Practical Machine Learning-Week4 Assignment

Ahad Zaman Khan

2022-09-26

Loading and processing The Data

I downloaded the data and reading it from local drive.

```
library(tidyverse)
list.files()
```

```
## [1] "Assignment.R"           "pml-testing.csv"        "pml-training.csv"
## [4] "Rplot.png"              "week4_assignment.html"  "week4_assignment.Rmd"
## [7] "week4_assignment_cache" "week4_assignment_files"
```

```
pml_train <- read_csv("pml-training.csv")
```

First column of the data is row numbers. Lets remove it from the data. I also check how the class variable is distributed.

```
pml_train <- pml_train[ , -1]

# names(pml_train)

length(unique(pml_train$classe))
```

```
## [1] 5
```

```
unique(pml_train$classe)
```

```
## [1] "A" "B" "C" "D" "E"
```

```
# find distribution of class variable-
pml_train %>% group_by(classe) %>%
  summarise( no_obs = n(), prcnt = 100* n()/nrow(.)  )
```

```
## # A tibble: 5 x 3
##   classe no_obs prcnt
##   <chr>   <int> <dbl>
## 1 A        5580  28.4
## 2 B        3797  19.4
## 3 C        3422  17.4
## 4 D        3216  16.4
## 5 E        3607  18.4
```

Lets check for missing values in the data-

```
# function to check a summary for "NA"s
na_count <-
  function(z) {
    num_obs = nrow(z)
    y =  map_df(z,  ~ sum(is.na(.x)))
```

```
    y = pivot_longer(y,
                     everything(),
                     names_to = "variable",
                     values_to = "num_na") %>%
      filter(num_na > 0)
    y$prcnt = y$num_na *100 /num_obs

    return(y)
  }

na_count(pml_train) %>% arrange(-prcnt)
```

```
## # A tibble: 100 x 3
##    variable               num_na prcnt
##    <chr>                   <int> <dbl>
##  1 skewness_roll_belt      19225  98.0
##  2 skewness_roll_dumbbell  19220  98.0
##  3 skewness_pitch_dumbbell 19217  97.9
##  4 kurtosis_roll_belt      19216  97.9
##  5 kurtosis_picth_belt     19216  97.9
##  6 kurtosis_yaw_belt       19216  97.9
##  7 skewness_roll_belt.1    19216  97.9
##  8 skewness_yaw_belt       19216  97.9
##  9 max_roll_belt           19216  97.9
## 10 max_picth_belt          19216  97.9
## # ... with 90 more rows
```

100 variables out of 159 variable has 97.93% to 97.97% records are missing "NA".I am omitting this variables and creating a new data named pml_train2. lets find out which variables does not have missing values:

```
pml_train2 <- pml_train %>%
  select( -na_count(pml_train)$variable   )
```

lets find the variables that are not numeric:

```
pml_train2 %>% select_if(~!is.numeric(.)) %>% names()
```

```
## [1] "user_name"      "cvtd_timestamp" "new_window"     "classe"
```

4 columns are not numerical. "classe" is dependent / predict variable. Ignoring other 3 variables.

```
pml_train2 <- pml_train2 %>%
  select( - c("user_name",  "cvtd_timestamp", "new_window") )
```

Partitioning the data

```
library(caret)

train_sample <- createDataPartition(y = pml_train2$classe, p = 0.7, list = FALSE)

training <- pml_train2[train_sample,]
testing <- pml_train2[-train_sample,]
```

Prediction models:

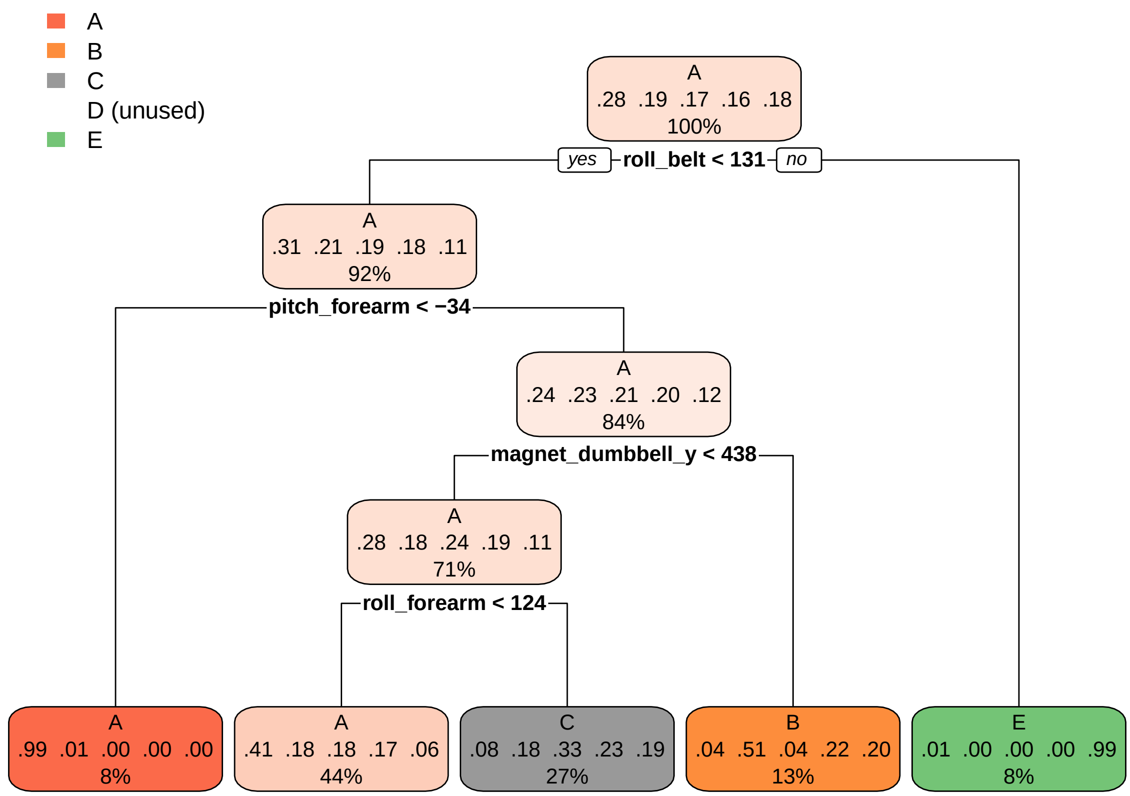
DECISION TREE

```
DT_model <- train(classe~.,
                  data = training, method = "rpart")

DT_prediction <- predict(DT_model, testing)

confusionMatrix(DT_prediction, as.factor(testing$classe))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1528  468  491  430  157
##          B   27  400   35  188  147
##          C  116  271  500  346  260
##          D    0    0    0    0    0
##          E    3    0    0    0  518
##
## Overall Statistics
##
##                Accuracy : 0.5006
##                  95% CI : (0.4877, 0.5135)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3469
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9128  0.35119  0.48733   0.0000  0.47874
## Specificity            0.6329  0.91635  0.79564   1.0000  0.99938
## Pos Pred Value         0.4971  0.50188  0.33490      NaN  0.99424
## Neg Pred Value         0.9481  0.85476  0.88024   0.8362  0.89485
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2596  0.06797  0.08496   0.0000  0.08802
## Detection Prevalence   0.5223  0.13543  0.25370   0.0000  0.08853
## Balanced Accuracy      0.7728  0.63377  0.64148   0.5000  0.73906
```

We can see prediction accuracy is only 52.05%.

```
rpart.plot:: rpart.plot(DT_model$finalModel,
                        roundint = FALSE)
```



RANDOM FOREST MDOEL

```r
rm_mdl <- train(classe~., data = training, method = "rf", ntree = 250)

rf_predict <- predict(rm_mdl, testing)
confusionMatrix(rf_predict, factor(testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1137    2    0    0
##          C    0    0 1024    1    0
##          D    0    0    0  962    0
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9978, 0.9996)
##     No Information Rate : 0.2845
##     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            1.0000   0.9982   0.9981   0.9979   1.0000
## Specificity            0.9995   0.9996   0.9998   1.0000   0.9998
## Pos Pred Value         0.9988   0.9982   0.9990   1.0000   0.9991
## Neg Pred Value         1.0000   0.9996   0.9996   0.9996   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1932   0.1740   0.1635   0.1839
## Detection Prevalence   0.2848   0.1935   0.1742   0.1635   0.1840
## Balanced Accuracy      0.9998   0.9989   0.9989   0.9990   0.9999
```

Gradient boosting model:

```r
gbm_model <- train(classe~., data = training, method = "gbm", verbose = FALSE )

gbm_model$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 55 predictors of which 55 had non-zero influence.
```

```r
gbm_predict <- predict(gbm_model, testing)

confusionMatrix(gbm_predict, factor(testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1135    3    0    0
##          C    0    2 1020    3    0
##          D    0    0    3  957    1
##          E    0    0    0    4 1081
##
## Overall Statistics
##
##                Accuracy : 0.9969
##                  95% CI : (0.9952, 0.9982)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9961
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9942   0.9927   0.9991
## Specificity            0.9995   0.9994   0.9990   0.9992   0.9992
## Pos Pred Value         0.9988   0.9974   0.9951   0.9958   0.9963
```

```
## Neg Pred Value         1.0000   0.9992   0.9988   0.9986   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1733   0.1626   0.1837
## Detection Prevalence   0.2848   0.1934   0.1742   0.1633   0.1844
## Balanced Accuracy      0.9998   0.9979   0.9966   0.9960   0.9991
```

I can see both Random forest and Gradient boosting model are highely accurate. However, Random forest is better performing and I will use this as my prediction model.

LOADING TESTING DATA and cleaning in same manner as test data

```
pml_test <- read_csv("pml-testing.csv")

pml_test <- pml_test[ ,-1]

pml_test <- pml_test %>%
  select( -na_count(pml_train)$variable   )

pml_test <- pml_test %>%
  select( - c("user_name",  "cvtd_timestamp", "new_window") )
```

Final Prediction:

```
rf_final_predition <- predict(rm_mdl, pml_test)

rf_final_predition
```

```
##  [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
```