Practical Machine Learning Assignment- Exercise Class Prediction

Amrendra Kumar

12 August 2018

Load packages

```
library(randomForest)
```

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

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

```
library(caret)
```

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

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

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

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

Load Training Data from URL

```
TrainURL="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
data <- read.csv(url(TrainURL))
dim(data)
```

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

Replace blank with NA

```
data[data==""] <- NA
```

Identifying columns that has NA value in more than 50% records

```
i=0
x=c()
for(i in 1:ncol(data)){
  if (sum(is.na(data[i]))<nrow(data)/2) {
    x=append(x,names(data[i]))
  }
}
```

Create a new dataset using the training data with the columns identified in the above step

```
data= data[,x]
dim(data)
```

```
## [1] 19622    60
```

Split data into train(80%) and test(20%)

```
rows=seq(1,nrow(data),1)
train_rows=sample(x=rows,size=(0.8*nrow(data)))
train_data=data[train_rows,]
test_data=data[-train_rows,]
```

Create model using random forest as it is one of the best model used for miulti classification. Exclude variable X and user_name as these have no significance in the prediction.

```
attach(train_data)
model = randomForest(classe~. -X -user_name, data=train_data, metric="Accura
cy", importance=TRUE, ntree=500)
```

Order the predictors based on their importance.

```
df_imp=importance(model)[,6:7]
df_imp=as.data.frame((df_imp))
df_imp[order(-df_imp$MeanDecreaseAccuracy),]
```

```
##                      MeanDecreaseAccuracy MeanDecreaseGini
## raw_timestamp_part_1           68.4995943     1314.9569445
## cvtd_timestamp                 60.8737550     1930.5530101
## num_window                     43.1449570      738.3167169
## yaw_belt                       41.8675693      483.3177292
## roll_belt                      38.8138461      703.8973635
## pitch_belt                     38.0898489      408.1248005
## magnet_dumbbell_y              31.5327854      403.0012290
## magnet_dumbbell_z              31.3077837      394.1117451
## accel_dumbbell_y               27.8444213      248.5528989
## pitch_forearm                  26.8754536      415.1685658
## roll_arm                       26.0892352      172.0266808
## magnet_forearm_z               25.5411485      128.7934267
## gyros_arm_y                    25.3562177       53.4967986
## magnet_belt_y                  24.5874629      284.7954415
## gyros_dumbbell_x               23.7048009       55.6691143
## gyros_forearm_y                23.6133971       48.7256722
## accel_dumbbell_z               23.5666541      178.6044785
## magnet_dumbbell_x              23.4037229      318.1942237
## roll_dumbbell                  23.3006581      254.1151541
## gyros_belt_z                   23.2291064      151.6597191
## gyros_dumbbell_z               22.9031459       28.3836393
## magnet_belt_z                  22.5059700      236.9060387
## gyros_forearm_z                22.4358893       33.4895463
## gyros_dumbbell_y               21.8034062      126.7855821
## accel_dumbbell_x               21.6295102      171.0662181
## magnet_belt_x                  21.3129667      144.5624624
## total_accel_dumbbell           21.3025053      172.9905168
## gyros_arm_x                    21.2562555       57.2666384
## magnet_arm_z                   21.0808213       72.2474702
## accel_arm_y                    20.9941254       68.7078139
## roll_forearm                   20.3742084      316.1963252
## accel_belt_z                   20.3695338      249.6805552
## accel_forearm_z                20.3425163      124.4650501
## yaw_dumbbell                   20.3120526      152.4210370
## magnet_forearm_y               19.9513567       96.7283626
## accel_forearm_x                19.8877478      177.9520959
## raw_timestamp_part_2           19.2071503       12.3720307
## accel_belt_x                   18.8818706       85.3104699
## yaw_arm                        18.6360725      101.9462224
## gyros_forearm_x                18.2420651       30.6043271
## accel_forearm_y                18.0257378       57.3168875
## gyros_belt_x                   17.9879866       49.8899381
## yaw_forearm                    17.8804455       71.2222578
## pitch_arm                      17.8113755       75.5990416
## accel_arm_z                    16.5522579       49.4673837
## total_accel_forearm            16.4157212       45.5633543
## magnet_forearm_x               16.0574271       91.0120062
## total_accel_arm                15.6256223       37.4991324
## accel_belt_y                   15.6171873       89.1774992
```

```
## total_accel_belt               15.0271577      169.0247280
## gyros_belt_y                   14.7363368       65.3347245
## accel_arm_x                    14.7108446      125.0529625
## gyros_arm_z                    14.4795571       21.8494161
## pitch_dumbbell                 14.4785353      104.0672744
## magnet_arm_y                   13.3168378       98.3471225
## magnet_arm_x                   13.0254888      114.2542085
## new_window                     -0.3534639        0.1640155
```

Create a model using top 10 predictors based on their importance and validate the model using the test data.

```
model1=randomForest(classe~yaw_belt+ num_window+ roll_belt + pitch_belt+  ma
gnet_dumbbell_y + magnet_dumbbell_z + accel_dumbbell_y+ pitch_forearm + roll
_arm + roll_dumbbell, metric="Accuracy" ,ntree=500)
model1
```

```
##
## Call:
##  randomForest(formula = classe ~ yaw_belt + num_window + roll_belt +
pitch_belt + magnet_dumbbell_y + magnet_dumbbell_z + accel_dumbbell_y +
pitch_forearm + roll_arm + roll_dumbbell, metric = "Accuracy",      ntree =
500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 0.18%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4458    1    1    1    0 0.000672495
## B    3 3020    2    1    0 0.001982816
## C    0    6 2709    2    0 0.002944424
## D    0    0    4 2577    0 0.001549787
## E    0    2    2    3 2905 0.002403846
```

```
pred1=predict(model1,test_data)
confusionMatrix(pred1,test_data$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1119    3    0    0    0
##          B    0  768    1    0    0
##          C    0    0  704    0    1
##          D    0    0    0  635    2
##          E    0    0    0    0  692
##
## Overall Statistics
##
##                Accuracy : 0.9982
##                  95% CI : (0.9963, 0.9993)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9977
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9961   0.9986   1.0000   0.9957
## Specificity            0.9989   0.9997   0.9997   0.9994   1.0000
## Pos Pred Value         0.9973   0.9987   0.9986   0.9969   1.0000
## Neg Pred Value         1.0000   0.9990   0.9997   1.0000   0.9991
## Prevalence             0.2851   0.1964   0.1796   0.1618   0.1771
## Detection Rate         0.2851   0.1957   0.1794   0.1618   0.1763
## Detection Prevalence   0.2859   0.1959   0.1796   0.1623   0.1763
## Balanced Accuracy      0.9995   0.9979   0.9991   0.9997   0.9978
```

K-fold Cross Validation

```
train_control <- trainControl(method="cv", number=10)

cv_model1 <- train(classe~yaw_belt+ num_window+ roll_belt + pitch_belt+  mag
net_dumbbell_y + magnet_dumbbell_z + accel_dumbbell_y+ pitch_forearm + roll_
arm+ roll_dumbbell, data=train_data, trControl=train_control, method="rf", m
etric="Accuracy", ntree=500)
cv_model1
```

```
## Random Forest
##
## 15697 samples
##    10 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 14126, 14127, 14126, 14127, 14129, 14127, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9975790  0.9969379
##    6    0.9975155  0.9968576
##   10    0.9963689  0.9954074
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
cv_pred1=predict(cv_model1,test_data)
confusionMatrix(cv_pred1,test_data$classe)$overall[1]
```

```
## Accuracy
## 0.997707
```

Create another model using top 4 predictors based on their importance and test the model using the test data.

```
model2=randomForest(classe~yaw_belt+ num_window+ roll_belt + pitch_belt ,dat
a=train_data, metric="Accuracy", ntree=500)
model2
```

```
##
## Call:
##  randomForest(formula = classe ~ yaw_belt + num_window + roll_belt +
pitch_belt, data = train_data, metric = "Accuracy", ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.05%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4461    0    0    0    0 0.0000000000
## B    1 3023    1    1    0 0.0009914078
## C    0    0 2717    0    0 0.0000000000
## D    0    1    1 2579    0 0.0007748935
## E    0    1    0    2 2909 0.0010302198
```

```
pred2=predict(model2,test_data)
confusionMatrix(pred2,test_data$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1119    0    0    0    0
##          B    0  771    0    0    0
##          C    0    0  705    0    0
##          D    0    0    0  635    4
##          E    0    0    0    0  691
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9974, 0.9997)
##     No Information Rate : 0.2851
##     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   1.0000   1.0000   1.0000   0.9942
## Specificity            1.0000   1.0000   1.0000   0.9988   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   0.9937   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   0.9988
## Prevalence             0.2851   0.1964   0.1796   0.1618   0.1771
## Detection Rate         0.2851   0.1964   0.1796   0.1618   0.1761
## Detection Prevalence   0.2851   0.1964   0.1796   0.1628   0.1761
## Balanced Accuracy      1.0000   1.0000   1.0000   0.9994   0.9971
```

K-fold cross validation

```
cv_model2 <- train(classe~yaw_belt+ num_window+ roll_belt + pitch_belt, data
=train_data, trControl=train_control, method="rf" , metric="Accuracy", ntree
=500)
cv_model2
```

```
## Random Forest
##
## 15697 samples
##     4 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 14128, 14127, 14126, 14128, 14127, 14128, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9994903  0.9993553
##   3     0.9993629  0.9991942
##   4     0.9991081  0.9988720
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
cv_pred2=predict(cv_model2,test_data)
confusionMatrix(cv_pred2,test_data$classe)$overall[1]
```

```
##  Accuracy
## 0.9989809
```

The prediction accuracy of model cv_model2 is the highest so we will use it for predicting the unseen data.