Predicting Class from variables

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Code has been developed to predict, based on the collected data what barbell lift was used.

Data

The data comes from the following study: 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

Six participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

The data consists of 160 variables, some of which are from the accelerometers, and 'classe' which tells us which type of lift was used. The training set provided has 19,622 observations. A provided test set of 20 observations do not have the class identified and will be used for the final test.

Can the values from the accelerometers predict the class of lifting?

Load and process training and testing data

```
## Warning: package 'AppliedPredictiveModeling' was built under R version
## 3.2.3
```

```
## Warning: package 'caret' was built under R version 3.2.3
```

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

```
## Warning: package 'ggplot2' was built under R version 3.2.3
```

```
## Warning: package 'randomForest' was built under R version 3.2.3
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

The data has been downloaded from the website.

```
pmltrain <-read.csv("pml-training.csv",na.strings=c("", "NA", "NULL"))
pml20test <-read.csv("pml-testing.csv",na.strings=c("", "NA", "NULL"))
dim(pmltrain)
```

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

```
dim(pml20test)
```

```
## [1]  20 160
```

Let's make sure we are only using variables that are useful. First, get rid of columns that are mostly NAs. Then the irrelevant variables like line number, datestamps and names.

```
# keep columns with no NAs
pmltrain <- pmltrain[ , colSums(is.na(pmltrain)) == 0]
pml20test <- pml20test[ , colSums(is.na(pml20test)) == 0]
#Remove irrelevant variables
irrelevant = c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2',
           'cvtd_timestamp', 'new_window', 'num_window')
pmltrain <- pmltrain[, -which(names(pmltrain) %in% irrelevant)]
pml20test <- pml20test[, -which(names(pml20test) %in% irrelevant)]

dim(pmltrain)
```

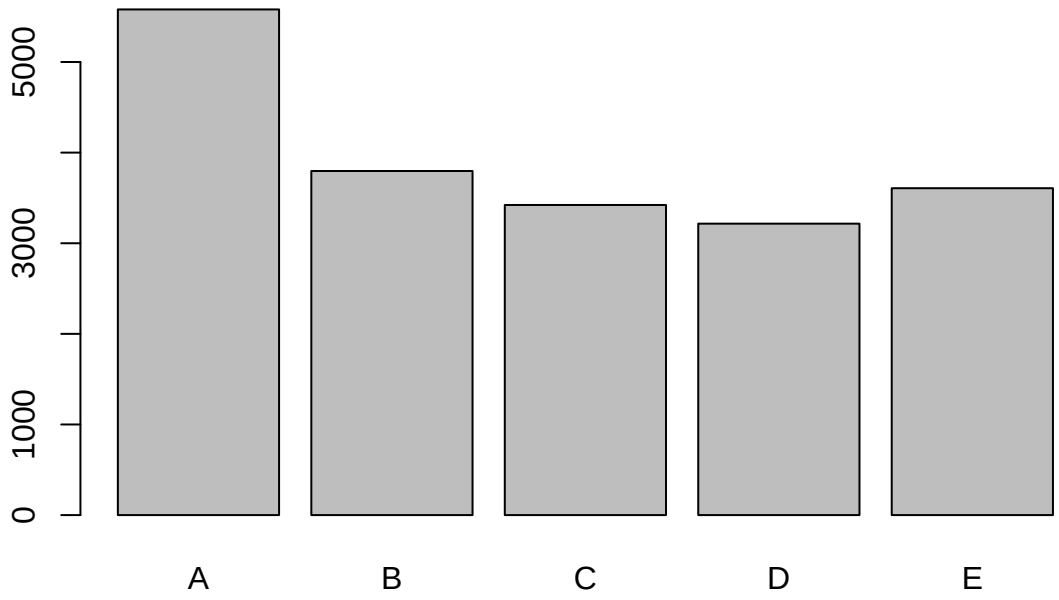
```
## [1] 19622    53
```

```
dim(pml20test)
```

```
## [1] 20 53
```

Take a quick look at the variance in the classe variable.

```
plot(pmltrain$classe)
```



Looks relatively even. Check the other variables for low variance.

```
nsvvalues <- nearZeroVar(pmltrain, saveMetrics=TRUE)
sum(nsvvalues[,"nzv"])
```

```
## [1] 0
```

No Low variance variables.

Dividing up the Data for Cross Validation

To validate the training algorithm, divide the training set into a training set and validation set.

```
set.seed(1010)
inTrain <- createDataPartition(pmltrain$classe, p=.65, list=FALSE)
trainset <- pmltrain[inTrain,]
valset <- pmltrain[-inTrain,]
dim(trainset)
```

```
## [1] 12757    53
```

```
dim(valset)
```

```
## [1] 6865   53
```

Random Forests

I decided to use the Random Forest algorithm since it is popular for it's accuracy. With this using the k-fold cross validation with K=4 to see how it does.

```
set.seed(1010)
RFModel <- train(trainset$classe ~ ., method="rf",
                 trControl=trainControl(method = "cv", number = 4), data=trainset)
```

```
print(RFModel, digits=3)
```

```
## Random Forest
##
## 12757 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 9567, 9570, 9567, 9567
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.990     0.987  0.001032     0.001307
##   27    0.989     0.986  0.000788     0.000996
##   52    0.984     0.979  0.000835     0.001055
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
predictions <- predict(RFModel, newdata=valset)
print(confusionMatrix(predictions, valset$classe), digits=3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1953    6    0    0    0
##          B    0 1317    2    0    0
##          C    0    5 1195   24    6
##          D    0    0    0 1101    6
##          E    0    0    0    0 1250
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.991, 0.995)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.992    0.998    0.979    0.990
## Specificity             0.999    1.000    0.994    0.999    1.000
## Pos Pred Value          0.997    0.998    0.972    0.995    1.000
## Neg Pred Value          1.000    0.998    1.000    0.996    0.998
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.174    0.160    0.182
## Detection Prevalence    0.285    0.192    0.179    0.161    0.182
## Balanced Accuracy       0.999    0.996    0.996    0.989    0.995
```

Results

This model has an accuracy of 99.3%. The Out of Sample error rate is (1-.993) which is .007. Since the error rate is so low for the Validation set, let's do our prediction for the test set.

```
testanswers<-predict(RFModel, newdata=pml20test)
testanswers
```

```
##  [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
```

These answers are 100% correct. It seems the data from the accelerometers can predict with high accuracy the class of weight lifting.