Course Project

AF

05/04/2021

Project Goal

In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise (`classe` variable in training set). They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

(1) Import data files and load relevant libraries

```
## load input data
training = read.csv("practical ML course/data/pml-training.csv",
                    header = T,
                    row.names = 1,
                    na.strings=c("", "#DIV/0!", "NA")) ## take care of empty cells as NA missing values
testing = read.csv("practical ML course/data/pml-testing.csv", header = T, row.names = 1)

## load relevant libraries
library(caret)
library(randomForest)
library(ggplot2)
library(corrplot) ## used for creating correlation matrix plot
library(cvms) ## used for creating confusion matrix plot
library(broom) ## used for creating confusion matrix plot
library(tibble) ## used for creating confusion matrix plot
```

(2) Data cleaning

(2a): Identify which columns/variables in the training dataset have missing values

```
percent_NA<-sapply(training, function (x){100*sum(is.na(x))/nrow(training)})
percent_NA_filtered<-as.data.frame(percent_NA[percent_NA>0])
names(percent_NA_filtered)<-"% NA"
percent_NA_filtered$VAR<-row.names(percent_NA_filtered)
rownames(percent_NA_filtered) <- NULL
head(percent_NA_filtered[order(-percent_NA_filtered$`% NA`),])
```

```
##    % NA                  VAR
## 3   100     kurtosis_yaw_belt
## 6   100     skewness_yaw_belt
## 53  100 kurtosis_yaw_dumbbell
## 56  100 skewness_yaw_dumbbell
## 78  100  kurtosis_yaw_forearm
## 81  100  skewness_yaw_forearm
```

```
paste0("The number of variables with > 97.9% of missing values is = ", nrow(percent_NA_filtered))
```

```
## [1] "The number of variables with > 97.9% of missing values is = 100"
```

(2b): Remove the variables identified above (given the high % of missing values they contain) from the training and testing sets

```
toRemove<-names(percent_NA[percent_NA>0])
training_clean1<-training
training_clean1[toRemove]<-NULL
testing_clean1<-testing
testing_clean1[toRemove]<-NULL

## counting N of predictors remaining
paste0("N of predictors is now = ", dim(training_clean1)[2]-1) # I remove the outcome variable from the co
unt
```

```
## [1] "N of predictors is now = 58"
```

(2c) Remove timestamp variables, username and variables with near zero variability, since they will not be used for prediction

```
## remove timestamp and username variables:
training_clean1<-training_clean1[-c(1:4)]
testing_clean1<-testing_clean1[-c(1:4)]

## identify and remove variables with near zero variability:
head(nearZeroVar(training_clean1, saveMetrics = T)) ## new_window variable had near-zero variance
```

```
##                 freqRatio percentUnique zeroVar   nzv
## new_window      47.330049    0.01019264   FALSE  TRUE
## num_window       1.000000    4.37264295   FALSE FALSE
## roll_belt        1.101904    6.77810621   FALSE FALSE
## pitch_belt       1.036082    9.37722964   FALSE FALSE
## yaw_belt         1.058480    9.97349913   FALSE FALSE
## total_accel_belt  1.063160    0.14779329   FALSE FALSE
```

```
training_final<-training_clean1
training_final$new_window<-NULL
testing_final<-testing_clean1
testing_final$new_window<-NULL

rm(training_clean1,testing_clean1)
## counting N of predictors remaining
paste0("N of predictors is now = ", dim(training_final)[2]-1) # I remove the outcome variable from the cou
nt
```

```
## [1] "N of predictors is now = 53"
```

(2d) The outcome variable needs to be transformed from character value into factor

```
training_final$classe<-as.factor(training_final$classe)
testing_final$problem_id<-as.factor(testing_final$problem_id)
#summary(training_final) ##checking the summary
```

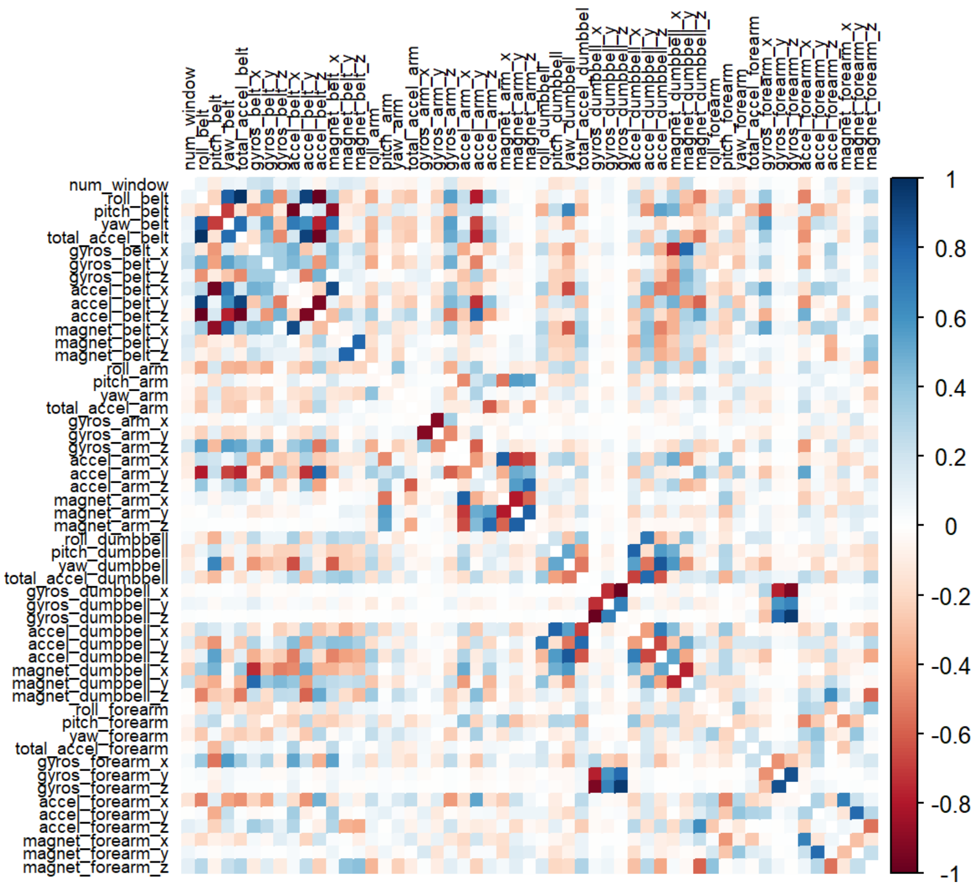
(3) Data exploration on the training set

(3a) Split the training dataset into a smaller training set (train_small: 70%) and a validating set (validate_small: 30%) I carry out this further set splitting because I want to use the "training_final" dataset as ultimate set for model prediction

```
set.seed(22519) # For allowing output reproducibility
inTrain <- createDataPartition(training_final$classe, p=0.70, list=F)
train_small <- training_final[inTrain, ]
validate_small <- training_final[-inTrain, ]
```

(3b) Check correlations between predictors in the train_small set

```
corrPlot <- cor(train_small[, -length(names(train_small))])
corrplot(corrPlot, method="color", tl.cex=0.60, tl.col="black", tl.srt = 90, diag = FALSE)
```



From the correlation matrix plot is evident that some predictors are highly correlated. In order to deal with them, I will set up two models, one with all predictors not pre-processed (Model 1) and another (Model 2) with predictors pre-processed using PCA.

(4) Data pre-processing, modelling and validation

(4a) Model 1: random forest using all predictors

Train on the `train_small` dataset:

```
control_rf <- trainControl(method="cv", 5)
model_rf <- train(classe ~ ., data=train_small, method="rf", trControl=control_rf, ntree=250)
model_rf
```

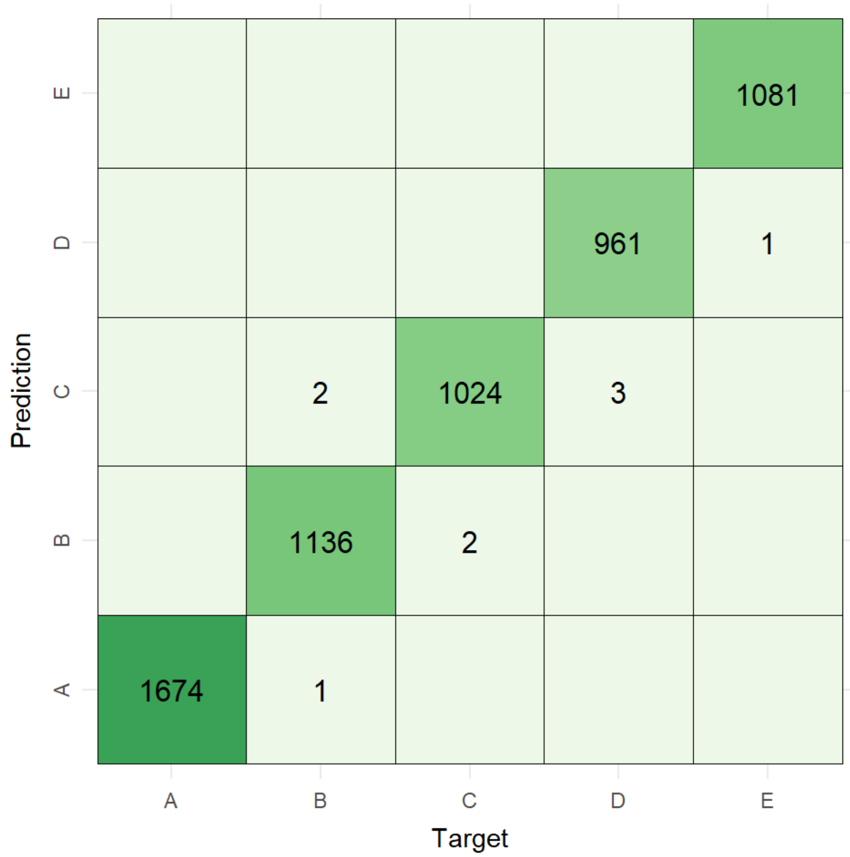
```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10988, 10989, 10989, 10991, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9930123  0.9911601
##   27    0.9973072  0.9965939
##   53    0.9951963  0.9939240
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Predict and calculate accuracy and confusion matrix on the `validate_small` dataset for Model 1 (`model_rf`):

```
confusionMatrix(validate_small$classe,predict(model_rf,newdata = validate_small))[3]
```

```
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9984707      0.9980656      0.9970989      0.9993005      0.2846219
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
t<-table(reference = validate_small$classe,prediction = predict(model_rf,newdata = validate_small))
plot_confusion_matrix(tidy(t),
                      target_col = "reference",
                      prediction_col = "prediction",
                      counts_col = "n",
                      place_x_axis_above = FALSE,
                      add_normalized = FALSE,
                      add_col_percentages = FALSE,
                      add_row_percentages = FALSE,
                      palette = "Greens",
                      tile_border_color = "black",
                      tile_border_size = 0.05)
```



Model 1 (random forest without covariates' pre-processing) is able to predict the outcome class with 99.8% of accuracy in the validation set

(4b) Model 2: random forest using predictors pre-processed using PCA

Train on the `train_small` dataset:

```
##keeping principal components able to explain 80% of variance
control_rf_PCA <- trainControl(preProcOptions=list(thresh=0.8),method="cv", 5)
model_rf_pca <- train(classe ~ ., data=train_small, method="rf",preProcess="pca", trControl=control_rf, nt
ree=250)
model_rf_pca
```

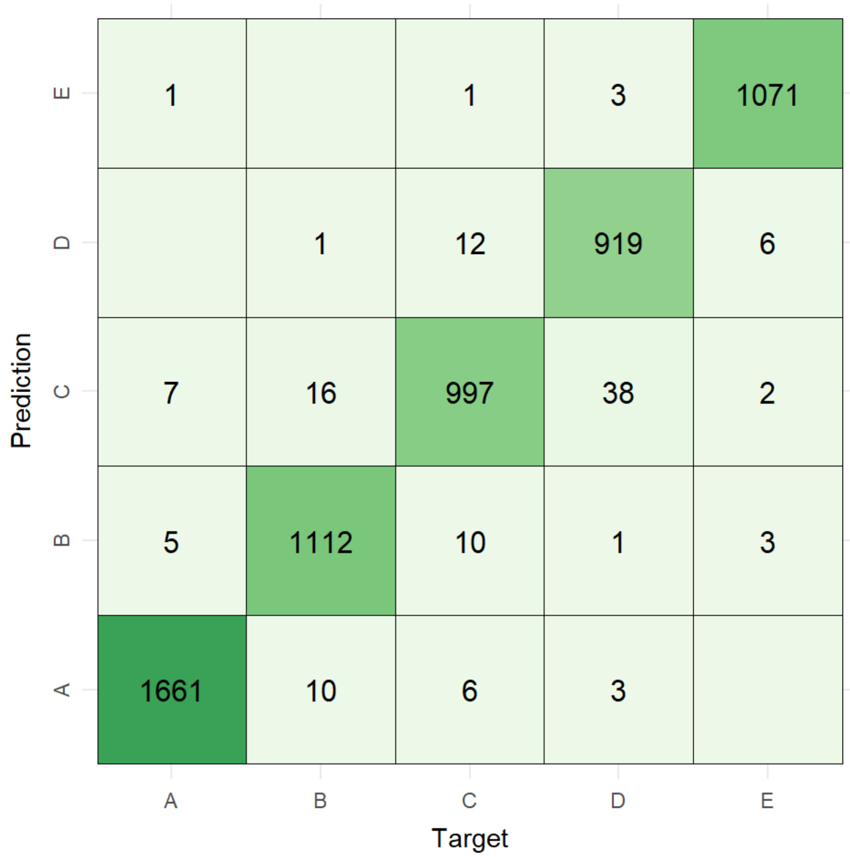
```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: principal component signal extraction (53), centered
##  (53), scaled (53)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10990, 10991, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9716100  0.9640858
##   27    0.9582157  0.9471466
##   53    0.9587980  0.9478834
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Predict and calculate accuracy and confusion matrix on the `validate_small` dataset for Model 2 (`model_rf_pca`):

```
confusionMatrix(validate_small$classe,predict(model_rf_pca,newdata = validate_small))[3]
```

```
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##    0.979099405    0.973559200    0.975113154    0.982600187    0.285471538
## AccuracyPValue  McnemarPValue
##    0.000000000    0.007347242
```

```
t<-table(reference = validate_small$classe,prediction = predict(model_rf_pca,newdata = validate_small))
plot_confusion_matrix(tidy(t),
                      target_col = "reference",
                      prediction_col = "prediction",
                      counts_col = "n",
                      place_x_axis_above = FALSE,
                      add_normalized = FALSE,
                      add_col_percentages = FALSE,
                      add_row_percentages = FALSE,
                      palette = "Greens",
                      tile_border_color = "black",
                      tile_border_size = 0.05)
```



Model 2 (random forest with covariates' PCA pre-processing) is able to predict the outcome class with 97.9% of accuracy in the validation set

Model 1 (`model_rf`) is more accurate then model 2 (`model_rf_pca`), so I will use it to predict the outcome classes in the `testing_final` dataset

(5) Calculate final predictions

```
final_predictions<-predict(model_rf,newdata=testing_final)
df<-cbind(case = testing_final$problem_id, pred = as.data.frame(final_predictions))
df
```

```
##    case final_predictions
## 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
```