author: May Robinson

Practical Machine Learning Prediction Assignment

The measurement data were collected from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. Participants performed barbell lifts, correctly and incorrectly, in 5 different ways. The five different ways corresponded to the “classe” variable with five levels (A B C D E).To calculate the out-of-sample error, the training data was split into a training set and a test set.
The goal of this assignment was to predict classe given the measurement data variables. The caret package was used. A tree model was computed initially with an accuracy of .5717. Next a random forest model was fitted, which resulted in a far better accuracy of .9980. The tree model took 4.22 seconds to compute, while the random forest model took 271.61 seconds. Random forest model was chosen to predict 20 different test cases.
==========

Read in training data

```
training <-read.csv("pml-training.csv")
testing <-read.csv("pml-testing.csv")
```

There were 160 variables in the training and test sets. Decide which variables to keep in the model. Build a new training data set and test set. Remove columns at the beginning of the training set (with names and timestamps) that do not contribute to the model.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
training<-training[,7:160]
testing<-testing[,7:160]
```

Remove columns with NAs, since Caret does not like NAs

```
complete_data<-apply(!is.na(training),2,sum)>19621
training<-training[,complete_data]
testing<-testing[,complete_data]
```

Remove columns with skewness, kurtosis, min, max and amplitudes These values were functions of other variables in the data and were not necessary for the model.

```
training <- select(training,-kurtosis_roll_belt,-kurtosis_picth_belt,
          -kurtosis_yaw_belt,-skewness_roll_belt,-skewness_roll_belt.1,
-skewness_yaw_belt,-max_yaw_belt,-min_yaw_belt,-amplitude_yaw_belt,
-kurtosis_roll_arm,-kurtosis_picth_arm,-kurtosis_yaw_arm,-skewness_roll_arm,
-skewness_pitch_arm,-skewness_yaw_arm, -kurtosis_roll_dumbbell,
-kurtosis_picth_dumbbell,-kurtosis_yaw_dumbbell,-skewness_roll_dumbbell,
-skewness_pitch_dumbbell,-skewness_yaw_dumbbell,-max_yaw_dumbbell
-min_yaw_dumbbell,amplitude_yaw_dumbbell,-max_yaw_dumbbell,-min_yaw_dumbbell,
-amplitude_yaw_dumbbell,-kurtosis_roll_forearm,-kurtosis_picth_forearm,
-kurtosis_yaw_forearm,-skewness_roll_forearm,-skewness_pitch_forearm,

-skewness_yaw_forearm,-max_yaw_forearm, -min_yaw_forearm,- amplitude_yaw_forearm)



testing <-select(testing,-kurtosis_roll_belt,-kurtosis_picth_belt,
                 -kurtosis_yaw_belt,-skewness_roll_belt,-skewness_roll_belt.1,
                 -skewness_yaw_belt,-max_yaw_belt,-min_yaw_belt,-amplitude_yaw_belt,
                 -kurtosis_roll_arm,-kurtosis_picth_arm,-kurtosis_yaw_arm,-skewness_roll_arm,
                 -skewness_pitch_arm,-skewness_yaw_arm, -kurtosis_roll_dumbbell,
                 -kurtosis_picth_dumbbell,-kurtosis_yaw_dumbbell,-skewness_roll_dumbbell,
                 -skewness_pitch_dumbbell,-skewness_yaw_dumbbell,-max_yaw_dumbbell
                 -min_yaw_dumbbell,-amplitude_yaw_dumbbell,-max_yaw_dumbbell,-min_yaw_dumbbell,
                 -amplitude_yaw_dumbbell,-kurtosis_roll_forearm,-kurtosis_picth_forearm,
                 -kurtosis_yaw_forearm,-skewness_roll_forearm,-skewness_pitch_forearm,

                 -skewness_yaw_forearm,-max_yaw_forearm, -min_yaw_forearm,
                 - amplitude_yaw_forearm)
dim(training)
```

```
## [1] 19622    54
```

```
dim(testing)
```

```
## [1] 20 54
```

Subset the training set into a training data set and test data set

```
set.seed(1234)
training_dat <- createDataPartition( training$classe, p = 0.7, list = FALSE)
trainData <- training[training_dat, ]
testData <- training[-training_dat, ]
dim(trainData)
```

```
## [1] 13737    54
```

```
dim(testData)
```

```
## [1] 5885   54
```

Fit a tree model using the caret package. To use cross validation, include the trControl specification, with “cv” specifying cross validation, 3 for the number of folds.

```
model_tree <- train(classe ~ ., data = trainData, method = 'rpart',
                  trControl=trainControl(method="cv", number=3, allowParallel=T))
system.time( train(classe ~ ., data = trainData, method = 'rpart',
                   trControl=trainControl(method="cv", number=3, allowParallel=T)))
```

```
##    user  system elapsed
##    5.10    0.13    5.23
```

```
print(model_tree)
```

```
## CART
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9158, 9158, 9158
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03885668  0.5716678  0.45347441
##   0.06092971  0.3679843  0.12762472
##   0.11738379  0.3390842  0.08360689
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03885668.
```

The accuracy of the tree model was .5717. The in-sample error of the tree model was 1- .5717= .4283*100= 42.83%. Calculate the out-of-sample error

```
prediction <- predict(model_tree, testData)
confusionMatrix(prediction, testData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  486  493  452  168
##          B   35  379   31  164  145
##          C  105  274  502  348  302
##          D    0    0    0    0    0
##          E    4    0    0    0  467
##
## Overall Statistics
##
##                Accuracy : 0.489
##                  95% CI : (0.4762, 0.5019)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3311
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140   0.3327   0.4893   0.0000  0.43161
## Specificity            0.6203   0.9210   0.7882   1.0000  0.99917
## Pos Pred Value         0.4890   0.5027   0.3279      NaN  0.99151
## Neg Pred Value         0.9478   0.8519   0.8797   0.8362  0.88641
## Prevalence             0.2845   0.1935   0.1743   0.1638  0.18386
## Detection Rate         0.2600   0.0644   0.0853   0.0000  0.07935
## Detection Prevalence   0.5317   0.1281   0.2602   0.0000  0.08003
## Balanced Accuracy      0.7671   0.6269   0.6388   0.5000  0.71539
```

Out-of-sample error was 1- .489 = .5110 * 100 = 51.10%. Try for a better accuracy by fitting a random forest model using the caret package.To use cross validation,include the “trControl=” specification in the caret train function, with “cv” for cross validation, 3 for the number of folds.

```
model_rf <- train(classe ~ ., data = trainData, method = 'rf',
                  trControl=trainControl(method="cv", number=3, allowParallel=T))
system.time(train(classe ~ ., data = trainData, method = 'rf',
                  trControl=trainControl(method="cv", number=3, allowParallel=T)))
```

```
##    user  system elapsed
##  277.81    1.29  279.10
```

Print out results of random forest model

```
print(model_rf)
```

```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9158, 9158, 9158
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9926476  0.9906988
##   27    0.9952683  0.9940147
##   53    0.9936667  0.9919889
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

The Accuracy of this model was .9924. From this calculate the in-sample error rate: (1 - .9924 = 0.0076 * 100)= 0.76%.

Use the test set to calculate the out-of-sample error rate.

```
pred_test <- predict(model_rf, testData)
outOfSampleError.accuracy <- sum(pred_test == testData$classe)/length(pred_test)

outOfSampleError.accuracy
```

```
## [1] 0.9976211
```

The out-of-sample accuracy was .9980 to double-check,use confusionMatrix to get out-of-sample accuracy.

```
confusionMatrix(pred_test, testData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1134    4    0    0
##          C    0    1 1022    1    0
##          D    0    0    0  963    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.996, 0.9987)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.997
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9956   0.9961   0.9990   0.9963
## Specificity            0.9991   0.9992   0.9996   0.9992   1.0000
## Pos Pred Value         0.9976   0.9965   0.9980   0.9959   1.0000
## Neg Pred Value         1.0000   0.9989   0.9992   0.9998   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1927   0.1737   0.1636   0.1832
## Detection Prevalence   0.2851   0.1934   0.1740   0.1643   0.1832
## Balanced Accuracy      0.9995   0.9974   0.9978   0.9991   0.9982
```

The out-of-sample accuracy was .998 out-Of-Sample Error = 1 - .9980 = .0020 The percentage of out-of-sample error was .0020 * 100 =0.20% .

Use the random forest model to predict 20 test cases.

```
testing1<- testing[ , -54]
predict_rf <- predict(model_rf, testing1)
predict_rf
```

```
##  [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
```