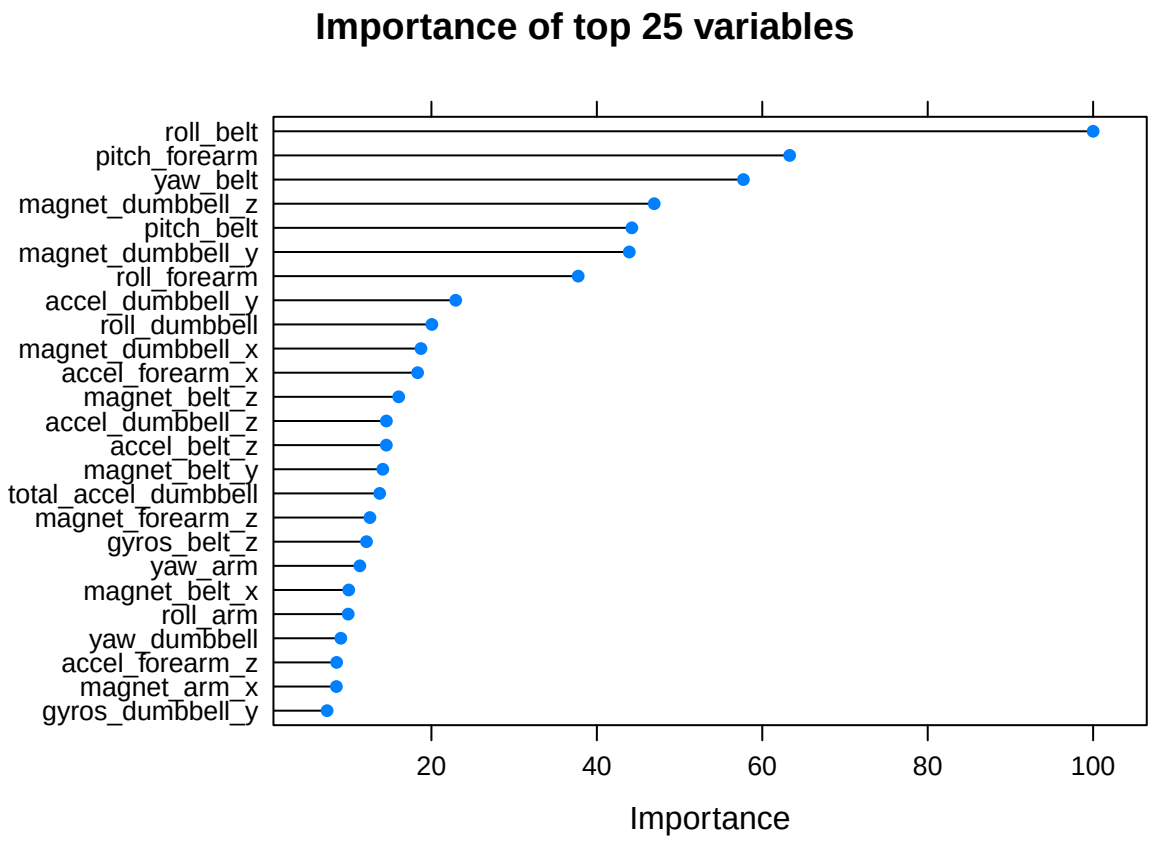
These results indicate that there is very little discrepancy in classifying the activity into the appropriate response class for the training data. But we are interested in the prediction accuracy of the model applied to any new data. Hence we now apply this to the validation set to cross-validate the accuracy obtained.

```
pvalidation = predict(mod, validation)
validation$classe = as.factor(validation$classe)
confusionMatrix(pvalidation, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    5    0    0    0
##          B    1 1127    5    0    0
##          C    0    7 1019   10    4
##          D    0    0    2  954    4
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9911, 0.9954)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9918
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9895   0.9932   0.9896   0.9926
## Specificity            0.9988   0.9987   0.9957   0.9988   1.0000
## Pos Pred Value         0.9970   0.9947   0.9798   0.9937   1.0000
## Neg Pred Value         0.9998   0.9975   0.9986   0.9980   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1915   0.1732   0.1621   0.1825
## Detection Prevalence   0.2851   0.1925   0.1767   0.1631   0.1825
## Balanced Accuracy      0.9991   0.9941   0.9944   0.9942   0.9963
```

The validation set accuracy is 99.35% with a very low p-value. This means that the Out-Of-Bag (OOB) error is very low.
Here the 25 variables with the highest importance in the predicted output are shown.

```
varobj = varImp(mod)
plot(varobj, main = "Importance of top 25 variables", top = 25)
```



Test set prediction

```
ptest = predict(mod, testing)
ptest
```

```
##  [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
```

Conclusion

The random forest method works best for this problem because:
1. It is suitable when handling a large number of predictors and their inter-relationship is unknown.
2. It's built-in cross-validation component gives an unbiased estimate.
3. The adaptive boosting or other combination of techniques are far more computationally intensive and are proven to produce similar or even lower accuracies in similar problems previously.