Practical Machine Learning

Hriday Patgiri

27/05/2020

Summary

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. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Loading Necessary Packages

```r
library(caret)
library(randomForest)
```

Reading the Data

```r
training <- read.csv("pml-training.csv",row.names=1,na.strings = "")
testing <- read.csv("pml-testing.csv",row.names=1,na.strings = "NA")
```

Processing Data

1. Remove near zero covariates

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

2. Remove variables with missing values

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

3. Remove unnecessary columns

```r
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","problem_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)

## [1] 19622    53

dim(testing_colRm)

## [1] 20 52
```

4. Split the processed 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.

1. Fit rf model

```
set.seed(1234)
rfFit <- train(classe ~ ., method = "rf", data = training_clean, importance =
T, trControl = trainControl(method = "cv", number = 4))
validation_pred <- predict(rfFit, newdata=validation_clean)
```

2. Check model performance

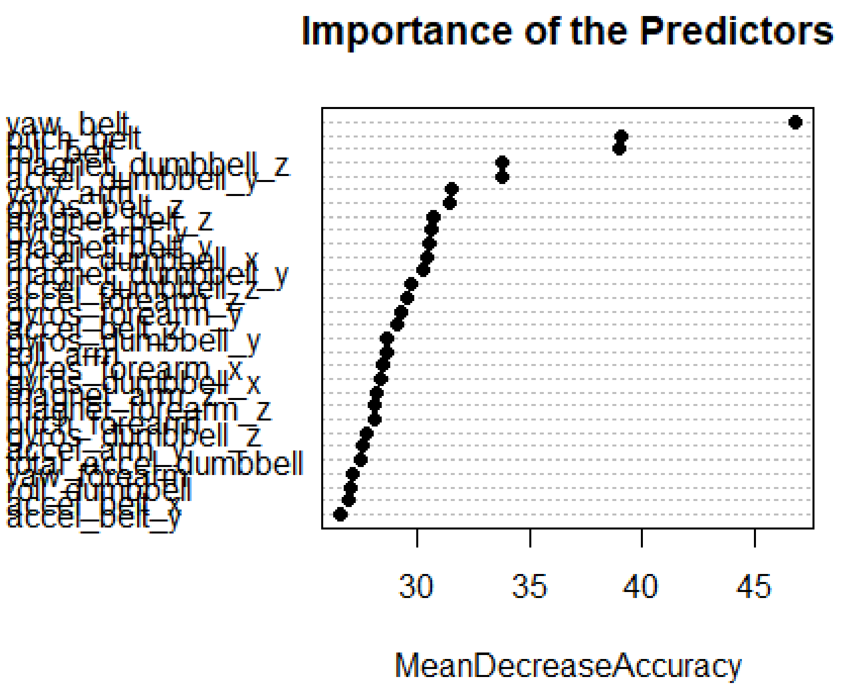
```
confusionMatrix(validation_pred,validation_clean$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    7    0    0    0
##          B    0 1132    3    0    0
##          C    0    0 1023   26    0
##          D    0    0    0  936    1
##          E    0    0    0    2 1081
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.991, 0.9953)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9916
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9939   0.9971   0.9710   0.9991
## Specificity            0.9983   0.9994   0.9946   0.9998   0.9996
## Pos Pred Value         0.9958   0.9974   0.9752   0.9989   0.9982
## Neg Pred Value         1.0000   0.9985   0.9994   0.9943   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1924   0.1738   0.1590   0.1837
## Detection Prevalence   0.2856   0.1929   0.1782   0.1592   0.1840
## Balanced Accuracy      0.9992   0.9966   0.9959   0.9854   0.9993
```

3. Checking important variable

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



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)
        }
}
testing_pred

##  [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

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