Practical Machine Learning - Course Project Week 4

Overview

The model building workflow adopted for this task follows the pattern outlined in lectures:

```
> question .. input .. features .. algorithm .. predict .. evaluation
```

Cross Validation has been used as a method for the trainControl function with 4 folds used.

The out of sample error was found to be 0.0037% when the model was applied to the test data derived from the training set.

Choices made at each step are described in the workflow below.

Setup

Due to size of the training sample (19622 observations and up to 60 variables), parallel processing was selected for model development

```
setwd("C:/Users/Husy Razool/Data Scientist/Assignments/PML Week 4")
#install.packages("doParallel")
#install.packages("randomForest")
#install.packages("e1071")
suppressWarnings(suppressMessages(library(caret)))
suppressWarnings(suppressMessages(library(randomForest)))
suppressWarnings(suppressMessages(library(e1071)))
set.seed(1603)
```

QUESTION

Create a model to predict the manner in which the subjects did the exercise using the accelerometer data as predictors.
The outcome to be predicted is the “classe” variable.

INPUT

Download source data

```
trainingFilename   <- 'pml-training.csv'
quizFilename       <- 'pml-testing.csv'
trainingUrl        <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
quizUrl            <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'


# download.file(trainingUrl, trainingFilename)
# download.file(quizUrl,quizFilename)
```

Data Cleansing

On inspection in Excel, found NA,#DIV/0! and blank values in the data. These are not valid observed values, so remove with na.strings parameter.

```
training.df     <-read.csv(trainingFilename, na.strings=c("NA","","#DIV/0!"))
training.df     <-training.df[,colSums(is.na(training.df)) == 0]
dim(training.df) #;head(training.df,3)
```

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

```
quiz.df         <-read.csv(quizFilename , na.strings=c("NA", "", "#DIV/0!"))
quiz.df         <-quiz.df[,colSums(is.na(quiz.df)) == 0]
dim(quiz.df) #;head(quiz.df,3)
```

```
## [1] 20 60
```

FEATURES

Reduce the number of variables

Remove the non-predictors from the training set. This includes the index, subject name, time and window variables.

```
Training.df   <-training.df[,-c(1:7)]
Quiz.df <-quiz.df[,-c(1:7)]
dim(Training.df)
```

```
## [1] 19622    53
```

Check for near zero values in training data

```
Training.nzv<-nzv(Training.df[,-ncol(Training.df)],saveMetrics=TRUE)
```

None found so display and count variables submitted for the train function

```
rownames(Training.nzv)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
```

```
## [52] "magnet_forearm_z"
```

```
dim(Training.nzv)[1]
```

```
## [1] 52
```

ALGORITHM

Partition the training data into a training set and a testing/validation set

```
inTrain     <- createDataPartition(Training.df$classe, p = 0.6, list = FALSE)
inTraining  <- Training.df[inTrain,]
inTest      <- Training.df[-inTrain,]
dim(inTraining);dim(inTest)
```

```
## [1] 11776    53
```

```
## [1] 7846   53
```

Construct the model using cross validation or reload using the cached model

Cross Validation achieved with trainControl method set to “cv”

```
myModelFilename <- "myModel.RData"
if (!file.exists(myModelFilename)) {


    # Parallel cores
```

```
    #require(parallel)
    library(doParallel)
    ncores <- makeCluster(detectCores() - 1)
    registerDoParallel(cores=ncores)
    getDoParWorkers() # 3

    # use Random Forest method with Cross Validation, 4 folds
    myModel <- train(classe ~ .
                , data = inTraining
                , method = "rf"
                , metric = "Accuracy"  # categorical outcome variable so choo
se accuracy
                , preProcess=c("center", "scale") # attempt to improve accura
cy by normalising
                , trControl=trainControl(method = "cv"
                                         , number = 4 # folds of the training
data
                                         , p= 0.60
                                         , allowParallel = TRUE
#                                        , seeds=NA # don't let workers set se
ed
                                         )
                )

    save(myModel, file = "myModel.RData")
    # 3:42 .. 3:49 without preProcess
    # 3:51 .. 3:58 with preProcess
    stopCluster(ncores)
} else {
    # Use cached model
    load(file = myModelFilename, verbose = TRUE)
}
```

```
## Loading objects:
##   myModel
```

```
print(myModel, digits=4)
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (52), scaled (52)
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 8832, 8832, 8832, 8832
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9870    0.9836
##   27    0.9885    0.9855
##   52    0.9820    0.9772
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

PREDICT

Predicting the activity performed using the training file derived test subset

```
predTest <- predict(myModel, newdata=inTest)
```

EVALUATION

Test

Check the accuracy of the model by comparing the predictions to the actual results

```
confusionMatrix(predTest, inTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    8    0    0    0
##          B    1 1507    9    0    2
##          C    0    3 1358    3    0
##          D    0    0    1 1282    1
##          E    0    0    0    1 1439
##
## Overall Statistics
##
##                Accuracy : 0.9963
##                  95% CI : (0.9947, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9953
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9928   0.9927   0.9969   0.9979
## Specificity            0.9986   0.9981   0.9991   0.9997   0.9998
## Pos Pred Value         0.9964   0.9921   0.9956   0.9984   0.9993
## Neg Pred Value         0.9998   0.9983   0.9985   0.9994   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
```

```
## Detection Rate         0.2843   0.1921   0.1731   0.1634   0.1834
## Detection Prevalence   0.2854   0.1936   0.1738   0.1637   0.1835
## Balanced Accuracy      0.9991   0.9954   0.9959   0.9983   0.9989
```

Out of Sample Error

The out-of-sample error of 0.0037 or 0.37%.

Accuracy is very high, at 0.9963, and this figure lies within the 95% confidence interval.

Final Model data and important predictors in the model

```
myModel$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.83%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3341    3    2    0    2 0.002090800
## B   17 2252    8    2    0 0.011847301
## C    0   16 2029    9    0 0.012171373
## D    0    1   23 1903    3 0.013989637
## E    0    1    4    7 2153 0.005542725
```

```
varImp(myModel)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## pitch_forearm          60.20
## yaw_belt               53.95
## magnet_dumbbell_y      45.12
## pitch_belt             44.87
## magnet_dumbbell_z      43.55
## roll_forearm           40.20
## accel_dumbbell_y       22.86
## roll_dumbbell          17.96
## magnet_dumbbell_x      16.65
## accel_forearm_x        16.34
## magnet_belt_z          14.89
## accel_belt_z           13.82
## magnet_forearm_z       13.75
## total_accel_dumbbell   13.64
## accel_dumbbell_z       13.52
## magnet_belt_y          12.64
## yaw_arm                11.49
## gyros_belt_z           10.78
## magnet_belt_x          10.07
```

27 variables were tried at each split and the reported OOB Estimated Error is a low 0.83%.

Overall we have sufficient confidence in the prediction model to predict classe for the 20 quiz/test cases.

Validation/Quiz

The accuracy of the model by predicting with the Validation/Quiz set supplied in the test file.

```
print(predict(myModel, newdata=Quiz.df))
```

```
##  [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
```

Results

Course Project Prediction Quiz
Passed: 20/20 points earned (100%)
Quiz passed!