Analysis on Exercise Data

SNTag

20 June, 2020

1 Introduction

This project will attempt to identify exercise regimes followed from biological data. This will give us a guide to what is necessary for proper development.

Goal: to identify the exercise regime identified in the classe variable.

```
library(tidyverse)
library(magrittr)
library(caret)

df_train <- read_csv("./data/pml-training.csv")
df_test  <- read_csv("./data/pml-testing.csv")

df_train$classe <- df_train$classe %>% as.factor

set.seed("1701")
```

2 Exploratory data analysis

The training data for this project has 19622 entries, belonging to 6. We are interested in using this data to predict how 20 entries should look.

In a quick test to check distribution of NAs, 2/3rds of the columns are nearly entirely NA, with only some data. None are entirely NAs though.

```
missing_values <- df_train %>% summarize_all(funs(sum(is.na(.))/dim(df_train)[1]))

missing_values <- gather(missing_values, key="feature", value="missing_pct")
missing_values %>%
    ggplot(aes(x=reorder(feature,-missing_pct),y=missing_pct)) +
    geom_bar(stat="identity",fill="red")+
    coord_flip()+theme_bw()
```

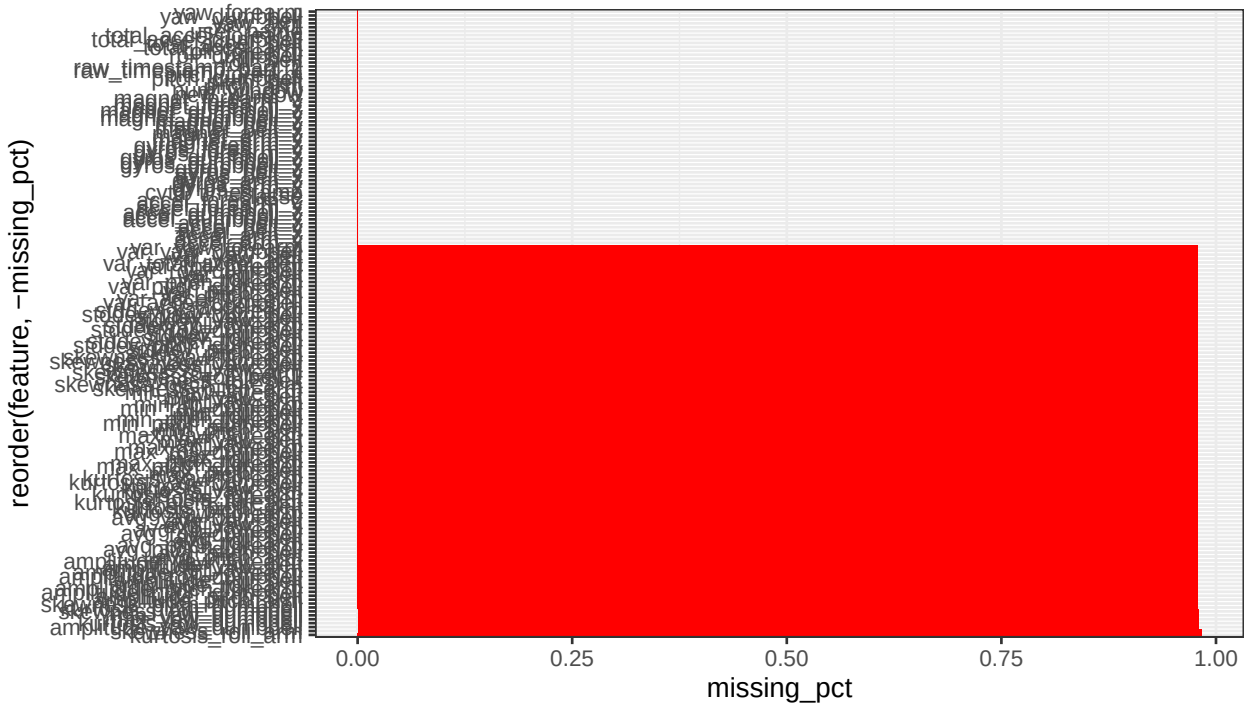

Figure 1: Percentage of rows per column that are NA.

Further examination of the data shows that the same rows have NAs (Fig. 1). If one row has NAs, others will too. I am unsure how to make the best use of this information so I will remove these NAs.

```
missing_values_logic <- sapply(df_test, function(x) any(is.na(x)))
df_train_cc          <- df_train[,!missing_values_logic]
df_train_cc$classe   <- df_train$classe %>% as.factor
```

And finally, to test my models capabilities, I will split the training data set 80-20. The model will be trained on the 80%, and tested on the 20%, before being applied to data in question (df_test).

```
trainIndex <- caret::createDataPartition(df_train_cc$classe, p = .8,
                                  list = FALSE,
                                  times = 1)
df_train_cc80 <- df_train_cc[trainIndex,]
df_train_cc20 <- df_train_cc[-trainIndex,]
```

3 Classification

We are interested in a model capable of predicting the classe. For this, random forest is a robost algorithim. This shall be used with the kappa metric.

```
library(doParallel)

if (exists("rf_fit")) {
    rf_fit <- readRDS("fit.rds")
} else {
cl <- makePSOCKcluster(5)
    registerDoParallel(cl)
```

```
    fitControl <- caret::trainControl(method = "cv",
                                      number = 10)
    rf_fit <- caret::train(classe~.,
                           data=df_train_cc80,
                           method="rf",
                           metric="Kappa",
                           trControl=fitControl)

    stopCluster(cl)
    saveRDS(rf_fit, file = "fit.rds")
}
```

looking at the resulting model, we get:

```
rf_fit
```

```
## Random Forest
##
## 15699 samples
##    59 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 14129, 14129, 14129, 14128, 14129, 14130, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9966876  0.9958102
##   41    0.9999363  0.9999194
##   81    0.9998726  0.9998389
##
## Kappa was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 41.
```

```
results <- c()
results$predicted <- predict(rf_fit,newdata = df_train_cc20)
results$classe <- df_train_cc20$classe
results <- as.data.frame(results)
print("number of incorrectly predicted rows:")
```

```
## [1] "number of incorrectly predicted rows:"
```

```
length(which(results$predicted != results$classe))
```

```
## [1] 1
```

```
confusionMatrix(df_train_cc20$classe, predict(rf_fit, newdata = df_train_cc20))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    1  758    0    0    0
##          C    0    0  684    0    0
##          D    0    0    0  643    0
```

```
##          E    0    0    0    0  721
##
## Overall Statistics
##
##                Accuracy : 0.9997
##                  95% CI : (0.9986, 1)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9997
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   0.9997   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   0.9987   1.0000   1.0000   1.0000
## Neg Pred Value         0.9996   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2847   0.1932   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1744   0.1639   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9996   0.9998   1.0000   1.0000   1.0000
```

The random forest model has a relatively high accuracy. This approach will proceed. It also seems to have high sensitivity and specificity when applied to the split data.

```
finalResults <- predict(rf_fit, newdata = df_test)
print(finalResults)
```

```
##  [1] A A A A A A A A A A A A A A A A A A A A
## Levels: A B C D E
```

The results suggest it is all A. This may seem odd, but as the significance from the earlier steps is high, it will accepted.