Preditive model for Weight Lifting Style using Accelerometer dataset

Abstract

This is produced as required by Coursera Practical Machine Learning Assignment.
Human Activity Recognition has emerged as a new key research due to the wide availability of wearable sensons and accelerometers from Jawbone Up, Nike FuelBand and Fitbit. The dataset collected from accelerometers on the belt, forearm, arm adn dumbell of 6 people.

In this paper we try to develop a predictive model to predict the manner in which they did the exercise. This is the "classe" variable in the training set.

Data Loading

The pml-training was downloaded from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv and saved in the project directory The pml-testing was downloaded from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv and saved in the project directory

```
## read the training dataset
raw_dataset = read.csv('pml-training.csv')
## read the validation dataset
pml_testing = read.csv('pml-testing.csv')
```

The pml-training dataset has 19622 rows and 160 variables/columns

Data Preparation for modeling

Partion the raw_dataset into two parts - one for training the model and the other for testing model

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.0.3
```

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

```
## Warning: package 'lattice' was built under R version 3.0.3
```

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

```
## Warning: package 'ggplot2' was built under R version 3.0.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.0.3
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
#summary(raw_dataset)
set.seed(5678)
trainIndex = createDataPartition(raw_dataset$classe, list=FALSE, p=0.7)
train_set = raw_dataset[trainIndex,]
test_set  = raw_dataset[-trainIndex,]
# since most of the stistic columns have no data, let us get rid of them
#goodCols = !sapply(strsplit(names(train_set),"_"), function(x) x[1]) %in% c("kurtosis","skewness","max
#train_set = train_set[,goodCols]
#test_set = test_set[,goodCols]
#summary(train_set)
```

Remove all indicators with near zero variance as they are not useful in modeling. After that remove any columns that are not numeric, but keep the classe varaible as this is the outcome variable

```
nzerovar = nearZeroVar(train_set)
train_set = train_set[-nzerovar]
test_set = test_set[-nzerovar]
pml_testing = pml_testing[-nzerovar]

numIndex = which(lapply(train_set,class) %in% c('numeric'))
train_set1 = train_set[,numIndex]
test_set1 = test_set[,numIndex]
pml_test1 = pml_testing[,numIndex]
```

Now, impute any missing values in the training data set

```
preModel = preProcess(train_set1, method=c('knnImpute'))
ptrain_set = predict(preModel, train_set1)
```

```
## Warning: package 'RANN' was built under R version 3.0.3
```

```
ptest_set  = predict(preModel, test_set1)
pml_test_set = predict(preModel, pml_test1)

# add the classe column
ptrain_set = cbind(train_set$classe, ptrain_set)
ptest_set  = cbind(test_set$classe, ptest_set)
# need to fix the column name
names(ptrain_set)[1] = "classe"
names(ptest_set)[1] = "classe"
```

Model Building

Using the "caret" package, build a random forest model, since computationally intensive will use optimized parameters Then check the accuracy of the model for in-sample data

```
RFmodel = randomForest(classe ~ ., ptrain_set, ntree=500, mtry=32)

# check accuracy of model for in-sample data
train_pred = predict(RFmodel, ptrain_set)
confusionMatrix(train_pred, ptrain_set$classe)
```

```
## Warning: package 'e1071' was built under R version 3.0.3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (1, 1)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.193    0.174    0.164    0.184
## Detection Prevalence    0.284    0.193    0.174    0.164    0.184
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

As we can see that the accuracy for the model is perfect for training data, which could sometimes indicate an overfit based on the training dataset. This can be verified by checking the accuracy for test data

```
test_pred = predict(RFmodel, ptest_set)
confusionMatrix(test_pred, ptest_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672   10    0    1    1
##          B    0 1118    1    1    1
```

```
##          C    1   10 1013   13    5
##          D    0    1   12  946    1
##          E    1    0    0    3 1074
##
## Overall Statistics
##
##                Accuracy : 0.989
##                  95% CI : (0.987, 0.992)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.987
##  Mcnemar's Test P-Value : 0.00323
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.982    0.987    0.981    0.993
## Specificity             0.997    0.999    0.994    0.997    0.999
## Pos Pred Value          0.993    0.997    0.972    0.985    0.996
## Neg Pred Value          1.000    0.996    0.997    0.996    0.998
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.190    0.172    0.161    0.182
## Detection Prevalence    0.286    0.190    0.177    0.163    0.183
## Balanced Accuracy       0.998    0.990    0.991    0.989    0.996
```

Validation of the Model

As we can see from the results that the accuracy of the model is close to 99%, let us use the model to predict for the sample test data set provided as part of the assignment

```
results = predict(RFmodel, pml_test_set)
print("The results:")
```

```
## [1] "The results:"
```

```
results
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

```
#confusionMatrix(results, pml_test_set$classe)
```

Conclusion

The model built using the caret and randomForest packages for the given dataset seem to work very well as it predicted with good accuracy for both test data as well as the validation data.

References:

Weight Lifting Dataset is provided as part of Human Activity Recognition in the paper referred below Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.