Practical Machine Learning Final Project

Katherine Williams

6/28/2021

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, 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. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Downloading Data

The data is downloaded in two datasets a training and testing set

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

trainurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
train <- downloadcsv(trainurl, c("", "NA", "#DIV/0!"))

testurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
test <- downloadcsv(testurl, c("", "NA", "#DIV/0!"))
```

Data Preprocessing

Training set partitioning

The training set are divided into a training and a validation set

```
library(caret)
```

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

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

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

Feature Selection

Variables that might cause problems in training our model, such as those with near zero variance, those with a large number of missing values, or descriptive feilds, will be excluded from the analysis.

```
# exclude features with near zero variance
nzvcol <- nearZeroVar(Training)
Training <- Training[, -nzvcol]

# exclude columns where 40% or more are missing values
# exclude descriptive columns like name etc
cntlength <- sapply(Training, function(x) {
    sum(!(is.na(x) | x == ""))
})
nullcol <- names(cntlength[cntlength < 0.6 * length(Training$classe)])
descriptcol <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
    "cvtd_timestamp", "new_window", "num_window")
excludecols <- c(descriptcol, nullcol)
Training <- Training[, !names(Training) %in% excludecols]
```

Model Developement

With our new training set we can work towards developing a model that predicts the classe of each observation in the data set.

Random Forest

We will be building the model using a Random Forest approch. Random Forest selects the most important variables for us, while still being robust to correlated covariates and outliers.

```
library(randomForest)
```

```
## 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
```

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

Model Validation

Testing the model on both the training set and the cross validation set.

Training set accuracy

Testing the model against the set used to build it should result in very high accuracy.

```
Training$classe<- factor(Training$classe)
library(e1071)

ptraining <- predict(rfModel, Training)
print(confusionMatrix(ptraining, Training$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
```

Training Set Results

As expected the model does an excellent job predicting the classes of the training set. But this is not necessarily proof that it will do well on the test set.

Validation Set Accuracy

The Validation set a portion of the original training set that is not included training the model. After the model is built it is used to determine the out-of-sample accuracy of the model. This can help to determine if the model has been over fit to the training set, or if it is more widely applicable to data of the same type.

```
Validation$classe<- factor(Validation$classe)

pvalidation <- predict(rfModel, Validation)
print(confusionMatrix(pvalidation, Validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    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 Results

For the Validation Set the model has a 99.5% Accuracy rate or an out-of-sample error of 0.5%. Therefore we can say that the model works very well on not just the training set but also on data not used to train the model.

Test set Prediction

Now that we have built and validated our model, we can run it on the testing set to determine our final results.

```
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
```

```
answers <- as.vector(ptest)
```