PracticalMachineLearningCourseProject

Vicente Hernandez

10/5/2020

Data acquisition and cleaning

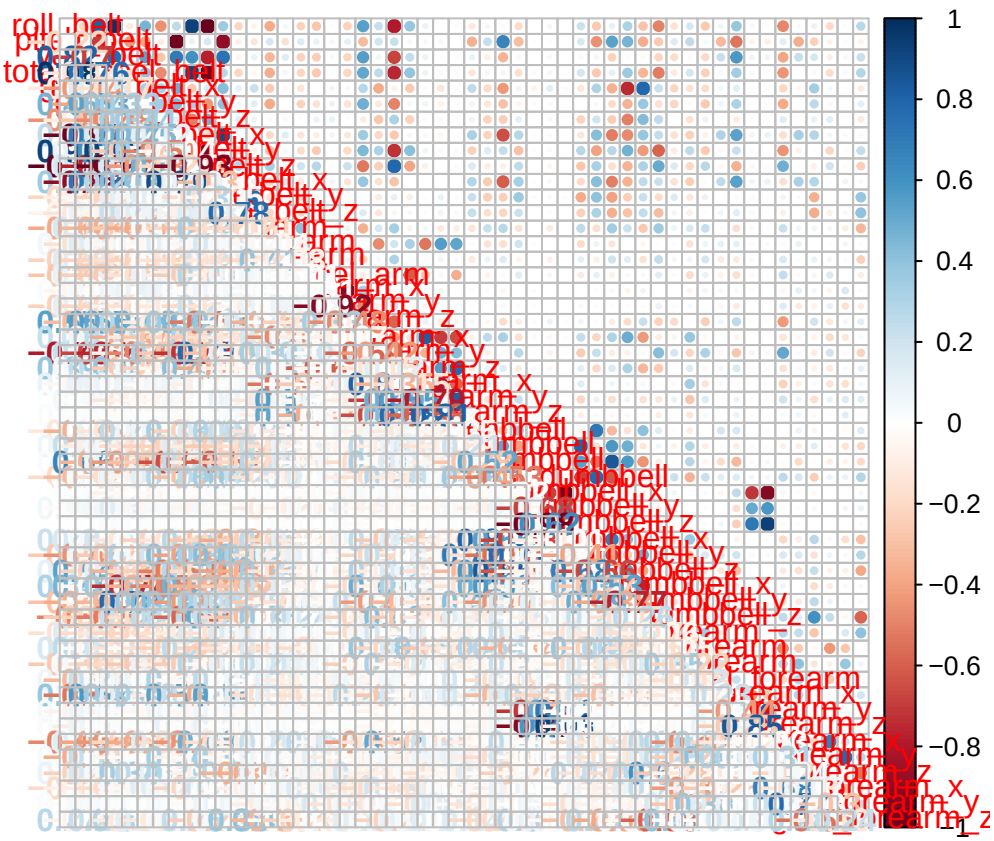
First it is needed to load the data and the libraries necessaries for the analysis. Then it begins the process to clean the data. First we eliminate the variables with more than 30% of observations been NA's. This is because there is not much that can be to done to fix a variable with so many NA's. In this case the variables that are eliminated have more than 90% of NA's values.Then unnecessary variables are eliminated, like Id variables, subject's name, etc. It is important later to check for redundancies in the data. This is when variables can be explained through others variables, in other words, variables that are highly correlated.

```
train <- read.csv("pml-training.csv",sep = ",",header = TRUE,na.strings=c("NA","#DIV/0!",""))
test <- read.csv("pml-testing.csv",sep = ",",header = TRUE,na.strings=c("NA","#DIV/0!",""))
```

```
train<-subset(train,select=-c(X,user_name,raw_timestamp_part_1,
                              raw_timestamp_part_2,cvtd_timestamp,
                              new_window,num_window))
test<-subset(test,select=-c(X,user_name,raw_timestamp_part_1,
                            raw_timestamp_part_2,cvtd_timestamp,
                            new_window,num_window))

train<-train[, colSums(is.na(train)) < 5887]
test<-test[, colSums(is.na(test)) <7]
```

```
CorrMat<-cor(subset(train,select = -c(classe)))
corrplot.mixed(CorrMat)
```



There are many variables that are highly correlated to other variables. Variables with a correlation of 0.9 or higher are eliminated, because this variables are not needed to explain the dependent variable, since other independent variables can explain the ones that are going to be eliminated.

```
train<-subset(train,select=-c(roll_belt,pitch_belt,total_accel_belt,
                              accel_belt_y,gyros_arm_x,
                              gyros_dumbbell_x,gyros_forearm_z))
test<-subset(test,select=-c(roll_belt,pitch_belt,total_accel_belt,
                              accel_belt_y,gyros_arm_x,
                              gyros_dumbbell_x,gyros_forearm_z))
```

Then we check the variance of the variables. Since variables with near zero variance are not going to contribute too much explanation to the model.

```
nzv <- nearZeroVar(train,saveMetrics=TRUE)
zero.var.ind <- sum(nzv$nzv)

if ((zero.var.ind>0))
  {
    train <- train[,nzv$nzv==FALSE]
}
```

There are no variables with near zero variance, meaning that there is no need to eliminate any variables because of this.

Data Partition

Before we train the model, it is needed to partition the data into the training data and the validation data.

```
train$classe<-factor(train$classe)
set.seed(1)
in.training <- createDataPartition(train$classe, p=0.70, list=F)
train <- train[in.training, ]
validation <- train[-in.training, ]
```

Predition model

The model that is going to be used is random forest. This is because is a robust model, it does not require many parameters, it has a trade-off of interpretability for predictive power compared to other models and in this case there is no need for interpretation of the variables.

```
ctrl <- trainControl(method="cv", 5)
set.seed(1)
rf <- train(classe ~ ., data=train, method="rf",trControl=ctrl, ntree=200)
rf
```

```
## Random Forest
##
## 13737 samples
##    45 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10990, 10989, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9882071  0.9850801
##   23    0.9890083  0.9860933
##   45    0.9844219  0.9802902
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 23.
```

The random forest model has a 0.989 accuracy when the trees are build using 23 variables of the 45 independent variables.

Validation performance

```
rf.predict <- predict(rf, validation)
confusionMatrix(validation$classe, rf.predict)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1163    0    0    0    0
##          B    0  807    0    0    0
##          C    0    0  722    0    0
##          D    0    0    0  667    0
##          E    0    0    0    0  768
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2818
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2818   0.1955   0.1749   0.1616   0.1861
## Detection Rate         0.2818   0.1955   0.1749   0.1616   0.1861
## Detection Prevalence   0.2818   0.1955   0.1749   0.1616   0.1861
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

The model it has an accuracy of 1 in the validation data, this means that the model was able to correctly assign every observation of the validation set. This is odd, considering that the model has a 0.989 accuracy in the training set, but it is certainly not impossible.

Prediction

Finally the model is used in the test set to make the final prediction.

```
results <- predict(rf, test[, -length(names(test))])
results
```

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