PML - Course Project

JJW

11/19/2020

```
## load libraries
library(caret)
library(parallel)
library(doParallel)
```

Download and Clean Data

```
## download and load data
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "pml-training.csv")
training <- read.csv("pml-training.csv")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", "pml-testing.csv")
testing <- read.csv("pml-testing.csv")
```

Variables with a high percentage of NA values are removed, as they will not provide enough information in training the models.

```
## remove variables high percentage of NA
maxNApercent = 0.5
maxNAcount <- nrow(training) * maxNApercent
removeVars <- which(colSums(is.na(training) | training =="") > maxNAcount)
training <- training[,-removeVars]
testing <- testing[,-removeVars]
```

Other irrelevant variables are also removed.

```
## remove irrelevant index, identifier, and time variables
testing <- testing[,-c(1:7)]
training <- training[, -c(1:7)]
```

The training data set is split to facilitate model stacking.

```
## split training into two data sets
set.seed(333)
inTrain <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
training1 <- training[inTrain,]
training2 <- training[-inTrain,]
```

Train models

The first step will be to train three models using random forest, boosted trees, and linear discriminant analysis.

```
## Use parallel processing to increase performance
cluster <- makeCluster(detectCores()-1)
registerDoParallel(cluster)
fitControl <- trainControl(method="cv", number=20, allowParallel = TRUE)

## random forest
modrf <- train(classe ~ ., method = "rf", data = training1, trControl = fitControl)
## boosted trees
modgbm <- train(classe ~ ., method = "gbm", data = training1, trControl = fitControl)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094            -nan     0.1000    0.2348
##      2        1.4600            -nan     0.1000    0.1628
##      3        1.3579            -nan     0.1000    0.1154
##      4        1.2820            -nan     0.1000    0.1127
##      5        1.2111            -nan     0.1000    0.0933
##      6        1.1523            -nan     0.1000    0.0773
##      7        1.1038            -nan     0.1000    0.0655
##      8        1.0622            -nan     0.1000    0.0606
##      9        1.0226            -nan     0.1000    0.0661
##     10        0.9818            -nan     0.1000    0.0456
##     20        0.7521            -nan     0.1000    0.0244
##     40        0.5251            -nan     0.1000    0.0094
##     60        0.4015            -nan     0.1000    0.0096
##     80        0.3187            -nan     0.1000    0.0052
##    100        0.2603            -nan     0.1000    0.0021
##    120        0.2187            -nan     0.1000    0.0023
##    140        0.1866            -nan     0.1000    0.0021
##    150        0.1729            -nan     0.1000    0.0013
```

```
## linear discriminant analysis
modlda <- train(classe ~ ., method = "lda", data = training1, trControl = fitControl)

## de-register parallel processing cluster
stopCluster(cluster)
registerDoSEQ()
```

The second step is to make predictions using the model and check their accuracy.

```
## make predictions on second training set for each  model
predrf <- predict(modrf,training2)
predgbm <- predict(modgbm,training2)
predlda <- predict(modlda,training2)

## check rf model accuracy
confusionMatrix(training2$classe, predrf)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    3    0    0    0
##          B    3 1134    2    0    0
##          C    0   10 1015    1    0
##          D    0    0   26  937    1
##          E    0    0    0   12 1070
## 
## Overall Statistics
## 
##                Accuracy : 0.9901
##                  95% CI : (0.9873, 0.9925)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9875
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9887   0.9732   0.9863   0.9991
## Specificity            0.9993   0.9989   0.9977   0.9945   0.9975
## Pos Pred Value         0.9982   0.9956   0.9893   0.9720   0.9889
## Neg Pred Value         0.9993   0.9973   0.9942   0.9974   0.9998
## Prevalence             0.2845   0.1949   0.1772   0.1614   0.1820
## Detection Rate         0.2839   0.1927   0.1725   0.1592   0.1818
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9987   0.9938   0.9854   0.9904   0.9983
```

```
## check gbm model accuracy
confusionMatrix(training2$classe, predgbm)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1641   26    4    2    1
##          B   35 1055   48    0    1
##          C    0   37  974   13    2
##          D    0    3   41  912    8
##          E    2   14    8   29 1029
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9534          
##                  95% CI : (0.9477, 0.9587)
##     No Information Rate : 0.2851          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9411          
##                                           
##  Mcnemar's Test P-Value : 6.305e-08       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9779   0.9295   0.9060   0.9540   0.9885
## Specificity            0.9922   0.9823   0.9892   0.9895   0.9891
## Pos Pred Value         0.9803   0.9263   0.9493   0.9461   0.9510
## Neg Pred Value         0.9912   0.9831   0.9792   0.9911   0.9975
## Prevalence             0.2851   0.1929   0.1827   0.1624   0.1769
## Detection Rate         0.2788   0.1793   0.1655   0.1550   0.1749
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9851   0.9559   0.9476   0.9717   0.9888
```

```
# check lda model accuracy
confusionMatrix(training2$classe, predlda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1389   42  109  131    3
##          B  178  705  146   50   60
##          C  116   99  675  105   31
##          D   50   39  115  713   47
##          E   49  206   84  121  622
##
## Overall Statistics
##
##                Accuracy : 0.6974
##                  95% CI : (0.6854, 0.7091)
##     No Information Rate : 0.3028
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6167
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7795   0.6462   0.5979   0.6366   0.8152
## Specificity            0.9305   0.9095   0.9262   0.9473   0.9102
## Pos Pred Value         0.8297   0.6190   0.6579   0.7396   0.5749
## Neg Pred Value         0.9067   0.9187   0.9066   0.9173   0.9706
## Prevalence             0.3028   0.1854   0.1918   0.1903   0.1297
## Detection Rate         0.2360   0.1198   0.1147   0.1212   0.1057
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8550   0.7778   0.7620   0.7920   0.8627
```

At about 99% accuracy, the random forest method seems to make the best predictions.

The third step is to stack the three models (using random forest) and check its accuracy.

```
## Use parallel processing to increase performance
cluster <- makeCluster(detectCores()-1)
registerDoParallel(cluster)
fitControl <- trainControl(method="cv", number=20, allowParallel = TRUE)

## stack the models
predDF <- data.frame(predrf, predgbm, predlda, classe = training2$classe)
modstacked <- train(classe ~ ., method = "rf", data=predDF, trControl = fitControl)

## de-register parallel processing cluster
stopCluster(cluster)
registerDoSEQ()

## check accuracy
predstacked <- predict(modstacked, predDF)
confusionMatrix(predDF$classe, predstacked)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    3    0    0    0
##          B    3 1134    2    0    0
##          C    0   10 1015    1    0
##          D    0    0   26  937    1
##          E    0    0    0   12 1070
##
## Overall Statistics
##
##                Accuracy : 0.9901
##                  95% CI : (0.9873, 0.9925)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9875
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9887   0.9732   0.9863   0.9991
## Specificity            0.9993   0.9989   0.9977   0.9945   0.9975
## Pos Pred Value         0.9982   0.9956   0.9893   0.9720   0.9889
## Neg Pred Value         0.9993   0.9973   0.9942   0.9974   0.9998
## Prevalence             0.2845   0.1949   0.1772   0.1614   0.1820
## Detection Rate         0.2839   0.1927   0.1725   0.1592   0.1818
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9987   0.9938   0.9854   0.9904   0.9983
```

The stacked model appears to be as accurate as the single random forest model, so the random forest model will be used as the final model.

Make out-of-sample predictions

```
## make predictions on testing data
predtesting <- predict(modrf, testing)
```

Given its high accuracy in predicting the second training data set, it is anticipated that the final model will make out-of-sample predictions with near-perfect accuracy.

]