PredictingClasse

Augusto Abe

03/11/2020

Loading the data sets

```
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
```

Looking at variables with NAs.

```
num_nas <- c()
num_rows <- nrow(training)
num_cols <- ncol(training)

for(i in 1:num_cols){
        nas <- is.na(training[,i])
        num_nas <- c(num_nas, length(nas[nas]))
}

as.numeric(levels(as.factor(num_nas)))/num_rows
```

```
## [1] 0.0000000 0.9793089
```

Variables or do not have NA or 98% of your observations are NA so this variables with NAs will be excluded

```
var_nas <- num_nas > 0
training <- training[,!var_nas]
```

Now we will do the same thing with empty values ("").

```
num_empties <- c()
num_cols <- ncol(training)

for(i in 1:num_cols){
        empties <- training[,i] == ""
        num_empties <- c(num_empties, length(empties[empties]))
}

as.numeric(levels(as.factor(num_nas)))/num_rows
```

```
## [1] 0.0000000 0.9793089
```

This is the same result as NA.

```r
var_empties <- num_empties > 0
training <- training[,!var_empties]
```

Now we will change the class of the variables or guarantee that they belong to the correct class and remove X column.

```r
training <- training[,-1]
training$user_name <- as.factor(training$user_name)
training$classe <- as.factor(training$classe)
training$cvtd_timestamp <- as.POSIXct(training$cvtd_timestamp, format = "%d/%m/%Y %H:%M")
training$new_window <- as.factor(training$new_window)

for(i in c(2, 3, 6:58 )){
        training[,i] <- as.numeric(training[,i])
}
```

Now our data set is cleaned.

As our variable to be predicted is categorical we try test three models: linear discriminant analysis, tree and random forest

Linear discriminant analysis:

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.3
```

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

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

```r
model_lda <- train(classe~., data = training, method = "lda",
               trControl = trainControl(method = "cv",
                                        number = 10))
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```r
print(model_lda)
```

```
## Linear Discriminant Analysis
##
## 19622 samples
##    58 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17659, 17660, 17660, 17660, 17660, 17659, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7448271  0.6768372
```

Tree:

```r
model_tree <- train(classe~., data = training, method = "rpart",
               trControl = trainControl(method = "cv",
                                        number = 10))

print(model_tree)
```

```
## CART
##
## 19622 samples
##    58 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17659, 17660, 17659, 17659, 17660, 17661, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03891896  0.5457707  0.41771525
##   0.05998671  0.4036642  0.18759267
##   0.11515454  0.3242362  0.06072321
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03891896.
```

Random Forest:

```r
model_rf <- train(classe~., data = training, method = "rf",
               trControl = trainControl(method = "cv",
                                        number = 10))

print(model_rf)
```

```
## Random Forest
##
## 19622 samples
##    58 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17659, 17662, 17659, 17659, 17660, 17658, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9978090  0.9972285
##   32    0.9993886  0.9992267
##   62    0.9985732  0.9981953
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 32.
```

From the three model the random forest has the highest precision so this will be the final model.

Processing the test set.

```r
testing <- testing[,!var_nas]
testing <- testing[,!var_empties]
testing <- testing[,-1]
testing$user_name <- as.factor(testing$user_name)
testing$cvtd_timestamp <- as.POSIXct(testing$cvtd_timestamp, format = "%d/%m/%Y %H:%M")
testing$new_window <- as.factor(testing$new_window)

for(i in c(2, 3, 6:58 )){
        testing[,i] <- as.numeric(testing[,i])
}
```

Now we can predict in the test set.

```r
predict(model_rf, testing)
```

```
##  [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
```