Prediction Assignment Writeup

Roxana Trejos Ramírez

12/18/2020

This project will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

GOAL: Predict the manner in which they did the exercise.

- This is the "classe" variable in the training set.
- Use any of the other variables to predict with.

This report is describing:

- How you built your model
- How you used cross validation
- What you think the expected out of sample error is
- Why you made the choices you did.
- Use prediction model to predict 20 different test cases

LOADING PACKAGES AND LIBRARIES

```
library(ggplot2)
#  A set of functions that attempt to streamline the process for creating predictive models.
library(caret)
#  Automated Feature Selection from 'caret'
library(fscaret)
#  Implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran cod
#  regression. It can also be used in unsupervised mode for assessing proximities among data points
library(randomForest)
#  Functions for latent class analysis, short time Fourier transform, fuzzy clustering, support vec
#  computation, bagged clustering, naive Bayes classifier, etc.
library(e1071)
```

READING DATA

```
# Loading training data from the csv file.
url_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(url_train, destfile = "pml-training.csv")
training <- read.table("pml-training.csv", sep = ",", header = TRUE)
```

```r
# Loading testing data from the csv file.
url_test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url_test, destfile = "pml-testing.csv")
testing <- read.table("pml-testing.csv", sep = ",", header = TRUE)
```

SPLITTING THE DATA AND SELECTING FEATURES

Create two partitions (80% and 20%) within the original training dataset.

```r
#  Seed to make the analysis reproducible
set.seed(1000)

#  Partitioning the training set into training
#  Data splitting functions
#  A series of training partitions are created
#  y = training$clase
inTrain <- createDataPartition(y=training$classe,
                               p=0.8,
                               list=F)

# inTraining = A list or matrix of row position integers corresponding to the training data.


# Splitting the original training set into the training set and a validation set.
training1 <- training[inTrain, ]
training2 <- training[-inTrain, ]
```

CLEANING TRAINING 1 AND 2

1. The two datasets (train_set and test_set) have a large number of NA values as well as near-zero-variance (NZV) variables. Both will be removed together with their ID variables.
2. Remove variables that are mostly NA.
3. Since columns 1 to 5 are identification variables only, they will be removed as well.

```r
# Removing Near Zero Variance Predictors
nzv <- nearZeroVar(training)
training1 <- training1[, -nzv]
training2 <- training2[, -nzv]

# Removing Predictors with NA values
training1 <- training1[, colSums(is.na(training1)) == 0]
training2 <- training2[, colSums(is.na(training2)) == 0]

# Removing columns unfit for prediction the cols:  ID, user_name, raw_timestamp_part_1 etc ...
training1 <- training1[, -(1:5)]
training2 <- training2[, -(1:5)]
```

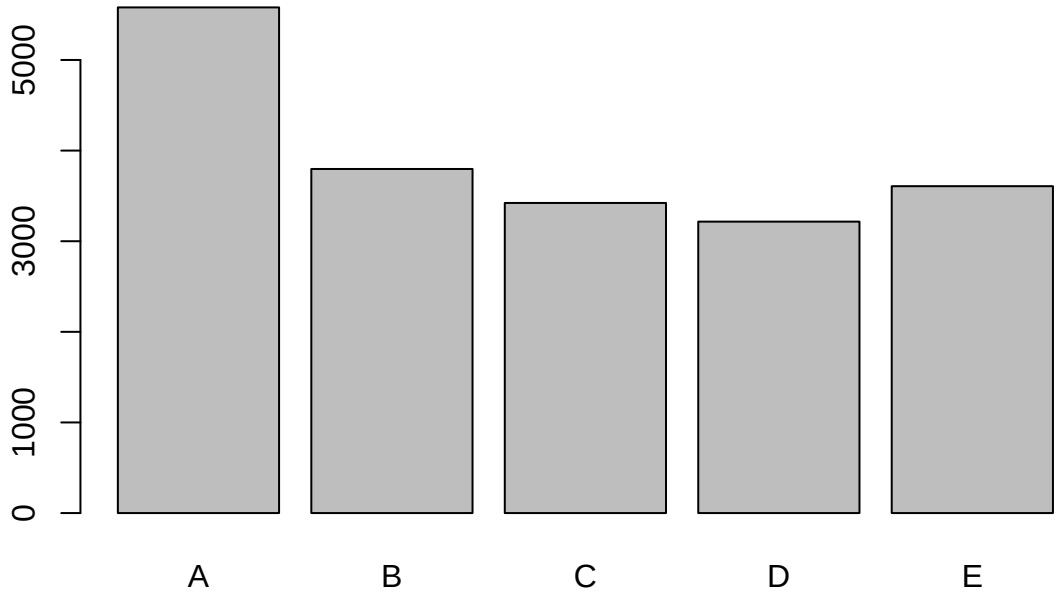
Quick exploration of the data (testing and training) to see if there is something in the data that might affect the model decision. Checking which column names are common among testing and training, so we can exclude the ones who are not common. Checking the classe balance in the training set to see whether

there is anything in particular we should be concerned with. Plotting the classe variable against the first 5 (example exploratory plot).

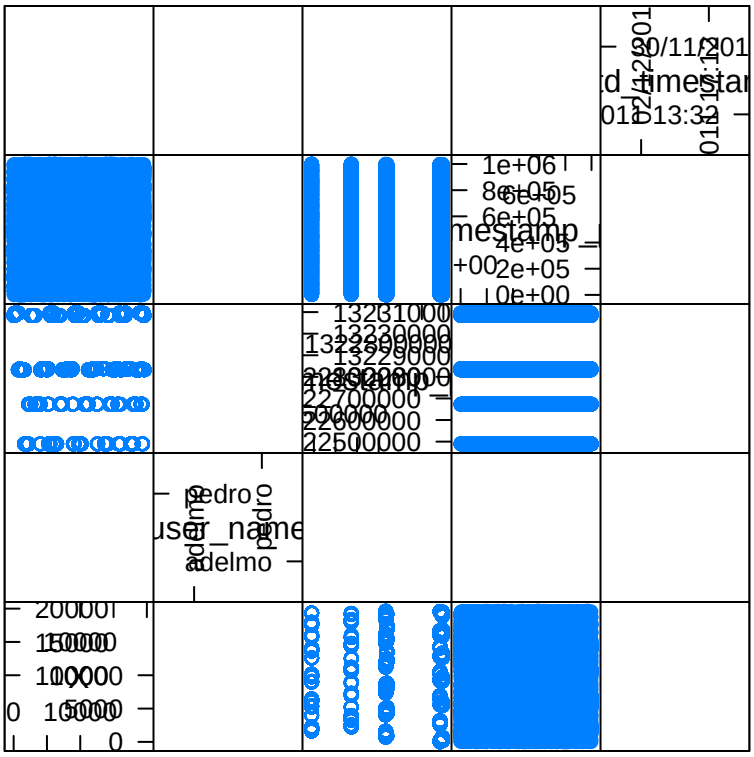
```
    length(intersect(colnames(training),colnames(testing)))
```

```
## [1] 159
```

```
    barplot(table(training$classe))
```




```
    splom(classe~training[1:5], data = training)
```



Scatter Plot Matrix

Conclusion:

- 159 variables in common, everyone except classe
- Target variable is balanced across the different classes

MODELING DATA WITH RANDOM FOREST

Fitting Predictive Models Over Different Tuning Parameters This function sets up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure. rf = Random Forest Model. This is one of the best model since it provides the most accurate results. The cross-validation is set to draw a subset of the data three different times.

```
modelo <- train(classe ~.,
                method = "rf",
                data = training1,
                verbose = TRUE,
                trControl = trainControl(method="cv"),
                number = 3)
```

Predictions of the random forest model on TRAINING 1

```
pred1 <- predict(modelo, training1)
confusionMatrix(pred1, factor(training1$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4464    0    0    0    0
##          B    0 3038    0    0    0
##          C    0    0 2738    0    0
##          D    0    0    0 2573    0
##          E    0    0    0    0 2886
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Predictions of the random forest model on TRAINING 2

```
pred2 <- predict(modelo, training2)
confusionMatrix(pred2, factor(training2$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    2    0    0    0
##          B    0  755    1    0    0
##          C    0    2  682    3    0
##          D    0    0    1  640    0
##          E    0    0    0    0  721
##
## Overall Statistics
##
##                Accuracy : 0.9977
##                  95% CI : (0.9956, 0.999)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9971
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9971   0.9953   1.0000
## Specificity            0.9993   0.9997   0.9985   0.9997   1.0000
## Pos Pred Value         0.9982   0.9987   0.9927   0.9984   1.0000
## Neg Pred Value         1.0000   0.9987   0.9994   0.9991   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1925   0.1738   0.1631   0.1838
## Detection Prevalence   0.2850   0.1927   0.1751   0.1634   0.1838
## Balanced Accuracy      0.9996   0.9972   0.9978   0.9975   1.0000
```

CLEANING TESTING DATA

```
    testing <- testing[, colSums(is.na(testing)) == 0]
    testing <- testing[, -(1:5)]
    nzvt <- nearZeroVar(testing)
    testing <- testing[, -nzvt]
```

TESTING RANDOM FOREST MODEL WITH TESTING DATA (20 different test cases)

```
    pred3 <- predict(modelo, testing)
    pred3
```

```
##  [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
```