Project Course

Presto

15 de noviembre de 2021

Predicting classe of excercise

Using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, which they were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Being the correct way the class A and classes B to E different incorrect ways of perform barbell lifts. With a dataset of 19622 observations and 160 variables I used the Caret package and tried different prediction methods until I found one that predict correctly 99.5 percent of the observations in the evaluation.

First I set the working directory and read the train and the test datasets and the Caret package

```
setwd("~/Curso/Practical Machine Learning")
train_set<-read.csv("pml-training.csv")
test_set<-read.csv("pml-testing.csv")
library(caret)
```

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

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

```
table(train_set$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Preprocessing

I realize that there are a lot of variables with missing values in the train dataset so I ran a table of missing values percentage.

```
perc_na<-sapply(train_set, function(x) sum(is.na (x)))/nrow(train_set)
table(round(perc_na))
```

```
##
##  0  1
## 93 67
```

Once I dropped the variables with missing values there are still variables with no information , so I run a zero variance control so I can drop variables without implications for the prediction model, I used the nearZeroVar function in the Caret package.

```
vector_names<- perc_na==0
train_set<-train_set[,vector_names]
test_set<-test_set[,vector_names]
nzv <- nearZeroVar(train_set)
nzv
```

```
##  [1]  6 12 13 14 15 16 17 18 19 20 43 44 45 46 47 48 52 53 54 55 56 57 58 59 60
## [26] 74 75 76 77 78 79 80 81 82
```

```
train_set<-train_set[,-nzv]
test_set<-test_set[,-nzv]
```

Then I drop the first six variables that are ID variables and not really a measure.

```
train_set<-train_set[,-c(1:6)]
test_set<-test_set[,-c(1:6)]
```

Next I check for variables with a lot of absolute correlation between themselves using the findCorrelation function from the Caret package. Leaving the response variable out I found 4 variables with absolute correlation greater than 0.95 and I drop them from both datasets.

```
descrCor <-  cor(train_set[,-53])
highlyCorDescr <- findCorrelation(descrCor, cutoff = .95)
train_set2 <- train_set[,-highlyCorDescr]
test_set2 <- test_set[,-highlyCorDescr]
```

Finally I standarize the predictors using the preProcess function applied to the train set. With that scaling I standarize both the train and the test datasets.

```
pp=preProcess(train_set2, method = c("center","scale"))
train_set2<-predict(pp, train_set2)
test_set2<-predict(pp, test_set2)
```

Training models

Once I had the data preprocessed I trained different models for classification, and checked the accuracy of the clasification in the validation set. I need to select the training control of the train sample, I use cross validation leaving a 25% of the sample for validation and used 10 different folds each time. The training is done with the 75% of the sample and is evaluated on the remaining 25% as if that were the test dataset so it gives an idea of the out of sample error.

```
set.seed(1111)
trcontrol<-trainControl(method = "cv",
                        number = 10,
                        p=0.75)
```

Now with that estimation strategy, I train 5 models, a bayesian generalized lineal model, a classification tree, a linear discriminant analysis, a regularized discriminant analysis and a random forest.

```
glm=train(data=train_set2,
          classe~.,
          method="bayesglm",
          trControl=trcontrol)
glm
```

```
## Bayesian Generalized Linear Model
##
## 19622 samples
##    48 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17660, 17659, 17659, 17661, 17660, 17659, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.3948115  0.2262212
```

```
cart=train(data=train_set2,
         classe~.,
         method="rpart",
         trControl=trcontrol)
cart
```

```
## CART
##
## 19622 samples
##    48 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17661, 17661, 17659, 17659, 17661, 17659, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02983905  0.5071864  0.3568734
##   0.03240279  0.4781383  0.3177892
##   0.06612306  0.4145387  0.2150731
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.02983905.
```

```
lasso=train(data=train_set2,
            classe~.,
            method="sparseLDA",
            trControl=trcontrol)
lasso
```

```
## Sparse Linear Discriminant Analysis
##
## 19622 samples
##    48 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17660, 17660, 17659, 17660, 17659, 17660, ...
## Resampling results across tuning parameters:
##
##   NumVars  lambda  Accuracy   Kappa
##    2       0e+00   0.4811940  0.3360718
##    2       1e-04   0.4861377  0.3427061
##    2       1e-01   0.4871578  0.3441521
##   25       0e+00   0.6099280  0.5074019
##   25       1e-04   0.6062588  0.5028675
##   25       1e-01   0.6133929  0.5118400
##   48       0e+00   0.6843328  0.6002246
##   48       1e-04   0.6843328  0.6002246
##   48       1e-01   0.6843328  0.6002246
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were NumVars = 48 and lambda = 0.1.
```

```
lda=train(data=train_set2,
            classe~.,
            method="lda",
            trControl=trcontrol)
lda
```

```
## Linear Discriminant Analysis
##
## 19622 samples
##    48 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17660, 17658, 17660, 17660, 17659, 17660, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6847922  0.6008304
```

```
rf=train(data=train_set2,
         classe~.,
         method="rf",
         trControl=trcontrol)
rf
```

```
## Random Forest
##
## 19622 samples
##    48 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17660, 17659, 17660, 17660, 17661, 17660, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9943432  0.9928441
##   25    0.9955662  0.9943914
##   48    0.9903170  0.9877508
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 25.
```

The accuracy starts below expected in the glm method, it rises to 50% in the classification tree, and to around 68% in the discriminant analyisis, is not very different from the regularized discriminant because I already dropped the variables with high autocorrelation, then rises up to 99.5% in the random forest. I stopped after this method because there isn't much gains to keep fitting models when you have an accuracy of 99.5%.

Finally I predict the classe of the test dataset applying the predict function to the random forest

```
prf=predict(rf, test_set2)
prf
```

```
##  [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
```