Predicting PM10 concentration in Le Havre, France

Describing the data set

Loading the data set:

```
library(VSURF)
A = VSURF::rep
str(A)
```

```
## 'data.frame':    1096 obs. of  18 variables:
##  $ PM10     : int  13 22 29 25 23 17 20 17 32 31 ...
##  $ NO       : int  15 25 31 36 72 53 31 28 47 44 ...
##  $ NO2      : int  25 32 42 36 56 54 45 41 47 50 ...
##  $ SO2      : int  5 3 10 4 15 NA 14 13 3 10 ...
##  $ T.min    : num  0.6 -0.6 -2 1.9 5.7 NA 6.3 7.4 7 7.3 ...
##  $ T.max    : num  8.4 7.6 1.2 6.5 8 NA 8.6 10.9 9.8 10.8 ...
##  $ T.moy    : num  5.025 3.333 -0.583 4.675 7.238 ...
##  $ DV.maxvv : int  210 330 190 270 220 NA 190 190 250 220 ...
##  $ DV.dom   : num  338 45 180 225 NA ...
##  $ VV.max   : int  16 8 6 7 9 NA 11 22 14 8 ...
##  $ VV.moy   : num  9.17 4.43 3.42 3.92 6.08 ...
##  $ PL.som   : int  19 5 0 0 0 0 0 4 1 0 ...
##  $ HR.min   : int  79 81 74 86 94 NA 76 75 85 89 ...
##  $ HR.max   : int  99 93 90 99 97 NA 92 95 95 97 ...
##  $ HR.moy   : num  90.3 85.7 83 96.4 95.2 ...
##  $ PA.moy   : num  1010 1018 1024 1022 1021 ...
##  $ GTrouen  : num  0.7 -0.5 2.4 2.7 2.4 0.6 -0.1 0 -0.1 1.2 ...
##  $ GTlehavre: num  -0.3 0 0.1 -0.2 -0.2 -0.3 -0.5 NA -0.5 0.1 ...
```

According to the data set description, PM10 concentrations were measured in Le Havre, France, by Air Normand, with the associated weather data provided by Meteo France, from 2004 to 2006. The monitoring station is situated in an urban site, close to traffic, and considered the most polluted in the region. We have **1096 observations.**

The data set description gives the following definitions for the **18 numeric variables** in the data set:

- PM10: Daily mean concentration of PM10, in $\mu g/m^3$.
- NO, NO2, SO2: Daily mean concentration of NO, NO_2 , SO_2 in $\mu g/m^3$.
- T.min, T.max, T.moy: Daily minimum, maximum and mean temperature, in degree Celsius.
- DV.maxvv, DV.dom: Daily maximum speed and dominant wind direction in degree.
- VV.max, VV.moy: Daily maximum and mean wind speed, in m/s.
- PL.som: Daily rainfall in mm.
- HR.min, HR.max, HR.moy: Daily minimum, maximum and mean relative humidity, in %.
- PA.moy: Daily mean air pressure in hPa.

- GTrouen, GTlehavre: Daily temperature gradient in degree Celsius in the cities of Rouen and Le Havre, respectively.

PM10 is the target variable we want to predict using the other 17 variables.

We have missing data:

```
sum(is.na(A))
```

```
## [1] 269
```

There is data missing in most columns:

```
colnames(A)[ apply(A, 2, anyNA)]
```

```
##  [1] "PM10"      "NO"        "NO2"       "SO2"       "T.min"     "T.max"
##  [7] "T.moy"     "DV.maxvv"  "DV.dom"    "VV.max"    "VV.moy"    "HR.min"
## [13] "HR.max"    "HR.moy"    "PA.moy"    "GTlehavre"
```

I will substitute the missing values for the mean of each variable:

```
#Calculating the mean for each variable
mean_columns <- apply(A[,colnames(A)],2, mean, na.rm =  TRUE)

#Substituing the missing values by the mean
for (c in names(A)){
  A[,c] = ifelse(is.na(A[,c]), mean_columns[c], A[,c])
}

#Checking that there are no missing values
sum(is.na(A))
```

```
## [1] 0
```

Methodology

We are interested in Regression. For variable selection, I will use Random Forests with the VSURF package. Linear, Decision Tree and Random Forest models will be constructed to make predictions. They will be compared by calculating the Root Mean Squared Error (RSME) and the Mean Absolute Error (MAE). The data set into Learning and Test sets before modelling.

Splitting the data set

The models will be trained using the Learning set. Around 20% of the data will be kept separately on the Test set to test the models' predictions.

```
splitProb <- c(0.8,0.2)
splitNames <-c("Learning","Test")

n = nrow(x=A)

splitVector<- sample( x=splitNames, size=n, prob=splitProb, replace=TRUE )

table(splitVector)/n
```

```
## splitVector
##  Learning      Test
## 0.8120438 0.1879562
```

```r
#getting the indeces
indeces<-list(
  learning = which(x=(splitVector=="Learning")),
  test=which(x=(splitVector=="Test"))
  )
#splitting the data set
learningSet<-A[indeces$learning,]
testSet<-A[indeces$test,]
```

Linear Model

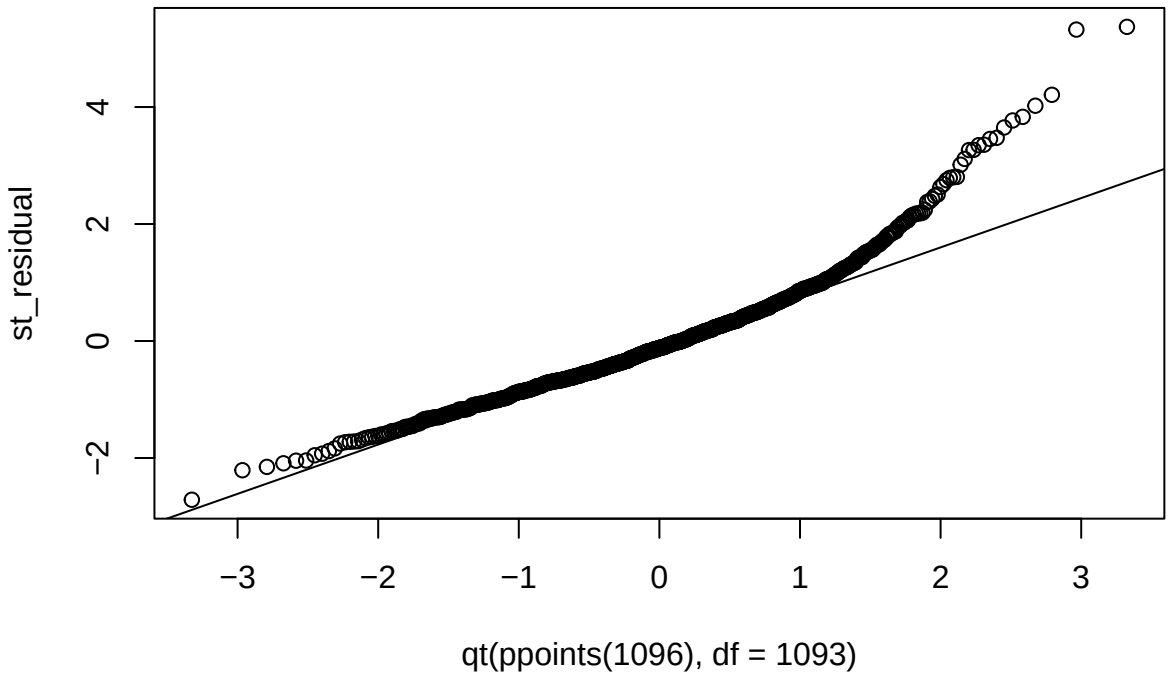
Linear model using all explanatory variables:

```r
L = lm(PM10~., data = learningSet)
summary(L)
```

```
##
## Call:
## lm(formula = PM10 ~ ., data = learningSet)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.616  -4.133  -0.773   3.076  33.121
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -79.833246  30.795470  -2.592  0.00969 **
## NO            0.113143   0.013013   8.694  < 2e-16 ***
## NO2           0.208701   0.021373   9.765  < 2e-16 ***
## SO2           0.157133   0.018243   8.613  < 2e-16 ***
## T.min        -0.126918   0.311165  -0.408  0.68346
## T.max         0.650320   0.340205   1.912  0.05626 .
## T.moy        -0.259607   0.584062  -0.444  0.65680
## DV.maxvv     -0.005132   0.002668  -1.923  0.05478 .
## DV.dom       -0.004184   0.002601  -1.608  0.10811
## VV.max       -0.334956   0.137504  -2.436  0.01505 *
## VV.moy        0.511213   0.187924   2.720  0.00665 **
## PL.som       -0.206352   0.052009  -3.968 7.86e-05 ***
## HR.min        0.079944   0.056139   1.424  0.15480
## HR.max        0.021515   0.077081   0.279  0.78021
## HR.moy       -0.075717   0.102846  -0.736  0.46179
## PA.moy        0.086342   0.029381   2.939  0.00338 **
## GTrouen      -0.204685   0.181655  -1.127  0.26015
## GTlehavre     1.098247   0.318457   3.449  0.00059 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.403 on 872 degrees of freedom
## Multiple R-squared:  0.5932, Adjusted R-squared:  0.5852
## F-statistic: 74.79 on 17 and 872 DF,  p-value: < 2.2e-16
```

To check the gaussian assumption of the noise, I will use a QQ-plot of the studentized residuals:

```r
#Studentized residuals - QQ plot
st_residual=rstudent(L)
#n=1096 #degrees of freedom n-3=1093
qqplot(qt(ppoints(1096), df=1093), st_residual)
qqline(st_residual, distribution=function(p){qt(p,df=1093)})
```



The QQ-plot is skewed and highly-tailed, so the noise does not appear to be Gaussian.

Using the Kolmogorov-Smirnov test to assess if the residuals have a student distribution.

```r
#Smirnov test
#n=1096 #degrees of freedom n-3=1093
#'pt' is for student distribution
ks.test(st_residual,'pt',1093)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  st_residual
## D = 0.073262, p-value = 0.0001419
## alternative hypothesis: two-sided
```

The p-value is very small.

The noise does not seem to have a Gaussian distribution. This means that most parameters given in the linear model summary are not meaningful. We still can look at its R^2 score, use the linear model for prediction and calculate errors, but we can't do much more.

Variable Selection

Many of the explanatory variables are highly dependent on each other.

For example, the temperature variables:

```
cor(learningSet[,c("T.min","T.max","T.moy")])
```

```
##           T.min     T.max     T.moy
## T.min 1.0000000 0.9299735 0.9769789
## T.max 0.9299735 1.0000000 0.9827103
## T.moy 0.9769789 0.9827103 1.0000000
```

Or humidity variables:

```
cor(learningSet[,c("HR.min","HR.max","HR.moy")])
```

```
##           HR.min    HR.max    HR.moy
## HR.min 1.0000000 0.6145385 0.9129258
## HR.max 0.6145385 1.0000000 0.8224679
## HR.moy 0.9129258 0.8224679 1.0000000
```

And several others.

Since we cannot make the Gaussian assumption of the noise, we cannot use the p-values given by the linear model to perform variable selection. Instead, I will perform variable selection using Random Forest to detect the most relevant variables for prediction.

```
library('VSURF')
#Three steps variable selection procedure based on random forests for
#supervised classification and regression problems.
#First step ("thresholding step") is dedicated to eliminate irrelevant
#variables from the dataset. Second step ("interpretation step")
#aims to select all variables related to the response for interpretation purpose.
#Third step ("prediction step") refines the selection by eliminating redundancy in
#the set of variables selected by the second step, for prediction purpose.
set.seed(221921186)
Vy<-VSURF(PM10~.,data=learningSet, nmj=1)
```

```
## Thresholding step
## Estimated computational time (on one core): 28.8 sec.
##   |                                                                              |
## Interpretation step (on 17 variables)
## Estimated computational time (on one core): between 5.1 sec. and  35.7 sec.
##   |                                                                              |
## Prediction step (on 13 variables)
## Maximum estimated computational time (on one core): 23.4 sec.
##   |                                                                              |
```

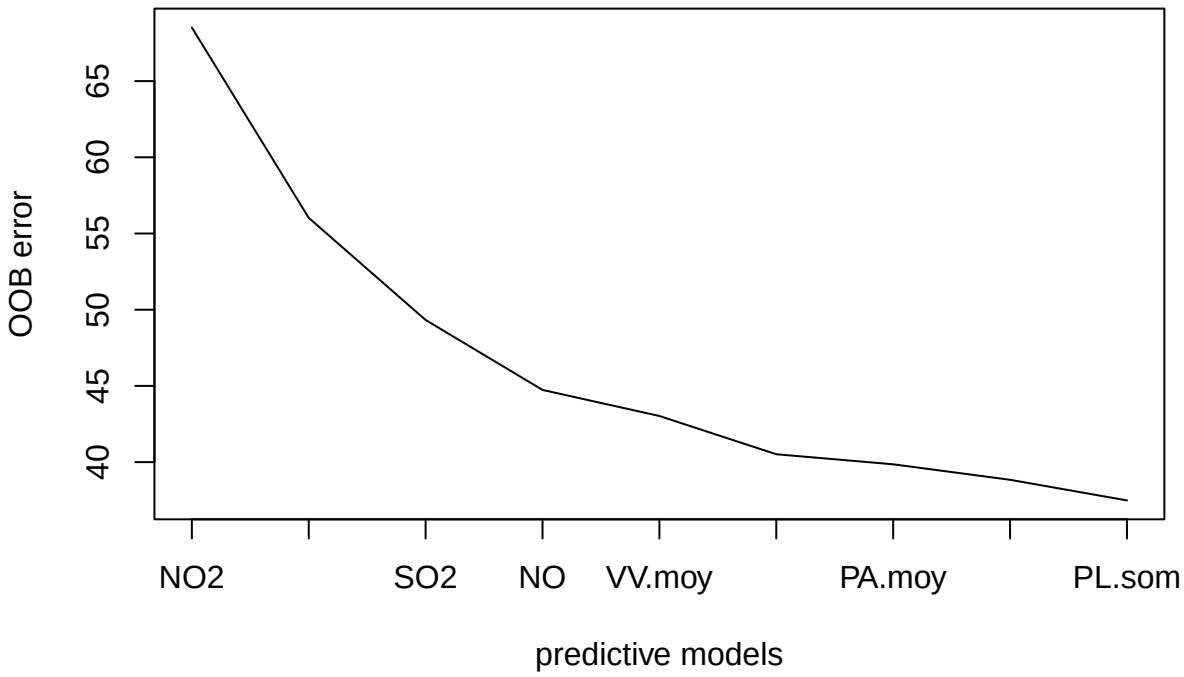
```
## Warning in VSURF.formula(PM10 ~ ., data = learningSet, nmj = 1): VSURF with a formula-type call outpu
## which are indices of the input matrix based on the formula:
## you may reorder these to get indices of the original data
```

```
summary(Vy)
```

```
##
##  VSURF computation time: 1.1 mins
##
##  VSURF selected:
```

```
##  17 variables at thresholding step (in 29.6 secs)
##  13 variables at interpretation step (in 22.8 secs)
##  9 variables at prediction step (in 11.9 secs)
```

```
plot(Vy,step="pred",var.names=TRUE)
```



The selected explanatory variables are:

```
variables = c()
for (i in Vy$varselect.pred){
  variables <- c(variables,colnames(learningSet)[i+1])
}
variables
```

```
## [1] "NO2"       "GTlehavre" "SO2"       "NO"        "VV.moy"    "T.moy"
## [7] "PA.moy"    "DV.maxvv"  "PL.som"
```

Removing other explanatory variables from Learning and Test sets:

```
learningSet= learningSet[,c(variables,"PM10")]
testSet= testSet[,c(variables,"PM10")]
```

Back to the Linear Model

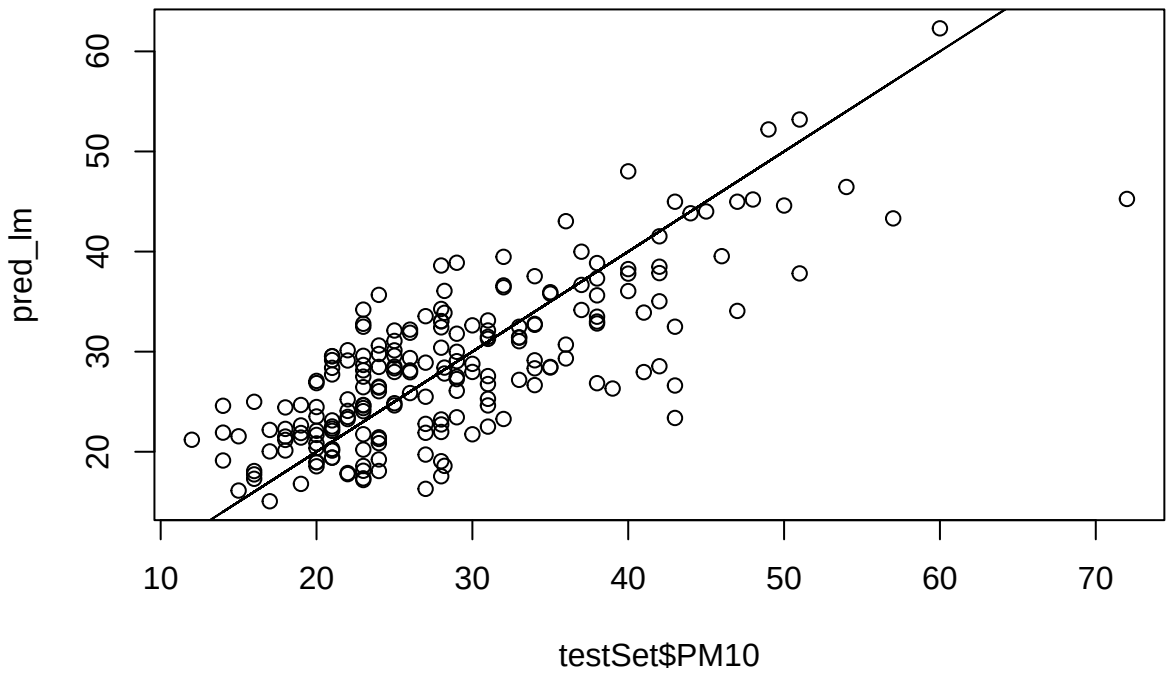
We can create a linear model using only the variables selected:

```
L2 = lm(PM10~., data = learningSet)
summary(L2)
```

```
##
## Call:
## lm(formula = PM10 ~ ., data = learningSet)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.202  -4.256  -0.898   3.131  33.682
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -87.648082  29.630822  -2.958  0.00318 **
## NO2           0.210051   0.020851  10.074  < 2e-16 ***
## GTlehavre     1.085095   0.243979   4.447 9.80e-06 ***
## SO2           0.169119   0.017495   9.667  < 2e-16 ***
## NO            0.101779   0.011872   8.573  < 2e-16 ***
## VV.moy        0.063534   0.086316   0.736  0.46188
## T.moy         0.271122   0.045264   5.990 3.06e-09 ***
## PA.moy        0.096420   0.028774   3.351  0.00084 ***
## DV.maxvv     -0.008025   0.002563  -3.131  0.00180 **
## PL.som       -0.219465   0.050233  -4.369 1.40e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.441 on 880 degrees of freedom
## Multiple R-squared:  0.5846, Adjusted R-squared:  0.5803
## F-statistic: 137.6 on 9 and 880 DF,  p-value: < 2.2e-16
```

The R^2 didn't change much, but the model is simpler. We still have the problem that the noise is not gaussian, but we can do some predictions. Using the linear model to predict the PM10 values in the Test Set:

```
#predictions for Linear Model
pred_lm <- predict(L2, newdata = testSet)
#true values vs predictions
plot(testSet$PM10,pred_lm)
lines(testSet$PM10,testSet$PM10)
```



The Root Mean Squared Error (RMSE) of the predictions is:

```
RMSE_lm = sqrt(1 /nrow(testSet)*sum((testSet$PM10-pred_lm)**2))
RMSE_lm
```

```
## [1] 6.014162
```

The Mean Absolute Error (MAE) of the predictions is:

```
MAE_lm = 1 /nrow(testSet)*sum(abs(testSet$PM10-pred_lm))
MAE_lm
```

```
## [1] 4.685296
```

Decision Tree

Since the linear model might not be ideal in our case, we can try some non linear models.

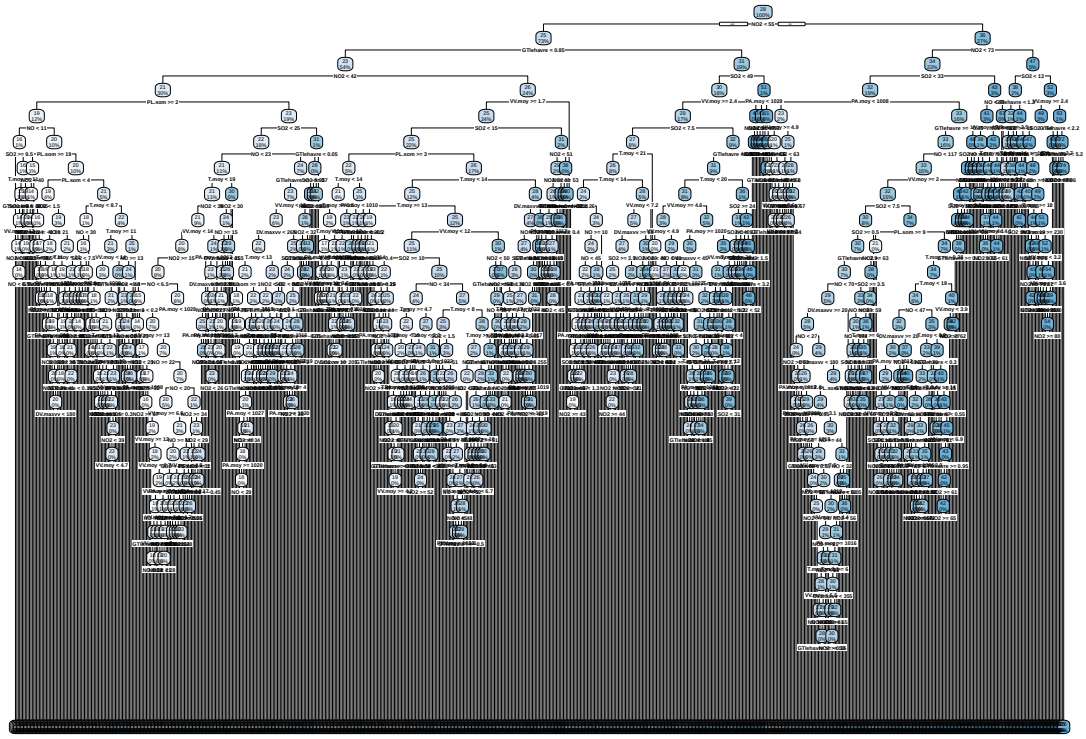
I will construct a decision tree using the explanatory variables selected before.

First, I obtain the maximal tree, maintaining the complexity parameter CP low.

The minimum number of observations in a node in order that an split is attempted is set as 2.

```
library(rpart)
library(rpart.plot)
#maximal tree
tree_max=rpart(PM10~.,data=learningSet, minsplit=2, cp = 10^(-9))
rpart.plot(tree_max)
```

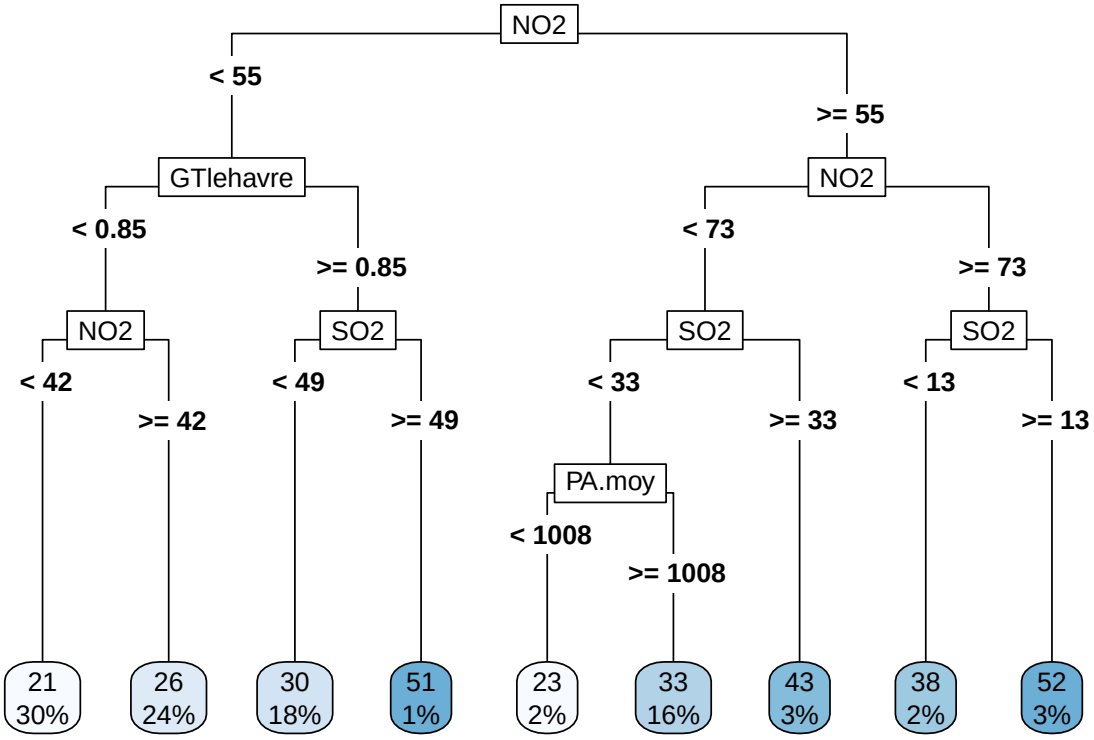
```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Then, I will prune this tree using the 1-SE rule.

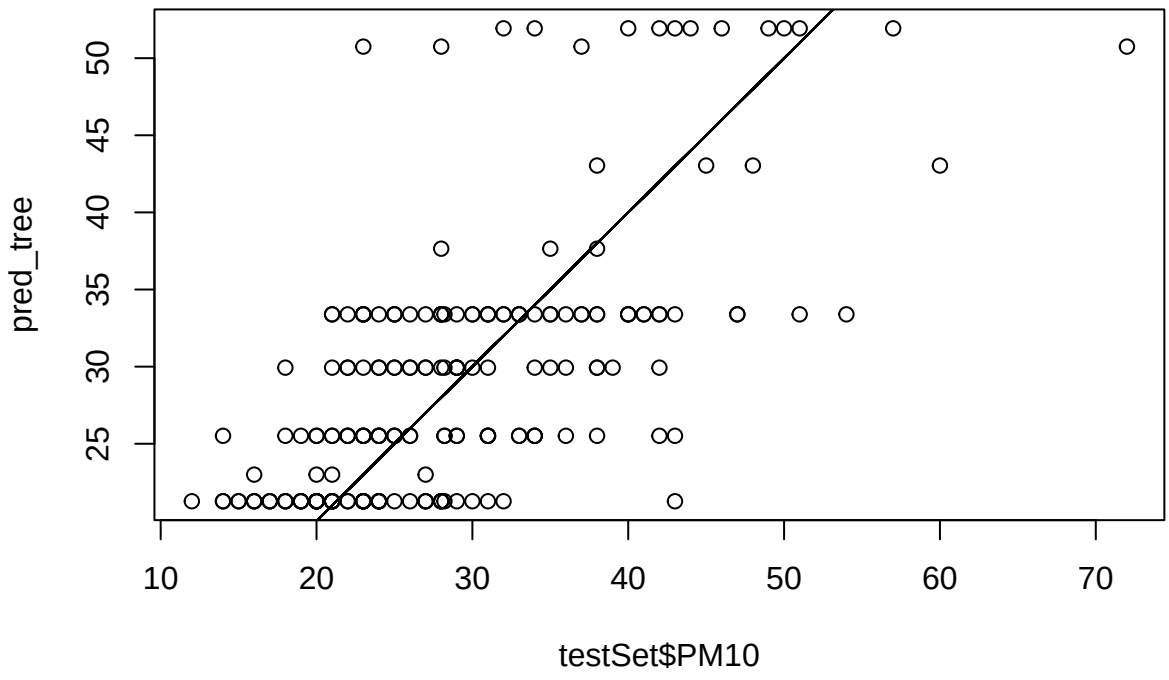
```
finalcart=function(T)
{
  P=printcp(T)
  CV=P[,4] #crossvalidation error
  a=which(CV==min(CV)) #finding the row with the smallest CV
  s=P[a,4]+P[a,5] #adding the standard deviation - the new threshold used in the 1SE rule
  ss=min(s) #in case s is a vector (several values are the min)
  b=which(CV<=ss)
  d=b[1] #selected value of cp
  Tf=prune(T,cp=P[d,1]) #pruning the maximal tree
  finalcart=Tf
}
tree = finalcart(tree_max)
```

```
rpart.plot(tree, type = 5)
```



Using the decision tree model to predict the PM10 values in the Test Set:

```
pred_tree<- predict(tree, newdata = testSet)
plot(testSet$PM10,pred_tree)
lines(testSet$PM10,testSet$PM10)
```



The RMSE error for this model:

```
RMSE_tree = sqrt(1 /nrow(testSet)*sum((testSet$PM10-pred_tree)**2))
RMSE_tree
```

```
## [1] 7.385158
```

The MAE error for this model:

```
MAE_tree = 1 /nrow(testSet)*sum(abs(testSet$PM10-pred_tree))
MAE_tree
```

```
## [1] 5.589259
```

Random Forest

Finally, we can also use a Random Forest model.

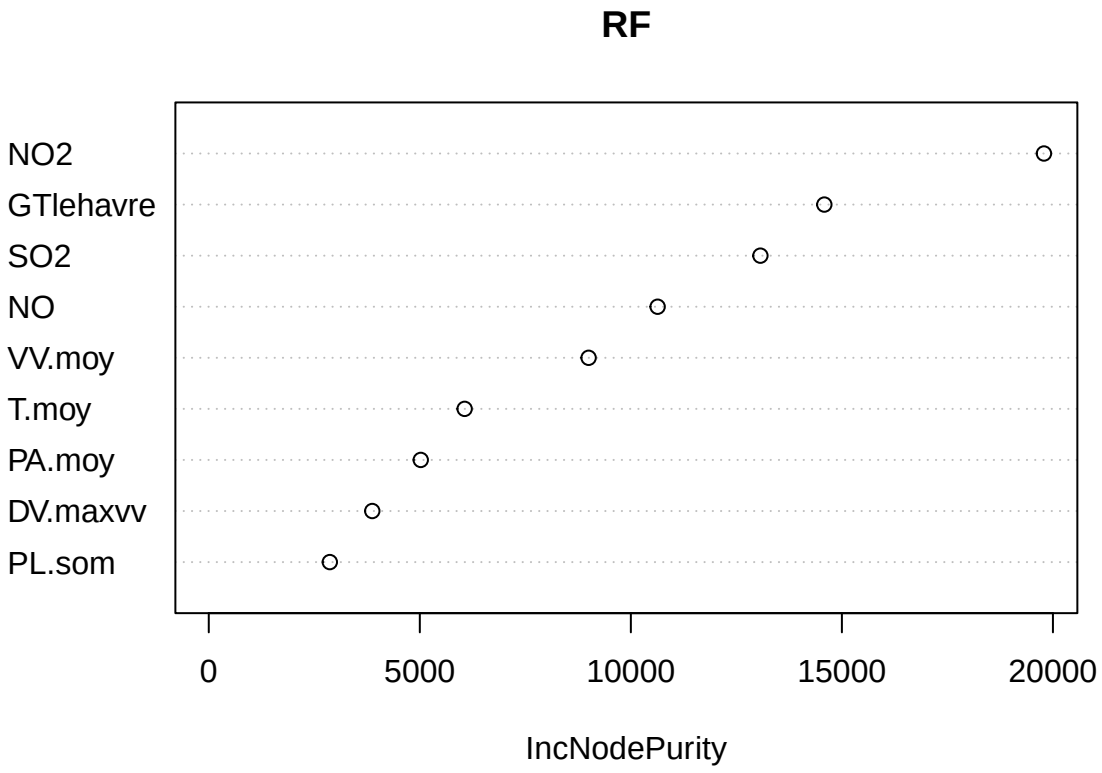
```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

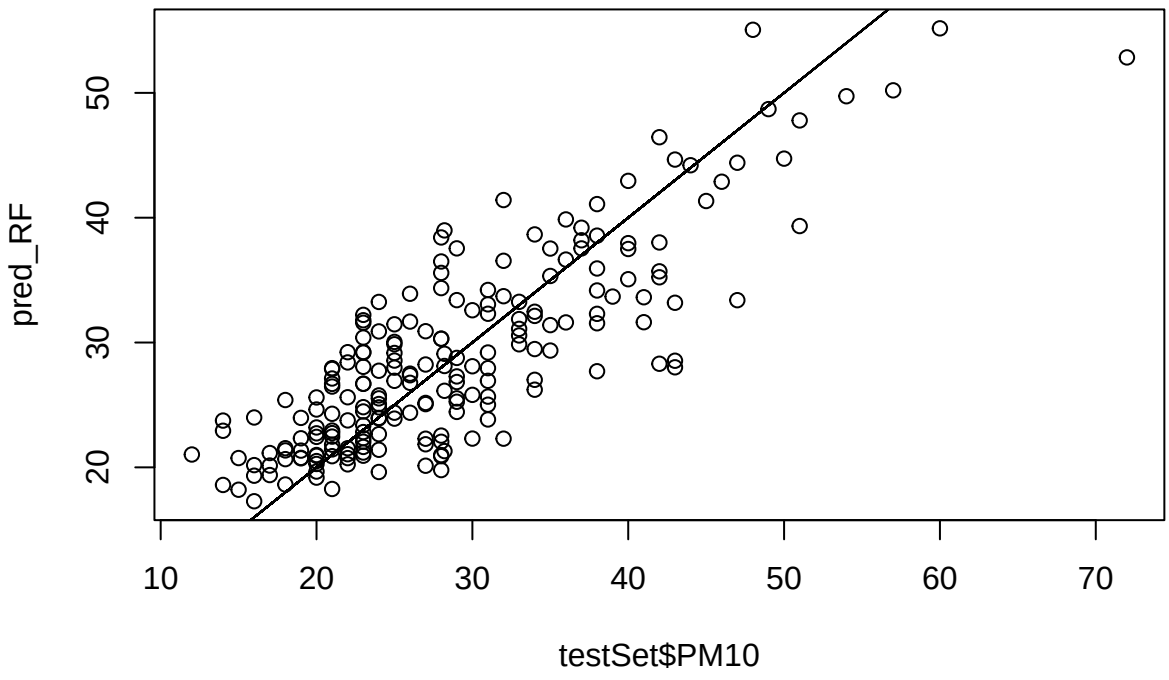
```
RF=randomForest(PM10~.,data=learningSet)
```

```
varImpPlot(RF)
```



Using the random forest model to predict the PM10 values in the Test Set:

```
pred_RF<- predict(RF, newdata = testSet)
plot(testSet$PM10,pred_RF)
lines(testSet$PM10,testSet$PM10)
```



The RMSE error for this model:

```
RMSE_RF = sqrt(1 /nrow(testSet)*sum((testSet$PM10-pred_RF)**2))
RMSE_RF
```

```
## [1] 5.258183
```

The MAE error for this model:

```
MAE_RF = 1 /nrow(testSet)*sum(abs(testSet$PM10-pred_RF))
MAE_RF
```

```
## [1] 4.141619
```

Discussion and Conclusion

```
results <- c(RMSE_lm, MAE_lm)
results <- rbind(results, c(RMSE_tree, MAE_tree))
results <- rbind(results,c(RMSE_RF, MAE_RF))
row.names(results) <- c("Linear Model", "Decision Tree", "Random Forest")
colnames(results) <- c("RMSE", "MAE")
results
```

```
##                   RMSE      MAE
## Linear Model  6.014162 4.685296
## Decision Tree 7.385158 5.589259
## Random Forest 5.258183 4.141619
```

The Random Forest model has predictions with the smallest RSME and MAE, followed by the Linear Model. The Decision Tree has the worst performance, but it does provide some nice explainability.

The concentration of other pollutants such as NO, NO_2 and SO_2 in the atmosphere is one of the main predictors of the concentration of PM10. It is also case of the temperature gradient in Le Havre, where this particular monitoring station is located. When this temperature gradient is high, there seems to be a higher concentration of PM10.