Machine_Learning_Assignment

Anita Datta Chowdhury

April 19, 2017

Machine Learning Assessment - Weight Lifting Exercise Manner Prediction Summary

This analysis was done to predict the manner in which the subjects performed weight lifting exercises. The data is collected from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The outcome variable has five classes and the total number of predictors are 159.

Set Environment

```
library(knitr)
opts_chunk$set(echo = TRUE, results = 'hold')
library(ElemStatLearn)
library(caret)
```

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

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

```
library(rpart)
library(randomForest)
```

```
## 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
```

```
set.seed(12345)
```

Loading the data

```
setwd("C:/Users/anita/OneDrive/Documents/DataSceince_Projects/Course8_Machine_Learning")
url1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
filename1 <- "pml-training.csv"
filename2 <- "pml-testing.csv"
download.file(url=url1, destfile=filename1)
download.file(url=url2, destfile=filename2)
training <- read.csv("pml-training.csv",row.names=1,na.strings=c("", "NA", "NULL"))
testing <- read.csv("pml-testing.csv",row.names=1,na.strings=c("", "NA", "NULL"))
```

Preprocessing

First we need to get rid of the variables that have close to zero variance in both training and testing data. Then remove the columns with missing values to avoid issues in training models. If the result is not good, we can add back those columns with missing values imputed.

Remove near zero covariates

```
nsv <- nearZeroVar(training,saveMetrics=TRUE)
training <- training[,!nsv$nzv]
testing <- testing[,!nsv$nzv]
```

Remove variables with missing values

```
training_filter_na <- training[,(colSums(is.na(training)) == 0)]
testing_filter_na <- testing[,(colSums(is.na(testing)) == 0)]
```

Remove unnecessary columns

```
colRm_train <- c("user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","num_window")
colRm_test <- c("user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","num_window","proble
m_id")
training_colRm <- training_filter_na[,!(names(training_filter_na) %in% colRm_train)]
testing_colRm <- testing_filter_na[,!(names(testing_filter_na) %in% colRm_test)]
dim(training_colRm)
dim(testing_colRm)
```

```
## [1] 19622    53
## [1] 20 52
```

Now we split the preprocessed training data into training set and validation set.

```
inTrain <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
training_clean <- training_colRm[inTrain,]
validation_clean <- training_colRm[-inTrain,]
```

In the new training set and validation set we just created, there are 52 predictors and 1 response. Check the correlations between the predictors and the outcome variable in the new training set. There doesn't seem to be any predictors strongly correlated with the outcome variable, so linear regression model may not be a good option. Random forest model may be more robust for this data.

```
cor <- abs(sapply(colnames(training_clean[, -ncol(training)]), function(x) cor(as.numeric(training_clean[, x]),
 as.numeric(training_clean$classe), method = "spearman")))
```

Random Forest Model

We try to fit a random forest model and check the model performance on the validation set.

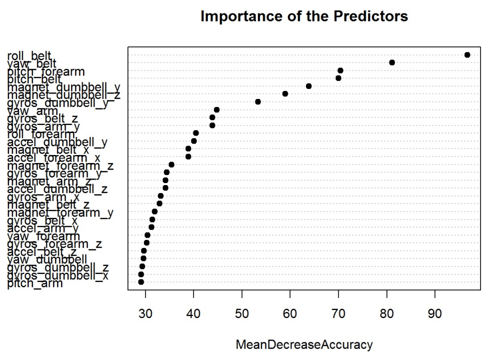
Fit rf model

```
rfFit <- train(classe ~ ., method = "rf", data = training_clean, importance = T,
               trControl =trainControl(method = "cv", number = 4))
validation_pred <- predict(rfFit, newdata=validation_clean)
confusionMatrix(validation_pred,validation_clean$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    9    0    1    0
##          B    1 1127    8    0    0
##          C    0    3 1013   13    1
##          D    0    0    5  949    4
##          E    0    0    0    1 1077
##
## Overall Statistics
##
##                Accuracy : 0.9922
##                  95% CI : (0.9896, 0.9943)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9901
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9895   0.9873   0.9844   0.9954
## Specificity            0.9976   0.9981   0.9965   0.9982   0.9998
## Pos Pred Value         0.9941   0.9921   0.9835   0.9906   0.9991
## Neg Pred Value         0.9998   0.9975   0.9973   0.9970   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1915   0.1721   0.1613   0.1830
## Detection Prevalence   0.2860   0.1930   0.1750   0.1628   0.1832
## Balanced Accuracy      0.9985   0.9938   0.9919   0.9913   0.9976
```

Check important variable

```
imp <- varImp(rfFit)$importance
varImpPlot(rfFit$finalModel, sort = TRUE, type = 1, pch = 19, col = 1, cex = 1, main = "Importance of the Predi
ctors")
```



The random forest algorithm generates a model with accuracy 0.9913. The out-of-sample error is 0.9%, which is pretty low. We don't need to go back and include more variables with imputations. The top 4 most important variables according to the model fit are 'roll_belt', 'yaw_belt', 'pitch_forearm' and 'pitch_belt'.

Prediction

The last step is to use the random forest model to predict on the testing set without the outcome variable and save the prediction output.

```
testing_pred <- predict(rfFit, newdata=testing_colRm)
write_files <- function(x) {
        n <- length(x)
        for (i in 1:n) {
                filename <- paste0("problem_id", i, ".txt")
                write.table(x[i], file=filename, quote=FALSE, row.names=FALSE,col.names=FALSE)
        }
}
write_files(testing_pred)
```

Results

We used 52 variables to build the random forest model with 4-fold cross validation. The out-of-sample error is approximately 0.9%.