Practical ML Course Project

Enrique Figueroa

25/10/2021

Background

Six people using accelerometers on their bodies are asked to perform barbell lifts in 5 different ways. It have been identified 5 different ways to do it, called Class A, B, C, D and E. Only the Class A is considered correct. Many measures have been taken and been registered at the indicated datasets. The task is to train a model that predict correctly the Class based on relevant columns of the datasets.

Loading libraries and datasets

```
rm(list=ls())

library(caret)
library(ggplot2);
library(dplyr)

training <- read.csv(file="pml-training.csv", header=T)
validation <- read.csv(file="pml-testing.csv", header=T)
```

Preprocesing

Since the training dataset has redundant summary fields we reduced to those that can have an impact on the outcome. Also, ignore some useless columns.

```
relevant_cols= c("accel_", "gyros_", "roll_", "pitch_","yaw_", "magnet_")
train_df = select(training, classe, starts_with(relevant_cols))
sort(colnames(train_df) )
```

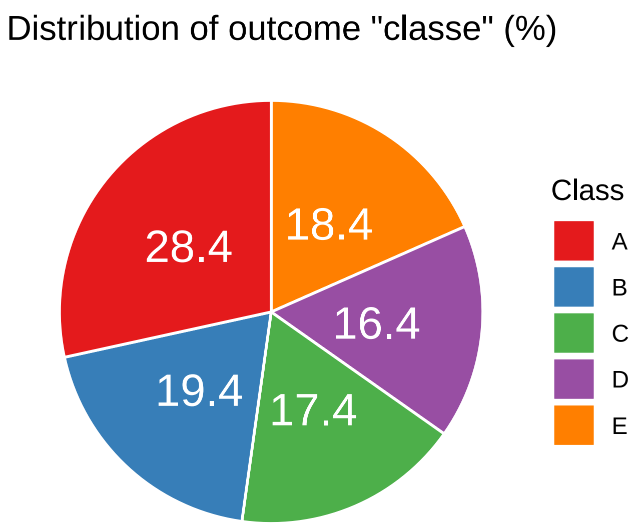
```
##  [1] "accel_arm_x"       "accel_arm_y"       "accel_arm_z"
##  [4] "accel_belt_x"      "accel_belt_y"      "accel_belt_z"
##  [7] "accel_dumbbell_x"  "accel_dumbbell_y"  "accel_dumbbell_z"
## [10] "accel_forearm_x"   "accel_forearm_y"   "accel_forearm_z"
## [13] "classe"            "gyros_arm_x"       "gyros_arm_y"
## [16] "gyros_arm_z"       "gyros_belt_x"      "gyros_belt_y"
## [19] "gyros_belt_z"      "gyros_dumbbell_x"  "gyros_dumbbell_y"
## [22] "gyros_dumbbell_z"  "gyros_forearm_x"   "gyros_forearm_y"
## [25] "gyros_forearm_z"   "magnet_arm_x"      "magnet_arm_y"
## [28] "magnet_arm_z"      "magnet_belt_x"     "magnet_belt_y"
## [31] "magnet_belt_z"     "magnet_dumbbell_x" "magnet_dumbbell_y"
```

```
## [34] "magnet_dumbbell_z" "magnet_forearm_x"  "magnet_forearm_y"
## [37] "magnet_forearm_z"  "pitch_arm"         "pitch_belt"
## [40] "pitch_dumbbell"    "pitch_forearm"     "roll_arm"
## [43] "roll_belt"         "roll_dumbbell"     "roll_forearm"
## [46] "yaw_arm"           "yaw_belt"          "yaw_dumbbell"
## [49] "yaw_forearm"
```

The pie below shows that the majority of participants perform incorrectly the barbell lifts (Class B, C, D, E). Only 28.4% perform the lifts correctly.

```
table_pie =as.data.frame(table(train_df$classe))
colnames(table_pie) = c("Class", "Freq")
table_pie = table_pie %>% arrange(desc(Class)) %>%
  mutate(prop = Freq / sum(table_pie$Freq) * 100) %>%
  mutate(ypos = cumsum(prop) - 0.5*prop )

ggplot(table_pie, aes(x="", y=prop, fill=Class)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) + theme_void() +
  geom_text(aes(y = ypos, label = round(prop,1)), color = "white", size=6) +
  scale_fill_brewer(palette="Set1") + ggtitle("Distribution of outcome \"classe\" (%)")
```



Partioning the training dataset

Now that we reduced the dataset from 160 columns to only 49 columns, we divide the huge training dataset in two parts: one for training the models (60%) and the other one for testing their performance (40%).

```
inTrain = createDataPartition(y=train_df$classe, p=0.6, list=FALSE)

train_df = train_df[inTrain, ];
test_df = train_df[-inTrain, ]
```

Training models

Five models will be tried: decision trees, random forest (rf), bagging (gbm), support vector machine (svm) and linear discriminant analysis(lda).

First, we set up the cross validation parameter to 2-fold. This parameter will be used in the train function for each model.

```
set.seed(7777)
control = trainControl(method="cv", number=2, verboseIter=F)
```

Decision Trees

```
mod_dt = train(classe ~ ., data=train_df, method="rpart", trControl = control)
```

Random Forests

```
mod_rf = train(classe~., data=train_df, method="rf", trControl = control)
```

Generalized Boosted Regression Modeling

```
mod_gbm = train(classe~., data=train_df, method="gbm", trControl = control, verbose = F)
```

Suppor Vector Machine

```
mod_svm = train(classe~., data=train_df, method="svmLinear", trControl = control, verbose = F)
```

Linear Discriminant Analysis

```
mod_lda = train(classe ~ ., data = train_df, method = "lda", trControl = control, verbose = F)
```

Prediction

Once the models have been trained we predict the classes (A, B, C, D or E) on the test_df to calculate some metrics. Since the procedure is the same for the five models a function is defined previously for calculating the confusion matrix.

```
pred_model = function(model, dataset){
  mod_predict = predict(model, dataset)
  conf_matrix = confusionMatrix(mod_predict, as.factor(dataset$classe))
  return(conf_matrix)
}
```

Then, we calculate each confusion matrix.

```
cm_dt = pred_model(mod_dt, test_df)
cm_rf = pred_model(mod_rf, test_df)
cm_gbm = pred_model(mod_gbm, test_df)
cm_svm = pred_model(mod_svm, test_df)
cm_lda = pred_model(mod_lda, test_df)
```

Results on testing dataset

Based upon the prior calculated confusion matrices, we build a table that shows the accuracy of each model.

```
res = round( rbind(
  "Decision Trees" = cm_dt$overall["Accuracy"],
  "Random Forest" = cm_rf$overall["Accuracy"],
  "Generalized Boosting " = cm_gbm$overall["Accuracy"],
  "Support Vector Machine" = cm_svm$overall["Accuracy"],
  "Linear Discriminant Analysis" = cm_lda$overall["Accuracy"]
), 2)

knitr::kable(res, caption = "**Overall model accuracy**", format="simple")
```

Table 1: **Overall model accuracy**

	Accuracy
Decision Trees	0.51
Random Forest	1.00
Generalized Boosting	0.98
Support Vector Machine	0.78
Linear Discriminant Analysis	0.71

Since what we are mostly interesed in calculating the Class A and not the other classes that represent mistakes, a table with relevant metrics for that Class A is composed with a function that takes the trained model as an argument.

```
table_row = function(model, modelName){
  result = rbind(
  "Bal Accuracy"= round(model$byClass[1,7],2),
  "Sensitivity"= round(model$byClass[1,1],2),
  "Specificity"= round(model$byClass[1,2],2),
  "Precision"= round(model$byClass[1,5],2),
  "Recall"= round(model$byClass[1,6],2))

  result = t(as.data.frame(result))
  rownames(result) = modelName

  return(result)
}

classA_result = rbind(
table_row(cm_dt, "Decision Trees"),
```

```r
table_row(cm_rf, "Random Forest"),
table_row(cm_gbm, "Generalized Boosting"),
table_row(cm_svm, "Support Vector Machine"),
table_row(cm_lda, "Linear Discriminant Analysis") )

knitr::kable(classA_result, caption = "**Class A metrics**", format="simple")
```

Table 2: **Class A metrics**

	Bal Accuracy	Sensitivity	Specificity	Precision	Recall
Decision Trees	0.65	0.91	0.63	0.51	0.91
Random Forest	1.00	1.00	1.00	1.00	1.00
Generalized Boosting	0.99	0.99	1.00	0.99	0.99
Support Vector Machine	0.86	0.92	0.92	0.82	0.92
Linear Discriminant Analysis	0.80	0.81	0.91	0.79	0.81

Prediction on validation dataset

We use random forest, the best model according to the out of sample metrics, to predict the 20 observations of the validation dataset.

```r
pred_val <- predict(mod_rf, validation)
print(pred_val)
```

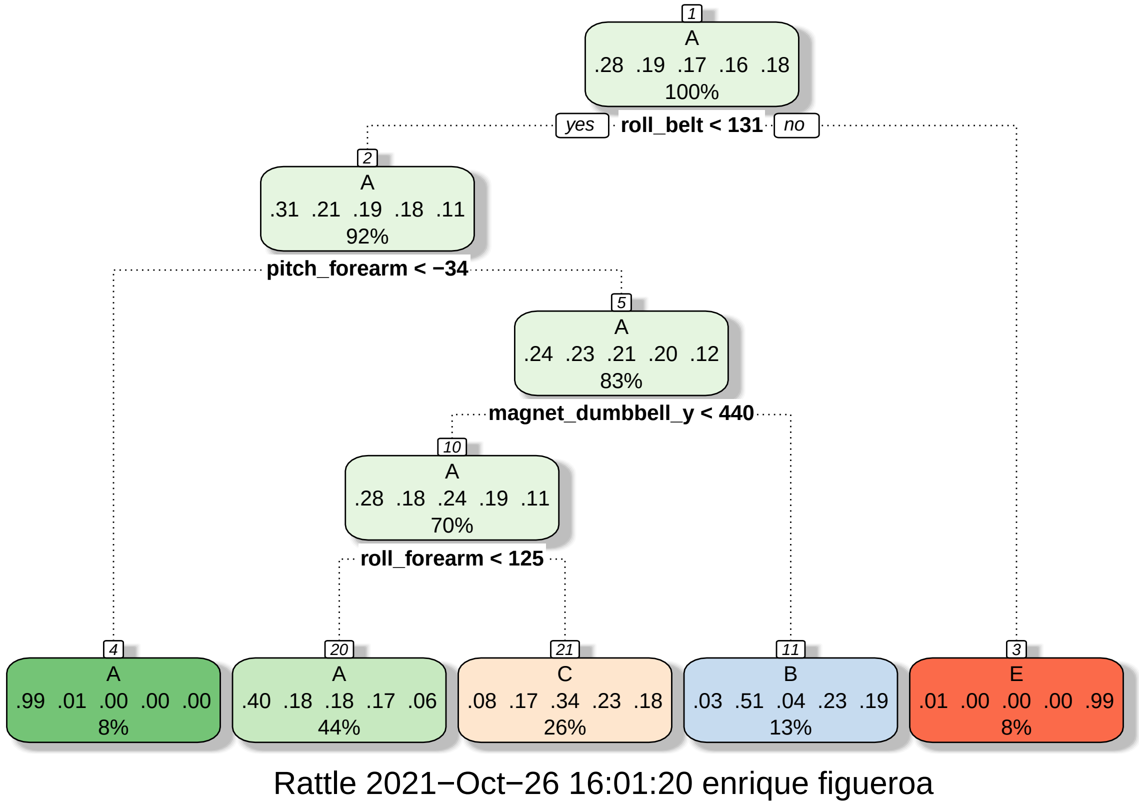
```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

Conclusion

The out of sample metrics clearly show that the best method is Random Forest, followed by Generalized Boosted Regression model.

Appendix

```r
library(rattle)
fancyRpartPlot(mod_dt$finalModel)
```


1
A
.28 .19 .17 .16 .18
100%
yes
roll_belt < 131
no
2
A
.31 .21 .19 .18 .11
92%
pitch_forearm < −34
5
A
.24 .23 .21 .20 .12
83%
magnet_dumbbell_y < 440
10
A
.28 .18 .24 .19 .11
70%
roll_forearm < 125
4
A
.99 .01 .00 .00 .00
8%
20
A
.40 .18 .18 .17 .06
44%
21
C
.08 .17 .34 .23 .18
26%
11
B
.03 .51 .04 .23 .19
13%
3
E
.01 .00 .00 .00 .99
8%
Rattle 2021−Oct−26 16:01:20 enrique figueroa