Practical ML Project

Tariq Naeem

Aug 05, 2023

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. 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, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise.

Libraries required

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.4
```

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

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

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.4
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Data Variables

```
training.url <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
test.cases.url <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'
```

Data Preprocessing

In this section:

1. Data is downloaded and processed.
2. Remove 'NA' values

```
downloadcsv <- function(url, nastrings) {
    temp <- tempfile()
    download.file(url, temp, method = "curl")
    data <- read.csv(temp, na.strings = nastrings)
    unlink(temp)
    return(data)
}

train <- downloadcsv(training.url, c("", "NA", "#DIV/0!"))

test <- downloadcsv(test.cases.url, c("", "NA", "#DIV/0!"))


# The training data has 19622 observations and 160 features, and the distribution of the five measured stances A,B,C,D,E is:

dim(train)
```

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

```
table(train$classe)
```

```
## 
##    A    B    C    D    E 
## 5580 3797 3422 3216 3607
```

```
set.seed(123456)
trainset <- createDataPartition(train$classe, p = 0.8, list = FALSE)
Training <- train[trainset, ]
Validation <- train[-trainset, ]
```

Feature Selection

```
# Check for near zero variance predictors and drop them if necessary
nonzerocol <- nearZeroVar(Training)
Training <- Training[, -nonzerocol]

# exclude columns with 40%  more missing values exclude descriptive columns

countlength <- sapply(Training, function(x) {
    sum(!(is.na(x) | x == ""))
})

nullCol <- names(countlength[countlength < 0.6 * length(Training$classe)])

descriptcol <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
    "cvtd_timestamp", "new_window", "num_window")

excludecolumns <- c(descriptcol, nullCol)

Training <- Training[, !names(Training) %in% excludecolumns]
```

Model Train & Model Validation

```r
rfModel <- randomForest(as.factor(classe)~ ., data = Training, importance = TRUE, ntrees = 10)

## Model Validation

ptraining <- predict(rfModel, Training)

# Using 'union' to ensure same level
u1 <- union(ptraining,Training$classe)
t1 <- table(factor(ptraining, u1), factor(Training$classe, u1))
print(confusionMatrix(t1))
```

```
## Confusion Matrix and Statistics
## 
## 
##        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
```

Our model performs good against training set, But we will cross validate against the held out set and check if we have avoided overfitting.

Validation (Out-of-Sample)

```r
pvalidation <- predict(rfModel, Validation)

# Using 'union' to ensure same level
u2 <- union(pvalidation,Validation$classe)
t2 <- table(factor(pvalidation, u2), factor(Validation$classe, u2))
print(confusionMatrix(t2))
```

```
## Confusion Matrix and Statistics
##
##
##       A    B    C    D    E
##   A 1116    1    0    0    0
##   B    0  758    0    0    0
##   C    0    0  684    4    0
##   D    0    0    0  638    3
##   E    0    0    0    1  718
##
## 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   0.9987   1.0000   0.9922   0.9958
## Specificity            0.9996   1.0000   0.9988   0.9991   0.9997
## Pos Pred Value         0.9991   1.0000   0.9942   0.9953   0.9986
## Neg Pred Value         1.0000   0.9997   1.0000   0.9985   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1744   0.1626   0.1830
## Detection Prevalence   0.2847   0.1932   0.1754   0.1634   0.1833
## Balanced Accuracy      0.9998   0.9993   0.9994   0.9957   0.9978
```

Cross validation accurracy is 99.7% & out-of-sample error is 0.3%. So our model performs good.

Test set prediction

Prediction of our algorithm for the test set is:

```
ptest <- predict(rfModel, test)
ptest
```

```
##  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
```