project week 4

Gjeffroy

May 17, 2017

Preparing the data

Loading and preparing dataset

Here we load the training (19622 observations) and testing data set (20 observations).

```r
training_messy <- read.csv("pml-training.csv")
testing_messy <- read.csv("pml-testing.csv")
dim(training_messy)
```

```
## [1] 19622   160
```

Each dataset hold 160 variables but many columns contain only NA's so lets start by removing all the empty columns.

```r
rm_empty_col<- function(df){
  j<- 1
  for(i in 1:160){
    if(sum(!is.na(df[,j]))/nrow(df) < 0.2)
       df <- df[,-j]
    else{j <- j +1}
  }
  df
}

training_messy[training_messy ==""]<- NA
training<- rm_empty_col(training_messy)
testing<- rm_empty_col(testing_messy)
```

We have now two clean datasets with the same variables (except the classe variable only in training and problem_id only in testing) so we can start thinking of a predication model.

Dividing training to get a validation set

In order to check our model, we divide the training set into two (training and validation set)

```r
inTrainingSet <- createDataPartition(training$classe, p = 0.7, list =F)
training_set <- training[inTrainingSet,]
validation_set <- training[-inTrainingSet,]
```

Training Random forest model

tuning model using caret::train()

We now train a random forest model using the caret::train() function. The parameters are low to avoid long time processing as the training dataset contains 13737 row and 60 columns.

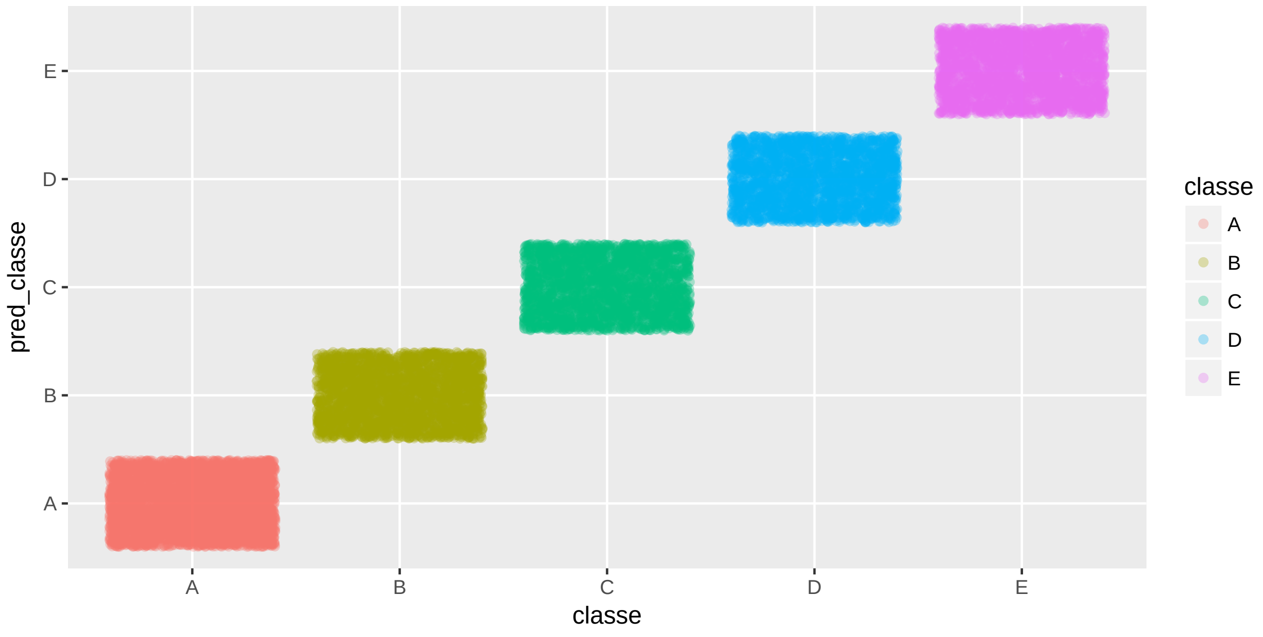
```
set.seed(123)
tunegrid <- expand.grid(.mtry=c(2))
control <- trainControl(method="repeatedcv", number=2, repeats=1, search="grid")
mod_RF <- train(classe ~ ., data = training_set,
                method = "rf",
                tuneGrid=tunegrid,
                trControl=control)
mod_RF
```

```
## Random Forest
##
## 13737 samples
##    59 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 1 times)
## Summary of sample sizes: 6868, 6869
## Resampling results:
##
##   Accuracy   Kappa
##   0.9905365  0.9880289
##
## Tuning parameter 'mtry' was held constant at a value of 2
```

good new! With only theses very simple parameters, the model give an accurancy of 0.992.So there is no need to increase k-fold(2 fold, 1 rep), or metry(2).

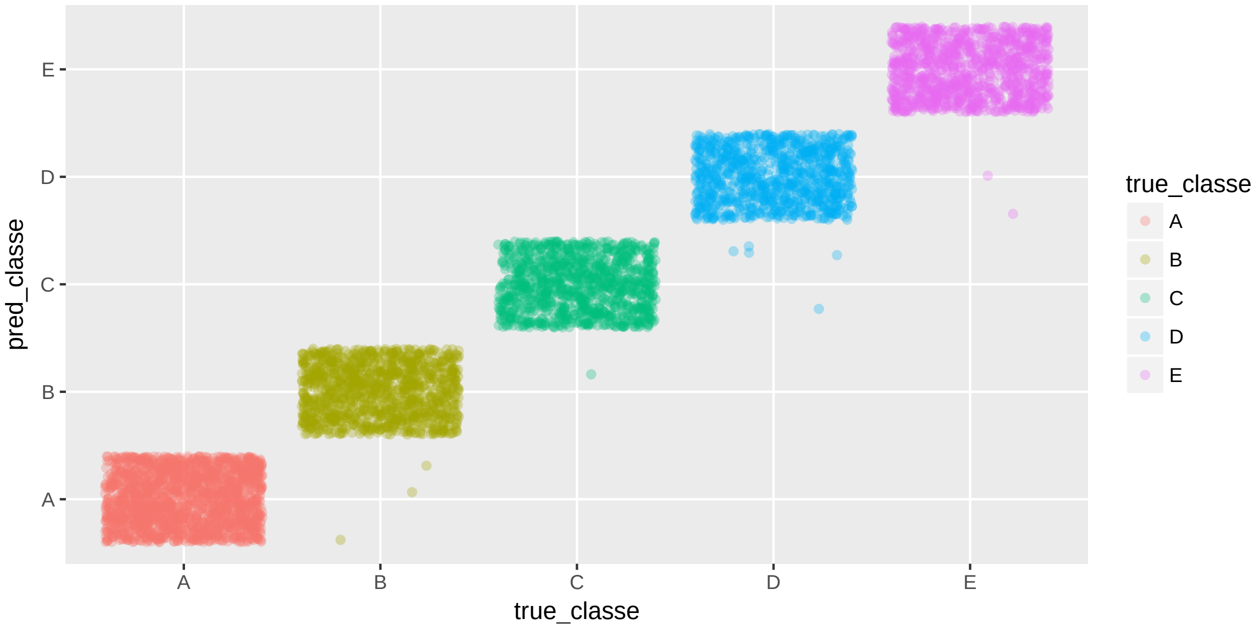
Plotting the predicted classes vs true classes

The model give almost 100% accuracy on the training set.



test mod_RF on the validation set

Here we evaluate the model using the validation set.



Looking at this plot, we see that very few observations got misclassified. Lets look at the confusion matrix to get clear figures.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    3 1136    0    0    0
##          C    0    1 1025    0    0
##          D    0    0    5  959    0
##          E    0    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9991   0.9951   0.9979   1.0000
## Specificity            1.0000   0.9994   0.9998   0.9990   0.9996
## Pos Pred Value         1.0000   0.9974   0.9990   0.9948   0.9982
## Neg Pred Value         0.9993   0.9998   0.9990   0.9996   1.0000
## Prevalence             0.2850   0.1932   0.1750   0.1633   0.1835
## Detection Rate         0.2845   0.1930   0.1742   0.1630   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9991   0.9992   0.9975   0.9985   0.9998
```

The random forest model does a very good job on the validation dataset.

- the accuracy is above 99.5%
- sensitivities for all classe are above 99%
- specificities for all classes are above 99%

Which means the model correctly predict the classe using our predictors. It is actually quite surprinsing how reliable this model is, probably due to the high number of predictor (59 predictors) for five classes only.

We can assume also that we are dealing with a high quality dataset containing that on top of having collected accurate measurement, have many observation for every class. That would explain why we get both excellent sensitivity and specificity.

Predicting classe on test set

Finally we predict the class for the test set. Results are shown below.

```r
test_RF <- predict(mod_RF, testing)
knitr::kable(data.frame(testing[,1:2],test_RF))
```

X	user_name	test_RF
1	pedro	B
2	jeremy	A
3	jeremy	B
4	adelmo	A
5	eurico	A
6	jeremy	E
7	jeremy	D
8	jeremy	B
9	carlitos	A
10	charles	A
11	carlitos	B
12	jeremy	C
13	eurico	B
14	jeremy	A
15	jeremy	E
16	eurico	E
17	pedro	A
18	carlitos	B
19	pedro	B
20	eurico	B