challenge B final

Maxime TRANCHARD , Emma GAILLAT

7 décembre 2017

Task 1B - Predicting house prices in Ames, Iowa

```
install.packages("tidyverse",repos = "http://cran.us.r-project.org")
install.packages("readr",repos = "http://cran.us.r-project.org")
install.packages("randomForest",repos = "http://cran.us.r-project.org")
install.packages("np",repos = "http://cran.us.r-project.org")
install.packages("data.table",repos = "http://cran.us.r-project.org")
library(randomForest)
library(data.table)
library(tidyverse)
library(readr)
library(dplyr)
library(np)
library(readxl)
```

STEP 1) we choose to run a randomForest prediction. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. This is better with a lot of observations. Here it's the case so we assume it suits.

STEP 2)

```
train<- read.csv("train.csv")
Test<- read.csv(file = "test.csv")
```

```
na_count<-sapply(Train, function(y) sum(length(which(is.na(y)))))
na_count
```

We remove the columns where there are a lot of missing variables, and only after that we remove the rows where there are missing variables.

```
Train$PoolQC <- NULL
Train$Fence <- NULL
Train$MiscFeature <- NULL
Train$Alley <- NULL
Train$FireplaceQu <- NULL
Train$LotFrontage <- NULL
Train<- na.omit(Train)
```

Creating our prediction with RandomForest

```
Train <-Train%>%mutate_if(is.character,as.factor)
names(Train)<-make.names(names(Train))
set.seed(1)
Train.fit<-randomForest(Train$SalePrice~., data=Train)
Predict.RandomForest<-predict(Train.fit,data=Test)
```

Step 3) Let's compare with a OLS

```
Train.lm<-lm(data=Train,SalePrice~.)
predict.lr<-predict(print(Train.lm), data=Test)
```

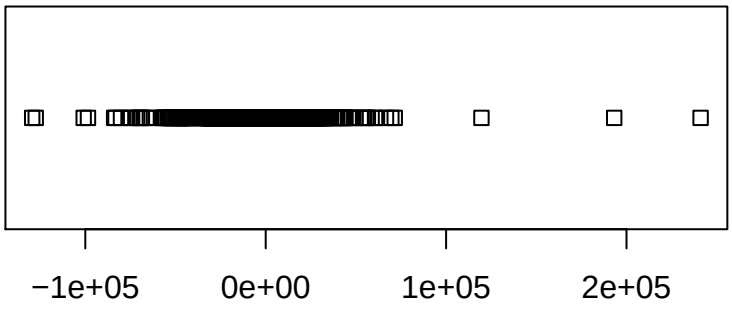
```
summary(Predict.RandomForest)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   87559  138234  170718  186451  216638  560225
```

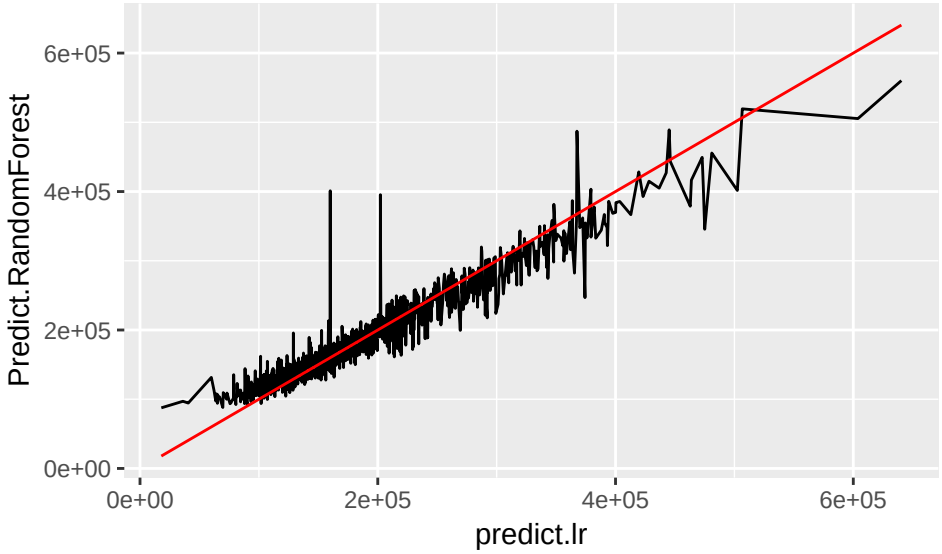
```
summary(predict.lr)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   17911  133717  169868  186762  220056  640453
```

```
diff<-as.data.frame(Predict.RandomForest-predict.lr)
plot(diff)
```




```
ggplot()+geom_line(aes(predict.lr,Predict.RandomForest))+geom_line(aes(predict.lr,predict.lr),col="red")
```



Even if we can think the difference are really big, when we focus and the mean (where the ols is good), both are pretty similar moreover regarding the variations of the observations a variation of 10^4 is not that important for a house price.

TASK 2B - Overfitting in Machine Learning remember first what we have done in Part A

```
set.seed(1)
xpaul <- 2
ypaul <- xpaul^3
x <- rnorm(n = 150)
e <- rnorm(n=150)
y <- x^3 + e
draws <- data.frame(y,x)
y <- function(x){x^3}
draws$idx <- sample(seq(FALSE, TRUE), size = nrow(draws), replace = TRUE, prob = c(.8, .2))
train <- draws[draws$idx == FALSE,]
test <- draws[draws$idx == TRUE,]
y_true<-train$x^3
```

We estimate a low-flexibility and a high-flexibility local linear model on the training data using the function npreg. STEP 1)

```
ll.fit.lowflex <- npreg(y ~ x, data = train, method = "ll", bws = 0.5)
summary(ll.fit.lowflex)
```

STEP 2)

```
ll.fit.highflex <- npreg(y ~ x, data = train, method = "ll", bws = 0.01)
summary(ll.fit.highflex)
```

STEP 3)

```
#First, we compute the predictions for ll.fit.highflex and ll.fit.lowflex
fitted.highflex <- fitted(ll.fit.highflex)
fitted.lowflex <- fitted(ll.fit.lowflex)

#And now, we are able to plot.
graph1 <- ggplot() + geom_point(data=train, mapping = aes(x = x, y = y)) + geom_line(data=train, mapping
graph1
```

STEP 4) when we look at the graph we see that the model ll.fit.highflex is the prediction most variable, which has the lowest bandwidth. Also, we can compute the variance of each prediction and check this statement.

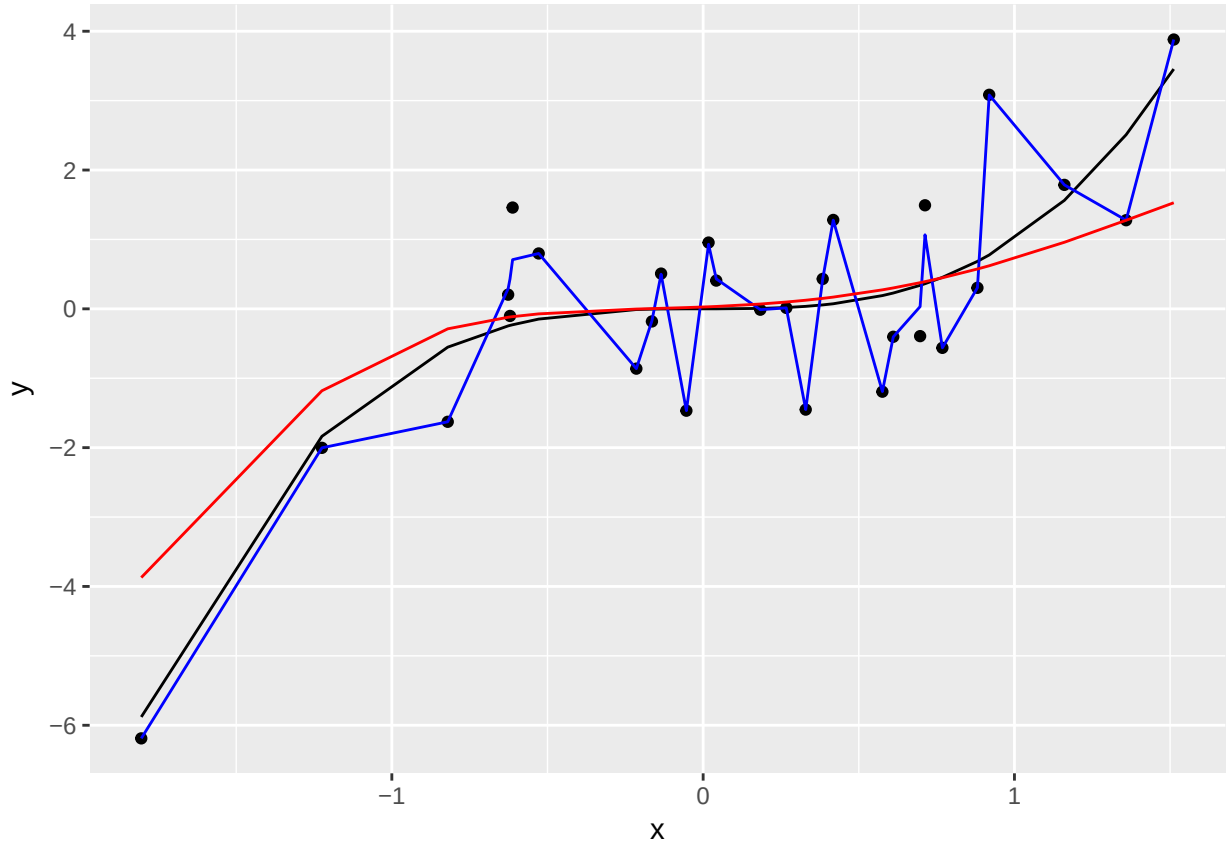
```
variance <- cbind(var(y_true), var(fitted.highflex), var(fitted.lowflex))
colnames(variance) <- c("true model","highflex","lowflex")
variance
bias <- cbind(mean(y_true-train$y), mean(fitted.highflex-train$y), mean(fitted.lowflex-train$y))
colnames(bias) <- c("true model","highflex","lowflex")
abs(bias)
```

thus the highflex model is the one with less biais.

STEP 5) same as before but with Data = test

```
ll.fit.lowflex.test <- npreg(y ~ x, data = test, method = "ll", bws = 0.5)
summary(ll.fit.lowflex.test)
ll.fit.highflex.test <- npreg(y ~ x, data = test, method = "ll", bws = 0.01)
summary(ll.fit.highflex.test)
y_true.test <- test$x^3
```

```
fitted.highflex.test <- fitted(ll.fit.highflex.test)
fitted.lowflex.test <- fitted(ll.fit.lowflex.test)

graph2 <- ggplot() +
  geom_point(data=test, mapping = aes(x = x, y = y)) +
  geom_line(data=test, mapping = aes(x = x, y = y_true.test)) +
  geom_line(data=test, mapping = aes(x = x, y = fitted.highflex.test), color = "blue") +
  geom_line(data=test, mapping = aes(x = x, y = fitted.lowflex.test), color = "red")
graph2
```




```
variance.test <- cbind(var(y_true.test), var(fitted.highflex.test), var(fitted.lowflex.test))
colnames(variance.test) <- c("true model","highflex","lowflex")

variance.test
bias.test <- cbind(mean(y_true.test-test$y), mean(fitted.highflex.test-test$y), mean(fitted.lowflex.tes
colnames(bias.test) <- c("true model","highflex","lowflex")

abs(bias.test)
```

highflex remain the best but the bias rise.

STEP 6) Creating a vector

```
bw <- seq(0.01, 0.5, by = 0.001)
```

STEP 7) Estimate a local linear model

```
vector.fit<-lapply(X = bw, FUN = function(bw) {npreg(y ~ x, data = train, method = "ll", bws = bw)})
```

STEP 8) we compute the MSE

```
train.mse <- function(fit.model){
predictions <- predict(object = fit.model, newdata = train)
train %>% mutate(squared.error = (y - predictions)^2) %>% summarize(mse = mean(squared.error))}
train.mse.output <- unlist(lapply(X = vector.fit, FUN = train.mse))
```

STEP 9) Lets do the same for the test Data.

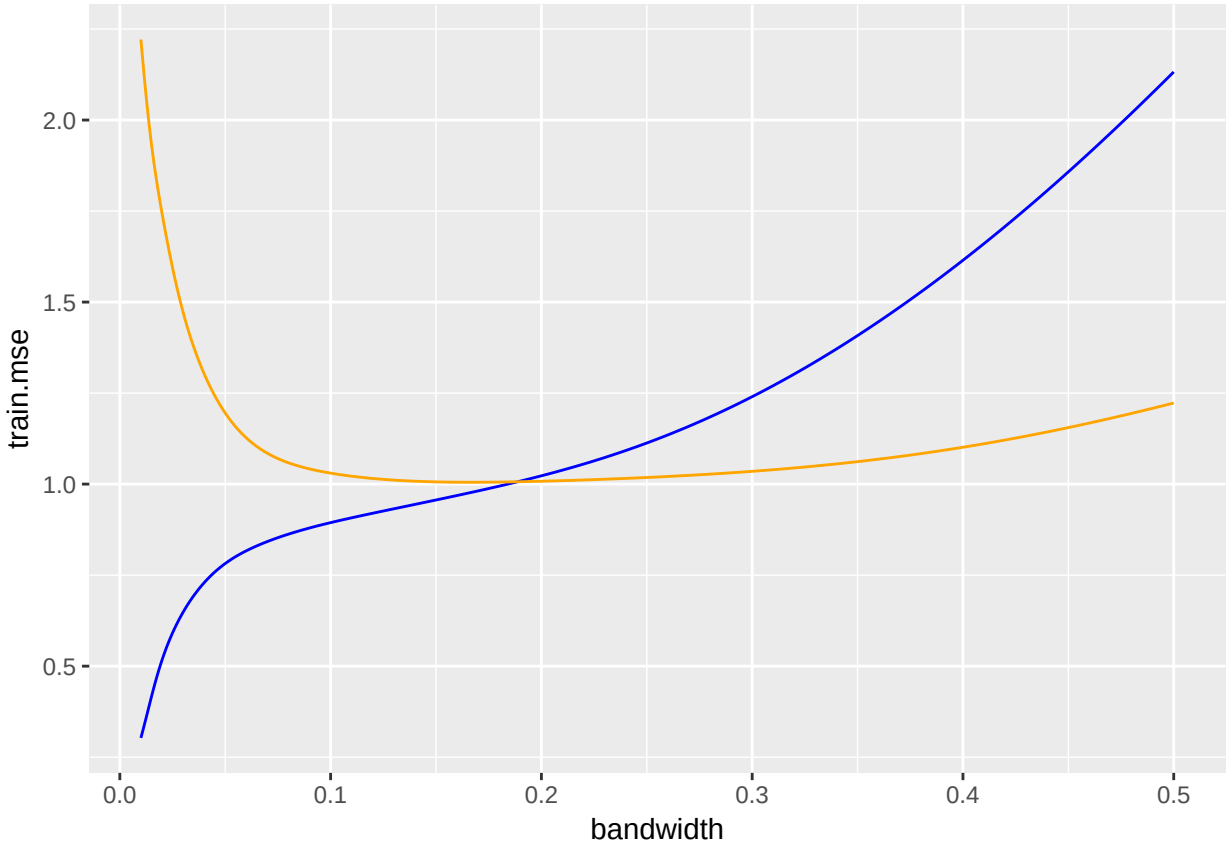
```
test.mse <- function(model.fit){
estimations <- predict(object = model.fit, newdata = test)
test %>% mutate(squared.error = (y - estimations)^2) %>% summarize(mse = mean(squared.error))
}
test.mse.output <- unlist(lapply(X = vector.fit, FUN = test.mse))
```

Step 10)

```
mse.table <- tbl_df(data.frame(bandwidth = bw, train.mse = train.mse.output, test.mse = test.mse.output)
mse.table
```

```
## # A tibble: 491 x 3
##    bandwidth train.mse test.mse
##        <dbl>     <dbl>    <dbl>
##  1     0.010 0.3028239 2.221189
##  2     0.011 0.3246634 2.156244
##  3     0.012 0.3472304 2.093529
##  4     0.013 0.3702386 2.035158
##  5     0.014 0.3933220 1.981847
##  6     0.015 0.4161016 1.933516
##  7     0.016 0.4382402 1.889650
##  8     0.017 0.4594798 1.849519
##  9     0.018 0.4796564 1.812336
## 10     0.019 0.4986945 1.777388
## # ... with 481 more rows
```

```
ggplot(mse.table) +
geom_line(mapping = aes(x = bandwidth, y = train.mse), color = "blue") +
geom_line(mapping = aes(x = bandwidth, y = test.mse), color = "orange")
```



0.2 is the point where we minimise our MSE for the test Data; No interest to go for a smaller one an decrease the MSE of the training data.

TASK 3 - PRIVACY REGULATION COMPLIANCE IN FRANCE

STEP 1) Let's import our DATA.

```
library(readr)
library(readxl)
CNIL<-cil <- read.csv("https://www.data.gouv.fr/s/resources/correspondants-informatique-et-libertes-cil/
SIREN <- read_delim("sirc-17804_9075_14211_2017340_E_Q_20171207_022339046.csv",
                                                                    ";", escape_double = FALSE, trim_ws =
```

STEP 2)

```
uniq.cil<-unique(cil1[c("SIREN","Dep")])
head(uniq.cil)
```

```
##       SIREN Dep
## 1 788349926  49
## 2 421715731  66
## 3 409869708  69
## 4 444600464  92
## 5 922002968  92
## 6 429621311  75
```

```
table.uniq<- table(unlist(duplicated(uniq.cil$SIREN)))
table.uniq
```

```
##
```

```
## FALSE  TRUE
## 17667   238
```

238 companies dont have a unique cil-responsible per department

STEP 3)

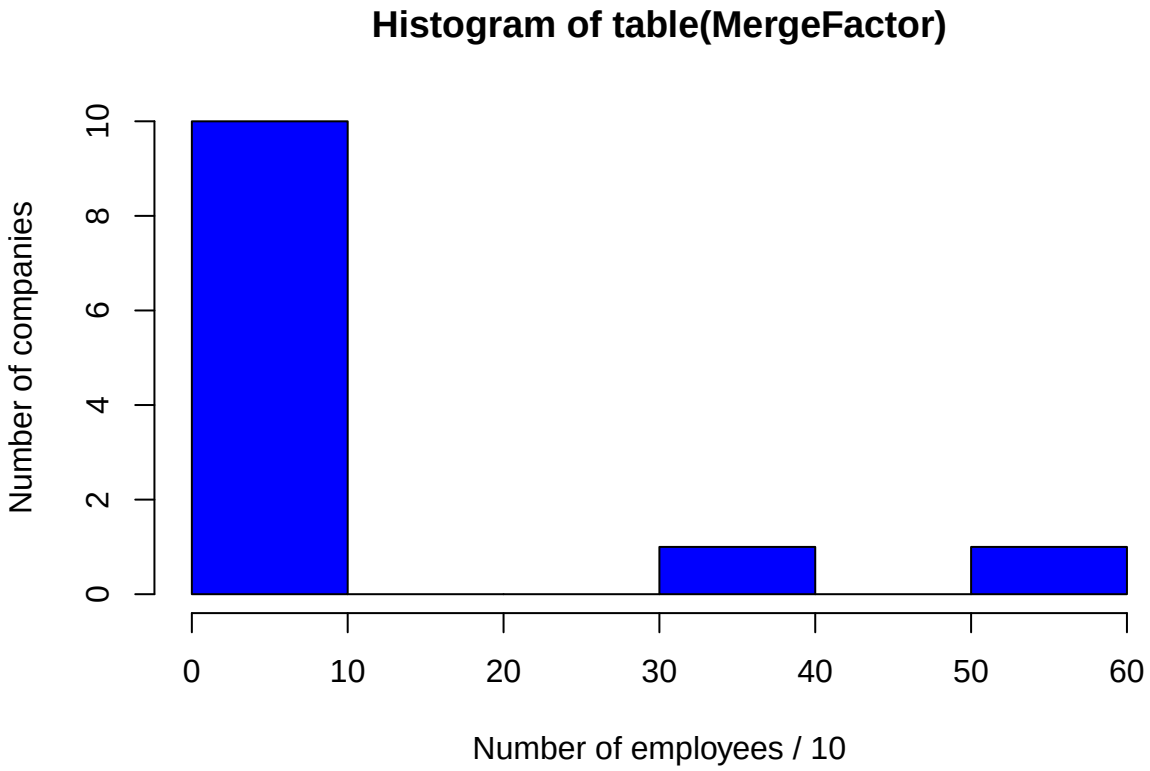
```
cil2<-setDT(cil1)

c<-as.character(cil2$SIREN)
cil3<-cbind(cil2,c)
colnames(cil3)<-c("SIREN1","Responsable","Adresse",
                  "Code_Postal","Ville","NAF",
                  "TypeCIL","Portee","Dep","SIREN")
Merge<-merge.data.frame(cil3,SIREN, by="SIREN")
```

STEP 4)

```
summary(Merge$TEFEN)
```

```
##    Length     Class      Mode
##       150 character character
```

```
MergeFactor <- as.factor(Merge$TEFEN)
hist(table(MergeFactor), freq=TRUE, ylab = "Number of companies", xlab="Number of employees / 10", col =
```



Barely all the companies who nominate a CIL were with less than 100 workers.