ML project

Tsygankov Nickita

05 06 2018

Overview

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

Aims

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Preparation and reproducibility procuring

First of all we should load the data. If u haven't got one of this packages, please, install it before starting.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

```
library(rpart)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(127) ## seed will guarantee reproducibility
```

Cleaning and exploring the data

Our data have a problem wih NA's - they exist in diffrent formats, and we need exclude this problem on the stage of loading data

```
training <- read.csv("pml-training.csv", na.strings=c("", "NA", "#DIV/0!"), row.names = 1)
testing <- read.csv("pml-testing.csv", na.strings=c("", "NA", "#DIV/0!"), row.names = 1)
training <- training[,!sapply(training,function(x) any(is.na(x)))]
testing <- testing[,!sapply(testing,function(x) any(is.na(x)))]
```

Several first columns contain information, which would be not useful for our analysis and can create noise for ML instruments. We should exclude them too.

```
training <-training[,-c(1:6)]
testing <-testing[,-c(1:6)]
```

Data spliting and training

We need cut our data for training and validation parts

```
indices <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
Train <- training[indices, ]
Validation <- training[-indices, ]
```

We have a large data set with about 20000 observation. This means that with high probability random forests should demonstrate best results.

```
model_forests <- randomForest(classe ~ ., data = Train)
```

Prediction

Let's test our prediction model!

```
pred_for <- predict(model_forests, Validation, type = "class")
confusionMatrix(pred_for, Validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  947    6    0    0
##          C    0    2  847    4    0
##          D    0    0    2  798    4
##          E    0    0    0    2  897
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9937, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9906   0.9925   0.9956
## Specificity            1.0000   0.9985   0.9985   0.9985   0.9995
## Pos Pred Value         1.0000   0.9937   0.9930   0.9925   0.9978
## Neg Pred Value         1.0000   0.9995   0.9980   0.9985   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1931   0.1727   0.1627   0.1829
## Detection Prevalence   0.2845   0.1943   0.1739   0.1639   0.1833
## Balanced Accuracy      1.0000   0.9982   0.9946   0.9955   0.9975
```

We got a few missing values, this fact means that our model works great! No we need to implement compare with confusion matrix of our first data part.

```
pred_for_train <- predict(model_forests, Train, type = "class")
confusionMatrix(pred_for_train, Train$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     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             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

efreshingly, random forests does even better on the training data, as it should. Note that if we keep all variables, we get even better performance and appearance of good predictions on the validation data, but can't predict on the test data when we get names of new people. So we've overfit to the names of the people. The performance is still quite good, though.

Given this, we expect near perfect accuracy on the test data (validation data in the data science specialization convention).

```
pred_for_test <- predict(model_forests, testing, type = "class")
pred_for_test
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

The result supprased our expectations and we not need to use another models and instruments.

Conclusion

Random forests predictably show a great perfomance on big data set and we got an exellent results with testing data. Thanks for attention!