Practical machine learning

Rajat Kumar

September 30, 2018

Practical Machine Learning Course project

Overview

This project consists of a dataset which has records of people doing different activities and assciated readings which were associated with the activities. In this project data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants are used to train and test different machine learning algorithms and find which one has the highes accuracy. The assignment is divided into following parts:

- Getting and cleaning data
- Exploratory analysis of data
- Model building and testing
- Random Forest
- Decision Tree
- Generalized boosted regression
- Support vector machines (linear and radial)

Getting and Cleaning Data

```
# Download part
train_url <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_url <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if(!file.exists("Module_8_Assignment_1")){dir.create("Module_8_Assignment_1")}
download.file(train_url, destfile = "./Module_4_Assignment_2/pml-training",method = "curl")
download.file(test_url, destfile = "./Module_4_Assignment_2/pml-testing",method = "curl")
pml_training <- read.csv("./Module_4_Assignment_2/pml-training", header = T,na.strings=c("","NA","#DIVO
pml_testing <- read.csv("./Module_4_Assignment_2/pml-testing", header = T,na.strings=c("","NA","#DIVO!"
```

This part removes the NAs and data points like name, timestamp and redundant data points.

```
#Data Exploration
na_records <-sapply(pml_training, function(x) sum(is.na(x))) ##Finding nas in columns
without_na <- na_records[na_records == 0] ## taking columns with zero nas
cols <- names(without_na) ## selecting the column names
training_full <- pml_training[,cols] ## Passing the column names to the training dataset
training <- training_full[,-c(1:6)]
```

Applying the same data transformation to the testing data set

```
#applying the same transformations to testing set
na_records_test <-sapply(pml_testing, function(x) sum(is.na(x))) ##Finding nas in columns
without_na_test <- na_records_test[na_records_test == 0] ## taking columns with zero nas
cols_test <- names(without_na_test) ## selecting the column names
testing<- pml_testing[,cols_test]
testing <- testing[,-c(60)]
##Removing data which has very little signinficance for activity calculation
testing <- testing[, -c(1:6)]##name and time of records
```

Dividing the training data set into training and validation dataset

```
## Model buiding
library(caret)
```

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

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

```
train_in <- createDataPartition(training$classe, p= 0.95)[[1]]
training = training[train_in,]
validation <- training[-train_in,]
```

Exploratory analysis

Since the predictor variables are 54, it will be difficult to visualise the correlation and other interaction between variables. So, we will just take a look at the various variables and theie class types. Our target variable is "classe" variable.

```
#str(training)
```

Model Building

Random Forest

Random forest doesn't require cross validation as it calculates and adapts on the basis of out of the bag error since it samples with replacement.

```
#Random Forest
library(randomForest)
```

```
## randomForest 4.6-14
```

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

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

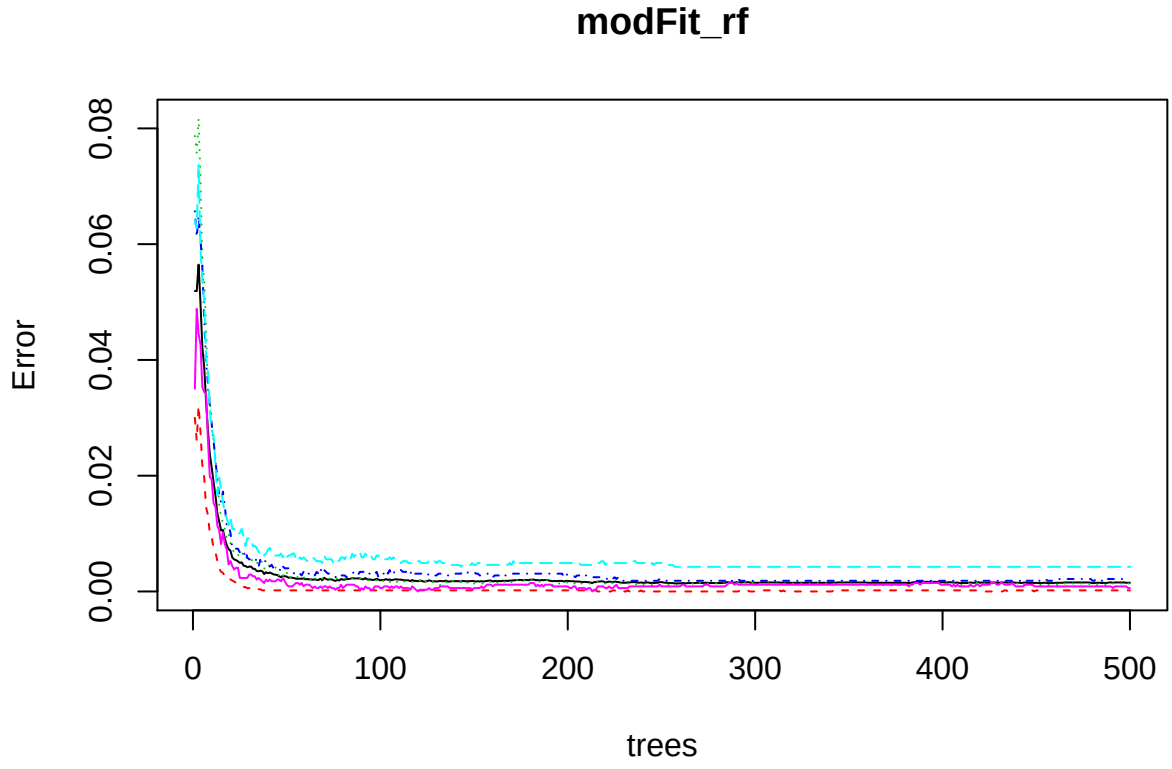
```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(12345)
modFit_rf <- randomForest(classe ~ ., data= training)
prediction_rf <- predict(modFit_rf, validation[,-54], type = "class")
cmrf <- confusionMatrix(prediction_rf,validation$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 264   0   0   0   0
##          B   0 184   0   0   0
##          C   0   0 158   0   0
##          D   0   0   0 150   0
##          E   0   0   0   0 175
##
## Overall Statistics
##
##                Accuracy : 1
```

```
##                  95% CI : (0.996, 1)
##     No Information Rate : 0.2836
##     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    1.000
## Specificity            1.0000   1.0000   1.0000   1.0000    1.000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Prevalence             0.2836   0.1976   0.1697   0.1611    0.188
## Detection Rate         0.2836   0.1976   0.1697   0.1611    0.188
## Detection Prevalence   0.2836   0.1976   0.1697   0.1611    0.188
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000    1.000
```

```
plot(modFit_rf)
```



Prediction with decision tree

```
#Prediction with Decision Trees
library(rpart)
set.seed(111)
```

```
modFit_trees <- train(classe ~ ., data = training, method = "rpart")
prediction_tress<- predict(modFit_trees, validation[,-54])
cmdt <- confusionMatrix(prediction_tress, validation$classe)
cmdt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 241  86  79  71  12
##          B   2  51   3  24  15
##          C  20  47  76  48  43
##          D   0   0   0   0   0
##          E   1   0   0   7 105
##
## Overall Statistics
##
##                Accuracy : 0.5081
##                  95% CI : (0.4754, 0.5406)
##     No Information Rate : 0.2836
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3573
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9129  0.27717  0.48101   0.0000   0.6000
## Specificity            0.6282  0.94110  0.79560   1.0000   0.9894
## Pos Pred Value         0.4928  0.53684  0.32479      NaN   0.9292
## Neg Pred Value         0.9480  0.84091  0.88235   0.8389   0.9144
## Prevalence             0.2836  0.19764  0.16971   0.1611   0.1880
## Detection Rate         0.2589  0.05478  0.08163   0.0000   0.1128
## Detection Prevalence   0.5252  0.10204  0.25134   0.0000   0.1214
## Balanced Accuracy      0.7705  0.60914  0.63831   0.5000   0.7947
```

Prediction using gbm

```
#prediction with gbm
set.seed(111)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modGBM  <- train(classe ~ ., data=training, method = "gbm", trControl = controlGBM, verbose = FALSE)
modGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 42 had non-zero influence.
```

```
print(modGBM)
```

```
## Stochastic Gradient Boosting
##
## 18643 samples
##    53 predictor
```

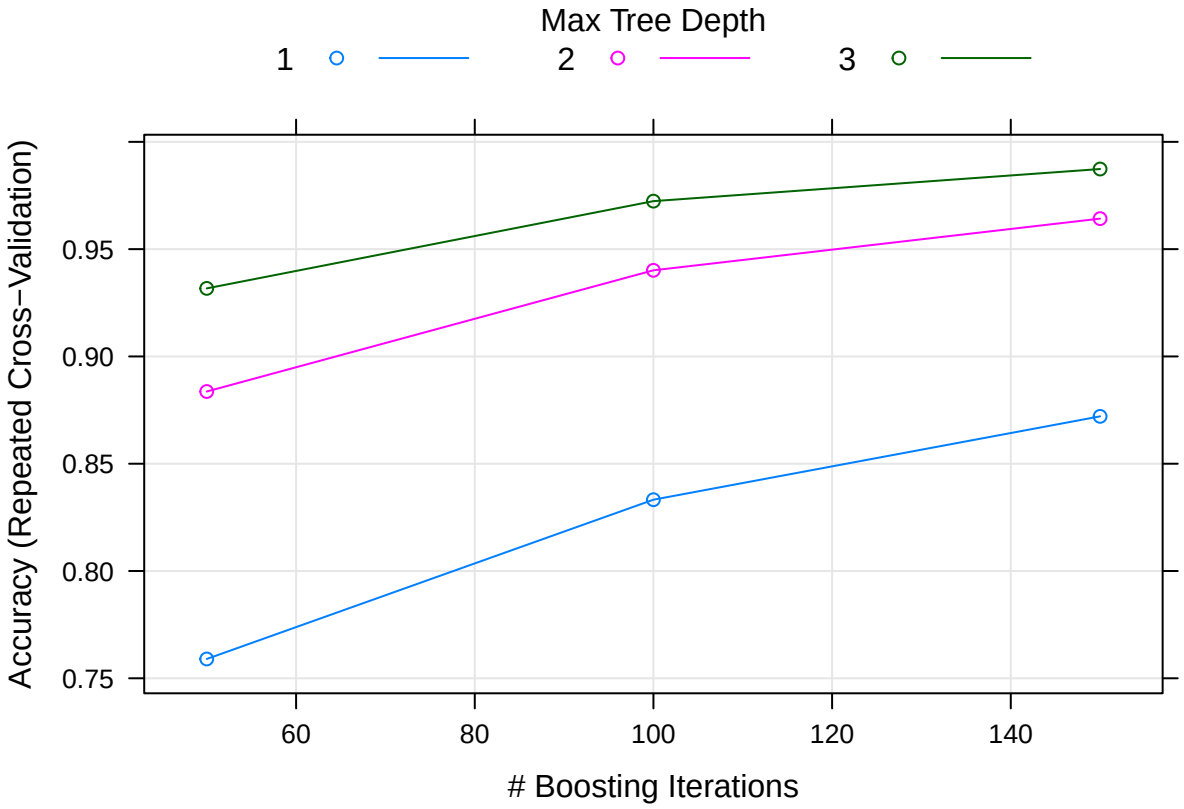
```
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 14914, 14915, 14915, 14913, 14915
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7589976  0.6942105
##   1                  100      0.8332353  0.7888206
##   1                  150      0.8720695  0.8380922
##   2                   50      0.8836562  0.8526374
##   2                  100      0.9401383  0.9242674
##   2                  150      0.9642224  0.9547391
##   3                   50      0.9317169  0.9135631
##   3                  100      0.9723760  0.9650510
##   3                  150      0.9873413  0.9839865
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
predictGBM <- predict(modGBM, newdata=validation[,-54])
cmGBM <- confusionMatrix(predictGBM, validation$classe)
cmGBM
```

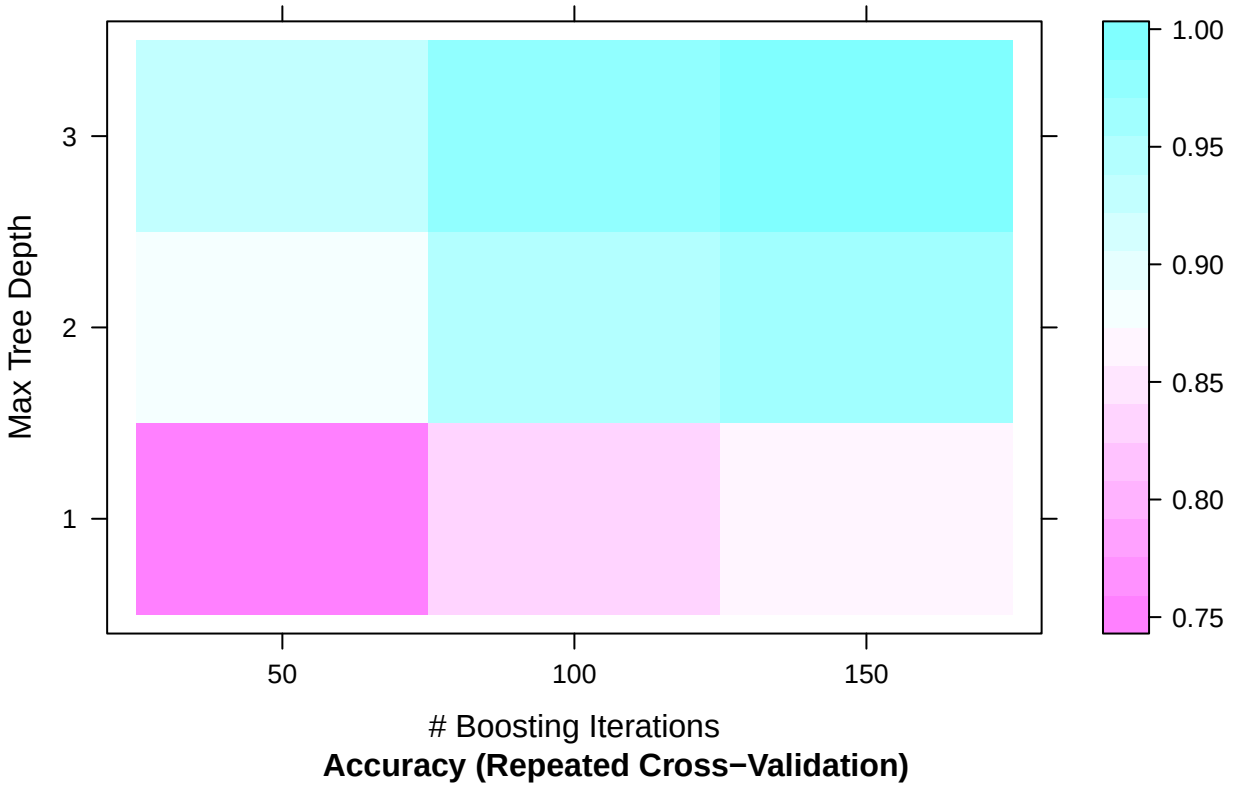
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 264   4   0   0   0
##          B   0 179   2   0   0
##          C   0   1 154   1   0
##          D   0   0   1 149   0
##          E   0   0   1   0 175
##
## Overall Statistics
##
##                Accuracy : 0.9893
##                  95% CI : (0.9803, 0.9948)
##     No Information Rate : 0.2836
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9864
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9728   0.9747   0.9933   1.0000
## Specificity            0.9940   0.9973   0.9974   0.9987   0.9987
## Pos Pred Value         0.9851   0.9890   0.9872   0.9933   0.9943
```

```
## Neg Pred Value         1.0000   0.9933   0.9948   0.9987   1.0000
## Prevalence             0.2836   0.1976   0.1697   0.1611   0.1880
## Detection Rate         0.2836   0.1923   0.1654   0.1600   0.1880
## Detection Prevalence   0.2879   0.1944   0.1676   0.1611   0.1890
## Balanced Accuracy      0.9970   0.9851   0.9860   0.9960   0.9993
```

```
plot(modGBM)
```




```
plot(modGBM, plotType = "level")
```



Prediction with support vector machine

```
#prediction with support vector machine
trctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
set.seed(111)

svm_Linear <- train(classe ~., data = training, method = "svmLinear",
                    trControl=trctrl,
                    preProcess = c("center", "scale"),
                    tuneLength = 10)
svm_Linear
```

```
## Support Vector Machines with Linear Kernel
##
## 18643 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (53), scaled (53)
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 14914, 14915, 14915, 14913, 14915
## Resampling results:
##
##   Accuracy   Kappa
##   0.7925229  0.7362397
##
```

```
## Tuning parameter 'C' was held constant at a value of 1
```

```
validate_pred <- predict(svm_Linear, newdata = validation[,-54])
cm_svml <- confusionMatrix(validate_pred, validation$classe)
cm_svml
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 245  29   9   6  11
##          B   4 126  17   6  19
##          C   2  11 121  16  12
##          D  10   3   8 114   8
##          E   3  15   3   8 125
##
## Overall Statistics
##
##                Accuracy : 0.7852
##                  95% CI : (0.7574, 0.8112)
##     No Information Rate : 0.2836
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7269
##  Mcnemar's Test P-Value : 1.846e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9280   0.6848   0.7658   0.7600   0.7143
## Specificity            0.9175   0.9384   0.9470   0.9629   0.9616
## Pos Pred Value         0.8167   0.7326   0.7469   0.7972   0.8117
## Neg Pred Value         0.9699   0.9236   0.9519   0.9543   0.9356
## Prevalence             0.2836   0.1976   0.1697   0.1611   0.1880
## Detection Rate         0.2632   0.1353   0.1300   0.1224   0.1343
## Detection Prevalence   0.3222   0.1847   0.1740   0.1536   0.1654
## Balanced Accuracy      0.9228   0.8116   0.8564   0.8614   0.8380
```

```
svm_Radial <- train(classe ~., data = training, method = "svmRadial",
                    trControl=trctrl,
                    preProcess = c("center", "scale"),
                    tuneLength = 10)

validate_pred_Radial <- predict(svm_Radial, newdata = validation[,-54])
cm_svmr<-confusionMatrix(validate_pred_Radial, validation$classe)
cm_svmr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 264   0   0   0   0
##          B   0 184   0   0   0
##          C   0   0 158   1   0
##          D   0   0   0 148   0
```

```
##          E   0   0   0   1 175
##
## Overall Statistics
##
##                Accuracy : 0.9979
##                  95% CI : (0.9923, 0.9997)
##     No Information Rate : 0.2836
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9973
##  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   0.9867   1.0000
## Specificity            1.0000   1.0000   0.9987   1.0000   0.9987
## Pos Pred Value         1.0000   1.0000   0.9937   1.0000   0.9943
## Neg Pred Value         1.0000   1.0000   1.0000   0.9974   1.0000
## Prevalence             0.2836   0.1976   0.1697   0.1611   0.1880
## Detection Rate         0.2836   0.1976   0.1697   0.1590   0.1880
## Detection Prevalence   0.2836   0.1976   0.1708   0.1590   0.1890
## Balanced Accuracy      1.0000   1.0000   0.9994   0.9933   0.9993
```

Conclusion

Comparing the different accuracies the random forest is the best algorithm to predict the test case with 100% accuracy

```
predictions<- predict(modFit_rf, testing, type = "class")
predictions
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```