ML project

Me

12/08/2021

Reading the given training and test sets

```
knitr::opts_chunk$set(echo = TRUE)
library(caret)
```

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

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

```
library(ggplot2)


pml_training<- read.csv('pml-training.csv')
pml_testing<- read.csv('pml-testing.csv')
```

Having a quick look at our training and test sets

```
head(pml_training)
str(pml_training)
head(pml_testing)
str(pml_testing)
```

Cleaning the data

Step 1: Dealing with NA values

NA strategy: if a variable or column has more than 50% NA values then we will omit it from our dataset. Any variable that satisfies this condition and has less than or equal to 50% NA values will be kept. After that we will impute the remaining NA values and check if it will be a viable predictor for our model building or not

```
cond<- colSums(is.na(pml_training))<= (0.50*nrow(pml_training))
my_cleanset<- pml_training[, cond] #subsetting our training set to include only those variabl
es that satisfy cond i.e. we will only keep a variable or column that has 50% or lesser NA va
lues
```

We are now checking for any integer or numeric variables that are being stored in character format and convert them into numeric class. We will apply the same condition from before i.e. we will only keep a variable or column that has 50% or lesser NA values

```
character_variables<- my_cleanset[,sapply(my_cleanset, class)=='character']
str(character_variables)
```

We will now apply our NA strategy to see which variables to keep and which variables to remove. "" is the same as NA value in character format

```
cond2<- colSums(character_variables=="")<= (0.50*nrow(character_variables)) #we want to find
 how many of the character variables can be converted into numeric class provided they have a
tleast 50% of the values i.e. we will tolerate maximum 50% NA values
cond2
character_variables_subset<- character_variables[,cond2] #subsetting character_variables so t
hat it satisfies our cond2 or the NA strategy
head(character_variables_subset) #so we don't need any of the columns aside from user_name, c
vtd_timestamp, new_window and classe as they did not satisfy cond2
```

Now doing the appropriate class conversions

```
library(dplyr)
```

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

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

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

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

```
character_variables_subset <- mutate(character_variables_subset, cvtd_timestamp= dmy_hm(cvtd_
timestamp))
str(character_variables_subset)
```

Now replacing the proper preprocessed columns in character_variables_subset to my_cleanset or our training set

```
my_cleanset2<- my_cleanset
my_cleanset2$cvtd_timestamp<- character_variables_subset$cvtd_timestamp
my_cleanset<- my_cleanset2
```

Now we are removing all the unnecessary character variables that did not satisfy cond2

```
char_var_to_be_removed<- names(character_variables[,!cond2])
my_cleanset<- select(my_cleanset, -char_var_to_be_removed)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(char_var_to_be_removed)` instead of `char_var_to_be_removed` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

Step 2: Data conversions and removing remaining unnecessary variables**

Converting classe and new_window into factor variables

```
my_cleanset$classe<- as.factor(my_cleanset$classe)
my_cleanset$new_window<- as.factor(my_cleanset$new_window)
```

We will use nearZeroVar() to find out the variables that have near zero variance and will omit them from our model building process

```
library(ISLR)
library(caret)
```

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

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

```
nearZeroVar(my_cleanset, saveMetrics = TRUE) #so the variable new_window has near zero variance. So it won't be a good predictor for our model.
```

```
##                      freqRatio percentUnique zeroVar   nzv
## X                     1.000000  100.00000000   FALSE FALSE
## user_name             1.100679    0.03057792   FALSE FALSE
## raw_timestamp_part_1  1.000000    4.26562022   FALSE FALSE
## raw_timestamp_part_2  1.000000   85.53154622   FALSE FALSE
## cvtd_timestamp        1.000668    0.10192641   FALSE FALSE
## 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
## gyros_belt_x          1.058651    0.71348486   FALSE FALSE
## gyros_belt_y          1.144000    0.35164611   FALSE FALSE
## gyros_belt_z          1.066214    0.86127816   FALSE FALSE
## accel_belt_x          1.055412    0.83579655   FALSE FALSE
## accel_belt_y          1.113725    0.72877383   FALSE FALSE
## accel_belt_z          1.078767    1.52379982   FALSE FALSE
## magnet_belt_x         1.090141    1.66649679   FALSE FALSE
## magnet_belt_y         1.099688    1.51870350   FALSE FALSE
## magnet_belt_z         1.006369    2.32901845   FALSE FALSE
## roll_arm             52.338462   13.52563449   FALSE FALSE
## pitch_arm            87.256410   15.73234125   FALSE FALSE
## yaw_arm              33.029126   14.65701763   FALSE FALSE
## total_accel_arm       1.024526    0.33635715   FALSE FALSE
## gyros_arm_x           1.015504    3.27693405   FALSE FALSE
## gyros_arm_y           1.454369    1.91621649   FALSE FALSE
## gyros_arm_z           1.110687    1.26388747   FALSE FALSE
## accel_arm_x           1.017341    3.95984099   FALSE FALSE
## accel_arm_y           1.140187    2.73672409   FALSE FALSE
## accel_arm_z           1.128000    4.03628580   FALSE FALSE
## magnet_arm_x          1.000000    6.82397309   FALSE FALSE
## magnet_arm_y          1.056818    4.44399144   FALSE FALSE
## magnet_arm_z          1.036364    6.44684538   FALSE FALSE
## roll_dumbbell         1.022388   84.20650290   FALSE FALSE
## pitch_dumbbell        2.277372   81.74498012   FALSE FALSE
## yaw_dumbbell          1.132231   83.48282540   FALSE FALSE
## total_accel_dumbbell  1.072634    0.21914178   FALSE FALSE
## gyros_dumbbell_x      1.003268    1.22821323   FALSE FALSE
## gyros_dumbbell_y      1.264957    1.41677709   FALSE FALSE
## gyros_dumbbell_z      1.060100    1.04984201   FALSE FALSE
## accel_dumbbell_x      1.018018    2.16593619   FALSE FALSE
## accel_dumbbell_y      1.053061    2.37488533   FALSE FALSE
## accel_dumbbell_z      1.133333    2.08949139   FALSE FALSE
## magnet_dumbbell_x     1.098266    5.74864948   FALSE FALSE
## magnet_dumbbell_y     1.197740    4.30129447   FALSE FALSE
## magnet_dumbbell_z     1.020833    3.44511263   FALSE FALSE
## roll_forearm         11.589286   11.08959331   FALSE FALSE
## pitch_forearm        65.983051   14.85577413   FALSE FALSE
## yaw_forearm          15.322835   10.14677403   FALSE FALSE
## total_accel_forearm   1.128928    0.35674243   FALSE FALSE
## gyros_forearm_x       1.059273    1.51870350   FALSE FALSE
## gyros_forearm_y       1.036554    3.77637346   FALSE FALSE
## gyros_forearm_z       1.122917    1.56457038   FALSE FALSE
## accel_forearm_x       1.126437    4.04647844   FALSE FALSE
## accel_forearm_y       1.059406    5.11160942   FALSE FALSE
## accel_forearm_z       1.006250    2.95586586   FALSE FALSE
```

```
## magnet_forearm_x      1.012346    7.76679238   FALSE FALSE
## magnet_forearm_y      1.246914    9.54031189   FALSE FALSE
## magnet_forearm_z      1.000000    8.57710733   FALSE FALSE
## classe                1.469581    0.02548160   FALSE FALSE
```

```
#We also no that the variable X contains just the serial numbers so we won't need it.
```

so removing X and new_window from our train set

```
my_cleanset_updated<- select(my_cleanset, -c(X, new_window))
str(my_cleanset_updated) #this is the final cleaned and formatted train set
```

```
## 'data.frame':    19622 obs. of  58 variables:
##  $ user_name           : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1: int  1323084231 1323084231 1323084231 1323084232 1323084232 13230
84232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 440390 4843
23 484434 ...
##  $ cvtd_timestamp      : POSIXct, format: "2011-12-05 11:23:00" "2011-12-05 11:23:00" ...
##  $ num_window          : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4
...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02
...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8
...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
```

```
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Building a model with classe as our outcome variable and random forest as our prediction method

Algorithm: We will use random forest here as our outcome variable classe is a categorical variable and random forests are good with non linear data

Now dividing the train set further into a smaller train (70%) and validation set(30%). We will then building a model with random forest method

```
set.seed(100)
project_train<- createDataPartition(my_cleanset_updated$classe, p=0.70, list = FALSE)
my_cleanset_training<- my_cleanset_updated[project_train,]
my_cleanset_validation<- my_cleanset_updated[-project_train,]
```

Usiing parallel processing for improving the processing time of random forest

```
#Step 1: Configure parallel processing
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)



#Configuring trainControl object.We will be doing cross validation with 10 folds

modControl<- trainControl(method = 'cv', number = 5, verboseIter = TRUE, allowParallel = TRUE
)



#Finally, building a model with random forest method



set.seed(100)
system.time(model_RF<- train(classe~., data= my_cleanset_training, method= 'rf', trControl= m
odControl))
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 31 on full training set
```

```
##    user  system elapsed
##   39.51    1.74  576.51
```

```
#Step 4: De-register parallel processing cluster

stopCluster(cluster)
registerDoSEQ()
```

Now evaluating the accuracy on our validation set

```
set.seed(100)
modelRF_predictions<- predict(model_RF, my_cleanset_validation)
confusionMatrix(my_cleanset_validation$classe, modelRF_predictions)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    2 1024    0    0
##          D    0    0    1  963    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9995
##                  95% CI : (0.9985, 0.9999)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9994
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9982   0.9990   1.0000   1.0000
## Specificity            1.0000   1.0000   0.9996   0.9998   1.0000
## Pos Pred Value         1.0000   1.0000   0.9981   0.9990   1.0000
## Neg Pred Value         1.0000   0.9996   0.9998   1.0000   1.0000
## Prevalence             0.2845   0.1939   0.1742   0.1636   0.1839
## Detection Rate         0.2845   0.1935   0.1740   0.1636   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   0.9991   0.9993   0.9999   1.0000
```

Conclusion

So we are getting an accuracy of 0.9985 or near 100% accuracy (approximately). So we can conclude that using a random forest model is giving us near perfect accuracy on our validation set

Prediciting on the test data set now

```
pml_testing_updated<- pml_testing[,-160] #removing problem ID from test set
#cleaning the test set
pml_testing_updated<- select(pml_testing_updated, names(my_cleanset_updated)[-58]) #selecting
only the variables in our final cleaned train set and removing classe as it does not exist th
e test set

pml_testing_updated<- mutate(pml_testing_updated, cvtd_timestamp= dmy_hm(cvtd_timestamp))
set.seed(100)
#now predicting on the test set
testing_predictions<- predict(model_RF, pml_testing_updated)
testing_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
```