# Air Quality Sensors: Prediction of Pollution Level

Mikhail Zevakin

08 01 2021

## Executive summary

### The goals of the project

The goals of this project are:

1) to explore the cross-reference of the sensors (when the sensor responds not only to the level of pollutant to which it is normally targeted, but to other pollutants), and
2) to build an efficient model to predict the actual level of pollutant using the response of sensors normally not targeted to this pollutant (for the situation when we do not have normally targeted to pollutant sensor at all, or the response of sensor has very weak correlation with the actual level of the pollutant).

### Dataset and variables

For this project we used public dataset "Air Quality": https://archive.ics.uci.edu/ml/datasets/Air+quality

The dataset contains hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. The device was located on the field in a significantly polluted area, at road level, within an Italian city. Data were recorded from March 2004 to February 2005 (one year) representing the longest freely available recordings of on field deployed air quality chemical sensor devices responses.

Ground Truth hourly averaged concentrations for CO, Non Metanic Hydrocarbons, Benzene, Total Nitrogen Oxides (NOx) and Nitrogen Dioxide (NO2) were provided by a co-located reference certified analyzer.

Here is the description of fields from the web page of dataset https://archive.ics.uci.edu/ml/datasets/Air+quality

1) Date - date (DD/MM/YYYY).
2) Time - time (HH.MM.SS).
3) CO(GT) - true hourly averaged concentration CO in mg/m3 (reference analyzer).
4) PT08.S1(CO) - hourly averaged sensor response (tin oxide, nominally CO targeted).
5) NMHC(GT) - true hourly averaged overall Non Metanic HydroCarbons concentration in microg/mˆ3 (reference analyzer).
6) PT08.S2(NMHC) hourly averaged sensor response (titania, nominally NMHC targeted).
7) C6H6(GT) - true hourly averaged Benzene concentration in microg/m3 (reference analyzer). we noted that we do not have any special sensor nominally targeted to Benzene.

8) NOx(GT) - true hourly averaged NOx concentration in ppb (reference analyzer). ppb is an abbreviation for Parts per billion. NOx is an abbreviation for NO and NO2.

9) PT08.S3(NOx) - hourly averaged sensor response (tungsten oxide, nominally NOx targeted).

10) NO2(GT) - true hourly averaged NO2 concentration in microg/m3 (reference analyzer).

11) PT08.S4(NO2) - hourly averaged sensor response (tungsten oxide, nominally NO2 targeted).

12) PT08.S5(O3) - hourly averaged sensor response (indium oxide, nominally O3 targeted). We noted that we do not have true hourly averaged 03 level data.

13) T - temperature in °C.

14) RH - relative humidity (%).

15) AH - absolute humidity.

### Key steps that were performed

1. Data downloading and cleaning.
2. Exploration of dataset to trace the correlation between actual levels of different pollutants and the response of different sensors both targeted to specific pollutant and not targeted).
3. Building several models to predict the level of NO2:

- linear model (LM) using the response of all sensors targeted to pollutants except the sensor targeted to No2 (due to weak correlation) - during this step an optimal division between training and testing data was also set;
- generalized linear model (GLM) using the response of two sensors normally targeted to different pollutants, but with the strongest correlation (positive and negative) to the actual level of NO2;
- random forest (RF) model using the response of all sensors targeted to pollutants except the sensor targeted to No2;
- linear model (LM) using the response of the sensor normally targeted to NO2 (with very weak correlation) and temperature sensor.

4. Building of the random forest model (as the most efficient) to predict the level of benzene (C6H6), for which no special sensor in the dataset was targeted.

## Analysis

### 1. Preparing and cleaning of the data

We install and include the packages that will be used.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")

library(caret)
library(tidyverse)
```

```r
library(rpart)
library(dplyr)
library(ggplot2)
library(lubridate)
```

Air Quality Data Set: https://archive.ics.uci.edu/ml/machine-learning-databases/00360/AirQualityUCI.zip

We downloaded the zip file and extract the data.

```r
dl <- tempfile()
download.file("https://archive.ics.uci.edu/ml/machine-learning-databases/00360/AirQualityUCI.zip", dl)
AirQuality <- read_csv2(unzip(dl, "AirQualityUCI.csv"))
```

```
## Warning: Missing column names filled in: 'X16' [16], 'X17' [17]
```

The downloaded dataset contains 17 columns instead of 15 as per the description of dataset due to incorrect amount of semicolons. We exclude the columns 16 and 17 that are empty and not mentioned in description, and check the structure of the dataset.

```r
AirQuality <- AirQuality %>% select(!X16) %>% select(!X17)
AirQuality
```

```
## # A tibble: 9,471 x 15
##    Date  Time  `CO(GT)` `PT08.S1(CO)` `NMHC(GT)` `C6H6(GT)` `PT08.S2(NMHC)`
##    <chr> <chr>    <dbl>         <dbl>      <dbl>      <dbl>           <dbl>
##  1 10/0~ 18.0~      2.6          1360        150       11.9            1046
##  2 10/0~ 19.0~      2            1292        112        9.4             955
##  3 10/0~ 20.0~      2.2          1402         88        9               939
##  4 10/0~ 21.0~      2.2          1376         80        9.2             948
##  5 10/0~ 22.0~      1.6          1272         51        6.5             836
##  6 10/0~ 23.0~      1.2          1197         38        4.7             750
##  7 11/0~ 00.0~      1.2          1185         31        3.6             690
##  8 11/0~ 01.0~      1            1136         31        3.3             672
##  9 11/0~ 02.0~      0.9          1094         24        2.3             609
## 10 11/0~ 03.0~      0.6          1010         19        1.7             561
## # ... with 9,461 more rows, and 8 more variables: `NOx(GT)` <dbl>,
## #   `PT08.S3(NOx)` <dbl>, `NO2(GT)` <dbl>, `PT08.S4(NO2)` <dbl>,
## #   `PT08.S5(O3)` <dbl>, T <dbl>, RH <dbl>, AH <dbl>
```

```r
str(AirQuality)
```

```
## tibble [9,471 x 15] (S3: tbl_df/tbl/data.frame)
##  $ Date         : chr [1:9471] "10/03/2004" "10/03/2004" "10/03/2004" "10/03/2004" ...
##  $ Time         : chr [1:9471] "18.00.00" "19.00.00" "20.00.00" "21.00.00" ...
##  $ CO(GT)       : num [1:9471] 2.6 2 2.2 2.2 1.6 1.2 1.2 1 0.9 0.6 ...
##  $ PT08.S1(CO)  : num [1:9471] 1360 1292 1402 1376 1272 ...
##  $ NMHC(GT)     : num [1:9471] 150 112 88 80 51 38 31 31 24 19 ...
##  $ C6H6(GT)     : num [1:9471] 11.9 9.4 9 9.2 6.5 4.7 3.6 3.3 2.3 1.7 ...
##  $ PT08.S2(NMHC): num [1:9471] 1046 955 939 948 836 ...
##  $ NOx(GT)      : num [1:9471] 166 103 131 172 131 89 62 62 45 -200 ...
##  $ PT08.S3(NOx) : num [1:9471] 1056 1174 1140 1092 1205 ...
```

```
##  $ NO2(GT)      : num [1:9471] 113 92 114 122 116 96 77 76 60 -200 ...
##  $ PT08.S4(NO2) : num [1:9471] 1692 1559 1555 1584 1490 ...
##  $ PT08.S5(O3)  : num [1:9471] 1268 972 1074 1203 1110 ...
##  $ T            : num [1:9471] 13.6 13.3 11.9 11 11.2 11.2 11.3 10.7 10.7 10.3 ...
##  $ RH           : num [1:9471] 48.9 47.7 54 60 59.6 59.2 56.8 60 59.7 60.2 ...
##  $ AH           : num [1:9471] 0.758 0.726 0.75 0.787 0.789 ...
```

We noted that the data is presented in "wide" format, with every column representing date, time, sensor value, actual value of pollutant etc.

While checking the limits of values we note that the data has some `NA` values.

```
max(AirQuality$`CO(GT)`)
```

```
## [1] NA
```

```
AirQuality %>% filter(is.na(`CO(GT)`))
```

```
## # A tibble: 114 x 15
##    Date  Time  `CO(GT)` `PT08.S1(CO)` `NMHC(GT)` `C6H6(GT)` `PT08.S2(NMHC)`
##    <chr> <chr>    <dbl>         <dbl>      <dbl>      <dbl>           <dbl>
##  1 <NA>  <NA>        NA            NA         NA         NA              NA
##  2 <NA>  <NA>        NA            NA         NA         NA              NA
##  3 <NA>  <NA>        NA            NA         NA         NA              NA
##  4 <NA>  <NA>        NA            NA         NA         NA              NA
##  5 <NA>  <NA>        NA            NA         NA         NA              NA
##  6 <NA>  <NA>        NA            NA         NA         NA              NA
##  7 <NA>  <NA>        NA            NA         NA         NA              NA
##  8 <NA>  <NA>        NA            NA         NA         NA              NA
##  9 <NA>  <NA>        NA            NA         NA         NA              NA
## 10 <NA>  <NA>        NA            NA         NA         NA              NA
## # ... with 104 more rows, and 8 more variables: `NOx(GT)` <dbl>,
## #   `PT08.S3(NOx)` <dbl>, `NO2(GT)` <dbl>, `PT08.S4(NO2)` <dbl>,
## #   `PT08.S5(O3)` <dbl>, T <dbl>, RH <dbl>, AH <dbl>
```

We exclude the missing lines (containing only "NA's"), update the dataframe and check the maximum values of the columns representing pollution level (`*(GT)`) and the response of the sensors (`PT08.S*`).

```
AirQuality <- AirQuality %>% filter(!is.na(`CO(GT)`))
AirQuality
```

```
## # A tibble: 9,357 x 15
##    Date  Time  `CO(GT)` `PT08.S1(CO)` `NMHC(GT)` `C6H6(GT)` `PT08.S2(NMHC)`
##    <chr> <chr>    <dbl>         <dbl>      <dbl>      <dbl>           <dbl>
##  1 10/0~ 18.0~      2.6          1360        150       11.9            1046
##  2 10/0~ 19.0~      2            1292        112        9.4             955
##  3 10/0~ 20.0~      2.2          1402         88        9               939
##  4 10/0~ 21.0~      2.2          1376         80        9.2             948
##  5 10/0~ 22.0~      1.6          1272         51        6.5             836
##  6 10/0~ 23.0~      1.2          1197         38        4.7             750
##  7 11/0~ 00.0~      1.2          1185         31        3.6             690
##  8 11/0~ 01.0~      1            1136         31        3.3             672
```

```
##  9 11/0~ 02.0~      0.9          1094         24        2.3             609
## 10 11/0~ 03.0~      0.6          1010         19        1.7             561
## # ... with 9,347 more rows, and 8 more variables: `NOx(GT)` <dbl>,
## #   `PT08.S3(NOx)` <dbl>, `NO2(GT)` <dbl>, `PT08.S4(NO2)` <dbl>,
## #   `PT08.S5(O3)` <dbl>, T <dbl>, RH <dbl>, AH <dbl>
```

```
max(AirQuality$`CO(GT)`)
```

```
## [1] 11.9
```

```
max(AirQuality$`PT08.S1(CO)`)
```

```
## [1] 2040
```

```
max(AirQuality$`NMHC(GT)`)
```

```
## [1] 1189
```

```
max(AirQuality$`PT08.S2(NMHC)`)
```

```
## [1] 2214
```

```
max(AirQuality$`C6H6(GT)`)
```

```
## [1] 63.7
```

```
max(AirQuality$`NOx(GT)`)
```

```
## [1] 1479
```

```
max(AirQuality$`PT08.S3(NOx)`)
```

```
## [1] 2683
```

```
max(AirQuality$`NO2(GT)`)
```

```
## [1] 340
```

```
max(AirQuality$`PT08.S4(NO2)`)
```

```
## [1] 2775
```

```
max(AirQuality$`PT08.S5(O3)`)
```

```
## [1] 2523
```

We note that the maximum sensor response among all the values associated with sensors (PT08.S*) is 2683. We know that any digital sensor has certain resolution in bits (1-bit sensor has 2 available values - "0" and "1"). We suppose that 12-bit sensors are used - they could measure 4096 values from 0 to 4095.

Now we have 9357 lines with the values.

We also note in the description of the dataframe that the value `-200` is assigned to missing values.

```
min(AirQuality$`CO(GT)`)
```

```
## [1] -200
```

```
min(AirQuality$`PT08.S1(CO)`)
```

```
## [1] -200
```

```
min(AirQuality$`NMHC(GT)`)
```

```
## [1] -200
```

```
min(AirQuality$`PT08.S2(NMHC)`)
```

```
## [1] -200
```

```
min(AirQuality$`C6H6(GT)`)
```

```
## [1] -200
```

```
min(AirQuality$`NOx(GT)`)
```

```
## [1] -200
```

```
min(AirQuality$`PT08.S3(NOx)`)
```

```
## [1] -200
```

```
min(AirQuality$`NO2(GT)`)
```

```
## [1] -200
```

```
min(AirQuality$`PT08.S4(NO2)`)
```

```
## [1] -200
```

```
min(AirQuality$`PT08.S5(O3)`)
```

```
## [1] -200
```

Every column with sensor value and actual level of pollutant contains missing data.

If we exclude all the lines containing at lease one missing value, we will lose most of the data (only 827 lines from 9357 will be left). We understand that the actual values of pollutants,
or sensor values could be only positive, thus, we could use the condition ">0" instead of ">-200".

```
AirQuality %>%
  filter(`CO(GT)` > 0) %>%
  filter(`PT08.S1(CO)` > 0) %>%
  filter(`NMHC(GT)` > 0) %>%
  filter(`PT08.S2(NMHC)` > 0) %>%
  filter(`C6H6(GT)` > 0) %>%
  filter(`NOx(GT)` > 0) %>%
  filter(`PT08.S3(NOx)` > 0) %>%
  filter(`NO2(GT)` > 0) %>%
  filter(`PT08.S4(NO2)` > 0) %>%
  filter(`PT08.S5(O3)` > 0)
```

```
## # A tibble: 827 x 15
##    Date  Time  `CO(GT)` `PT08.S1(CO)` `NMHC(GT)` `C6H6(GT)` `PT08.S2(NMHC)`
##    <chr> <chr>    <dbl>         <dbl>      <dbl>      <dbl>           <dbl>
##  1 10/0~ 18.0~      2.6          1360        150       11.9            1046
##  2 10/0~ 19.0~      2            1292        112        9.4             955
##  3 10/0~ 20.0~      2.2          1402         88        9               939
##  4 10/0~ 21.0~      2.2          1376         80        9.2             948
##  5 10/0~ 22.0~      1.6          1272         51        6.5             836
##  6 10/0~ 23.0~      1.2          1197         38        4.7             750
##  7 11/0~ 00.0~      1.2          1185         31        3.6             690
##  8 11/0~ 01.0~      1            1136         31        3.3             672
##  9 11/0~ 02.0~      0.9          1094         24        2.3             609
## 10 11/0~ 05.0~      0.7          1066          8        1.1             512
## # ... with 817 more rows, and 8 more variables: `NOx(GT)` <dbl>,
## #   `PT08.S3(NOx)` <dbl>, `NO2(GT)` <dbl>, `PT08.S4(NO2)` <dbl>,
## #   `PT08.S5(O3)` <dbl>, T <dbl>, RH <dbl>, AH <dbl>
```

In order not to lose the lines data in the next chapters we will make a separate dataset for each pollutant with only necessary columns filtered to exclude the missing data.

The description of data says that the certain censors are nominally targeted to certain pollutants, but there are also the evidences of cross-reference.

We will make the separate dataframes for each pollutant to preserve maximum amount of lines (otherwise many lines could be lost if any column contains missing data), clean the data from missing values and check the correlation between the level of pollutant and the response of sensors.

To be able to use both date and time as an axis (for diagramms) we add the column “timestamp” (this column will show the number of hours passed from the epoch, January, 1, 1970).

First, we set the vector containing the number of hour from each line (the data is collected hourly). Then we extract the date as a timestamp (number of days since January 1, 1970 used as the epoch). Than we add the vector “timestamp”, which shows the number of hours from the epoch, create the empty column “Timestamp” and assign the values from the vector “timestamp” to this column.

```
time <- as.factor(AirQuality$Time)
hour = as.numeric(format(as.POSIXct(time,format="%H.%M.%S"),"%H"))

date <- as.numeric(as.Date(AirQuality$Date, "%d/%m/%Y"))

timestamp = date * 24 + hour
```

```r
AirQuality <- AirQuality %>% mutate(Timestamp = 0)
AirQuality$Timestamp <- timestamp
```

## 2. Exploration of the dataset

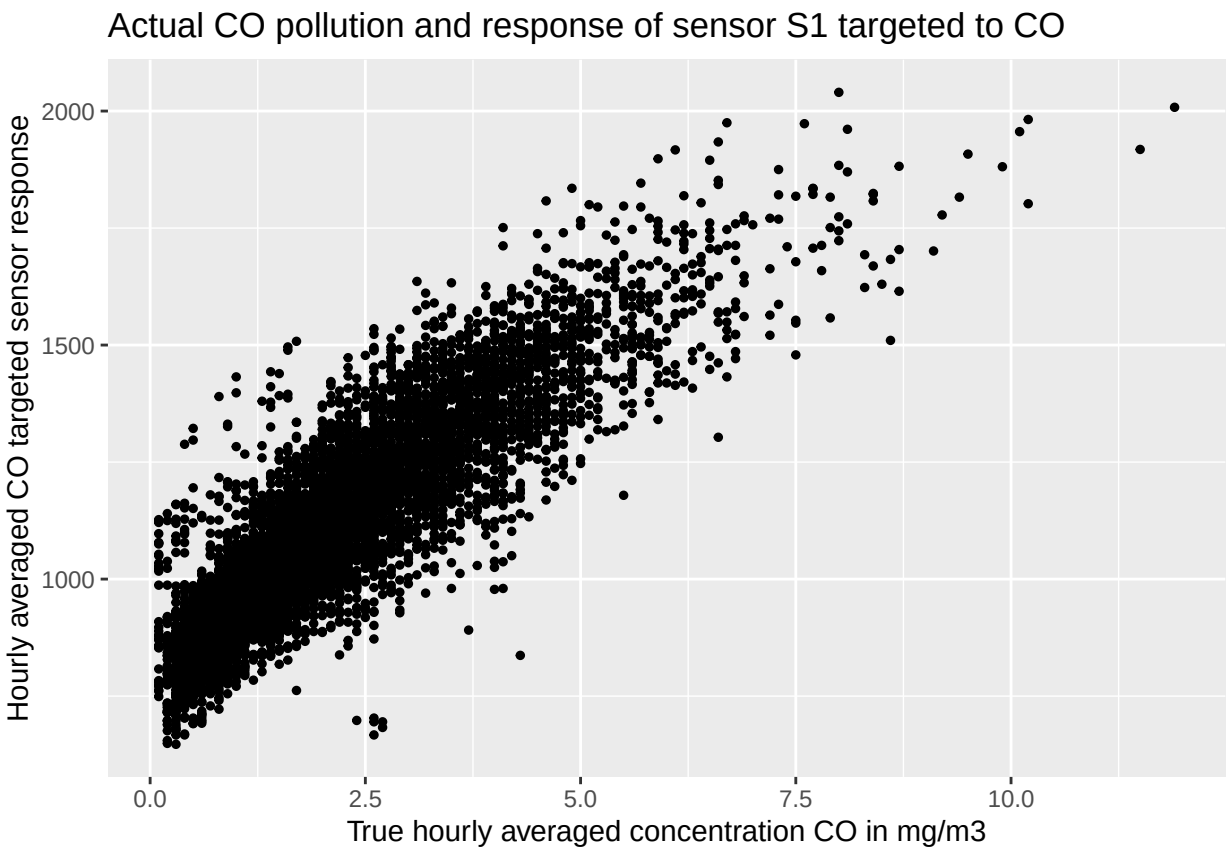
### 2.1 CO pollution

```r
CO_pollution <- AirQuality %>%
  select(Timestamp, `CO(GT)`, `PT08.S1(CO)`) %>%
  filter(`CO(GT)` > 0) %>%
  filter(`PT08.S1(CO)` >0)

cor(CO_pollution$`CO(GT)`, CO_pollution$`PT08.S1(CO)`)
```

```
## [1] 0.8792883
```

```r
CO_pollution %>%
  ggplot(aes(`CO(GT)`, `PT08.S1(CO)`)) +
  geom_point(size = 1) +
  xlab("True hourly averaged concentration CO in mg/m3") +
  ylab("Hourly averaged CO targeted sensor response") +
  ggtitle("Actual CO pollution and response of sensor S1 targeted to CO")
```



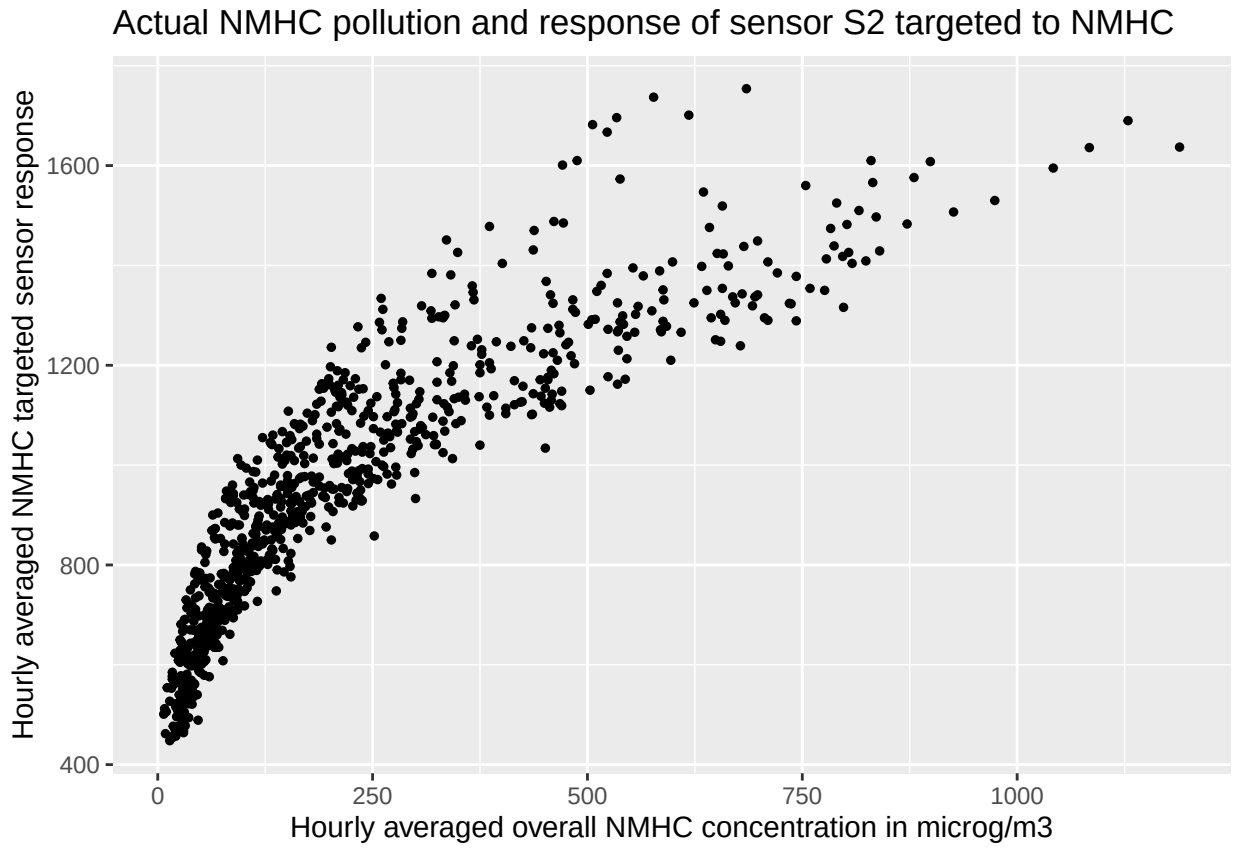
### 2.2 NMHC pollution

```
NMHC_pollution <- AirQuality %>%
  select(Timestamp, `NMHC(GT)`, `PT08.S2(NMHC)`) %>%
  filter(`NMHC(GT)` > 0) %>%
  filter(`PT08.S2(NMHC)` >0)

cor(NMHC_pollution$`NMHC(GT)`, NMHC_pollution$`PT08.S2(NMHC)`)
```

```
## [1] 0.8776957
```

```
NMHC_pollution %>%
  ggplot(aes(`NMHC(GT)`, `PT08.S2(NMHC)`)) +
  geom_point(size = 1) +
  xlab("Hourly averaged overall NMHC concentration in microg/m3") +
  ylab("Hourly averaged NMHC targeted sensor response") +
  ggtitle("Actual NMHC pollution and response of sensor S2 targeted to NMHC")
```




### 2.3 NOx pollution

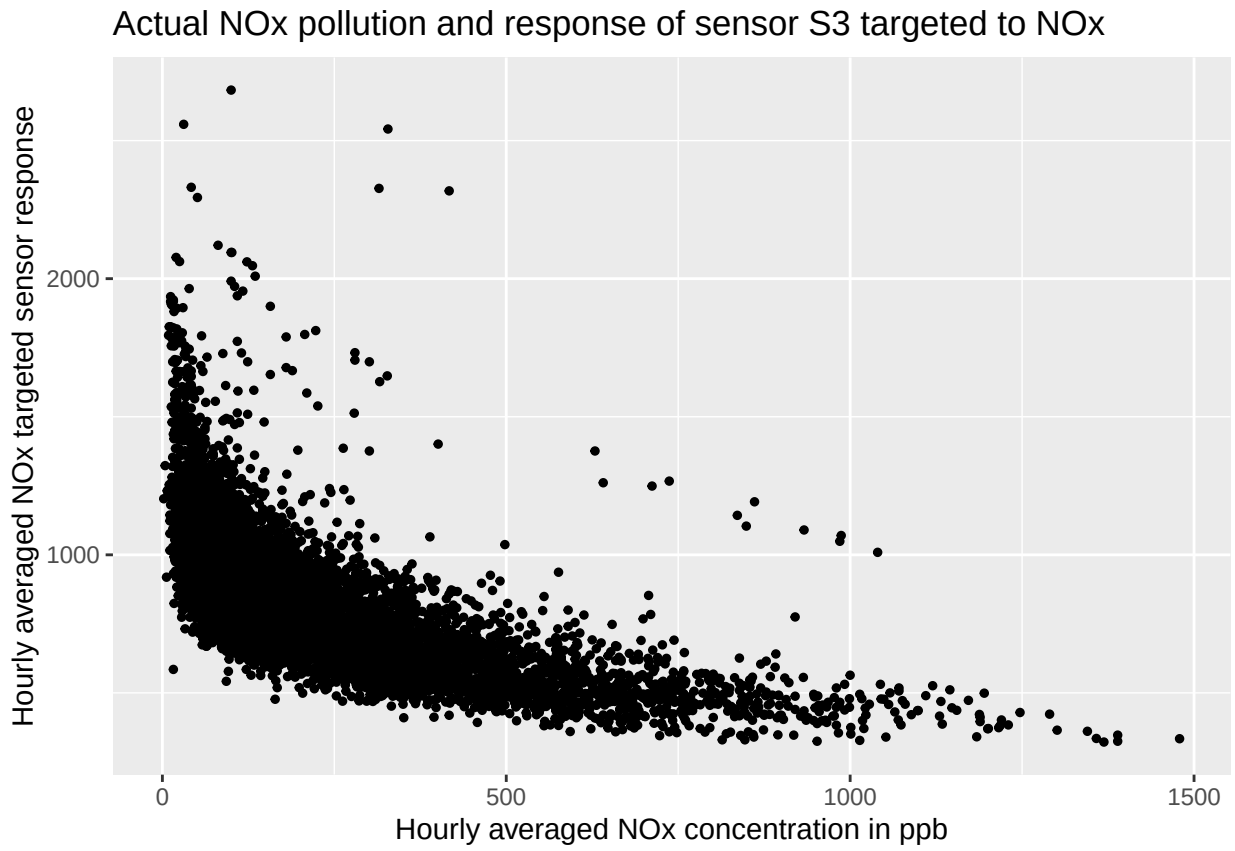
```
NOx_pollution <- AirQuality %>%
  select(Timestamp, `NOx(GT)`, `PT08.S3(NOx)`) %>%
```

```
  filter(`NOx(GT)` > 0) %>%
  filter(`PT08.S3(NOx)` >0)

cor(NOx_pollution$`NOx(GT)`, NOx_pollution$`PT08.S3(NOx)`)
```

```
## [1] -0.6557071
```

```
NOx_pollution %>%
  ggplot(aes(`NOx(GT)`, `PT08.S3(NOx)`)) +
  geom_point(size = 1) +
  xlab("Hourly averaged NOx concentration in ppb") +
  ylab("Hourly averaged NOx targeted sensor response") +
  ggtitle("Actual NOx pollution and response of sensor S3 targeted to NOx")
```



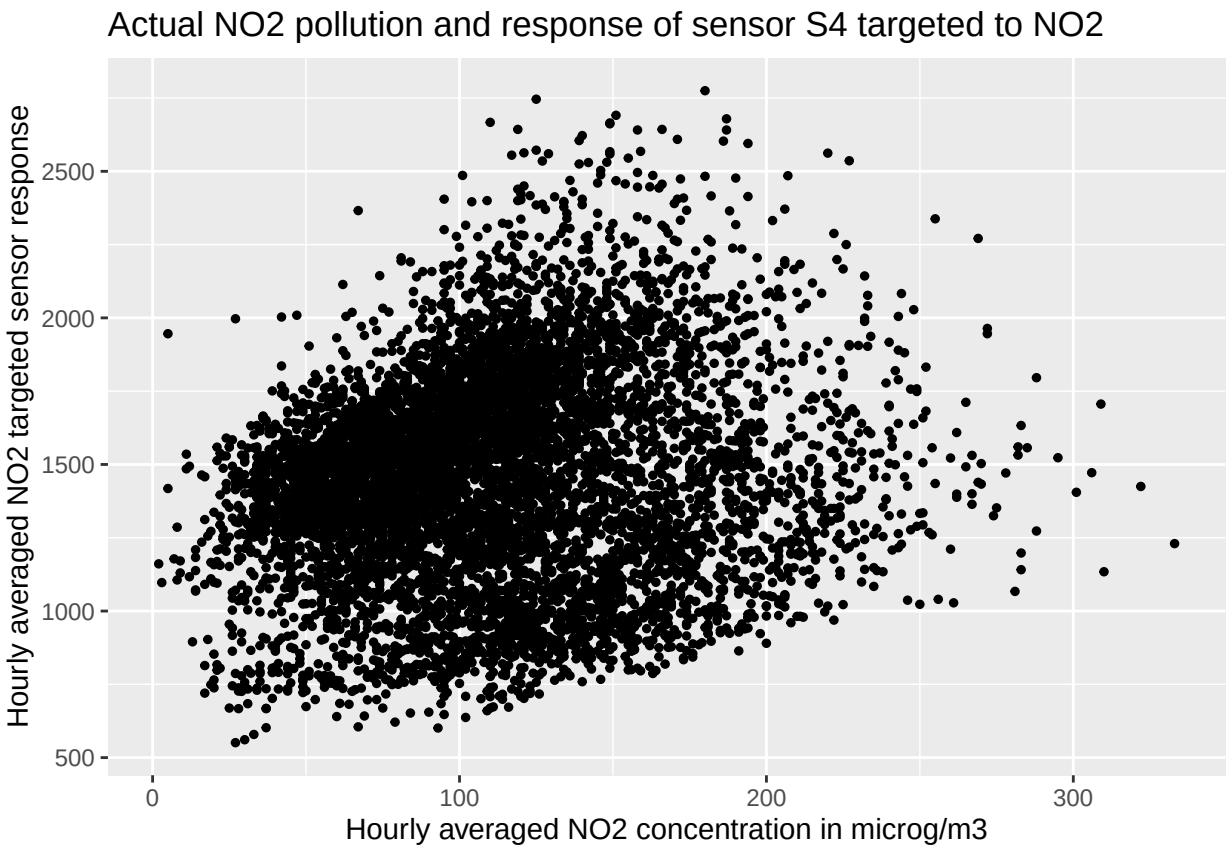

### 2.4 NO2 pollution

```
NO2_pollution <- AirQuality %>%
  select(Timestamp, `NO2(GT)`, `PT08.S4(NO2)`, T) %>%
  filter(`NO2(GT)` > 0) %>%
  filter(`PT08.S4(NO2)` >0)

cor(NO2_pollution$`NO2(GT)`, NO2_pollution$`PT08.S4(NO2)`)
```

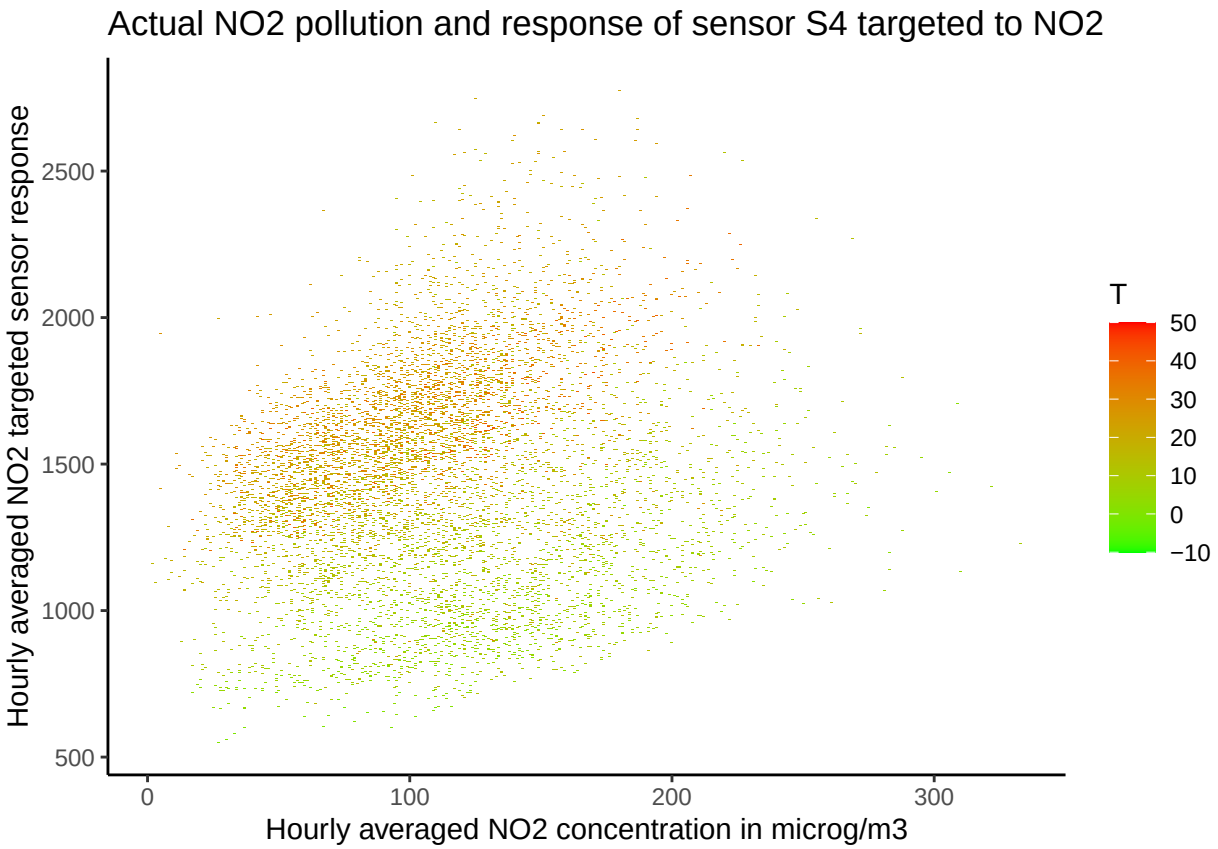
```
## [1] 0.1573603
```

```
NO2_pollution %>%
  ggplot(aes(`NO2(GT)`, `PT08.S4(NO2)`)) +
  geom_point(size = 1) +
  xlab("Hourly averaged NO2 concentration in microg/m3") +
  ylab("Hourly averaged NO2 targeted sensor response") +
  ggtitle("Actual NO2 pollution and response of sensor S4 targeted to NO2")
```




We see very weak correlation between the level of NO2 and the response of the sensor nominally targeted to NO2 Now we check if the temperature also influences the response of the sensor.

```
NO2_pollution %>%
  ggplot(aes(`NO2(GT)`, `PT08.S4(NO2)`)) +
  geom_tile(aes(fill = T)) +
  theme_classic() +
  scale_fill_gradient(low="green", high="red", limits=c(-10, 50)) +
  xlab("Hourly averaged NO2 concentration in microg/m3") +
  ylab("Hourly averaged NO2 targeted sensor response") +
  ggtitle("Actual NO2 pollution and response of sensor S4 targeted to NO2")
```




```
cor(NO2_pollution$T, NO2_pollution$`PT08.S4(NO2)`)
```

```
## [1] 0.5755162
```

We could see that higher temperature shifts the range of response of the sensor up for the same level of pollution. We also found that the correlation between the temperature and the response of the sensor is much stronger than the correlation between the pollution level of NO2 and the response of the sensor (that means that the sensor is more likely to respond to the change of temperature than to change of level of NO2, to which it is nominally targeted).

Let's check if there is a correlation between the response of other sensors (not nominally targeted to NO2) and the level of NO2.

```
NO2_pollution_other <- AirQuality %>%
  select(Timestamp, `NO2(GT)`, `PT08.S1(CO)`, `PT08.S2(NMHC)`,
         `PT08.S3(NOx)`, `PT08.S4(NO2)`,  `PT08.S5(O3)`) %>%
  filter(`NO2(GT)` > 0) %>%
  filter(`PT08.S1(CO)` >0) %>%
  filter(`PT08.S2(NMHC)` >0) %>%
  filter(`PT08.S3(NOx)` >0) %>%
  filter(`PT08.S4(NO2)` >0) %>%
  filter(`PT08.S5(O3)` >0)

cor(NO2_pollution_other$`NO2(GT)`, NO2_pollution_other$`PT08.S1(CO)`)
```

```
## [1] 0.6415291
```

```r
cor(NO2_pollution_other$`NO2(GT)`, NO2_pollution_other$`PT08.S2(NMHC)`)
```

```
## [1] 0.6462453
```

```r
cor(NO2_pollution_other$`NO2(GT)`, NO2_pollution_other$`PT08.S3(NOx)`)
```

```
## [1] -0.652083
```

```r
cor(NO2_pollution_other$`NO2(GT)`, NO2_pollution_other$`PT08.S5(O3)`)
```

```
## [1] 0.7081276
```

### 2.5 Insights

1. The responses of sensors nominally targeted to CO and NHMC have strong correlation with actual levels of CO and NMHC correspondingly. The response of sensor nominally targeted to NOx has medium negative correlation with actual level of NOx.
2. The response of sensor nominally targeted to NO2 has very weak correlation with actual level of NO2.
3. The responses of sensors nominally targeted to 03, CO, NHMC and NOx have medium correlation with the actual level of NO2 (cross-reference).
4. We could try to use the cross-reference of the sensors to predict the levels of pollutants to which the sensors are not nominally targeted.

## 3. Developing of the models

### 3.1 LM for NO2 pollution with different split of data

Here we will develop linear models (LM) to predict NO2 pollution using the sensors nominally targeted to other pollutants (O3, CO, NHMC and NOx). We will also try different splits of data between train set and test set and select the best variant

We could try to make the model for prediction of NO2 level using the sensors nominally targeted to other pollutants (O3, CO, NHMC and NOx).

We noted that the correlation coefficient could be used to measure how our algorithm works. We will use a threshold 0.75 as a minimal required correlation.

We will also use RMSE (root mean square error) as a secondary measure for evaluation of the algorithm.

```r
RMSE <- function(true_concentration, predicted_concentration){
  sqrt(mean((true_concentration - predicted_concentration)^2))
}
```

We will try 3 variants of split of data between training and test sets (90%-10%, 70%-30% and 50%-50%) and choose the best variant to find the balance between sufficient amount of data for training and overtraining (when the accuracy of algorithm on train set is much higher than on test set).

First, we try to build the algorithm using 10% of NO2_pollution_other dataframe for validation and, 90% for training (train_set). We will try linear Model (LM) using the sensors nominally targeted to other pollutants (O3, CO, NHMC and NOx).

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = NO2_pollution_other$`NO2(GT)`, times = 1, p = 0.1,
                                  list = FALSE)
train_set_NO2 <- NO2_pollution_other[-test_index,]
test_set_NO2 <- NO2_pollution_other[test_index,]

train_lm <- train(`NO2(GT)` ~ `PT08.S1(CO)` + `PT08.S2(NMHC)` + `PT08.S3(NOx)` + `PT08.S5(O3)`,
                  method = "lm", data = train_set_NO2)

#prediction on train set
train_set_NO2 <- train_set_NO2 %>%
  mutate(predicted_NO2_lm = predict(train_lm, train_set_NO2))

RMSE(train_set_NO2$`NO2(GT)`, train_set_NO2$predicted_NO2_lm)
```

```
## [1] 32.61922
```

```
cor(train_set_NO2$`NO2(GT)`, train_set_NO2$predicted_NO2_lm)
```

```
## [1] 0.7269481
```

```
#prediction on test set
test_set_NO2 <- test_set_NO2 %>%
  mutate(predicted_NO2_lm = predict(train_lm, test_set_NO2))

RMSE_NO2_10p <- RMSE(test_set_NO2$`NO2(GT)`, test_set_NO2$predicted_NO2_lm)
cor_NO2_10p <- cor(test_set_NO2$`NO2(GT)`, test_set_NO2$predicted_NO2_lm)

cor_results <- tibble(Method = "Linear model, sensors S1, S2, S3, S5, test set 10%",
                      Correlation = cor_NO2_10p, RMSE = RMSE_NO2_10p)

cor_results
```

```
## # A tibble: 1 x 3
##   Method                                                Correlation  RMSE
##   <chr>                                                       <dbl> <dbl>
## 1 Linear model, sensors S1, S2, S3, S5, test set 10%          0.699  34.9
```

Correlation coefficient on test set is much lower than on train set (and RMSE is higher). Looks like the algorithm was a bit overtrained and showed much better results on train set than on unused for training data.

Let's use 70% of data for training (instead of 90%) and 30% of data for validation in the same model.

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = NO2_pollution_other$`NO2(GT)`, times = 1, p = 0.3,
                                  list = FALSE)
train_set_NO2_30p <- NO2_pollution_other[-test_index,]
test_set_NO2_30p <- NO2_pollution_other[test_index,]

train_lm_NO2_30p <- train(`NO2(GT)` ~ `PT08.S1(CO)` + `PT08.S2(NMHC)` +
                            `PT08.S3(NOx)` + `PT08.S5(O3)`,
```

```
                  method = "lm", data = train_set_NO2_30p)

#prediction on test set
train_set_NO2_30p <- train_set_NO2_30p %>%
  mutate(predicted_NO2_lm = predict(train_lm_NO2_30p, train_set_NO2_30p))

RMSE(train_set_NO2_30p$`NO2(GT)`, train_set_NO2_30p$predicted_NO2_lm)
```

```
## [1] 33.1204
```

```
cor(train_set_NO2_30p$`NO2(GT)`, train_set_NO2_30p$predicted_NO2_lm)
```

```
## [1] 0.7154569
```

```
#prediction on test set
test_set_NO2_30p <- test_set_NO2_30p %>%
  mutate(predicted_NO2_lm = predict(train_lm_NO2_30p, test_set_NO2_30p))

RMSE_NO2_30p <- RMSE(test_set_NO2_30p$`NO2(GT)`, test_set_NO2_30p$predicted_NO2_lm)
cor_NO2_30p <- cor(test_set_NO2_30p$`NO2(GT)`, test_set_NO2_30p$predicted_NO2_lm)

cor_results <- bind_rows(cor_results,
                         tibble(Method = "Linear model, sensors S1, S2, S3, S5, test set 30%",
                                Correlation = cor_NO2_30p, RMSE = RMSE_NO2_30p))

cor_results
```

```
## # A tibble: 2 x 3
##   Method                                                Correlation  RMSE
##   <chr>                                                       <dbl> <dbl>
## 1 Linear model, sensors S1, S2, S3, S5, test set 10%          0.699  34.9
## 2 Linear model, sensors S1, S2, S3, S5, test set 30%          0.743  32.2
```

Now our algorithm has better RMSE and correlation coefficient on test set.

Finally, we will try to use only 50% data for training, leaving 50% for testing.

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = NO2_pollution_other$`NO2(GT)`, times = 1, p = 0.5,
                                  list = FALSE)
train_set_NO2_50p <- NO2_pollution_other[-test_index,]
test_set_NO2_50p <- NO2_pollution_other[test_index,]

train_lm_NO2_50p <- train(`NO2(GT)` ~ `PT08.S1(CO)` + `PT08.S2(NMHC)` +
                            `PT08.S3(NOx)` + `PT08.S5(O3)`,
                          method = "lm", data = train_set_NO2_50p)

#prediction on train set
train_set_NO2_50p <- train_set_NO2_50p %>%
  mutate(predicted_NO2_lm = predict(train_lm_NO2_50p, train_set_NO2_50p))

RMSE(train_set_NO2_50p$`NO2(GT)`, train_set_NO2_50p$predicted_NO2_lm)
```

```
## [1] 33.11278
```

```
cor(train_set_NO2_50p$`NO2(GT)`, train_set_NO2_50p$predicted_NO2_lm)
```

```
## [1] 0.7189593
```

```
#prediction on test set
test_set_NO2_50p <- test_set_NO2_50p %>%
  mutate(predicted_NO2_lm = predict(train_lm_NO2_50p, test_set_NO2_50p))

RMSE_NO2_50p <- RMSE(test_set_NO2_50p$`NO2(GT)`, test_set_NO2_50p$predicted_NO2_lm)
cor_NO2_50p <- cor(test_set_NO2_50p$`NO2(GT)`, test_set_NO2_50p$predicted_NO2_lm)

cor_results <- bind_rows(cor_results,
                         tibble(Method = "Linear model, sensors S1, S2, S3, S5, test set 50%",
                                Correlation = cor_NO2_50p, RMSE = RMSE_NO2_50p))

cor_results
```

```
## # A tibble: 3 x 3
##   Method                                              Correlation  RMSE
##   <chr>                                                     <dbl> <dbl>
## 1 Linear model, sensors S1, S2, S3, S5, test set 10%        0.699  34.9
## 2 Linear model, sensors S1, S2, S3, S5, test set 30%        0.743  32.2
## 3 Linear model, sensors S1, S2, S3, S5, test set 50%        0.729  32.6
```

We diminished the amount of data for training and this worsened the resulting RMSE and correlation coefficient

We find the split of data as 70% for training and 30% for testing as the most optimal and we now we are ready to try a couple of other models.

### 3.2 GLM for NO2 pollution

We will try to develop Generalized Linear Model (GLM) using only two sensors nominally targeted to other pollutants with the strongest positive and negative correlation with the level of NO2 (these are the sensors targeted to NOx and O3).

```
train_glm <- train(`NO2(GT)` ~ `PT08.S3(NOx)` + `PT08.S5(O3)`,
                   method = "glm", data = train_set_NO2_30p)

#prediction on train set
train_set_NO2_30p <- train_set_NO2_30p %>%
  mutate(predicted_NO2_glm = predict(train_glm, train_set_NO2_30p))

RMSE(train_set_NO2_30p$`NO2(GT)`, train_set_NO2_30p$predicted_NO2_glm)
```

```
## [1] 33.15343
```

```
cor(train_set_NO2_30p$`NO2(GT)`, train_set_NO2_30p$predicted_NO2_glm)
```

```
## [1] 0.7147758
```

```
#prediction on test set
test_set_NO2_30p <- test_set_NO2_30p %>%
  mutate(predicted_NO2_glm = predict(train_glm, test_set_NO2_30p))

RMSE_NO2_glm_30p <- RMSE(test_set_NO2_30p$`NO2(GT)`, test_set_NO2_30p$predicted_NO2_glm)
cor_NO2_glm_30p <- cor(test_set_NO2_30p$`NO2(GT)`, test_set_NO2_30p$predicted_NO2_glm)

cor_results <- bind_rows(cor_results,
                         tibble(Method = "Generalized Linear Model, sensors S3, S5, test set 30%",
                                Correlation = cor_NO2_glm_30p, RMSE = RMSE_NO2_glm_30p))

cor_results
```

```
## # A tibble: 4 x 3
##   Method                                                Correlation  RMSE
##   <chr>                                                       <dbl> <dbl>
## 1 Linear model, sensors S1, S2, S3, S5, test set 10%          0.699  34.9
## 2 Linear model, sensors S1, S2, S3, S5, test set 30%          0.743  32.2
## 3 Linear model, sensors S1, S2, S3, S5, test set 50%          0.729  32.6
## 4 Generalized Linear Model, sensors S3, S5, test set 30%      0.744  32.2
```

The result is a bit better than in linear model with all the sensors except one nominally targeted to NO2, but correlation coefficient is still lower than our threshold (0.75).

### 3.3 RF for NO2 pollution

Now we will try to develop Random Forest (RF) Model using the sensors nominally targeted to other pollutants (O3, CO, NHMC and NOx). Random Forest model will take much time.

```
train_rf <- train(`NO2(GT)` ~ `PT08.S1(CO)` + `PT08.S2(NMHC)` + `PT08.S3(NOx)` + `PT08.S5(O3)`,
                  method = "rf", data = train_set_NO2_30p)

#prediction on train set
train_set_NO2_30p <- train_set_NO2_30p %>%
  mutate(predicted_NO2_rf = predict(train_rf, train_set_NO2_30p))

RMSE(train_set_NO2_30p$`NO2(GT)`, train_set_NO2_30p$predicted_NO2_rf)
```

```
## [1] 13.28845
```

```
cor(train_set_NO2_30p$`NO2(GT)`, train_set_NO2_30p$predicted_NO2_rf)
```

```
## [1] 0.9660633
```

```
#prediction on test set
test_set_NO2_30p <- test_set_NO2_30p %>%
  mutate(predicted_NO2_rf = predict(train_rf, test_set_NO2_30p))

RMSE_NO2_rf_30p <- RMSE(test_set_NO2_30p$`NO2(GT)`, test_set_NO2_30p$predicted_NO2_rf)
cor_NO2_rf_30p <- cor(test_set_NO2_30p$`NO2(GT)`, test_set_NO2_30p$predicted_NO2_rf)

cor_results <- bind_rows(cor_results,
                         tibble(Method = "Random Forest, sensors S1, S2, S3, S5, test set 30%",
                                Correlation = cor_NO2_rf_30p, RMSE = RMSE_NO2_rf_30p))

cor_results
```

```
## # A tibble: 5 x 3
##   Method                                                Correlation  RMSE
##   <chr>                                                       <dbl> <dbl>
## 1 Linear model, sensors S1, S2, S3, S5, test set 10%          0.699  34.9
## 2 Linear model, sensors S1, S2, S3, S5, test set 30%          0.743  32.2
## 3 Linear model, sensors S1, S2, S3, S5, test set 50%          0.729  32.6
## 4 Generalized Linear Model, sensors S3, S5, test set 30%      0.744  32.2
## 5 Random Forest, sensors S1, S2, S3, S5, test set 30%         0.797  29.1
```

Using the Random Forest method and the values of sensors nominally targeted to CO. NHMC, NOx and O3, we predicted the level of NO2 with the highest correlation and the lowest RMSE than using linear models.

### 3.4 LM for NO2 pollution using targeted sensor

We found that the response of NO2 targeted sensor has very weak correlation with the actual level of NO2. We also found that change of temperature shifted the slope of correlation. What if we try to predict the level of NO2 using the sensor nominally targeted to NO2, and the temperature? We will build Linear Model (LM) using the sensor nominally targeted to NO and the values of temperature.

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = NO2_pollution$`NO2(GT)`, times = 1, p = 0.3,
                                  list = FALSE)
train_set_NO2_T <- NO2_pollution[-test_index,]
test_set_NO2_T <- NO2_pollution[test_index,]

train_lm_NO2_T <- train(`NO2(GT)` ~ `PT08.S4(NO2)` + T,
                        method = "lm", data = train_set_NO2_T)

#prediction on train set
train_set_NO2_T <- train_set_NO2_T %>%
  mutate(predicted_NO2_lm = predict(train_lm_NO2_T, train_set_NO2_T))

RMSE(train_set_NO2_T$`NO2(GT)`, train_set_NO2_T$predicted_NO2_lm)
```

```
## [1] 44.04742
```

```
cor(train_set_NO2_T$`NO2(GT)`, train_set_NO2_T$predicted_NO2_lm)
```

```
## [1] 0.3696866
```

```
#prediction on test set
test_set_NO2_T <- test_set_NO2_T %>%
  mutate(predicted_NO2_lm = predict(train_lm_NO2_T, test_set_NO2_T))

RMSE_NO2_T <- RMSE(test_set_NO2_T$`NO2(GT)`, test_set_NO2_T$predicted_NO2_lm)
cor_NO2_T <- cor(test_set_NO2_T$`NO2(GT)`, test_set_NO2_T$predicted_NO2_lm)

cor_results <- bind_rows(cor_results,
     tibble(Method = "Linear model, sensor S4 (targeted to NO2) and Temperature, test set 30%",
                                     Correlation = cor_NO2_T, RMSE = RMSE_NO2_T))

cor_results
```

```
## # A tibble: 6 x 3
##   Method                                                    Correlation  RMSE
##   <chr>                                                           <dbl> <dbl>
## 1 Linear model, sensors S1, S2, S3, S5, test set 10%              0.699  34.9
## 2 Linear model, sensors S1, S2, S3, S5, test set 30%              0.743  32.2
## 3 Linear model, sensors S1, S2, S3, S5, test set 50%              0.729  32.6
## 4 Generalized Linear Model, sensors S3, S5, test set 30%          0.744  32.2
## 5 Random Forest, sensors S1, S2, S3, S5, test set 30%             0.797  29.1
## 6 Linear model, sensor S4 (targeted to NO2) and Temperature, ~    0.382  44.5
```

The predictions made by this model have the lowest correlation with actual values in spite of use of the values of the sensor nominally targeted to NO2.

### 3.5 RF for benzene (C6H6) pollution

Random Forest model provided the best results for NO2. We could try to predict the level of benzene with the random forest model using the available sensors (we do not have the sensor nominally targeted to benzene in the dataset).

We make the dataframe containing the actual levels of benzene and also the responses of all the sensors, and check the correlation between the responses of each sensor and actual level of pollutant benzene.

```
Benzene_pollution <- AirQuality %>%
  select(Timestamp, `C6H6(GT)`, `PT08.S1(CO)`, `PT08.S2(NMHC)`, `PT08.S3(NOx)`,
         `PT08.S4(NO2)`,  `PT08.S5(O3)`) %>%
  filter(`C6H6(GT)` > 0) %>%
  filter(`PT08.S1(CO)` >0) %>%
  filter(`PT08.S2(NMHC)` >0) %>%
  filter(`PT08.S3(NOx)` >0) %>%
  filter(`PT08.S4(NO2)` >0) %>%
  filter(`PT08.S5(O3)` >0)

cor(Benzene_pollution$`C6H6(GT)`, Benzene_pollution$`PT08.S1(CO)`)
```

```
## [1] 0.8837951
```

```
cor(Benzene_pollution$`C6H6(GT)`, Benzene_pollution$`PT08.S2(NMHC)`)
```

```
## [1] 0.9819503
```

```
cor(Benzene_pollution$`C6H6(GT)`, Benzene_pollution$`PT08.S3(NOx)`)
```

```
## [1] -0.7357441
```

```
cor(Benzene_pollution$`C6H6(GT)`, Benzene_pollution$`PT08.S4(NO2)`)
```

```
## [1] 0.7657314
```

```
cor(Benzene_pollution$`C6H6(GT)`, Benzene_pollution$`PT08.S5(O3)`)
```

```
## [1] 0.8656885
```

The responses of all 5 sensors show relatively high correlation with the level of benzene (positive or negative). We will use Random Forest Model which showed relatively good performance in predicting of NO2 level, and we will use response of sensors nominally targeted to other pollutants (O3, CO, NHMC and NOx).

Test set will be 30% of Benzene_pollution dataframe, the other data (70%) will be used for training (train_set).

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = Benzene_pollution$`C6H6(GT)`, times = 1, p = 0.3,
                                  list = FALSE)
train_set_benzene <- Benzene_pollution[-test_index,]
test_set_benzene <- Benzene_pollution[test_index,]

train_benzene_rf <- train(`C6H6(GT)` ~ `PT08.S1(CO)` + `PT08.S2(NMHC)` +
                            `PT08.S3(NOx)` + `PT08.S4(NO2)` + `PT08.S5(O3)`,
                  method = "rf", data = train_set_benzene)

#prediction on train set
train_set_benzene <- train_set_benzene %>%
  mutate(predicted_benzene_rf = predict(train_benzene_rf, train_set_benzene))

RMSE(train_set_benzene$`C6H6(GT)`, train_set_benzene$predicted_benzene_rf)
```

```
## [1] 0.03218301
```

```
cor(train_set_benzene$`C6H6(GT)`, train_set_benzene$predicted_benzene_rf)
```

```
## [1] 0.9999908
```

```
#prediction on test set
test_set_benzene <- test_set_benzene %>%
  mutate(predicted_benzene_rf = predict(train_benzene_rf, test_set_benzene))

RMSE(test_set_benzene$`C6H6(GT)`, test_set_benzene$predicted_benzene_rf)
```

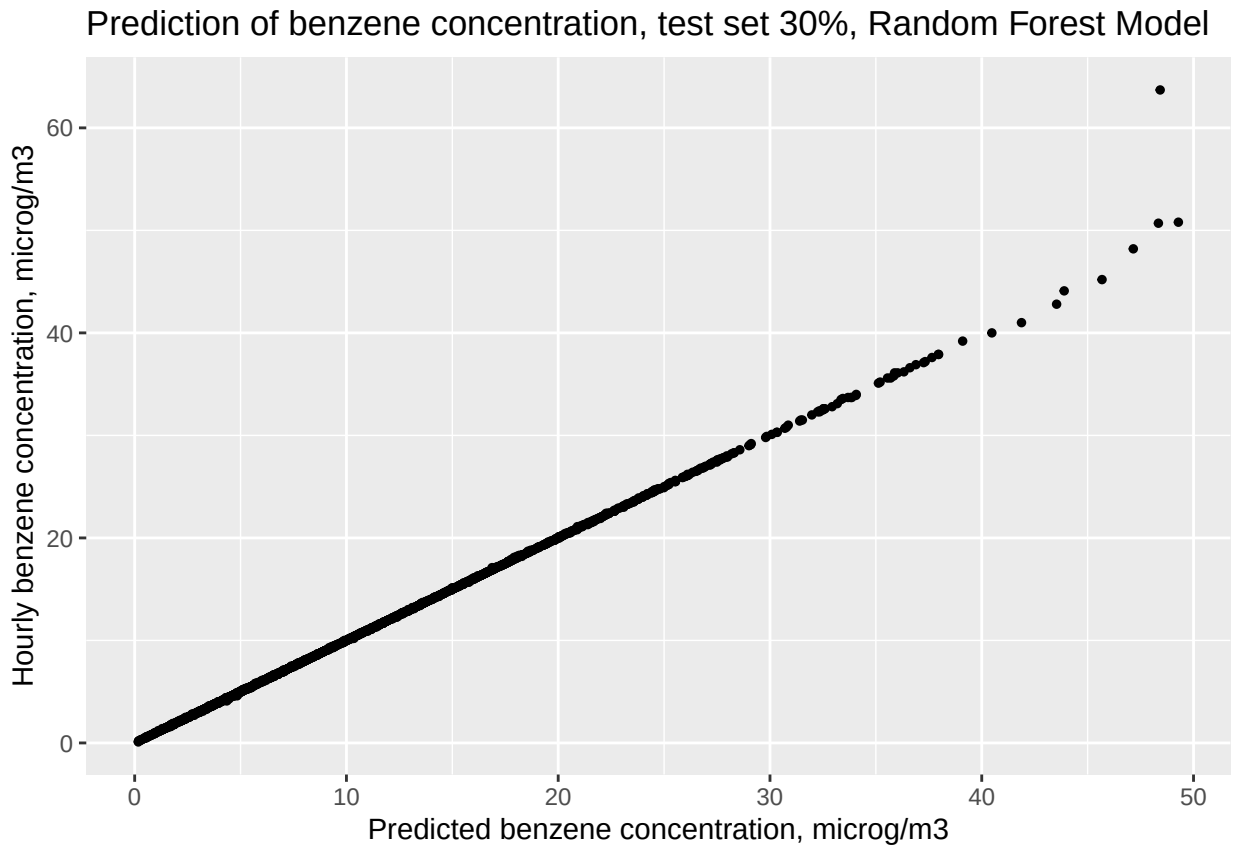
```
## [1] 0.302306
```

```
cor(test_set_benzene$`C6H6(GT)`, test_set_benzene$predicted_benzene_rf)
```

```
## [1] 0.9991759
```

We built the algorithm to predict the level of pollutant benzene with the correlation over 0.999 without use of any sensor nominally targeted to benzene.

```
test_set_benzene %>%
  ggplot(aes(predicted_benzene_rf, `C6H6(GT)`)) +
  geom_point(size = 1) +
  xlab("Predicted benzene concentration, microg/m3") +
  ylab("Hourly benzene concentration, microg/m3") +
  ggtitle("Prediction of benzene concentration, test set 30%, Random Forest Model")
```




## Results

The results of this project are:

1. Proved cross-reference of the response of the sensors and the actual level of pollutants, for which there was no normally targeted sensor at all (benzene) or prediction with the response of normally targeted sensor gave not very good results (NO2).

2. Proved division between train set and test set for this dataset (70% / 30%) as a balance between use of sufficient amount of data from training purposes, but avoiding overtraining of the model.

3. Efficient model to predict the level of NO2 using the effect of cross-reference of sensors normally targeted to different pollutants (random forest model, correlation $> 0.75$).

4. Very efficient model to predict the level of benzene (C6H6) using the effect of cross-reference of sensors with no sensor normally targeted to benzene (random forest model, correlation $> 0.999$).

# Conclusion

We were able to build the models to predict the actual levels of pollution (NO2 and benzene) using a set of sensors nominally targeted to different pollutants with relatively high accuracy.

This could help to develop wide-spread pollution alarm systems using metal oxide multisensors and software to convert the responses of sensors to the level of pollution on the fly instead of use certified analyzers. For example, personal weather stations connected to internet with such sensors owned by multiple users across the country could be used to monitor the pollution in addition with a national pollution monitoring system, and the certified sensors could be used to double check the levels of pollution if several weather stations send the data which is treated as high pollution level.

Nevertheless the results of this project could be used with a certain (unknown) type of metal oxide sensors, and cross-reference of the sensors in this case is a feature, on which the idea is based. Different types of sensors could have different cross-reference, or even do not have cross-reference at all, thus, the results could not be applied to all the sensors available to mass market to monitor the pollution.

We did not use humidity sensors response in our project, for the sensors with low correlation to the level of pollutant to which the sensor is nominally targeted, this could also be a good help to make more accurate predictions regarding the level of pollutant (this is one of the ideas for future work with the model).