Course Project

Oty Rosario

Project background and instructions

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).

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Loading the caret library

```
library(caret)
```

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

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

Loading the pml_training dataset

```
pml_training <- read.csv("pml-training.csv",
                         na.strings = c("NA","#DIV/0!",""))
```

Splitting the pml_training dataset into a training and testing dataset

```
set.seed(20210107)
inTrain <- createDataPartition(y = pml_training$classe,
                               p = 0.8,
                               list = FALSE)
training <- pml_training[inTrain,]
testing <- pml_training[-inTrain,]
```

Removing first 7 variables

```
training_1 <- training[,-(1:7)]
testing_1 <- testing[,-(1:7)]
```

Removing variables with more than 20% of NA or empty values

```
na_empty_count = sapply(training_1, function(x) {sum(is.na(x) | x == '')})
vars_to_remove = names(na_empty_count[na_empty_count / dim(training_1)[1] > 0.20])
training_2 = training_1[, !names(training_1) %in% vars_to_remove]
testing_2 <- testing_1[, !names(training_1) %in% vars_to_remove]
```

Removing variables with high correlation

If two variables have a high correlation, it has been removed the variable with the largest mean absolute correlation. It has been defined a cutoff of 0.8, meaning that all correlations above this number were considered as high correlation.

```
corr_matrix <- cor(training_2[-(dim(training_2)[2])])
vars_to_remove2 <- findCorrelation(corr_matrix,
                                   cutoff = .8,
                                   exact = TRUE,
                                   names = TRUE)
training_3 = training_2[, !names(training_2) %in% vars_to_remove2]
testing_3 = testing_2[, !names(training_2) %in% vars_to_remove2]
```

Preprocessing training and testing dataset with range method

The datasets were normalized, or in other words they were scaled into the range of [0, 1].

```
train_stats <- preProcess(training_3,
                          method = "range")
training_4 <- predict(train_stats, training_3)
testing_4 <- predict(train_stats, testing_3)
```

Training a K-Nearest Neighbours (KNN) model

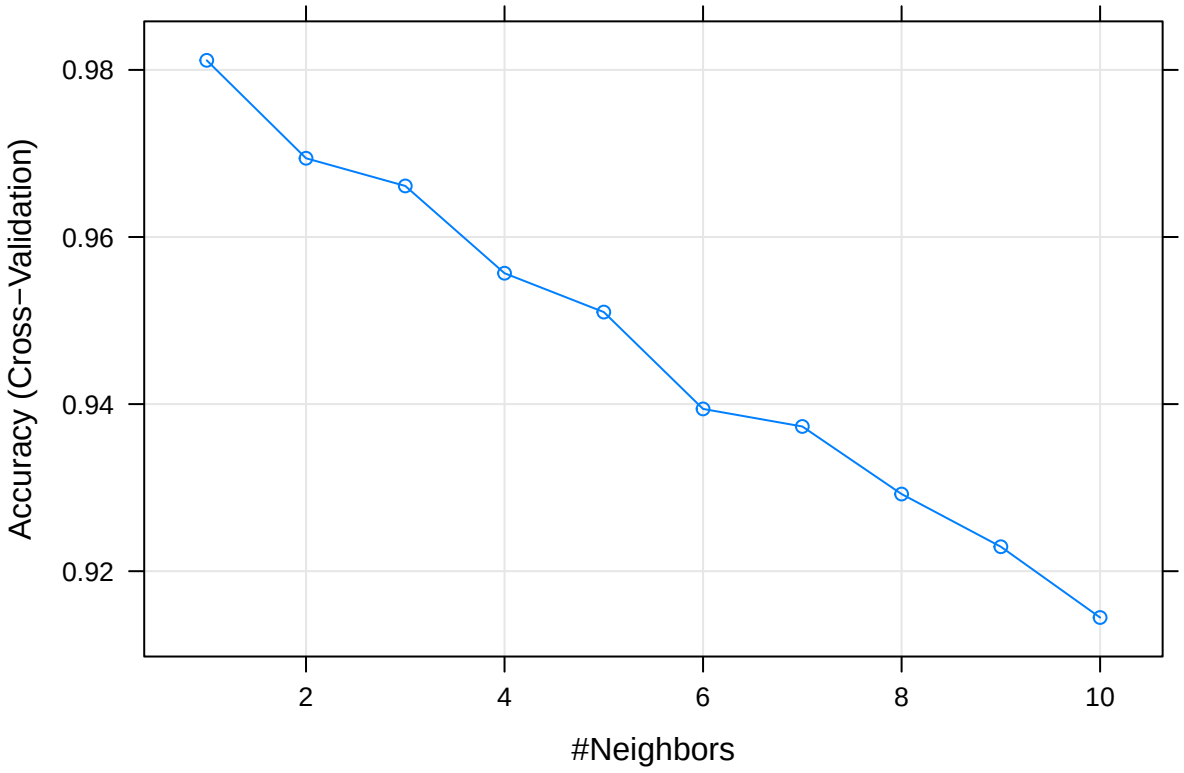
To choose the K for the KNN model we have used k-fold cross-validation, splitting the data into 5 equal parts, and we have chosen the K with the highest accuracy.

```
trControl <- trainControl(method  = "cv",
                          number  = 5)

knn_model <- train(classe ~ .,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:10),
             trControl  = trControl,
```

```
             metric     = "Accuracy",
             data       = training_4)

knn_model
```

```
## k-Nearest Neighbors
##
## 15699 samples
##    39 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 12560, 12558, 12560, 12558, 12560
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9811458  0.9761534
##    2  0.9694254  0.9613279
##    3  0.9661134  0.9571295
##    4  0.9556663  0.9439153
##    5  0.9510163  0.9380311
##    6  0.9394233  0.9233598
##    7  0.9373218  0.9206857
##    8  0.9292324  0.9104507
##    9  0.9229263  0.9024585
##   10  0.9144544  0.8917395
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```
plot(knn_model)
```



Cross-validating the model

```
prediction2 <- predict(knn_model, testing_4)
table(prediction2, testing_4$classe)
```

```
##
## prediction2    A    B    C    D    E
##           A 1113    1    0    0    0
##           B    2  746    5    2    5
##           C    0   10  670    9    1
##           D    1    2    7  630    3
##           E    0    0    2    2  712
```

```
knn_mod_acc <- mean(prediction2 == testing_4$classe)
```

The accuracy of this model is 0.9867448 and the expected sample error is 0.0132552.

Predicting 20 different test cases

```
pml_testing <- read.csv("pml-testing.csv",
                        na.strings = c("NA","#DIV/0!",""))
```

```
pml_testing <- pml_testing[,-(1:7)]
pml_testing <- pml_testing[, !names(pml_testing) %in% vars_to_remove]
pml_testing <- pml_testing[, !names(pml_testing) %in% vars_to_remove2]
pml_testing <- predict(train_stats, pml_testing)
prediction <- predict(knn_model,pml_testing)
pml_testing_pred <- as.data.frame(pml_testing$problem_id)
pml_testing_pred$classe_pred <- prediction
pml_testing_pred
```

```
##    pml_testing$problem_id classe_pred
## 1                       1           B
## 2                       2           A
## 3                       3           B
## 4                       4           A
## 5                       5           A
## 6                       6           E
## 7                       7           D
## 8                       8           B
## 9                       9           A
## 10                     10           A
## 11                     11           B
## 12                     12           C
## 13                     13           B
## 14                     14           A
## 15                     15           E
## 16                     16           E
## 17                     17           A
## 18                     18           B
## 19                     19           B
## 20                     20           B
```