Sport Wearable Devices and Activity Performance

Riccardo Finotello

24 June 2020

Summary

In this analysis we will consider data of sport wearable devices and we will tackle the task of predicting the performance of dumbell exercises as perceived from the device. In other words we will try to predict the assigned class of the exercise from motion sensor data.

Cleaning the Data

We will first access the training data:

```
data <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                 na.strings = c("#DIV/0!", "", "NA"),
                 stringsAsFactors = FALSE
                )
```

We then take a look at the number of entries and features in the dataset:

```
print(paste("No. of entries:", nrow(data)))
```

```
## [1] "No. of entries: 19622"
```

```
print(paste("No. of columns<:", ncol(data)))
```

```
## [1] "No. of columns<: 160"
```

From the dataset we remove information related to the specific user and keep only numerical features and non empty columns:

```
data <- data[,-(1:7)]
data$classe <- as.factor(data$classe) # convert classe to factor
```

We then compute the fraction of **NA** data in each column and remove those variables with more than 50% of missing data. This shows that we will keep only a fraction of the original variables:

```
# find fraction of NA values
data.na.fraction <- sapply(data, function(x) {mean(is.na(x))})
data.cols <- (data.na.fraction <= 0.5)
print(paste("Fraction of retained variables:", round(mean(data.cols),2)))
```

```
## [1] "Fraction of retained variables: 0.35"
```

```
# select the relevant columns
train.data <- data[, data.cols]
```

We therefore have a tidy dataset containing 100% of complete cases and new dimensions:

```
print(paste("No. of entries:", nrow(train.data)))
```

```
## [1] "No. of entries: 19622"
```

```
print(paste("No. of columns:", ncol(train.data)))
```

```
## [1] "No. of columns: 53"
```

We finally divide the features we use for training from the labels we try to predict:

```
# shuffle the dataset
train.data <- train.data[sample(nrow(train.data)),]

# store the id of the classes (last column)
labels <- c(ncol(train.data))
```

As an additional step before the exploratory data analysis, we divide the training set into a further partition for testing:

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## Warning: package 'lattice' was built under R version 3.6.3
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
train.part <- createDataPartition(train.data[,ncol(train.data)], p=0.8, list=FALSE)
train <- train.data[train.part,]
test <- train.data[-train.part,]
```

Exploratory Data Analysis

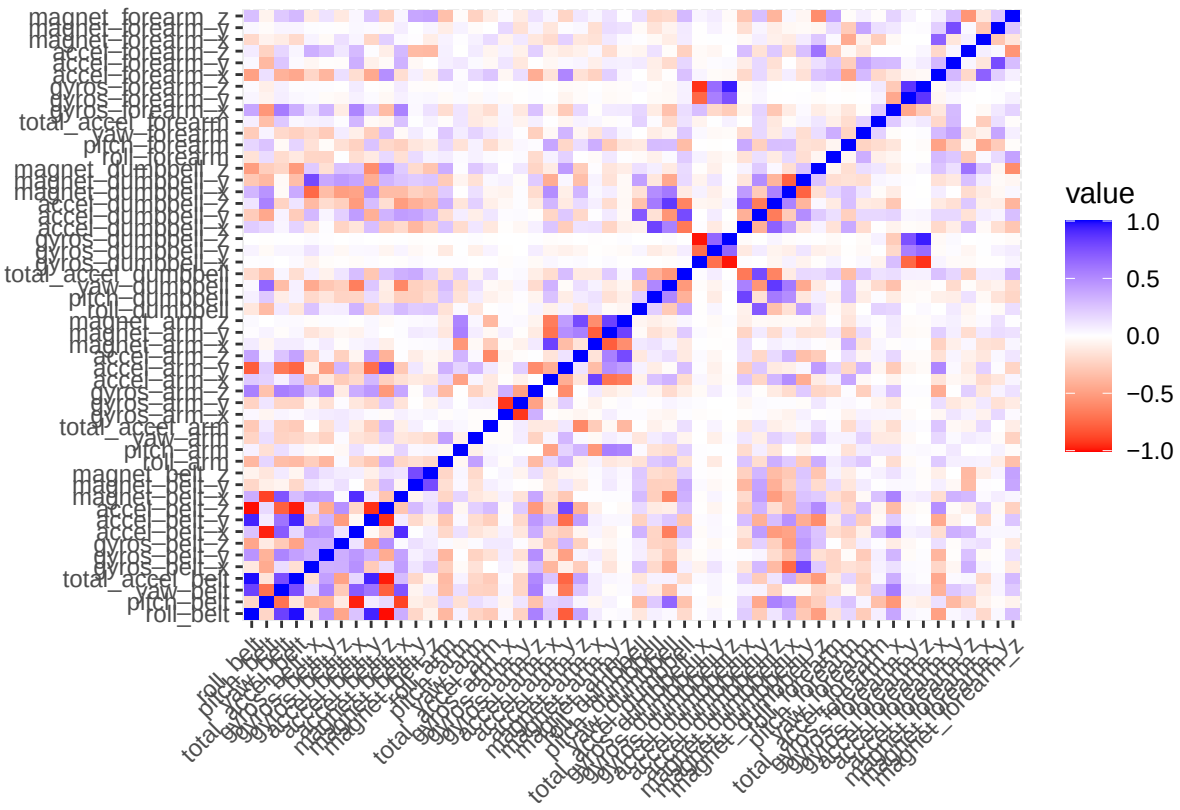
To better understand the distribution of the variables, we study their correlations properties (**we consider only the training partition and we do not look at the labels to avoid biasing the strategy**):

```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 3.6.3
```

```
corr.mat <- cor(train[,-labels])
corr.mat.melt <- melt(corr.mat)

# plot the correlation matrix
library(ggplot2)
g <- ggplot(data = corr.mat.melt, aes(Var1, Var2, fill = value)) +
     geom_tile() +
     xlab("") +
     ylab("") +
     scale_fill_gradient2(low = "red", mid= "white", high = "blue",
                          midpoint = 0, limit = c(-1,1)
                         ) +
    theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
print(g)
```



As we can see the variables are in general not correlated, apart from (mainly) the accelerometers and gyroscope (partly as expected). We may expect them to play a greater part in the prediction or at least we will see that they will be more interacting.

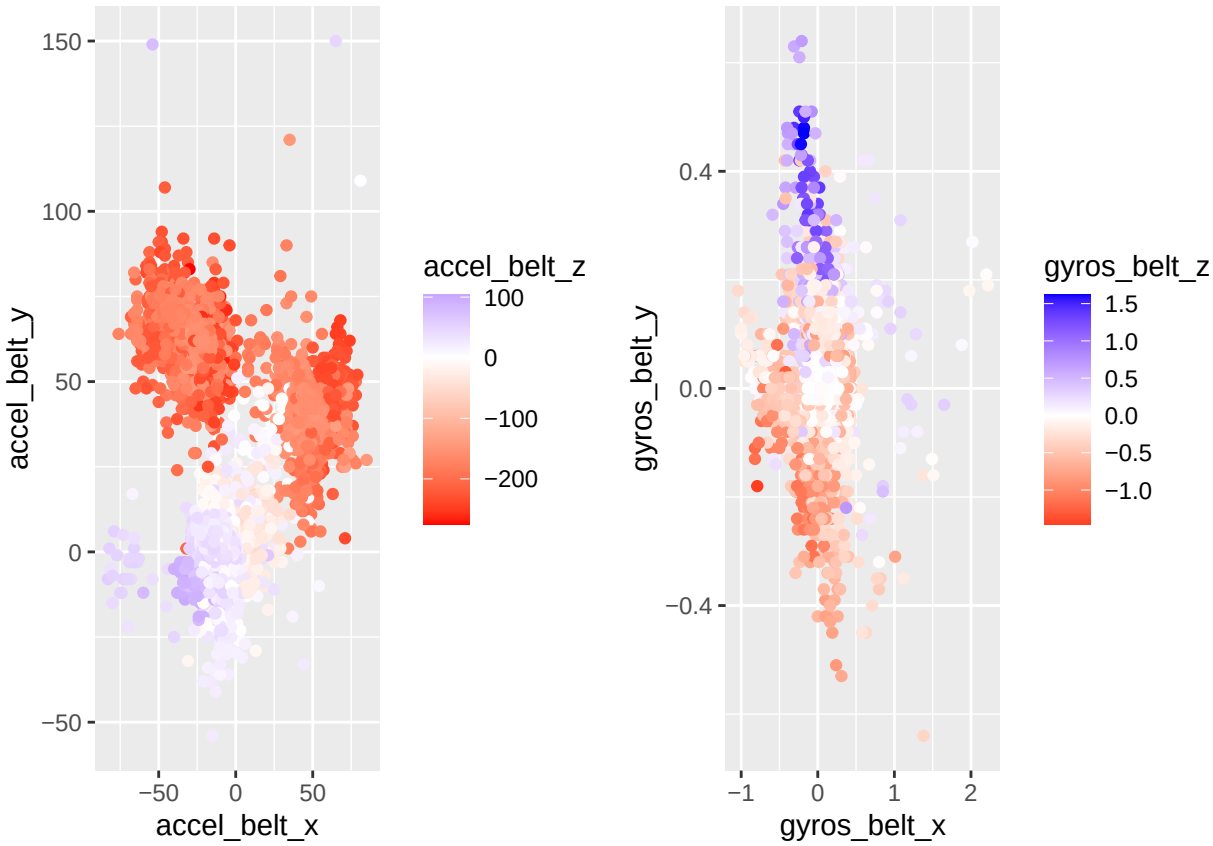
In fact we can dig a bit more on the subject and visualise the distribution of these sensors with respect to the `classe` label:

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.6.3
```

```
g1 <- ggplot(data = train,
             aes(x = accel_belt_x, y = accel_belt_y, col = accel_belt_z)) +
      geom_point() +
      scale_color_gradient2(low = "red", mid = "white", high = "blue",
                            midpoint = 0)
g2 <- ggplot(data = train,
             aes(x = gyros_belt_x, y = gyros_belt_y, col = gyros_belt_z)) +
      geom_point() +
      scale_color_gradient2(low = "red", mid = "white", high = "blue",
                            midpoint = 0)

grid.arrange(g1, g2, nrow = 1)
```



We then divide the training variables from the labels:

```
x.train <- train[,-labels]
y.train <- train[,labels]
x.test  <- test[,-labels]
y.test  <- test[,labels]
```

The summary of the training variables can then be useful to understand the next step in the analysis:

```
library(skimr)
```

```
## Warning: package 'skimr' was built under R version 3.6.3
```

```
skim(x.train)
```

Table 1: Data summary

Name	x.train
Number of rows	15699
Number of columns	52
Column type frequency:	
numeric	52
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
roll_belt	0	1	64.47	62.74	-28.90	1.09	114.00	123.00	162.00
pitch_belt	0	1	0.31	22.37	-55.80	1.70	5.28	15.00	60.30
yaw_belt	0	1	-11.11	95.17	-179.00	-88.30	-12.60	12.60	179.00
total_accel_belt	0	1	11.32	7.73	0.00	3.00	17.00	18.00	29.00
gyros_belt_x	0	1	-0.01	0.21	-1.04	-0.03	0.03	0.11	2.22
gyros_belt_y	0	1	0.04	0.08	-0.64	0.00	0.02	0.11	0.64
gyros_belt_z	0	1	-0.13	0.24	-1.46	-0.20	-0.10	-0.02	1.62
accel_belt_x	0	1	-5.58	29.68	-83.00	-21.00	-15.00	-5.00	85.00
accel_belt_y	0	1	30.19	28.58	-54.00	3.00	35.00	61.00	150.00
accel_belt_z	0	1	-72.67	100.45	-275.00	-162.00	-152.00	28.00	104.00
magnet_belt_x	0	1	55.67	64.04	-52.00	9.00	35.00	59.00	485.00
magnet_belt_y	0	1	593.64	36.05	354.00	581.00	601.00	610.00	673.00
magnet_belt_z	0	1	-345.50	65.78	-623.00	-375.00	-320.00	-306.00	293.00
roll_arm	0	1	18.19	72.58	-180.00	-31.40	0.00	77.50	180.00
pitch_arm	0	1	-4.70	30.61	-88.80	-26.00	0.00	10.90	88.50
yaw_arm	0	1	-0.52	71.21	-180.00	-42.80	0.00	45.40	180.00
total_accel_arm	0	1	25.44	10.58	1.00	17.00	27.00	33.00	66.00
gyros_arm_x	0	1	0.05	1.99	-6.37	-1.32	0.08	1.57	4.87
gyros_arm_y	0	1	-0.26	0.85	-3.44	-0.80	-0.26	0.14	2.84
gyros_arm_z	0	1	0.27	0.55	-2.33	-0.07	0.23	0.72	3.02
accel_arm_x	0	1	-60.00	181.66	-404.00	-241.00	-43.00	83.00	437.00
accel_arm_y	0	1	32.70	109.92	-318.00	-54.00	14.00	140.00	308.00
accel_arm_z	0	1	-70.78	134.62	-636.00	-142.00	-46.00	24.00	292.00
magnet_arm_x	0	1	192.00	443.82	-584.00	-299.00	289.00	638.00	782.00
magnet_arm_y	0	1	157.04	201.86	-392.00	-8.00	201.00	323.00	583.00
magnet_arm_z	0	1	307.23	325.26	-597.00	136.00	443.00	544.00	693.00
roll_dumbbell	0	1	23.80	69.83	-153.71	-18.17	48.09	67.31	153.55
pitch_dumbbell	0	1	-10.80	37.13	-149.59	-41.07	-20.99	17.49	149.40
yaw_dumbbell	0	1	1.64	82.59	-150.87	-77.70	-3.52	79.97	154.95
total_accel_dumbbell	0	1	13.77	10.23	0.00	5.00	10.00	20.00	58.00
gyros_dumbbell_x	0	1	0.16	1.68	-204.00	-0.03	0.14	0.37	2.22
gyros_dumbbell_y	0	1	0.05	0.64	-2.10	-0.14	0.05	0.21	52.00
gyros_dumbbell_z	0	1	-0.13	2.55	-2.38	-0.31	-0.13	0.03	317.00
accel_dumbbell_x	0	1	-28.70	67.52	-419.00	-51.00	-9.00	11.00	235.00
accel_dumbbell_y	0	1	52.78	80.98	-182.00	-8.00	42.00	112.00	315.00
accel_dumbbell_z	0	1	-38.43	109.60	-334.00	-142.00	-1.00	38.00	318.00
magnet_dumbbell_x	0	1	-327.50	341.17	-643.00	-535.00	-479.00	-305.50	592.00
magnet_dumbbell_y	0	1	219.78	327.83	-3600.00	231.00	310.00	390.00	633.00
magnet_dumbbell_z	0	1	45.99	139.73	-262.00	-45.00	14.00	95.00	452.00
roll_forearm	0	1	34.73	107.78	-180.00	-0.37	22.60	140.00	180.00
pitch_forearm	0	1	10.73	28.08	-72.50	0.00	9.16	28.30	89.80
yaw_forearm	0	1	19.59	103.00	-180.00	-68.10	0.00	110.00	180.00
total_accel_forearm	0	1	34.74	10.06	0.00	29.00	36.00	41.00	108.00
gyros_forearm_x	0	1	0.16	0.65	-22.00	-0.21	0.05	0.58	3.52
gyros_forearm_y	0	1	0.08	3.29	-7.02	-1.46	0.03	1.62	311.00
gyros_forearm_z	0	1	0.15	1.94	-8.09	-0.18	0.08	0.49	231.00
accel_forearm_x	0	1	-60.93	180.97	-498.00	-178.00	-56.00	78.00	477.00
accel_forearm_y	0	1	164.54	199.79	-595.00	58.00	201.00	313.00	923.00
accel_forearm_z	0	1	-55.47	138.22	-446.00	-181.00	-40.00	26.00	291.00
magnet_forearm_x	0	1	-312.71	346.66	-1280.00	-616.00	-378.00	-74.00	666.00
magnet_forearm_y	0	1	382.33	507.97	-896.00	15.00	593.00	737.00	1480.00
magnet_forearm_z	0	1	396.40	369.01	-966.00	195.00	513.00	655.00	1090.00

Machine Learning Predictions

we first preprocess the data and standardise the training features:

```
preprocess <- preProcess(x.train, method = c("center", "scale"))
x.new.train <- predict(preprocess, newdata = x.train)
x.new.test  <- predict(preprocess, newdata = x.test)
```

We finally apply a **random forest** algorithm to the training data. We use a 10-fold cross-validation asstrategy (partly to prevent overfit and improve predictions):

```
# set multithreading
library(doParallel)
```

```
## Warning: package 'doParallel' was built under R version 3.6.3
```

```
## Warning: package 'foreach' was built under R version 3.6.3
```

```
## Warning: package 'iterators' was built under R version 3.6.3
```

```
cl <- makeCluster(detectCores())
registerDoParallel(cl)

# begin training
set.seed(42)
train.ctrl <- trainControl(method = "cv", number = 10, search = "grid")
train.rf <- train(x.new.train, y.train, trControl = train.ctrl, method = "rf")

# stop multithreading
stopCluster(cl)
```

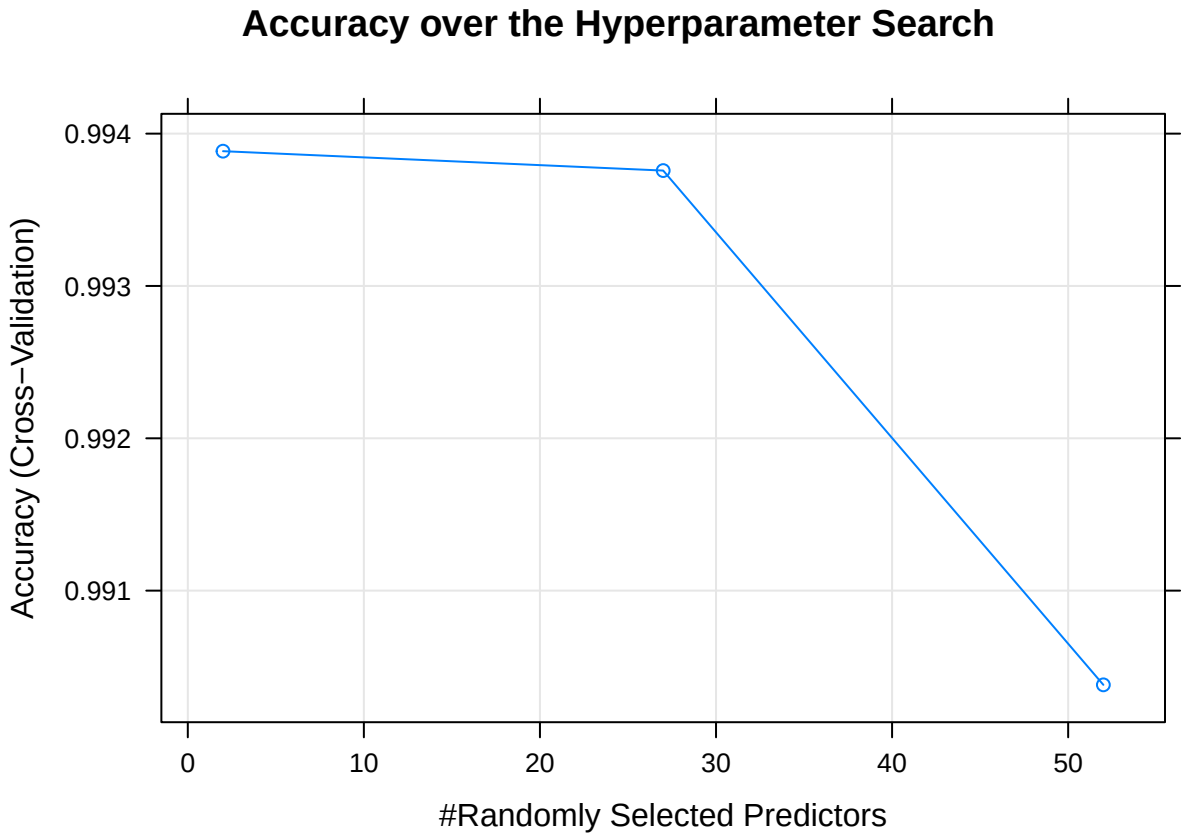
We the show the summary of the training procedure:

```
print(train.rf)
```

```
## Random Forest
##
## 15699 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 14130, 14128, 14129, 14130, 14128, 14130, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9938848  0.9922642
##   27    0.9937576  0.9921033
##   52    0.9903816  0.9878328
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

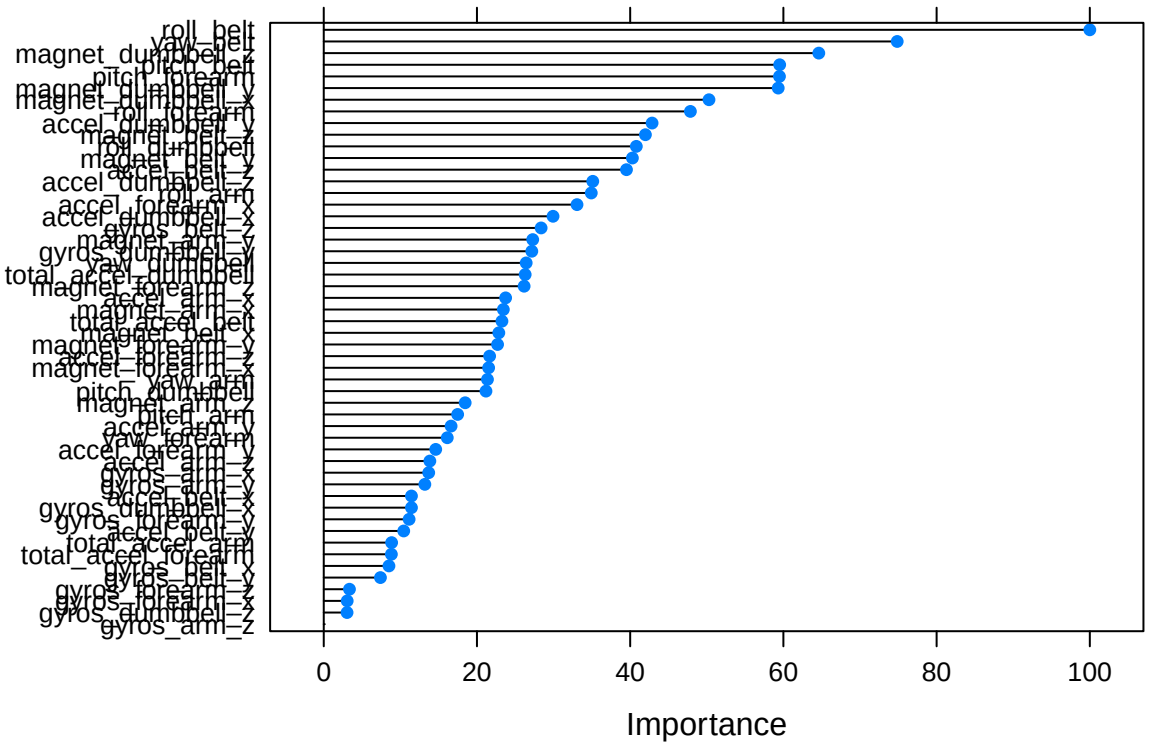
In training we performed a hyperparameter grid search in order to improve the accuracy of the prediction. In fact we show the plot of such search:

```
plot(train.rf, main="Accuracy over the Hyperparameter Search")
```



We finally plot the feature importance assigned by the algorithm to check that our analysis (including the exploratory analysis and dataset cleaning) was in fact meaningful:

```
var.imp <- varImp(train.rf)
plot(var.imp, main = "Variable Ranking (random forest)")
```



Finally we show the training performance:

```
train.pred <- predict(train.rf, newdata = x.new.train)
confusionMatrix(data = train.pred, reference = y.train, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4464    1    0    0    0
##          B    0 3037    5    0    0
##          C    0    0 2731   15    0
##          D    0    0    2 2557    0
##          E    0    0    0    1 2886
##
## Overall Statistics
##
##                Accuracy : 0.9985
##                  95% CI : (0.9977, 0.999)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9981
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9997   0.9974   0.9938   1.0000
## Specificity            0.9999   0.9996   0.9988   0.9998   0.9999
## Pos Pred Value         0.9998   0.9984   0.9945   0.9992   0.9997
## Neg Pred Value         1.0000   0.9999   0.9995   0.9988   1.0000
## Precision              0.9998   0.9984   0.9945   0.9992   0.9997
## Recall                 1.0000   0.9997   0.9974   0.9938   1.0000
## F1                     0.9999   0.9990   0.9960   0.9965   0.9998
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1740   0.1629   0.1838
## Detection Prevalence   0.2844   0.1938   0.1749   0.1630   0.1839
## Balanced Accuracy      1.0000   0.9996   0.9981   0.9968   1.0000
```

and for the test data:

```
test.pred <- predict(train.rf, newdata = x.new.test)
confusionMatrix(data = test.pred, reference = y.test, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    0  759    3    0    0
##          C    0    0  681    2    0
##          D    0    0    0  641    0
##          E    0    0    0    0  721
##
## Overall Statistics
##
##                Accuracy : 0.9987
##                  95% CI : (0.997, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9984
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9956   0.9969   1.0000
## Specificity            1.0000   0.9991   0.9994   1.0000   1.0000
## Pos Pred Value         1.0000   0.9961   0.9971   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   0.9991   0.9994   1.0000
## Precision              1.0000   0.9961   0.9971   1.0000   1.0000
## Recall                 1.0000   1.0000   0.9956   0.9969   1.0000
## F1                     1.0000   0.9980   0.9963   0.9984   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1736   0.1634   0.1838
## Detection Prevalence   0.2845   0.1942   0.1741   0.1634   0.1838
## Balanced Accuracy      1.0000   0.9995   0.9975   0.9984   1.0000
```

We finally plot the distribution of the **validation set** and compare the predictions:

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.6.3
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following objects are masked from 'package:reshape2':
##
##     dcast, melt
```

```
test.pred <- data.table(test.pred)
test.pred.n <- test.pred %>% count(test.pred)
y.test <- data.table(y.test)
y.test.n <- y.test %>% count(y.test)

# plot the counts
g <- ggplot() +
     geom_bar(data = test.pred , aes(test.pred, fill = test.pred), width = 0.75) +
     geom_bar(data = y.test, aes(y.test), width = 0.3, fill = "white") +
     xlab("classe") +
     ylab("count") +
     ggtitle("Validation set predictions and true values")
print(g)
```

Test Set Predictions

We finally consider the final test set:

```
final.test <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                 na.strings = c("#DIV/0!", "", "NA"),
                 stringsAsFactors = FALSE)

# perform the same transformations
final.test <- final.test[,-(1:7)]
final.test$problem_id <- as.factor(final.test$problem_id)
```

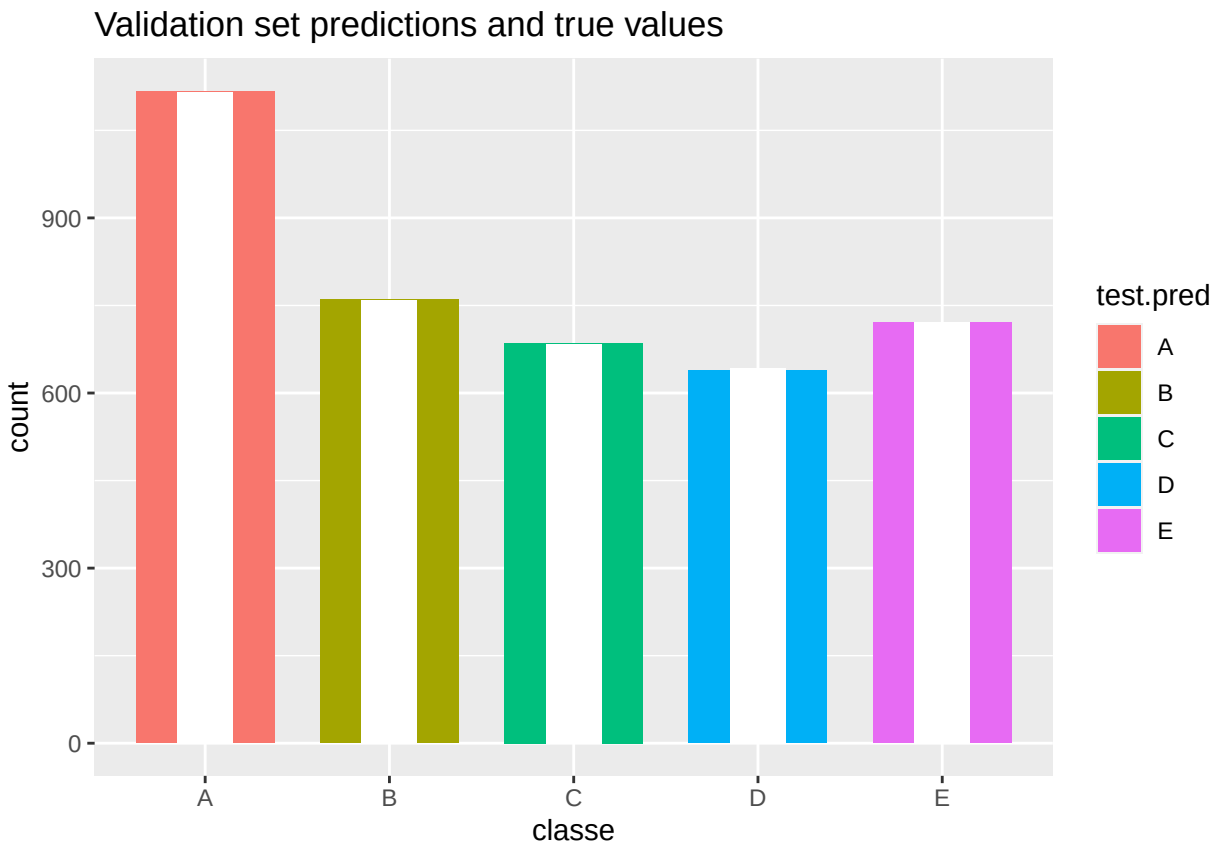


Figure 1: Predictions on the validation set: coloured bars represent true values and white superimposed values represent predictions.

```
test.data.na.fraction <- sapply(final.test, function(x) {mean(is.na(x))})
test.data.cols <- (test.data.na.fraction <= 0.5)

# select the relevant columns
test.data <- final.test[, test.data.cols]
test.data <- test.data[,-ncol(test.data)]

# apply pre-process transformation
test.data <- predict(preprocess, newdata = test.data)
```

We can now make the test set predictions:

```
final.predictions <- predict(train.rf, newdata = test.data)
print(final.predictions)
```

```
##  [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
```

Conclusions

We showed that we were able to make meaningful predictions on a cleaned version of the wearable database, achieving a very high level of accuracy using a random forest approach.