Statistical Adaptation of an Ozone Peak Prediction Model

Machine Learning for environement

06/10/2024

ilyas DAHAOUI

Contents

1 Introduction

The objective of this project is to improve the deterministic forecast of ozone concentration provided by Météo-France through the MOCAGE model for certain measurement stations.
This falls within the scope of a statistical adaptation problem with the objective of refinering local forecasts obtained from large-scale models by incorporating additional meteorological variables (temperature, wind speed, etc.) also provided by Météo-France but at a smaller scale.
This approach represents an example of hybrid AI combining a deterministic model with machine learning techniques.

Two types of variables can be predicted: the ozone concentration (quantitative) or the exceedance of a threshold set at $150\,\mu g/m^3$ (qualitative)
Two strategies are considered:
-Ozone concentration prediction: First modeling the ozone concentration (regression), and then deriving the potential exceedance of the threshold. -Direct exceedance prediction: Directly modeling the occurrence of the exceedance (binary classification or logistic regression).

The question, therefore, is: what are the most effective methods and strategies to predict, on one hand, the ozone concentration for the following day, and on the other hand, the occurrence of a pollution peak?

2 Descriptive statistics and linear models

2.1 Data munging

The data were extracted and formatted by the relevant department of Météo France. They are described by the following variables :

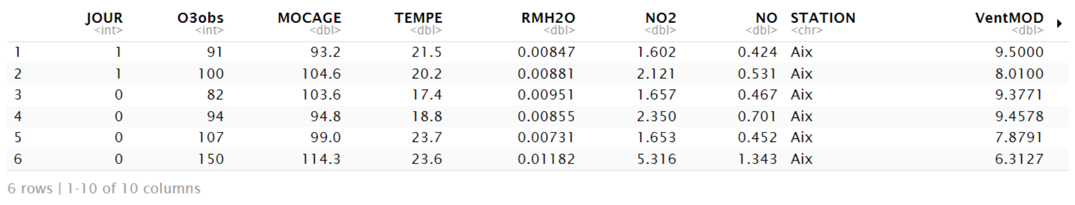
Variable	**Description**
JOUR	Type of day: holiday (1) or not (0)
O3obs	Observed ozone concentration at 5 pm the next day
MOCAGE	Forecast ozone concentration (deterministic model)
TEMPE	Forecast temperature at 5 pm the next day
RMH2O	Humidity ratio
NO2	Nitrogen dioxide concentration
NO	Nitrogen monoxide concentration
STATION	Observation location (e.g., Aix-en-Provence, etc.)
WindMOD	Wind strength
WindANG	Wind direction

Table 1: Description of variables used for ozone concentration forecasting

These are "clean" data, without missing values, well coded and small in size. They are therefore primarily of an educational nature, as they allow us to use and compare all the regression and supervised classification approaches.
Even if the data are of good quality, a preliminary exploratory study is always necessary to become familiar with the data and prepare them for the modeling phase.

2.1.1 Data Exploration

2.1.2 Reading the Data:



	JOUR <int>	O3obs <int>	MOCAGE <dbl>	TEMPE <dbl>	RMH2O <dbl>	NO2 <dbl>	NO <dbl>	STATION <chr>	VentMOD <dbl>
1	1	91	93.2	21.5	0.00847	1.602	0.424	Aix	9.5000
2	1	100	104.6	20.2	0.00881	2.121	0.531	Aix	8.0100
3	0	82	103.6	17.4	0.00951	1.657	0.467	Aix	9.3771
4	0	94	94.8	18.8	0.00855	2.350	0.701	Aix	9.4578
5	0	107	99.0	23.7	0.00731	1.653	0.452	Aix	7.8791
6	0	150	114.3	23.6	0.01182	5.316	1.343	Aix	6.3127

6 rows | 1-10 of 10 columns

Figure 1: Ozon Peak Data

2.1.3 Unidimensional statistics:

we can specify the nature of the different variables and the aspects to consider when studying their distribution:

Type of Variable	Variables and Characteristics
Categorical variables	**JOUR, STATION** – Their distribution is non-numerical, so symmetry isn't relevant.
Continuous numerical variables	**O3obs, MOCAGE, TEMPE, RMH2O, NO2, NO, WindMOD** – Study their distribution for symmetry or skewness (especially right-skewness in pollutant concentrations and weather variables like humidity or wind).
Circular variable	**WindANG** – Requires specific methods to assess distribution (not symmetry in the traditional sense).

Table 2: Summary of variable types and characteristics

The distribution of variables:

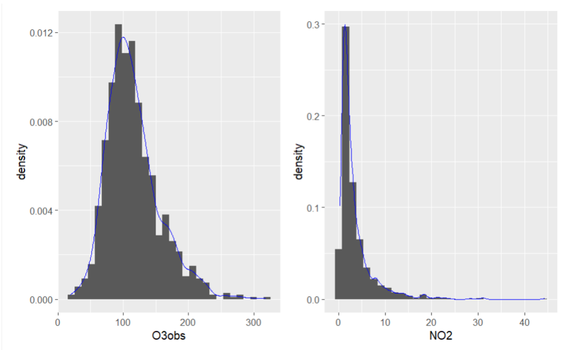


Figure 2: Density of variables

Transformations are suggested to make certain distributions more symmetric and closer to a Gaussian shape by using sqrt or log functions. This is important for some

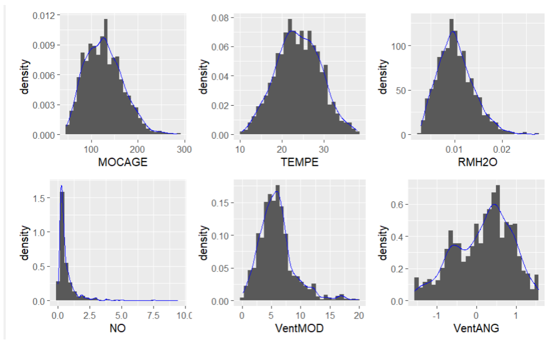
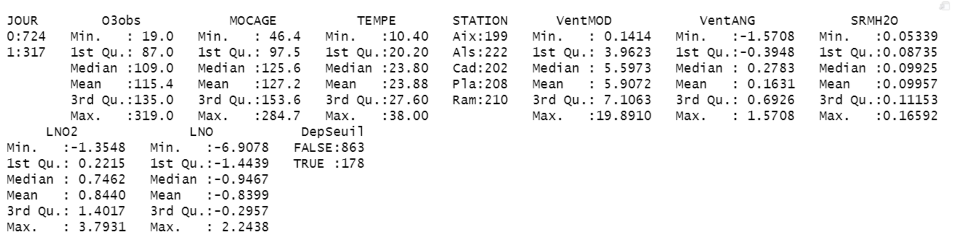


Figure 3: Density of variables

upcoming modeling methods, such as linear models, but not essential for others, like decision trees.

We now create the DepSeuil threshold variable, which will be used in the final dataset. This variable is based on whether the observed ozone concentration (O3obs) exceeds the threshold of 150 $\mu g/m^3$, which transforms the problem into a binary classification: exceedance (TRUE) or no exceedance (FALSE).



```
JOUR        O3obs           MOCAGE          TEMPE        STATION      VentMOD          VentANG            SRMH2O
0:724   Min.   : 19.0   Min.   : 46.4   Min.   :10.40   Aix:199   Min.   : 0.1414   Min.   :-1.5708   Min.   :0.05339
1:317   1st Qu.: 87.0   1st Qu.: 97.5   1st Qu.:20.20   Als:222   1st Qu.: 3.9623   1st Qu.:-0.3948   1st Qu.:0.08735
        Median :109.0   Median :125.6   Median :23.80   Cad:202   Median : 5.5973   Median : 0.2783   Median :0.09925
        Mean   :115.4   Mean   :127.2   Mean   :23.88   Pla:208   Mean   : 5.9072   Mean   : 0.1631   Mean   :0.09957
        3rd Qu.:135.0   3rd Qu.:153.6   3rd Qu.:27.60   Ram:210   3rd Qu.: 7.1063   3rd Qu.: 0.6926   3rd Qu.:0.11153
        Max.   :319.0   Max.   :284.7   Max.   :38.00             Max.   :19.8910   Max.   : 1.5708   Max.   :0.16592
     LNO2               LNO            DepSeuil
Min.   :-1.3548   Min.   :-6.9078   FALSE:863
1st Qu.: 0.2215   1st Qu.:-1.4439   TRUE :178
Median : 0.7462   Median :-0.9467
Mean   : 0.8440   Mean   :-0.8399
3rd Qu.: 1.4017   3rd Qu.:-0.2957
Max.   : 3.7931   Max.   : 2.2438
```

Figure 4: Variables after all transformations

2.1.4 Correlations of the variables:

the relationship of the variables 2 to 2:

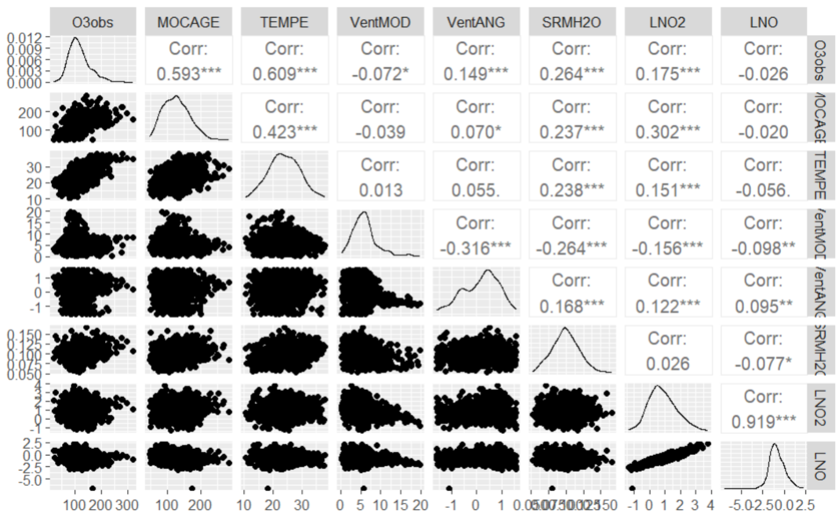


Figure 5: Correlation of the variables

The strongest positive correlations are observed between O3obs and MOCAGE (0.593), TEMPE (0.609), and SRMH2O (0.264), while NO2 and NO have an extremely high correlation (0.919)

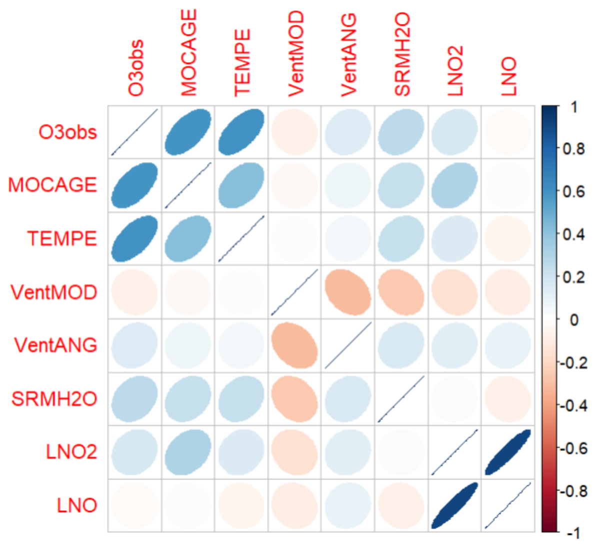


Figure 6: Correlation of the variables using corrplot

While corrplot() provides useful insights into linear correlations, its limitation lies in measuring only linear correlations (Pearson correlation by default), and it may overlook non-linear relationships. Other correlation measures, such as Spearman or Kendall, might be more appropriate when dealing with ordinal data or non-linear associations.

2.1.5 Principal Components Analysis:

The following commands perform a principal component analysis (PCA) on the quantitative variables only, excluding the target variable O3obs (observed ozone concentration) from the analysis.

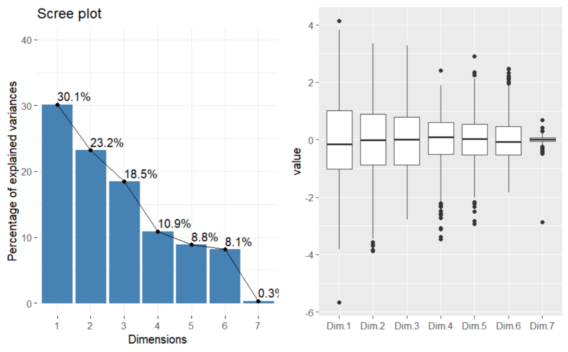


Figure 7: POEV and Boxplots by dimensions

together in Dimensions 1 to 3, they account for 71.8% of the total variance (30.1% + 23.2% + 18.5%). This indicates that these first three dimensions capture a significant portion of the underlying variability in the data, making them highly informative.

Dimension 1 and Dimension 2 show a good spread and variability, suggesting they capture meaningful information. While Dimension 3 still shows variability, it starts to flatten out and has more outliers indicating that it might not be as strong as the first two dimensions.

we should still retain Dimensions 1, 2, and 3, but we might also want to consider

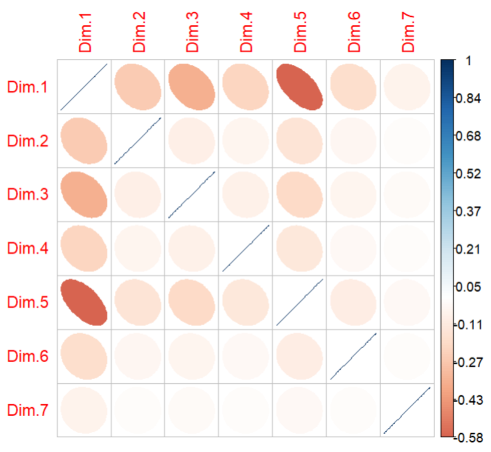


Figure 8: The correlation of variables by dimensions

Dimension 5 for certain modeling tasks due to its strong negative correlation with Dim. 1 and its ability to capture additional variance patterns.

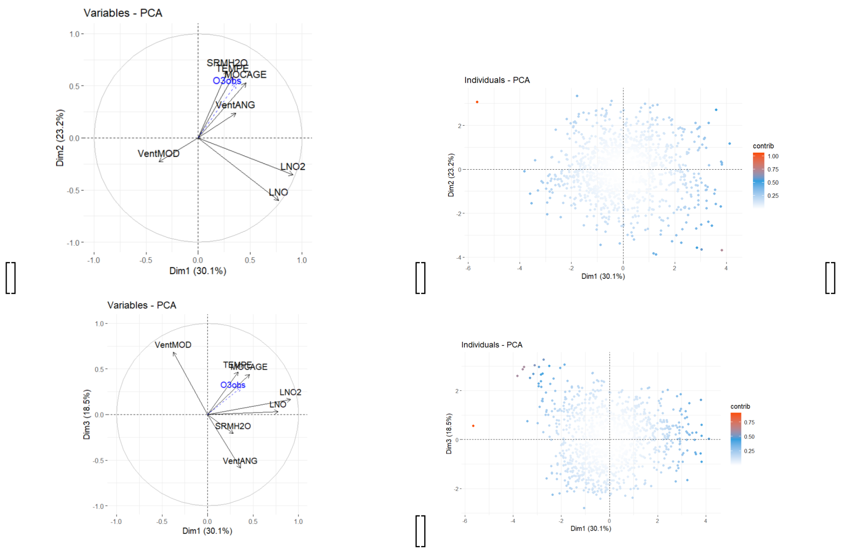


Figure 9: the contribution of each individual by dimension.

In the variable plots, we observe that the contributions of specific variables, such as O3obs, are highlighted in blue, indicating they have a significant impact on the first two principal components. This suggests that the observed ozone concentration is closely linked to the dimensions capturing the majority of the variance in the data.

In the individuals' plots, the distribution of observations is represented with colors indicating their contributions. The darker shades represent higher contributions, revealing that certain observations are more influential in defining the principal components. These observations may represent outliers or significant data points that exhibit unique characteristics within the data set.

Variable Contributions: The variable plots indicate that certain variables, particularly O3obs, MOCAGE, and TEMPE, contribute significantly to the first two principal components. This suggests that these variables are crucial in explaining the variance in the dataset. The strong relationships between these variables may imply that they interact closely in the context of ozone concentration.

Dimensionality Reduction: The PCA effectively reduces the dimensionality of the data while retaining a significant amount of variance. The first two dimensions account for a combined total of over 50% of the explained variance, making them essential for understanding the data's structure.

Correlation Structure: The correlation structure identified in the variable plots aligns with the contributions observed in the individuals' plots. This coherence reinforces the idea that the variables that explain the most variance are closely related, potentially indicating common underlying factors influencing ozone levels.
Atypical Values: The presence of certain observations that stand out in the individuals' plot may warrant further investigation. These atypical values could provide insights into specific events or conditions that lead to elevated ozone levels or deviations from the expected patterns.

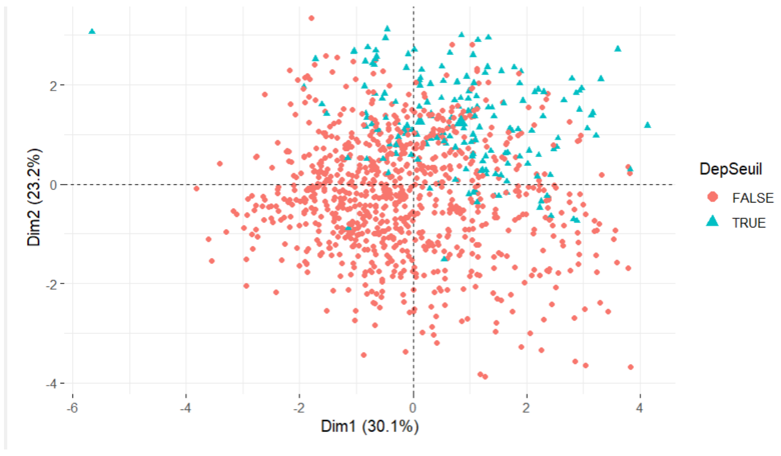


Figure 10: dimension 1 and 2 / DeepSeuil

The graph clearly indicates a separation between the two groups represented by TRUE (blue triangles) and FALSE (red circles). This suggests that the underlying dimensions represented by Dim1 and Dim2 are effective in distinguishing between these two conditions.

3 Model developpement

3.1 Prediction by linear Gaussian model:

The initial model to be evaluated is a simple linear Gaussian model; however, due to the presence of qualitative variables, it is treated as an analysis of covariance (ANCOVA). Additionally, we want to determine whether interactions should be considered in the analysis. Consequently, the model is extended to a quadratic form, incorporating polynomial terms of degree 2.

The linear model integrates categorical variables; it is in this case an analysis of covariance estimated by the aov function better adapted to this model.

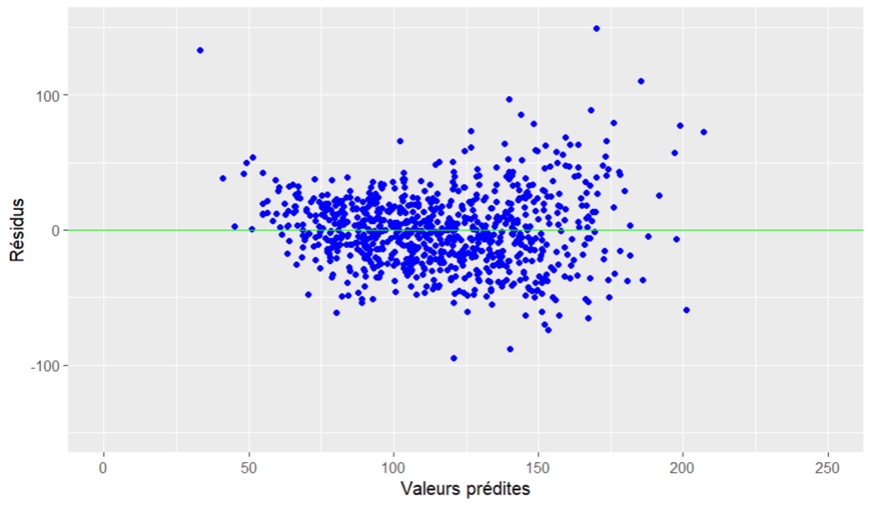
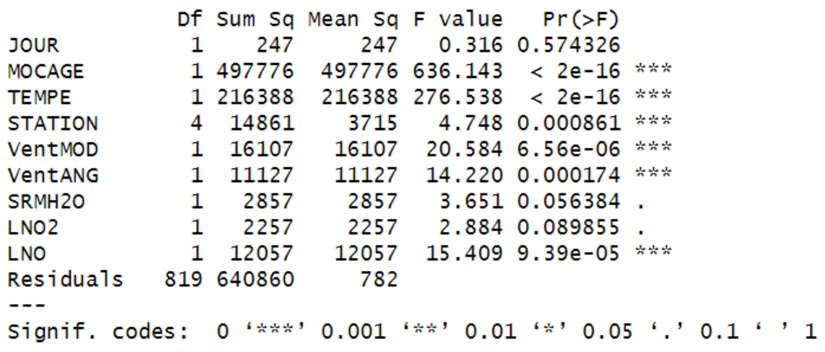


Figure 11: ANCOVA

To analyze the residuals of a regression model, we typically look at a residual plot, which plots the residuals (errors) against the predicted values. This allows us to check the assumptions of linear regression, particularly linearity and homoscedasticity. Let's break down the analysis based on residual plot.
The residuals are plotted against the predicted values, and the spread appears to be somewhat symmetrical around the horizontal line at zero. This is a good indication that the model is capturing the underlying data well, as the residuals are centered around zero.
The points appear to be randomly scattered without any apparent pattern, which suggests that the linearity assumption of the model is likely valid
Given that the residuals appear to be randomly distributed around zero and show no signs of non-linearity or heteroscedasticity, we can conclude that the model's assumptions are satisfied. This supports the validity of the linear regression model used. The linear regression model can be considered a good fit for the data.

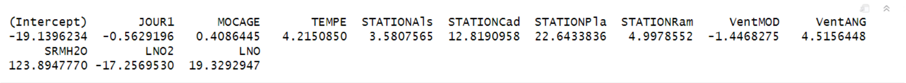
Summary;



```
               Df Sum Sq Mean Sq F value    Pr(>F)    
JOUR            1    247     247   0.316 0.574326    
MOCAGE          1 497776  497776 636.143  < 2e-16 ***
TEMPE           1 216388  216388 276.538  < 2e-16 ***
STATION         4  14861    3715   4.748 0.000861 ***
VentMOD         1  16107   16107  20.584 6.56e-06 ***
VentANG         1  11127   11127  14.220 0.000174 ***
SRMH2O          1   2857    2857   3.651 0.056384 .  
LNO2            1   2257    2257   2.884 0.089855 .  
LNO             1  12057   12057  15.409 9.39e-05 ***
Residuals     819 640860     782                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

[h]

Regression coefficients:



```
(Intercept)       JOUR1      MOCAGE       TEMPE  STATIONAls  STATIONCad  STATIONPla  STATIONRam     VentMOD     VentANG 
-19.1396234  -0.5629196   0.4086445   4.2150850   3.5807565  12.8190958  22.6433836   4.9978552  -1.4468275   4.5156448 
     SRMH2O        LNO2         LNO 
123.8947770 -17.2569530  19.3292947 
```

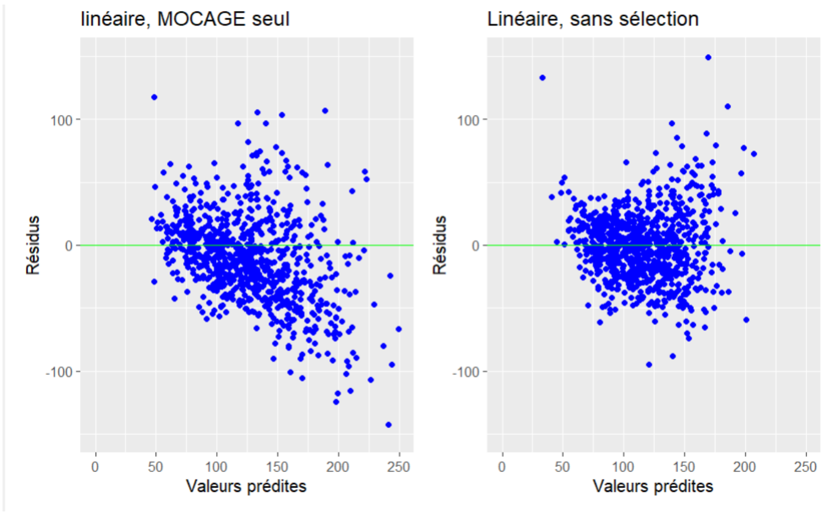


Figure 12: deterministic forecast MOCAGE Vs Linear Model

The left plot displays residuals for the MOCAGE model, showing a discernible pattern. Specifically, there is a noticeable negative trend where the residuals tend to decrease as the predicted values increase In contrast, the right plot displays the residuals from the model without selection, which appear more centered around zero without any evident trends.

This pattern suggests potential issues with model fit, indicating that the MOCAGE model may not fully capture the complexity of the data. The presence of this trend implies that the model could be underestimating the dependent variable at higher values, indicating a systematic bias.

the comparison between the two models illustrates that the model without selection is more reliable, with residuals showing no systematic bias, whereas the MOCAGE model exhibits a concerning trend in the residuals. This suggests a need for refinement in the MOCAGE model to enhance its predictive accuracy.

3.1.1 Variable selection with L1 regularisation (LASSO):

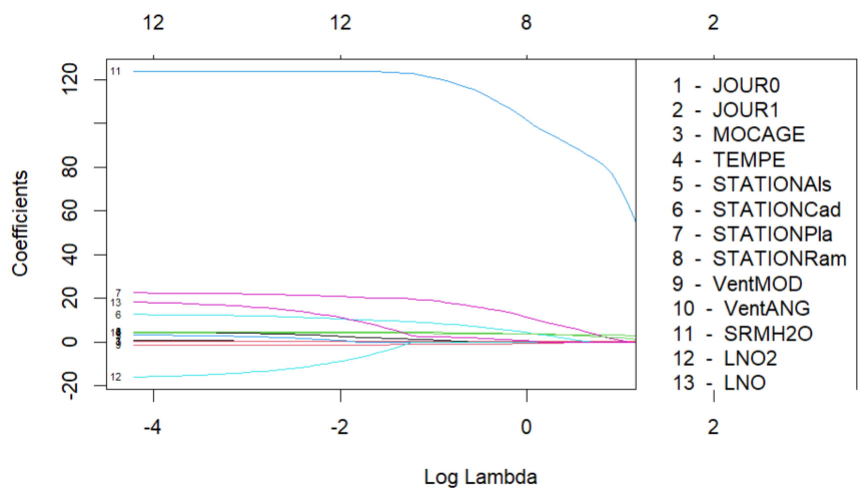


Figure 13: Lasso Regression: Coefficient Paths Across Lambda Values

As λ increases we can observe that Most coefficients decrease in magnitude, indicating that they are being shrunk toward zero due to the penalty applied by Lasso. This is a key feature of Lasso regression, which helps in feature selection by driving some coefficients exactly to zero.

Certain predictors, especially those with less predictive power, become exactly zero for larger values of λ. This indicates that these predictors are not contributing to the model at those λ values and can be effectively eliminated from the model.

-The bold dots on the plot represent the **Mean Squared Error** (MSE) values for different values of the regularization parameter λ, plotted on a logarithmic scale ($\log(\lambda)$). Each dot corresponds to the mean cross-validated error for a specific value of λ.

- As λ changes, the model becomes more or less regularized, leading to different levels of fit. For small λ values (left side), the model is less regularized, allowing more coefficients to have non-zero values, which can lead to *overfitting*. - As λ increases (right side), more regularization is applied, shrinking the coefficients toward zero, which may

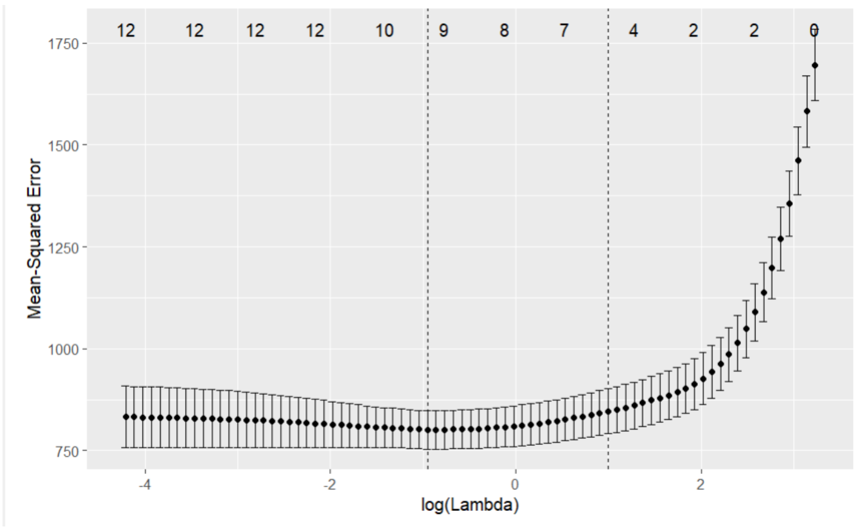
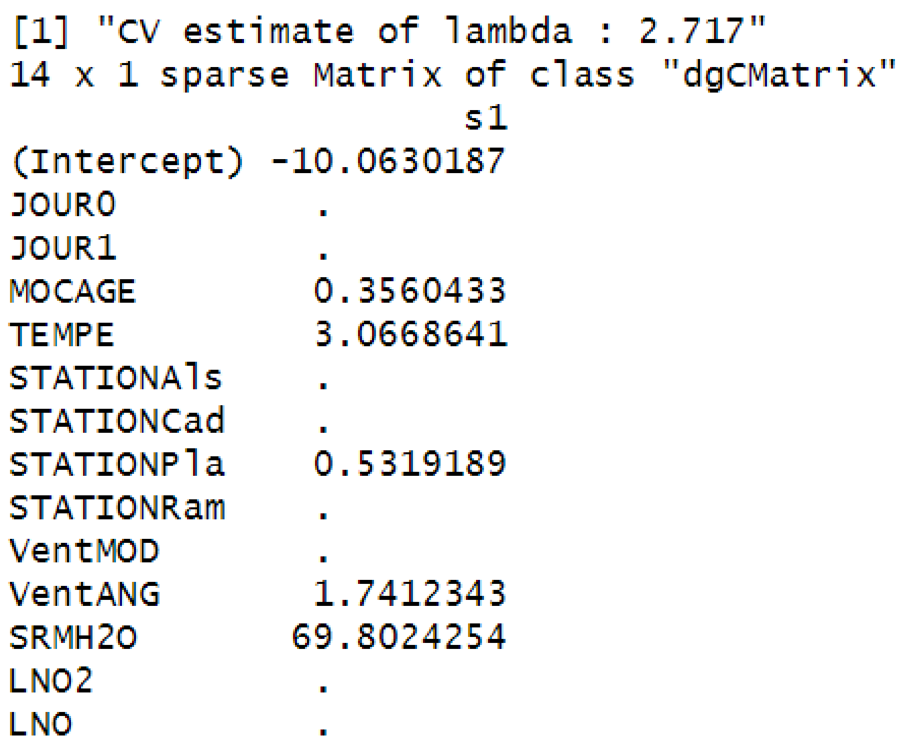


Figure 14: MSE Vs Log(lambda)

cause *underfitting.*

In the final model with =2.717, only 5 variables retain non-zero coefficients, and the remaining ones are shrunk to zero due to the regularization, meaning they are effectively excluded from the model.



```
[1] "CV estimate of lambda : 2.717"
14 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept) -10.0630187
JOUR0          .
JOUR1          .
MOCAGE         0.3560433
TEMPE          3.0668641
STATIONAls     .
STATIONCad     .
STATIONPla     0.5319189
STATIONRam     .
VentMOD        .
VentANG        1.7412343
SRMH2O        69.8024254
LNO2           .
LNO            .
```

Figure 15: coeff estimated values

for λmin:

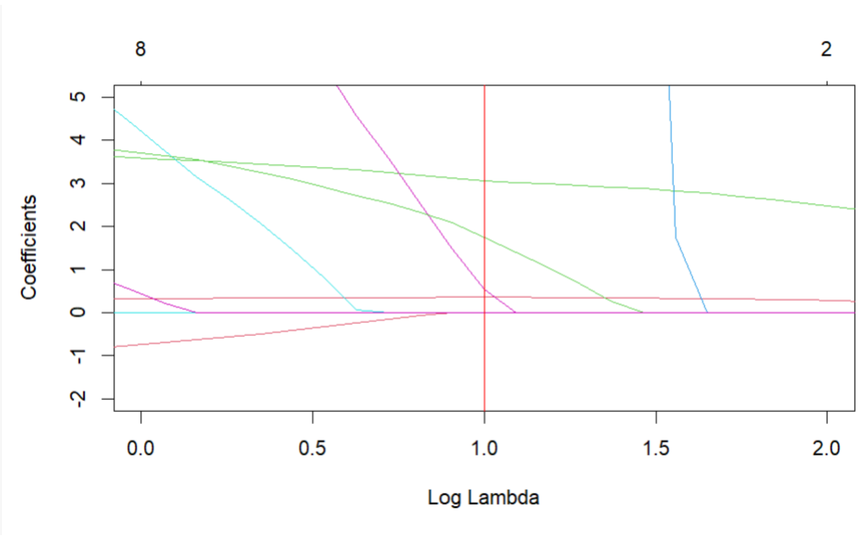
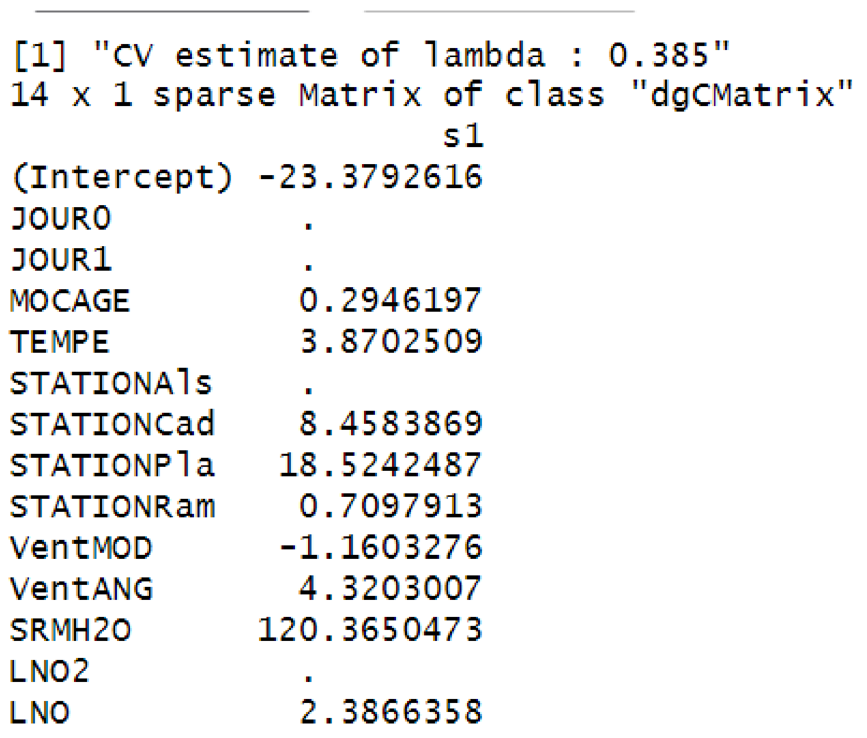


Figure 16: Regularization Paths for LASSO Regression with Optimal λ

We then plot the residuals against the predicted values:

In this model, no penalty is applied, so all predictors are used with their full influence. The plot of residuals vs predicted values may show overfitting, with residuals potentially more scattered or showing patterns if the model fits too closely to the training data, leading to poor generalization. The plot could also highlight heteroscedasticity or non-linearity in the residuals, indicating issues with the model assumptions of constant variance or linearity.

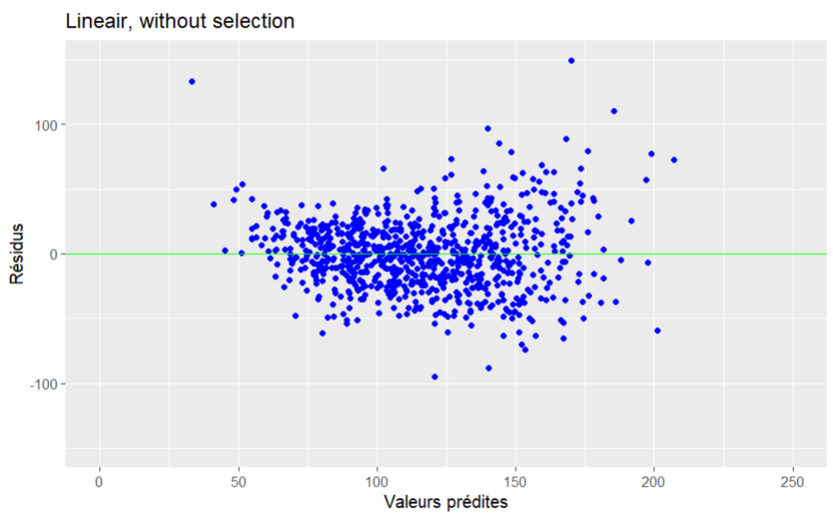


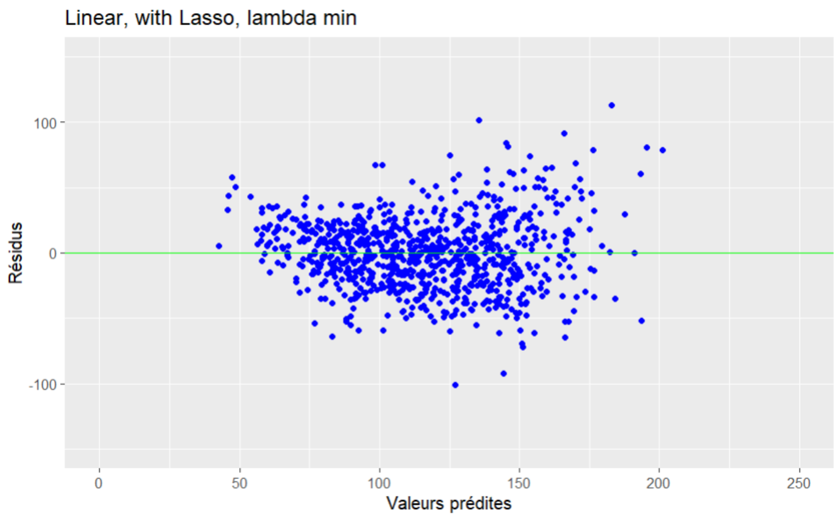
```
[1] "CV estimate of lambda : 0.385"
14 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept) -23.3792616
JOUR0         .
JOUR1         .
MOCAGE        0.2946197
TEMPE         3.8702509
STATIONAls    .
STATIONCad    8.4583869
STATIONPla   18.5242487
STATIONRam    0.7097913
VentMOD      -1.1603276
VentANG       4.3203007
SRMH2O      120.3650473
LNO2          .
LNO           2.3866358
```

This is the point where the mean-squared error (MSE) from cross-validation is minimized. The resulting model is more regularized than the unpenalized model, meaning some coefficients are shrunk or set to zero. The plot of residuals vs predicted values with min should show an improvement over the ”no selection” model. The residuals are likely to be more randomly scattered, indicating a better fit that avoids overfitting. However, the model may still include more variables than strictly necessary.

-This is the point where the cross-validated MSE is within one standard error of the minimum. The model is even more regularized than the λ min model, often selecting fewer variables to avoid overfitting.

-In the residuals vs predicted values plot for λse we expect fewer predictors, and as a result, the residuals should still be randomly distributed. However, the model may sacrifice some predictive accuracy in exchange for greater simplicity and robustness.



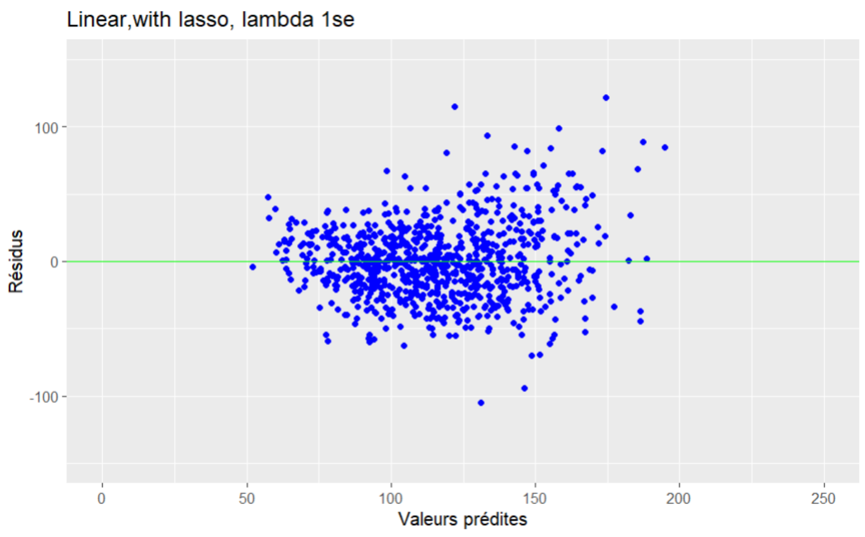


We computed the Mean Squared Error (MSE) for three different cases,The results are presented in the table below:

Model	**MSE**
Without Selection	770.26
LASSO (λ_{min})	783.31
LASSO (λ_{1se})	836.39

Table 3: Mean Squared Error for different models

The MSE obtained by LASSO is slightly worse than the model without selection because LASSO introduces a regularization penalty to shrink some coefficients, potentially reducing the model's ability to fit the training data as closely as an unregularized model. However, this trade-off is intentional: LASSO aims to prevent overfitting and improve the model's performance on new, unseen data by encouraging sparsity (setting some coefficients to zero). As a result, the MSE on the training data might be higher, but the



model could generalize better on test data.

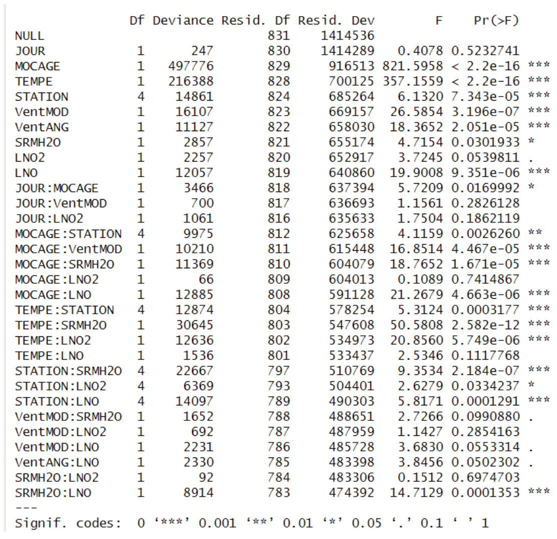
3.2 Prediction By Quadratic model

The following study incorporates all second-order interactions between the variables, effectively creating a quadratic regression model. This model is estimated using the ‘glm()‘ function, which supports automatic model selection. The backward elimination method is applied, although the stepwise selection method could also be utilized. However, this type of procedure is not available in Python.

3.2.1 Variable selection with AICA (Akaïke Information Criterion)

Backward selection involves comparing the current model at each step to all possible submodels, which are created by removing either one interaction or one variable ensuring the variable isn’t part of an interaction. The variable or interaction that, when deleted, results in the greatest reduction of the AIC (Akaike Information Criterion) is selected for removal.

In the Gaussian case with known residual variance, another criterion equivalent to AIC is the Bayesian Information Criterion (BIC). Like AIC, BIC is used for model selection, but it introduces a stronger penalty for the number of parameters, which makes it more conservative when choosing models with a higher number of variables. Both AIC and BIC aim to balance model fit and complexity, but BIC tends to prefer simpler models when the sample size is large.

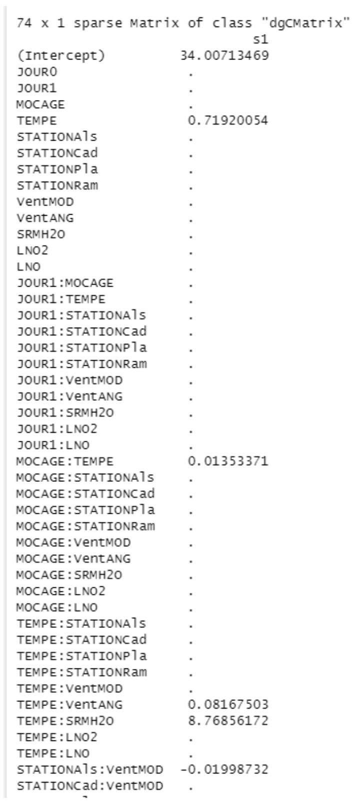


```
                Df Deviance Resid. Df Resid. Dev        F    Pr(>F)    
NULL                              831    1414536                       
JOUR             1      247       830    1414289   0.4078 0.5232741    
MOCAGE           1   497776       829     916513 821.5958 < 2.2e-16 ***
TEMPE            1   216388       828     700125 357.1559 < 2.2e-16 ***
STATION          4    14861       824     685264   6.1320 7.343e-05 ***
VentMOD          1    16107       823     669157  26.5854 3.196e-07 ***
VentANG          1    11127       822     658030  18.3652 2.051e-05 ***
SRMH2O           1     2857       821     655174   4.7154 0.0301933 *  
LNO2             1     2257       820     652917   3.7245 0.0539811 .  
LNO              1    12057       819     640860  19.9008 9.351e-06 ***
JOUR:MOCAGE      1     3466       818     637394   5.7209 0.0169992 *  
JOUR:VentMOD     1      700       817     636693   1.1561 0.2826128    
JOUR:LNO2        1     1061       816     635633   1.7504 0.1862119    
MOCAGE:STATION   4     9975       812     625658   4.1159 0.0026260 ** 
MOCAGE:VentMOD   1    10210       811     615448  16.8514 4.467e-05 ***
MOCAGE:SRMH2O    1    11369       810     604079  18.7652 1.671e-05 ***
MOCAGE:LNO2      1       66       809     604013   0.1089 0.7414867    
MOCAGE:LNO       1    12885       808     591128  21.2679 4.663e-06 ***
TEMPE:STATION    4    12874       804     578254   5.3124 0.0003177 ***
TEMPE:SRMH2O     1    30645       803     547608  50.5808 2.582e-12 ***
TEMPE:LNO2       1    12636       802     534973  20.8560 5.749e-06 ***
TEMPE:LNO        1     1536       801     533437   2.5346 0.1117768    
STATION:SRMH2O   4    22667       797     510769   9.3534 2.184e-07 ***
STATION:LNO2     4     6369       793     504401   2.6279 0.0334237 *  
STATION:LNO      4    14097       789     490303   5.8171 0.0001291 ***
VentMOD:SRMH2O   1     1652       788     488651   2.7266 0.0990880 .  
VentMOD:LNO2     1      692       787     487959   1.1427 0.2854163    
VentMOD:LNO      1     2231       786     485728   3.6830 0.0553314 .  
VentANG:LNO      1     2330       785     483398   3.8456 0.0502302 .  
SRMH2O:LNO2      1       92       784     483306   0.1512 0.6974703    
SRMH2O:LNO       1     8914       783     474392  14.7129 0.0001353 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 17: Model coeff after variable selectionn by AIC

3.2.2 Variable selection by L1 regularisation (LASSO):

after application of L1-regularization the result is on the figure below:



```
74 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept)          34.00713469
JOUR0                  .
JOUR1                  .
MOCAGE                 .
TEMPE                 0.71920054
STATIONAls             .
STATIONCad             .
STATIONPla             .
STATIONRam             .
VentMOD                .
VentANG                .
SRMH2O                 .
LNO2                   .
LNO                    .
JOUR1:MOCAGE           .
JOUR1:TEMPE            .
JOUR1:STATIONAls       .
JOUR1:STATIONCad       .
JOUR1:STATIONPla       .
JOUR1:STATIONRam       .
JOUR1:VentMOD          .
JOUR1:VentANG          .
JOUR1:SRMH2O           .
JOUR1:LNO2             .
JOUR1:LNO              .
MOCAGE:TEMPE          0.01353371
MOCAGE:STATIONAls      .
MOCAGE:STATIONCad      .
MOCAGE:STATIONPla      .
MOCAGE:STATIONRam      .
MOCAGE:VentMOD         .
MOCAGE:VentANG         .
MOCAGE:SRMH2O          .
MOCAGE:LNO2            .
MOCAGE:LNO             .
TEMPE:STATIONAls       .
TEMPE:STATIONCad       .
TEMPE:STATIONPla       .
TEMPE:STATIONRam       .
TEMPE:VentMOD          .
TEMPE:VentANG         0.08167503
TEMPE:SRMH2O          8.76856172
TEMPE:LNO2             .
TEMPE:LNO              .
STATIONAls:VentMOD   -0.01998732
STATIONCad:VentMOD     .
```

Figure 18: Model Coeff after Lasso regularization

to compare the results of regularization we can visualize the graphs below:

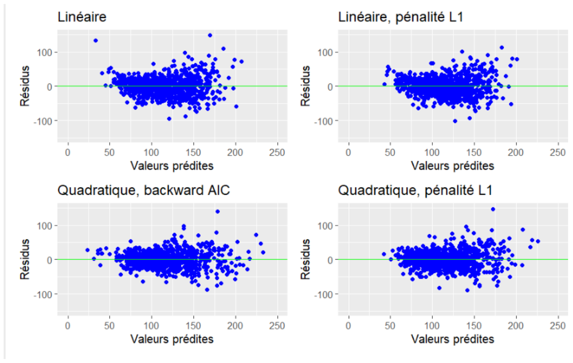


Figure 19: trend of the model for each regularization

We notice that the presence of some interactions or variables are relevant according to the Akaïke criterion but not significant according to the Fisher test. This presence in the model could be more finely analyzed by considering an estimate of the error by cross-validation. The idea would be to remove one by one the least significant variables or interactions to see how the cross-validation behaves. On the other hand, if the stepwise procedure leads to a different model, the estimation of the error by cross-validation also allows to optimize the choice.

These refinements are not efficient on these data. The model obtained by minimizing the AIC criterion is kept.

3.2.3 Prediction of the test simple:

Model	**MSE**
Without Selection	770.26
LASSO (lambda.min)	783.31
LASSO (lambda.1se)	836.39
Regression Error (AIC)	**611.28**
Quadratic Error (MOCAGE)	**1496.41**

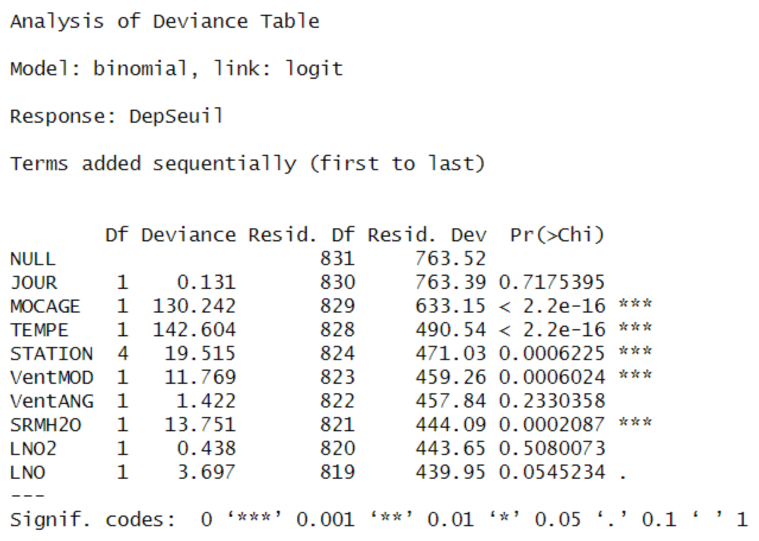
Table 4: Comparison of MSE for different models

-Regression Error (AIC): This model achieves the lowest MSE (611.28), which suggests it has the best performance in terms of predictive accuracy among the models tested.
-LASSO Models: Both LASSO models (lambda.min and lambda.1se) have higher MSE values compared to the regression error (AIC), indicating they are less accurate in this case. The regularization imposed by LASSO may have resulted in some loss of predictive power, despite reducing the model complexity.
-Quadratic Error for MOCAGE: The quadratic error for MOCAGE is the highest (1496.41), implying it has the worst performance in terms of MSE, possibly due to overfitting or a poor fit with the data.

3.3 Binomial Model Prediction

Rather than predicting the concentration and then the exceedance, we can ask ourselves if it would not be relevant to directly predict the presence or absence of an exceedance. Since the variable to be modeled is binary, logistic regression will be used. As for regression, different model selection strategies can be used and compared before estimating the prediction error on the test sample.

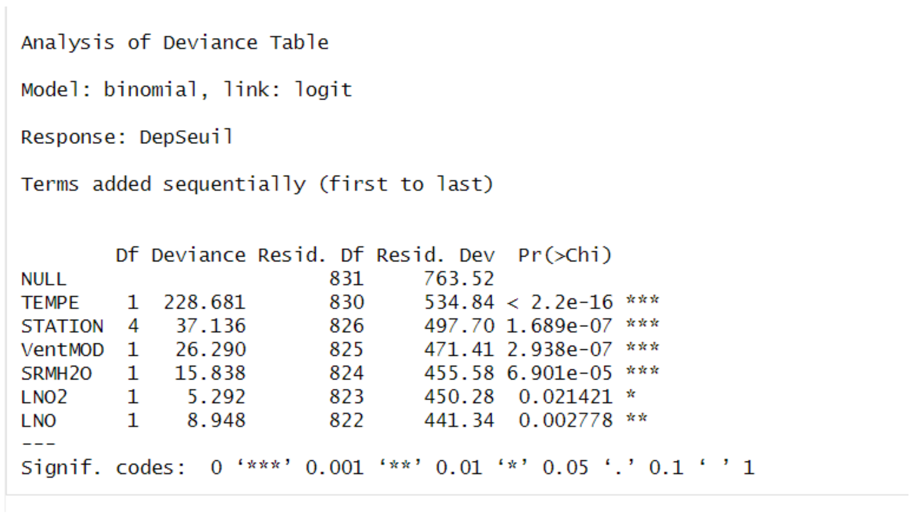
3.3.1 Logistic regression without interaction:



```
Analysis of Deviance Table

Model: binomial, link: logit

Response: DepSeuil

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                      831     763.52
JOUR     1    0.131       830     763.39 0.7175395
MOCAGE   1  130.242       829     633.15 < 2.2e-16 ***
TEMPE    1  142.604       828     490.54 < 2.2e-16 ***
STATION  4   19.515       824     471.03 0.0006225 ***
VentMOD  1   11.769       823     459.26 0.0006024 ***
VentANG  1    1.422       822     457.84 0.2330358
SRMH2O   1   13.751       821     444.09 0.0002087 ***
LNO2     1    0.438       820     443.65 0.5080073
LNO      1    3.697       819     439.95 0.0545234 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 20: full Model estimation:

After optimizing the previews model by taking advantges of the Akaïke Information Criterion (AIC) we arrive to the optimal model with the parameters as defined below:

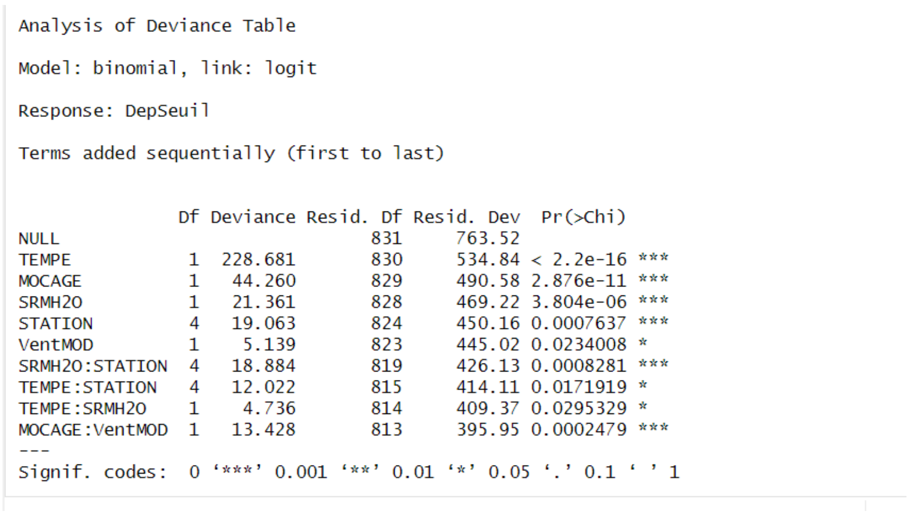


```
Analysis of Deviance Table

Model: binomial, link: logit

Response: DepSeuil

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                      831     763.52
TEMPE    1  228.681       830     534.84 < 2.2e-16 ***
STATION  4   37.136       826     497.70 1.689e-07 ***
VentMOD  1   26.290       825     471.41 2.938e-07 ***
SRMH2O   1   15.838       824     455.58 6.901e-05 ***
LNO2     1    5.292       823     450.28  0.021421 *
LNO      1    8.948       822     441.34  0.002778 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 21: AIC Optimal mmodel

3.3.2 Logistic regression with interactions

With so many variables and interactions, and therefore parameters, the estimation of the complete logistic regression model encounters problems and displays warnings because some well-fitted probabilities (0 or 1) cause divisions by 0. Here a forward or better stepwise procedure for selecting the variables and interactions leads to reasonable results. A method with L1 penalization can also be used.



```
Analysis of Deviance Table

Model: binomial, link: logit

Response: DepSeuil

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                             831     763.52
TEMPE           1  228.681       830     534.84 < 2.2e-16 ***
MOCAGE          1   44.260       829     490.58 2.876e-11 ***
SRMH2O          1   21.361       828     469.22 3.804e-06 ***
STATION         4   19.063       824     450.16 0.0007637 ***
VentMOD         1    5.139       823     445.02 0.0234008 *
SRMH2O:STATION  4   18.884       819     426.13 0.0008281 ***
TEMPE:STATION   4   12.022       815     414.11 0.0171919 *
TEMPE:SRMH2O    1    4.736       814     409.37 0.0295329 *
MOCAGE:VentMOD  1   13.428       813     395.95 0.0002479 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 22: AIC Optimal Model with interactions

3.3.3 Confusion Matrix:

	FALSE	**TRUE**
FALSE	168	20
TRUE	6	15

Table 5: Confusion Matrix

3.3.4 ROC curv:

It is also possible to construct a ROC curve in association with the prediction obtained from a Gaussian linear model. Indeed, the variation of the theoretical threshold of over-taking (150) will vary the respective proportions of the true and false positive rates. This is the same as varying the threshold of a ”proba” for the forecast values divided by 300.

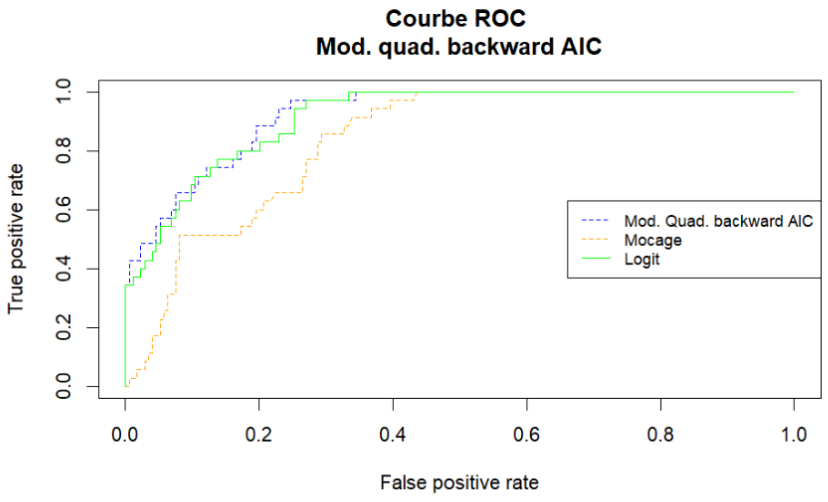


Figure 23: ROC curv

$$\text{Sensitivity} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

Sensitivity measures the proportion of actual positives that are correctly identified by the model.

$$\text{Specificity} = \frac{\text{True Negatives (TN)}}{\text{True Negatives (TN)} + \text{False Positives (FP)}}$$

Specificity measures the proportion of actual negatives that are correctly identified by the model.

The performances of the Gaussian and binomial approaches differ based on the type of problem. The Gaussian approach (linear regression) is ideal for continuous predictions, assuming a linear relationship between variables, and its performance is typically evaluated using metrics like Mean Squared Error (MSE) or Akaike Information Criterion (AIC). However, it may struggle with non linear relationships. The binomial approach (logistic regression) is designed for binary classification tasks, predicting probabilities of outcomes like threshold exceedances, and is evaluated using metrics such as sensitivity, specificity, and the Area Under the ROC Curve (AUC). While the Gaussian model is better suited for continuous data, the binomial model excels in classification tasks, particularly when analyzing whether a certain threshold is exceeded. Therefore, their performances are different and not directly comparable, each excelling in its respective domain.

3.4 Discriminant Analysis, KNN, SVM

3.4.1 Discriminant Analysis

The objective is to compare the three discriminant analysis methods available in R:lda linear parametric lda (homoscedasticity), quadratic parametric qda (heteroscedasticity) under Gaussian assumption and non-parametric nearest neighbors K.

In Linear Discriminant Analysis (LDA), the assignment criterion is based on the maximization of the posterior probability. Specifically, an observation is assigned to the class with the highest posterior probability, which is calculated using Bayes' theorem. LDA assumes that the data follows a multivariate normal distribution with class-specific means and a shared covariance matrix.

The decision rule can be written as:

$$\hat{y} = \arg\max_{k} \left[P(y = k|x)\right]$$

where $P(y = k|x)$ is the posterior probability of class k, given the feature vector x.

In practical terms, the criterion for assigning an observation to a class is the largest discriminant score, which is a function of the class means, covariance, and prior probabilities of each class.

Homoscedasticity refers to the assumption that the variance of the residuals in a regression model is constant across all levels of the independent variable(s). This is crucial for valid statistical inference as it ensures that the estimates are efficient and standard errors are reliable.

In contrast, heteroscedasticity occurs when the variance of the residuals varies with the level of the independent variable(s). This can lead to inefficient estimates and biased standard errors, resulting in invalid hypothesis tests.

To detect heteroscedasticity, one can plot the residuals against fitted values; patterns like funnel shapes indicate non-constant variance. If heteroscedasticity is found, solutions include transforming the dependent variable, using weighted least squares, or employing robust standard errors.

For a more detailed discussion, please refer to the previous section on the assumptions of homoscedasticity and heteroscedasticity.

The nearest neighbors algorithm estimates the conditional expectation function $E[Y|X]$, where Y is the response variable and X is the predictor variable. This non parametric approach does not assume a specific functional form for the relationship between X and Y. Instead, it predicts the expected value of Y based on the values of Y associated with the k nearest neighbors in the data set, using a distance metric such as the Euclidean distance.

Note: these techniques (Discriminant Analysis) only accept quantitative explanatory or predictive variables. However, a qualitative variable with two modalities, for example the type of day, can be considered as quantitative in the form of an indicator function taking its values in . In this last case, one should not try to interpret the discrimination functions, just consider prediction errors. The Station variable is not taken into account.

The standard R library MASS for discriminant analysis does not propose an automatic procedure for choosing a variable, but in this example, the variables are few

3.4.2 Estimation of the models:Estimation of the prediction error by cross-validation with caret Library or Not

	FALSE	**TRUE**
FALSE	660	29
TRUE	74	69

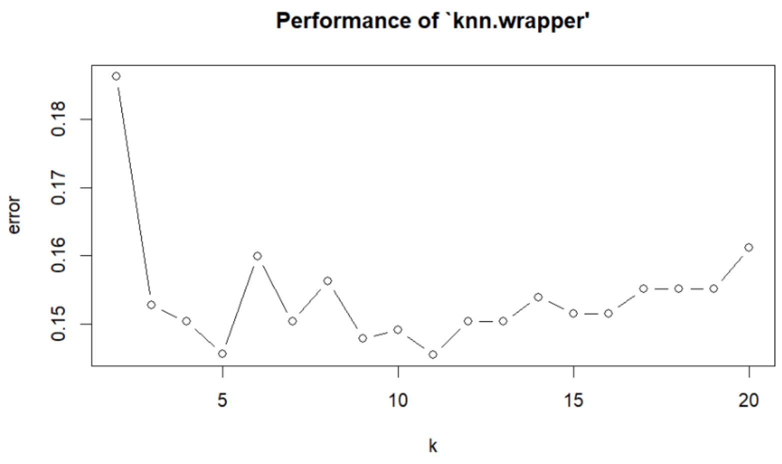
Table 6: Confusion Matrix

3.4.3 quadratic discriminant analysis

	FALSE	**TRUE**
FALSE	649	40
TRUE	86	57

Table 7: Confusion Matrix

3.4.4 KNN optimization:



with caret Library:

	FALSE	TRUE
FALSE	661	28
TRUE	72	71

Table 8: Confusion Matrix

without caret Library:

	FALSE	TRUE
FALSE	653	36
TRUE	77	66

Table 9: Confusion Matrix

3.4.5 Prediction of the test sample/Confussion matrices:

	FALSE	TRUE
FALSE	661	28
TRUE	72	71

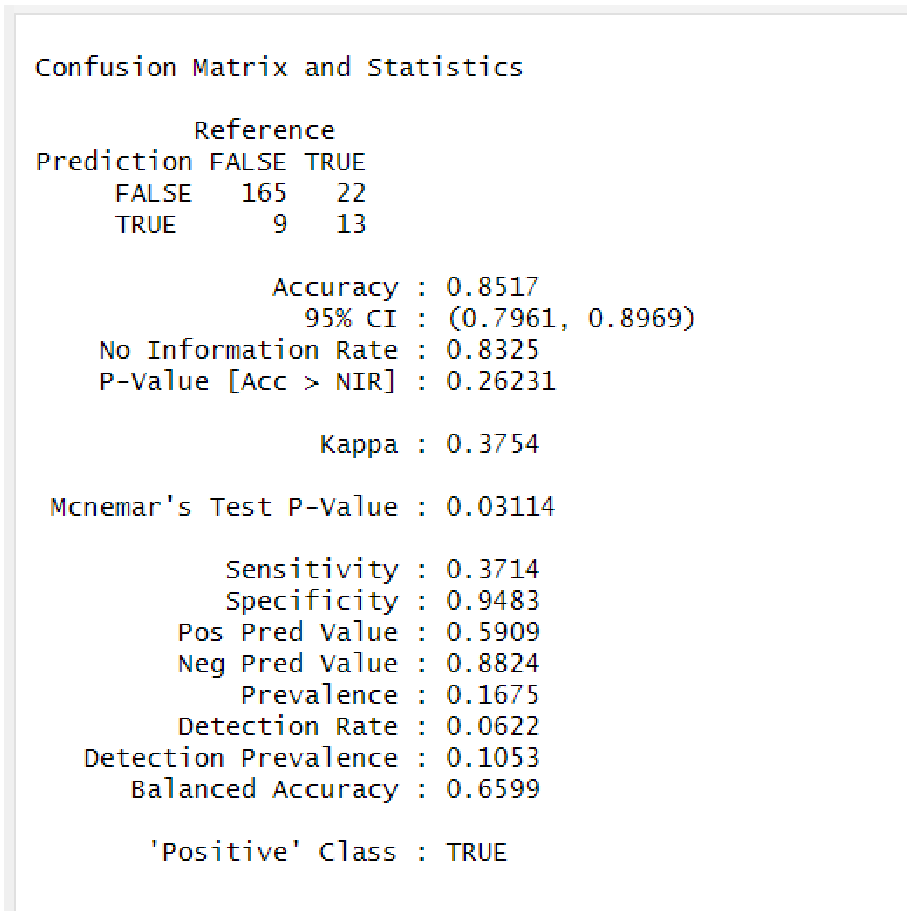
Table 10: Confusion Matrix LDA

	FALSE	TRUE
FALSE	653	36
TRUE	77	66

Table 11: Confusion Matrix QDA

	FALSE	TRUE
FALSE	653	36
TRUE	77	66

Table 12: Confusion Matrix KNN



```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE   165   22
     TRUE      9   13

               Accuracy : 0.8517
                 95% CI : (0.7961, 0.8969)
    No Information Rate : 0.8325
    P-Value [Acc > NIR] : 0.26231

                  Kappa : 0.3754

 Mcnemar's Test P-Value : 0.03114

            Sensitivity : 0.3714
            Specificity : 0.9483
         Pos Pred Value : 0.5909
         Neg Pred Value : 0.8824
             Prevalence : 0.1675
         Detection Rate : 0.0622
   Detection Prevalence : 0.1053
      Balanced Accuracy : 0.6599

       'Positive' Class : TRUE
```

3.4.6 ROC curves:

Here we will compare via the ROC curve the discriminant analysis methods LDA, QDA, KNN and logistic regression,

Each method—LDA, QDA, KNN, and Logit—shows similar performance, but they each excel at slightly different points:

Logit appears to perform slightly better at the lower false positive rates, as it stays closest to the top left corner early on, indicating a high true positive rate with low FPR. KNN and QDA follow closely, with KNN performing slightly better at mid-range FPRs compared to QDA. LDA performs similarly to Logit and QDA across most of the curve but doesn't seem to outperform the others consistently.

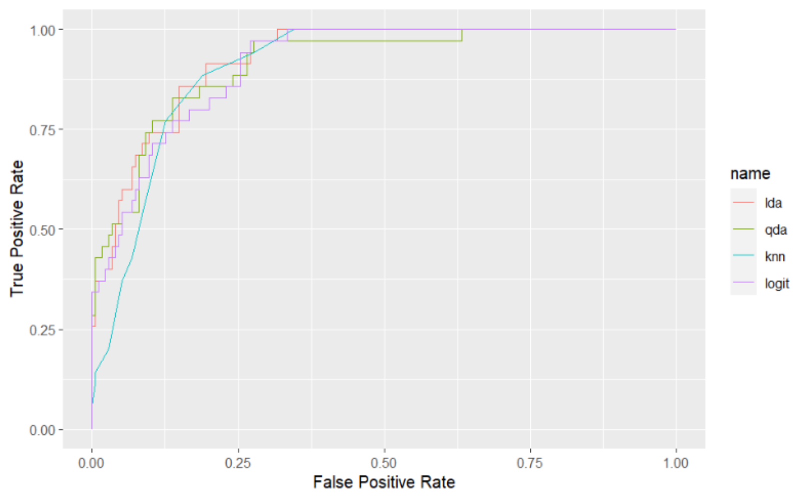


Figure 24: ROC Curve

3.5 Support Vector Machine SVM:

Despite the theoretical guarantees concerning this type of algorithm, the results depend strongly on the choice of parameters. We will first limit ourselves to the Gaussian kernel (default choice); the function tune.svm allows to easily test several situations by estimating the prediction quality by cross-validation on a grid. The execution time in R is a bit long..

The execution time for Support Vector Machines (SVM) is more sensitive to the number of observations than the number of variables. This is because SVM training involves solving a quadratic programming problem, which typically scales quadratically or cubically with the number of data points, making it computationally expensive as the dataset grows. Additionally, kernel functions used in SVM require pairwise comparisons between observations, further increasing the time complexity. While the number of features also impacts the computation, especially in high-dimensional spaces, its influence on execution time is less significant compared to the effect of the number of observations. -regression: Although initially developed in the case of a binary variable, SVMs have been extended to regression problems. The estimation and optimization of the penalty coefficient are obtained by the following commands.

The goal is to find the combination of 'gamma' and 'cost' that minimizes the cost, thereby improving the model's predictive accuracy.

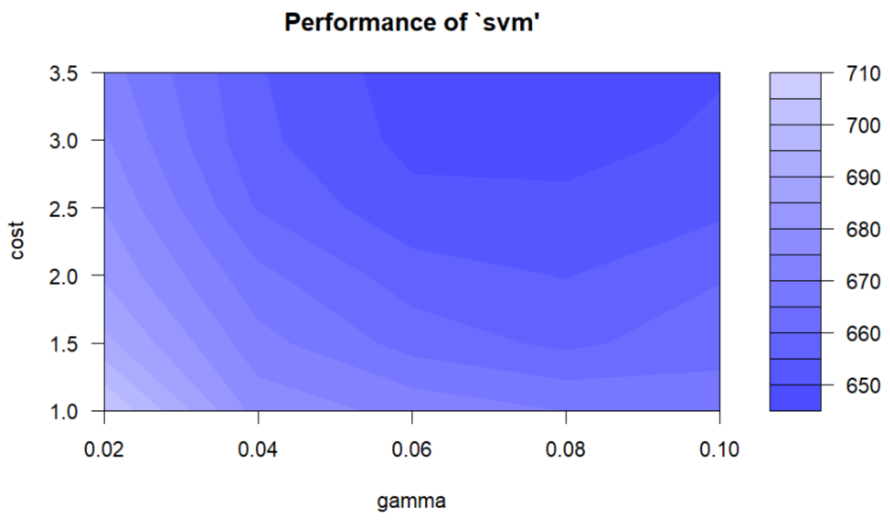


Figure 25: Performance of svm

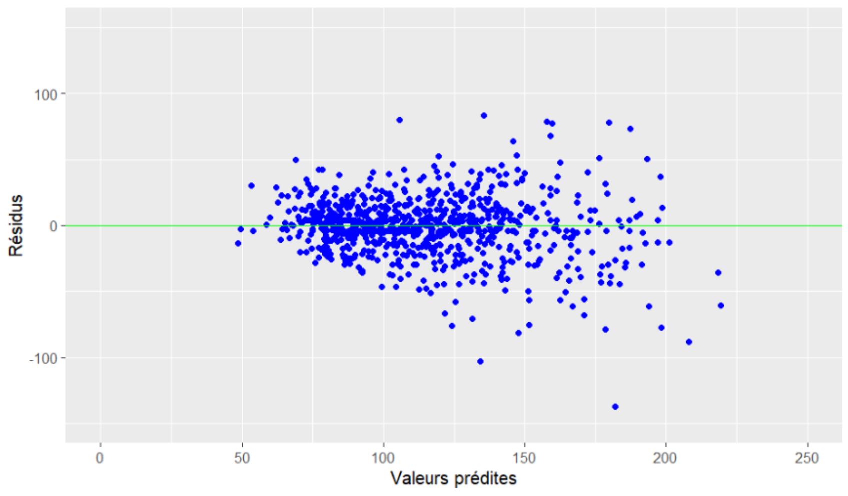


Figure 26: The Residuals

The residuals, which are the differences between the observed and predicted values, are mostly centered around zero, indicating that the model's predictions are generally accurate. The spread of residuals appears to be consistent across the range of predicted values, suggesting that the model does not exhibit significant bias at different levels of prediction. This pattern is a good sign of a well-fitted model, as it implies that the errors are randomly distributed and not systematically over- or under-predicting.

3.5.1 Discrimination:

After SVM Training we get: γ =0.1 and Cost=2

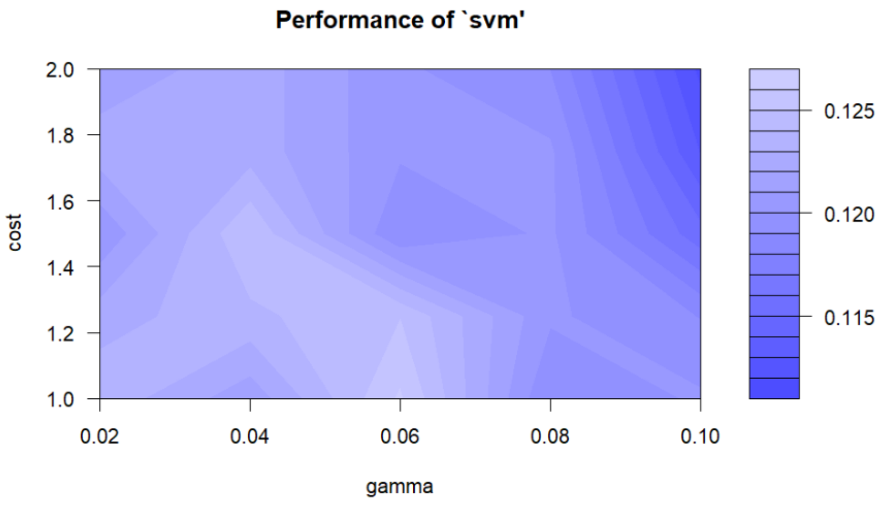


Figure 27: Enter Caption

3.5.2 Prediction on the test sample:

-Regression error: MSE = 579.0746

-Classification error:

	FALSE	**TRUE**
FALSE	165	18
TRUE	9	17

Table 13: Confusion Matrix

-Discrimination:

	FALSE	**TRUE**
FALSE	168	19
TRUE	6	16

Table 14: Confusion Matrix

3.5.3 ROC Curve

Here we will compare via the ROC curve the discriminant analysis methods LDA, QDA, KNN and logistic regression

we can see a significant improvement using the SVM model

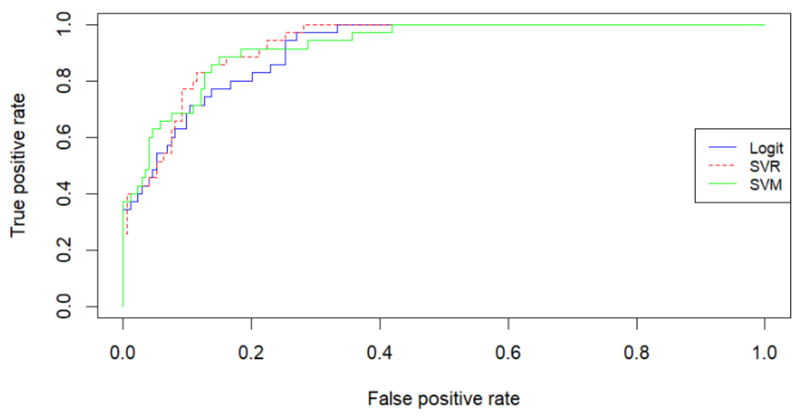


Figure 28: ROC curve

3.6 CART, Agreggation of models

3.6.1 Binary decision tree:

The rpart library is the most commonly used for the construction of decision trees. Two types of trees can be estimated depending on whether the variable to be modeled is the ozone concentration (regression tree) or directly the threshold overflow (discrimination or decision tree). Different parameters control the execution of the algorithm: the minimum penalty (cp) for the construction of the maximum tree, the minimum number of observations per node, the number of cross-validations (by default 10)... see the online help (?rpart.control) for more details but it is not very explicit on some parameters.

NB. A sequence of values of the cp penalty is associated to a sequence of nested trees.

When creating a node in a decision tree, the optimized criterion is typically a measure of impurity in the data at that node. The goal is to split the data in a way that results in groups that are as pure as possible. Two common impurity criteria used are:

-Gini Impurity: Used in classification trees (e.g., in CART algorithm). Gini impurity measures how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the set.

-Information Gain (Entropy): Used in algorithms like ID3 or C4.5. It measures the reduction in entropy (disorder) achieved after the dataset is split based on an attribute.

For regression trees, the optimized criterion is usually the reduction in variance, where the algorithm tries to minimize the variance of the target variable within each node.

3.6.2 Estimation and pruning of the regression tree

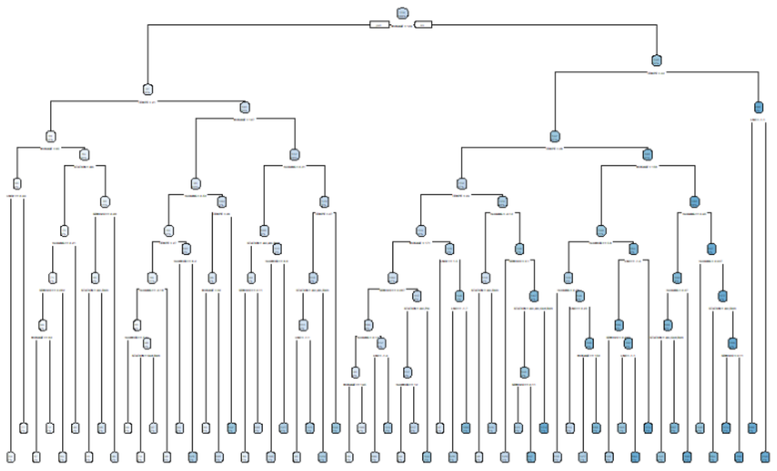


Figure 29: the Artificial Tree

The tree is unreadable and has too many leaves for a good prediction (overlearning), it is necessary to reduce the number by pruning. The following commands compute the predictions obtained by 10-fold cross-validation for each pruned tree according to the successive values of the complexity coefficient. The sequence of these values is implicitly the one provided by rpart.

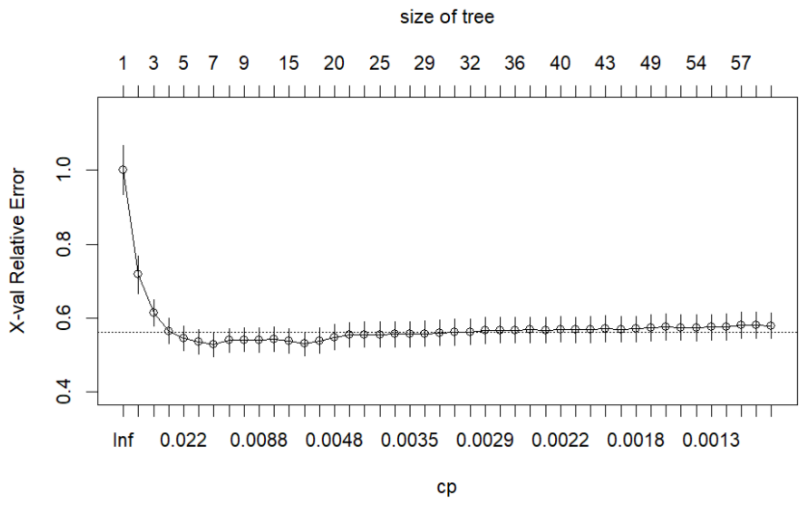


Figure 30: Cross Validation

cp (Complexity Parameter) represents a threshold for pruning the tree. As you grow the tree, each split is evaluated for whether it significantly improves the model's performance, considering both accuracy and complexity. The complexity parameter is used to

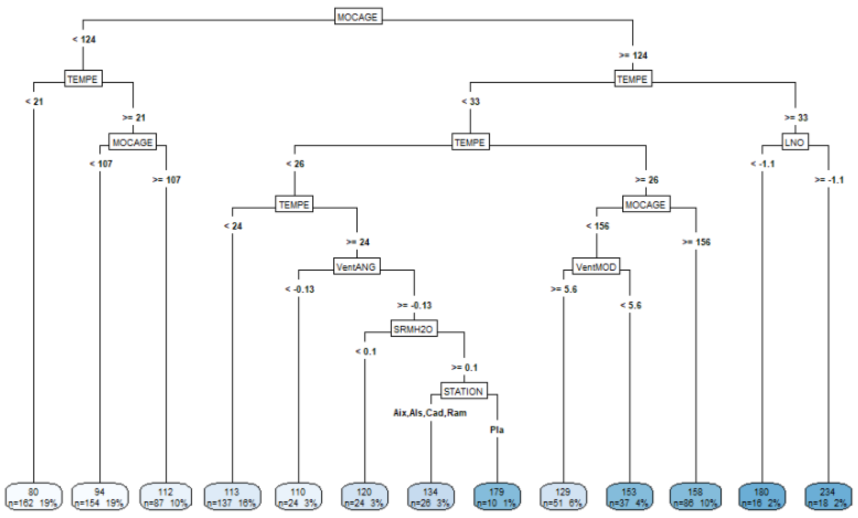
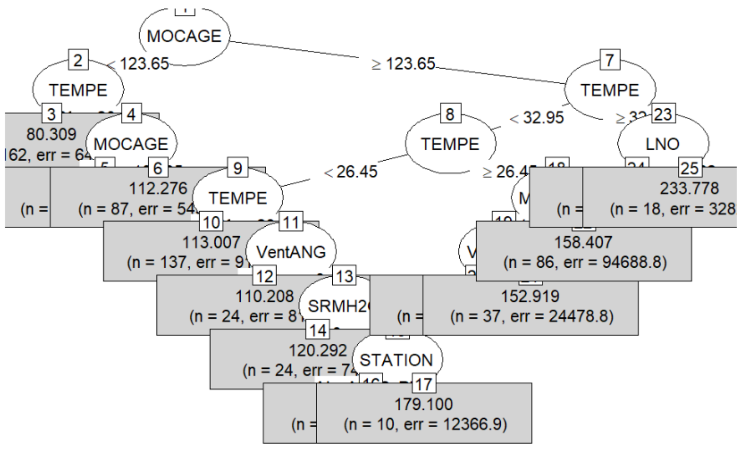


Figure 31: The Tree with minimal error

penalize the addition of new splits that do not reduce the model's error significantly. looking to the plot above we can find that for a cp 0.079 the error is minimal then we use this value to build the tree:

Based on the tree below, we can say that the variable that contributes the most to prediction is

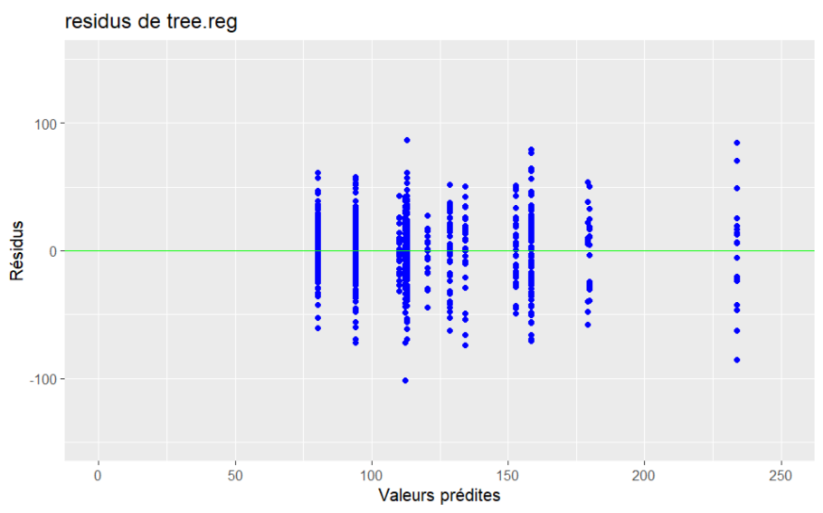


[h]

3.6.3 Prediction of the test sample

The structure of this residuals plot reveals some important characteristics about the model and the data:

The residuals are grouped into vertical bands, which indicates that the predicted



values (on the x-axis) take on a limited set of discrete values. This is typical of models like regression trees, which produce piecewise constant predictions. Since a decision tree generates predictions for different regions of the feature space, all the observations in the same region are assigned the same predicted value, leading to vertical bands in the residuals.

The residuals seem to show varying spread across different predicted values. This means that the variance of the residuals isn't constant across all levels of the predicted values. For smaller predicted values (around 100), the spread seems more concentrated, while for larger predicted values (above 200), the residuals seem more dispersed. This could indicate **heteroscedasticity**, where the variability of the errors depends on the magnitude of the prediction, which can affect model performance and inference.

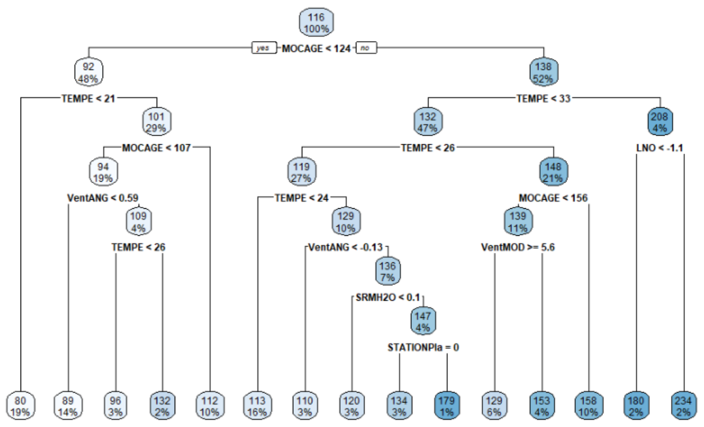
The residuals seem symmetrically distributed around zero, which is a positive sign. This symmetry suggests that the model doesn't systematically over- or under-predict, though the larger variance for certain predicted values should be noted.

4. Extreme Residuals: There are some points with high residuals (greater than 100 or less than -100). These could represent outliers or cases where the model significantly under- or over-predicted the true values.

Let's so prune the tree to mitigate this problem:

3.6.4 Estimation and pruning of a discrimination tree

In the case of discrimination, the default criterion is the Gini concentration index; it is possible to specify another criterion (split="information") as well as weights on the obser-

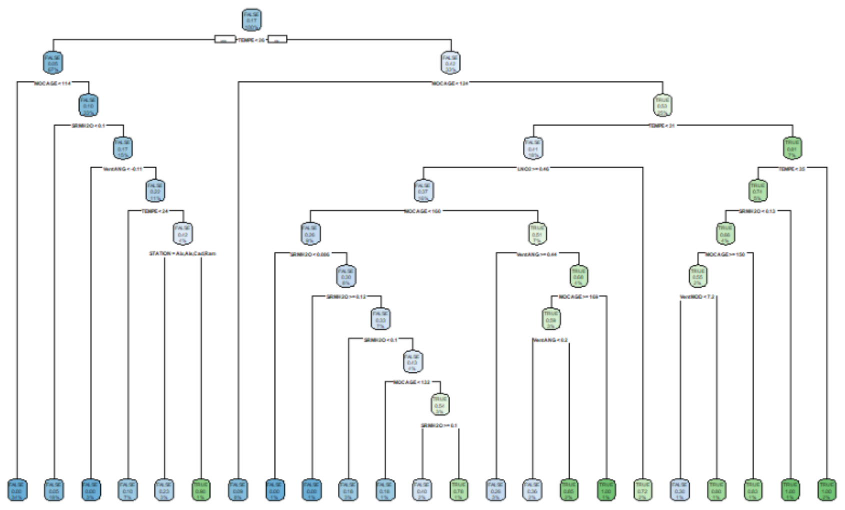


vations, a misclassification cost matrix and a priori probabilities (?rpart for more details).

Another commonly used heterogeneity criterion in decision trees is the Entropy or Information Gain, which is based on the concept of information theory. This criterion is often used in classification tasks, particularly when using algorithms like ID3, C4.5, and their extensions.

Entropy measures the uncertainty or impurity in a dataset. The goal of a decision tree is to reduce entropy by splitting the data into subsets that are more homogeneous in terms of the target variable.

Information Gain quantifies the reduction in entropy (or uncertainty) after a dataset is split based on a particular feature. The split with the highest information gain is chosen for the next node.



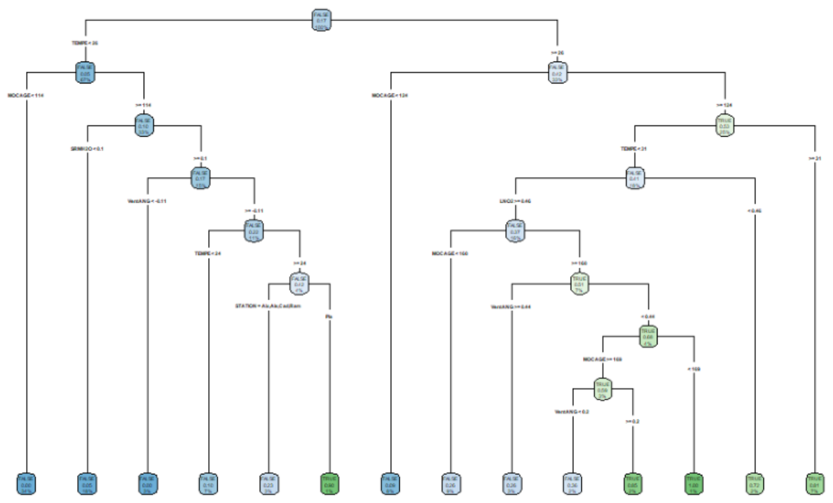
Another commonly used heterogeneity criterion in decision tree algorithms is entropy, particularly when employing the information gain metric. This is often used in conjunction with the ID3 and C4.5 algorithms.
Entropy and Information Gain: - Entropy measures the impurity or disorder of the data. A node with high entropy contains a mix of different classes, while a node with low entropy contains predominantly one class. - Information Gain is a measure of the reduction in entropy when a dataset is split on an attribute. It is computed as the difference between the entropy of the original dataset and the weighted entropy of the resulting subsets after the split.

The same cross-validation pruning procedure is implemented but with a different expression of the prediction error : misclassification rate rather than squared error.

the result:

$$\begin{bmatrix} 0.543123543 & 0.069462305 & 0.048448974 & 0.024224487 & 0.012509471 & 0.008845532 & 0.002644429 \\ 0.1718750 & 0.1454327 & 0.1454327 & 0.1418269 & 0.1430288 & 0.1406250 & 0.1538462 \end{bmatrix}$$

the model's performance varies across the different folds, with some folds showing high error rates and others showing quite low error rates. This variability might indicate that the model could be overfitting or underperforming in certain partitions of the data. By training a decision tree model using 10-fold cross-validation to tune the cp (complexity parameter) of the tree. we find the best value of cp that gives the highest accuracy.



Finally:

cp=0.0736105999263894

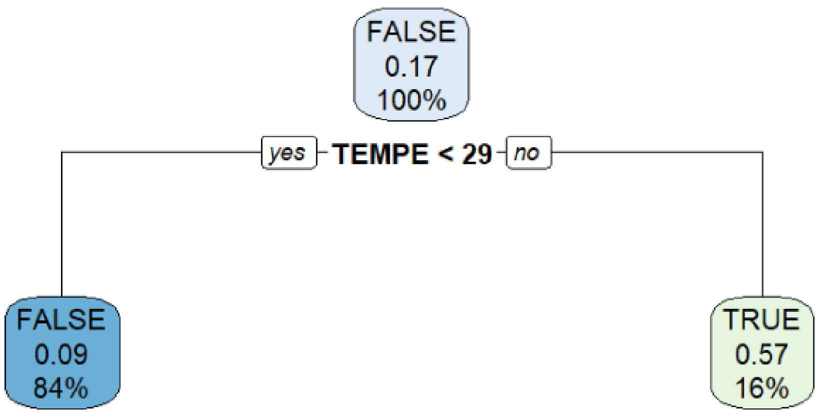


Figure 32: the decision tree using the best cp value

3.6.5 Prediction of the test sample:

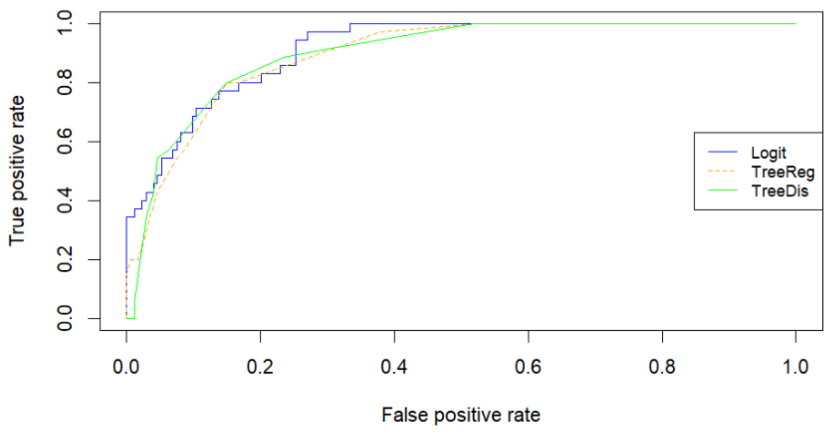
Different predictions are considered along with the estimated errors on the test sample. Quantitative concentration prediction, exceedance prediction from the quantitative prediction and directly the exceedance prediction from the decision tree.

regression error: we calculate the MSE=794.8491 Classififcaton error:

	FALSE	**TRUE**
FALSE	161	16
TRUE	13	19

Table 15: Confusion Matrix

3.6.6 ROC curve

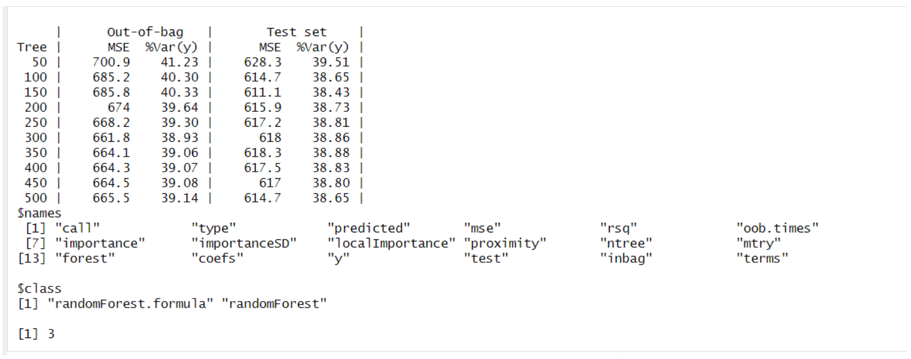


Based on the ROC curves, it appears that the Logit method has a slight edge over the TreeReg and TreeDis methods. The ROC curve for Logit is consistently above the other two, indicating a higher true positive rate for the same false positive rate across most thresholds.

However, without the exact Area Under the Curve (AUC) values, it's challenging to definitively conclude which method is superior. Generally, the method with the highest AUC is considered the best in terms of predictive quality. From the visual inspection, Logit seems to perform better, but a precise comparison would require those AUC values.

3.7 Random forests:

for a regression with 500 estimators:



```
     |      Out-of-bag   |       Test set    |
Tree |      MSE  %Var(y) |      MSE  %Var(y) |
  50 |    700.9    41.23 |    628.3    39.51 |
 100 |    685.2    40.30 |    614.7    38.65 |
 150 |    685.8    40.33 |    611.1    38.43 |
 200 |      674    39.64 |    615.9    38.73 |
 250 |    668.2    39.30 |    617.2    38.81 |
 300 |    661.8    38.93 |      618    38.86 |
 350 |    664.1    39.06 |    618.3    38.88 |
 400 |    664.3    39.07 |    617.5    38.83 |
 450 |    664.5    39.08 |      617    38.80 |
 500 |    665.5    39.14 |    614.7    38.65 |
$names
 [1] "call"            "type"            "predicted"       "mse"             "rsq"             "oob.times"
 [7] "importance"      "importanceSD"    "localImportance" "proximity"       "ntree"           "mtry"
[13] "forest"          "coefs"           "y"               "test"            "inbag"           "terms"

$class
[1] "randomForest.formula" "randomForest"

[1] 3
```

let's plot the residuals

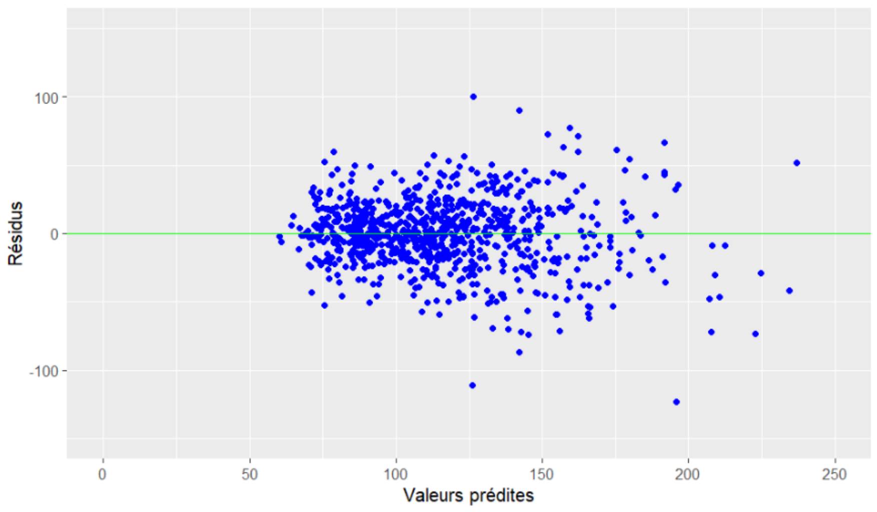
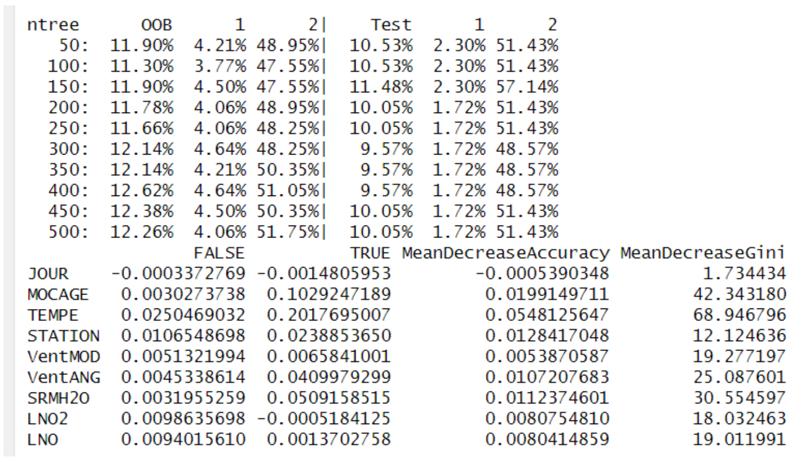


Figure 33: Random forest model

for discrimination using 500 estimators :

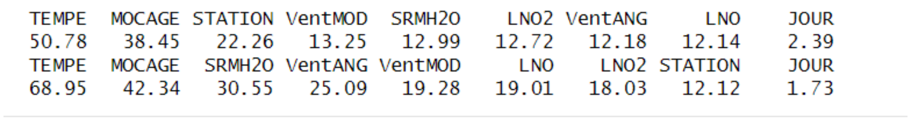


```
ntree      OOB      1      2|    Test      1      2
   50:  11.90%  4.21% 48.95%|  10.53%  2.30% 51.43%
  100:  11.30%  3.77% 47.55%|  10.53%  2.30% 51.43%
  150:  11.90%  4.50% 47.55%|  11.48%  2.30% 57.14%
  200:  11.78%  4.06% 48.95%|  10.05%  1.72% 51.43%
  250:  11.66%  4.06% 48.25%|  10.05%  1.72% 51.43%
  300:  12.14%  4.64% 48.25%|   9.57%  1.72% 48.57%
  350:  12.14%  4.21% 50.35%|   9.57%  1.72% 48.57%
  400:  12.62%  4.64% 51.05%|   9.57%  1.72% 48.57%
  450:  12.38%  4.50% 50.35%|  10.05%  1.72% 51.43%
  500:  12.26%  4.06% 51.75%|  10.05%  1.72% 51.43%
                FALSE          TRUE MeanDecreaseAccuracy MeanDecreaseGini
JOUR     -0.0003372769 -0.0014805953        -0.0005390348         1.734434
MOCAGE    0.0030273738  0.1029247189         0.0199149711        42.343180
TEMPE     0.0250469032  0.2017695007         0.0548125647        68.946796
STATION   0.0106548698  0.0238853650         0.0128417048        12.124636
VentMOD   0.0051321994  0.0065841001         0.0053870587        19.277197
VentANG   0.0045338614  0.0409979299         0.0107207683        25.087601
SRMH2O    0.0031955259  0.0509158515         0.0112374601        30.554597
LNO2      0.0098635698 -0.0005184125         0.0080754810        18.032463
LNO       0.0094015610  0.0013702758         0.0080414859        19.011991
```

3.7.1 Variable Importance

The resulting model is uninterpretable, but coefficients estimate the contributions of the variables in their participation in discrimination. Compare with the variables selected by the other models in episode 1. Two importance criteria are proposed.

In Random Forests, variable importance is measured using two common methods: Mean Decrease in Impurity (MDI) and Mean Decrease in Accuracy (MDA). MDI, also known as Gini Importance, quantifies the contribution of each variable to reducing impurity (such as Gini impurity for classification or variance for regression) across the trees in the forest. Variables that lead to significant reductions in impurity are considered more important. MDA, or Permutation Importance, assesses how much a variable impacts model performance by randomly shuffling its values and observing the resulting decrease in accuracy. A significant drop in performance indicates a highly important variable. Both measures provide insights into the role of each feature in the model's predictive decisions.



```
TEMPE  MOCAGE STATION VentMOD  SRMH2O    LNO2 VentANG     LNO    JOUR
50.78   38.45   22.26   13.25   12.99   12.72   12.18   12.14    2.39
TEMPE  MOCAGE  SRMH2O VentANG VentMOD     LNO    LNO2 STATION    JOUR
68.95   42.34   30.55   25.09   19.28   19.01   18.03   12.12    1.73
```

Figure 34: Variable importance

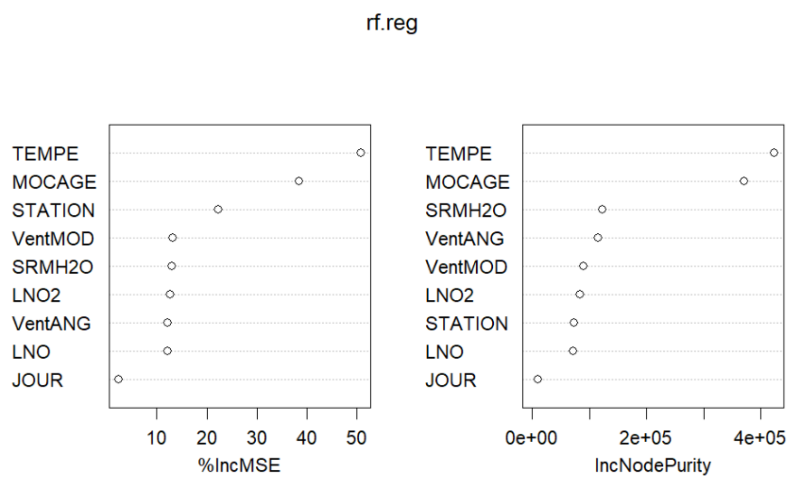


Figure 35: Variable Importance by measure method for regression

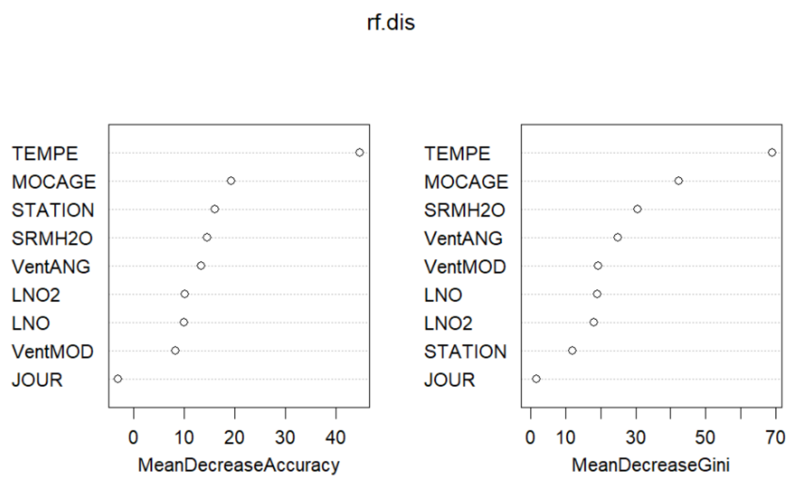


Figure 36: Variable Imporatnce by measure method for discrimination

3.7.2 Prediction of the test sample

Regression: for that regression task by random forest we find a MSE of 614.6802 and for the task of discrimination;

	FALSE	**TRUE**
FALSE	170	16
TRUE	4	19

Table 16: Confusion Matrix

3.7.3 Boosting:

Two libraries offer relatively sophisticated versions of boosting algorithms in R. The boost library offers 4 approaches: adaboost, bagboost and two logitboost. Developed for a particular problem: the analysis of genomic expression data, it may not be completely adapted to the data studied; it is limited to quantitative predictors and may provide strange results. The gbm library is preferred; it also offers several versions depending on the chosen cost function. A more recent library xgboost integrates parallelization features (not under Windows) and uses several other parameters.

The variable to predict must be numerically coded (0-1) for this implementation. The number of iterations, or number of trees, is set as well as a shrinkage coefficient (shrinkage).

In boosting, shrinkage refers to the technique of reducing the impact of each individual tree in the ensemble by scaling down the predictions made by each tree. This is typically accomplished by multiplying the contribution of each tree by a factor known as the learning rate (or shrinkage parameter).

Mathematically, if the prediction from the m-th tree is $T_m(x)$, the boosted prediction after adding this tree is updated as:

$$F_{m+1}(x) = F_m(x) + \lambda T_m(x)$$

where $F_m(x)$ is the current ensemble prediction, $T_m(x)$ is the prediction from the newly added tree, and λ is the learning rate (a value between 0 and 1). A smaller value of λ (stronger shrinkage) slows down the learning process, allowing more trees to be added while avoiding overfitting and helping the model generalize better.

In summary, **shrinkage** allows boosting models to take smaller, more controlled steps toward fitting the data, making the process more robust and helping to prevent overfitting.

We can ensure the absence of a critical overfitting phenomenon by calculating and plotting the evolution of the error on the test sample as a function of the number of trees in the model. The error remains stable around the number of trees selected and shown by the vertical line.

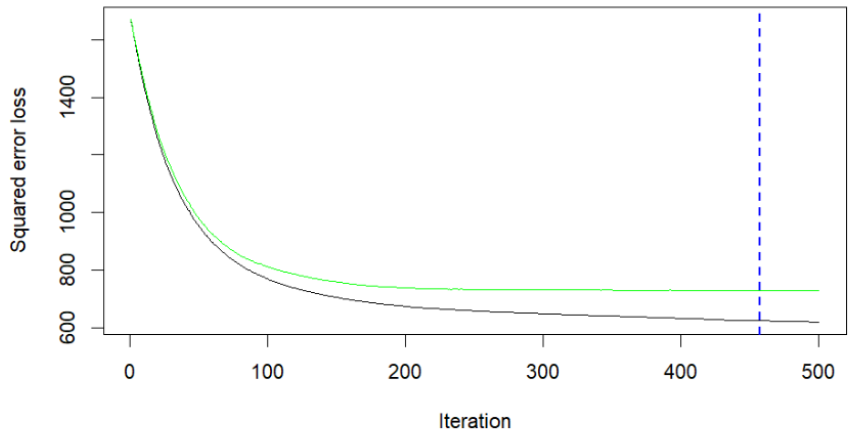


Figure 37: optimal iteration number by cross validation

-Discrimination:

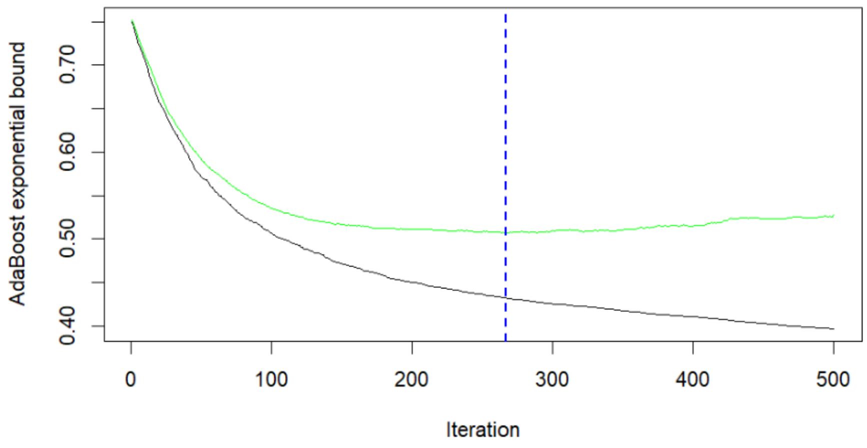


Figure 38: Discrimination:optimal iteration number by cross validation

-Model training:

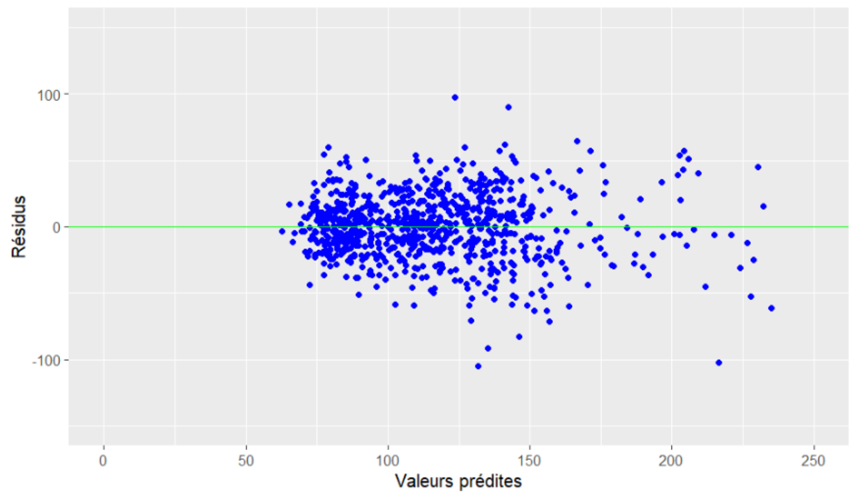


Figure 39: Residuals

3.7.4 Prediction of the test sample:

-Regression: MSE= 741.0894 -Classification:

	FALSE	**TRUE**
FALSE	170	22
TRUE	4	13

Table 17: Confusion Matrix

3.7.5 ROC curves:

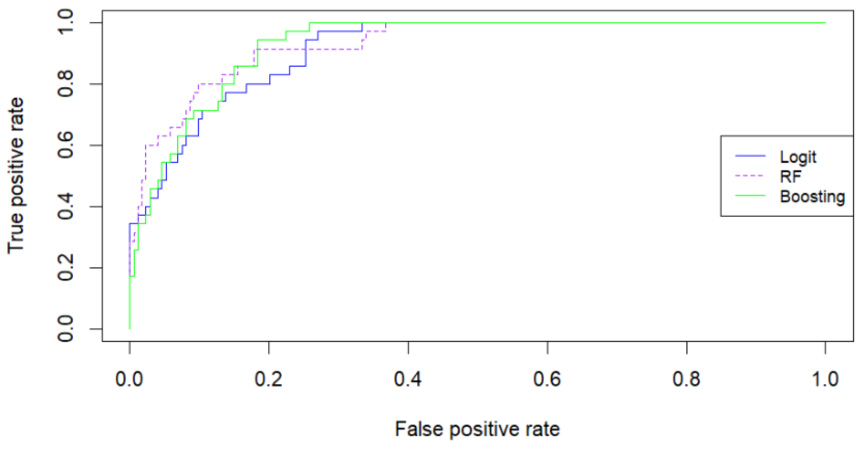


Figure 40: ROC Curves

3.8 Neural Networks:

t consists in estimating a perceptron type model with qualitative or quantitative variables as input and the variable to be predicted as output. R functions for learning an elementary perceptron have been realized by different authors and are available on the network. The nnet library of (Ripley, 1999), is limited to the single layer perceptron. It is not ”deep learning”! but it is sufficient in many cases. An R library associated with the eponymous H2O software proposes multi-layer and ”convolutional” networks.

As for trees, the variable to be explained is either quantitative or qualitative; the activation function of the output neuron of a network must be adapted accordingly.

In neural networks, the choice of the number of hidden layer neurons is typically determined based on the complexity of the task and the dataset. While there is no universal default, common heuristics suggest setting the number of neurons between the size of the input and output layers, or using the average of both. However, the exact number of neurons is often determined experimentally through techniques like cross-validation. In most libraries such as Keras, TensorFlow, and PyTorch, users are required to specify the hidden layer size manually, while some automated tools may adjust this based on the problem at hand.

The decay parameter in the nnet function (from the nnet package in R) is a regularization term that controls weight decay in the model. Weight decay helps prevent overfitting by adding a penalty to the size of the network's weights. This is done by modifying the cost function to include a term that penalizes large weight values. Specifically, the decay parameter applies L2 regularization, which shrinks the weights towards zero, leading to simpler models that generalize better to unseen data.

In formula terms, the cost function being minimized in the neural network is:

$$\text{Cost Function} = \text{Loss Function} + \lambda \sum w_i^2$$

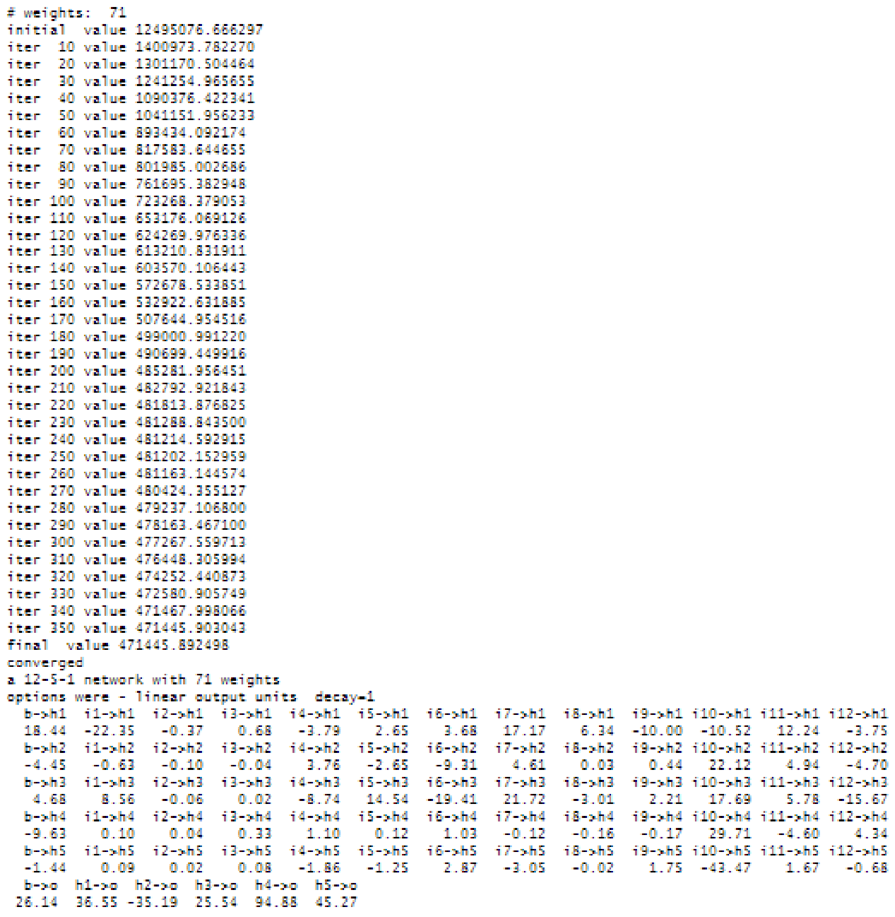
Where:

- λ is the decay parameter (regularization strength),
- w_i are the weights of the network.

A higher decay value increases the penalty on large weights, encouraging the model to learn smaller weights and thus reducing overfitting.
Another method to prevent overfitting is through the use of cross-validation. Cross-validation splits the dataset into multiple subsets (folds) and trains the model on different combinations of these subsets, while using the remaining data for validation. This ensures that the model generalizes well and is not overfitting to a specific subset of the data. A commonly used technique is k-fold cross-validation, where the dataset is divided into k parts, and the model is trained k times, each time validating on a different fold.

3.8.1 Regression case:

Model Training: The optimization of the parameters still requires a cross-validation pro-



```
# weights:  71
initial  value 12495076.666297
iter  10 value 1400973.782270
iter  20 value 1301170.504464
iter  30 value 1241254.965655
iter  40 value 1090376.422341
iter  50 value 1041151.956233
iter  60 value 893434.092174
iter  70 value 817583.644655
iter  80 value 801985.002686
iter  90 value 761695.382948
iter 100 value 723268.379053
iter 110 value 653176.069126
iter 120 value 624269.976336
iter 130 value 613210.831911
iter 140 value 603570.106443
iter 150 value 572678.533851
iter 160 value 532922.631885
iter 170 value 507644.954516
iter 180 value 499000.991220
iter 190 value 490699.449916
iter 200 value 485281.956451
iter 210 value 482792.921843
iter 220 value 481813.876825
iter 230 value 481288.843500
iter 240 value 481214.592915
iter 250 value 481202.152959
iter 260 value 481163.144574
iter 270 value 480424.355127
iter 280 value 479237.106800
iter 290 value 478163.467100
iter 300 value 477267.559713
iter 310 value 476448.305994
iter 320 value 474252.440873
iter 330 value 472580.905749
iter 340 value 471467.998066
iter 350 value 471445.903043
final  value 471445.892498
converged
a 12-5-1 network with 71 weights
options were - linear output units  decay=1
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1 i10->h1 i11->h1 i12->h1
  18.44  -22.35   -0.37    0.68   -3.79    2.65    3.68   17.17    6.34  -10.00  -10.52   12.24   -3.75
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2 i10->h2 i11->h2 i12->h2
  -4.45   -0.63   -0.10   -0.04    3.76   -2.65   -9.31    4.61    0.03    0.44   22.12    4.94   -4.70
  b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i9->h3 i10->h3 i11->h3 i12->h3
   4.68    8.56   -0.06    0.02   -8.74   14.54  -19.41   21.72   -3.01    2.21   17.69    5.78  -15.67
  b->h4  i1->h4  i2->h4  i3->h4  i4->h4  i5->h4  i6->h4  i7->h4  i8->h4  i9->h4 i10->h4 i11->h4 i12->h4
  -9.63    0.10    0.04    0.33    1.10    0.12    1.03   -0.12   -0.16   -0.17   29.71   -4.60    4.34
  b->h5  i1->h5  i2->h5  i3->h5  i4->h5  i5->h5  i6->h5  i7->h5  i8->h5  i9->h5 i10->h5 i11->h5 i12->h5
  -1.44    0.09    0.02    0.08   -1.86   -1.25    2.87   -3.05   -0.02    1.75  -43.47    1.67   -0.68
  b->o  h1->o  h2->o  h3->o  h4->o  h5->o
 26.14  36.55 -35.19  25.54  94.88  45.27
```

cedure. There is no function in the nnet library allowing to do this but the tune.nnet function of the e1071 library is adapted to this approach.

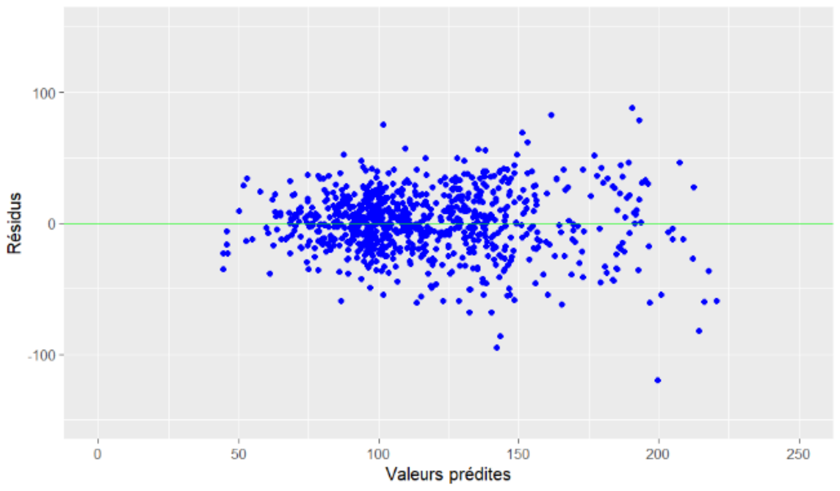
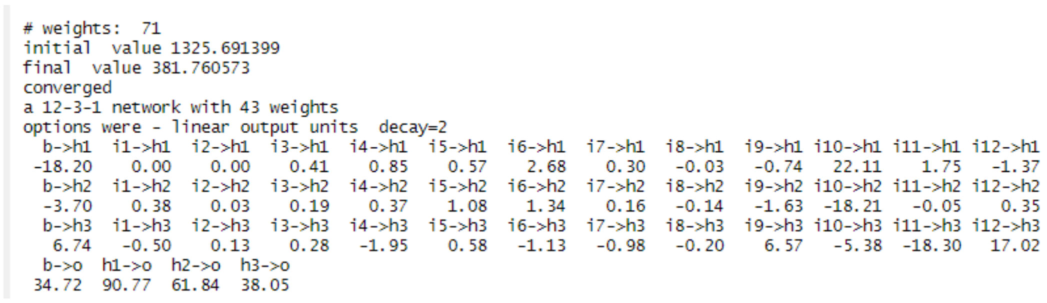


Figure 41: Residuals

3.8.2 Discrimination case:

-Model Training:



```
# weights:  71
initial  value 1325.691399
final  value 381.760573
converged
a 12-3-1 network with 43 weights
options were - linear output units  decay=2
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1 i10->h1 i11->h1 i12->h1
 -18.20    0.00    0.00    0.41    0.85    0.57    2.68    0.30   -0.03   -0.74   22.11    1.75   -1.37
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2 i10->h2 i11->h2 i12->h2
  -3.70    0.38    0.03    0.19    0.37    1.08    1.34    0.16   -0.14   -1.63  -18.21   -0.05    0.35
  b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i9->h3 i10->h3 i11->h3 i12->h3
   6.74   -0.50    0.13    0.28   -1.95    0.58   -1.13   -0.98   -0.20    6.57   -5.38  -18.30   17.02
  b->o  h1->o  h2->o  h3->o
 34.72  90.77  61.84  38.05
```

The cross-validation is always necessary in order to try to optimize the choices in presence: number of neurons, decay and possibly the maximum number of iterations.

The initialization of the training of a neural network as well as the estimation of the error by cross-validation are random. Each execution therefore gives different results. At this level, it would be interesting to build a two-factor design (here, the size and decay parameters) of each of the three levels. Several realizations for each combination of levels followed by a classical anova test would give a better idea of the influence of these factors on the error.

-Model Training:

3.8.3 Prediction of the test sample

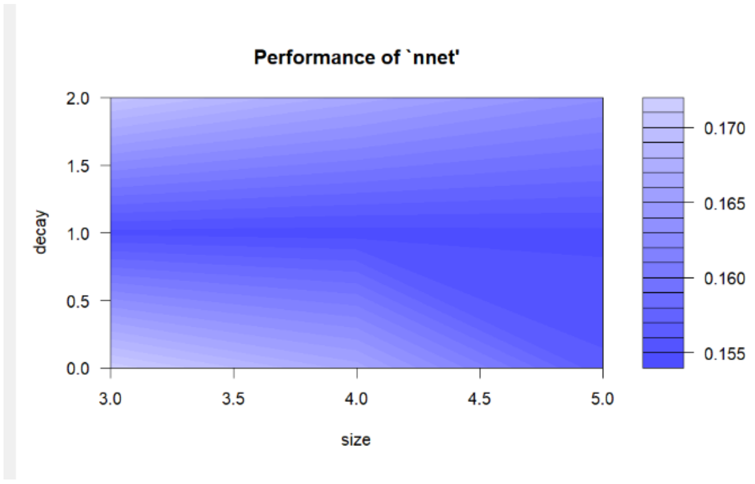
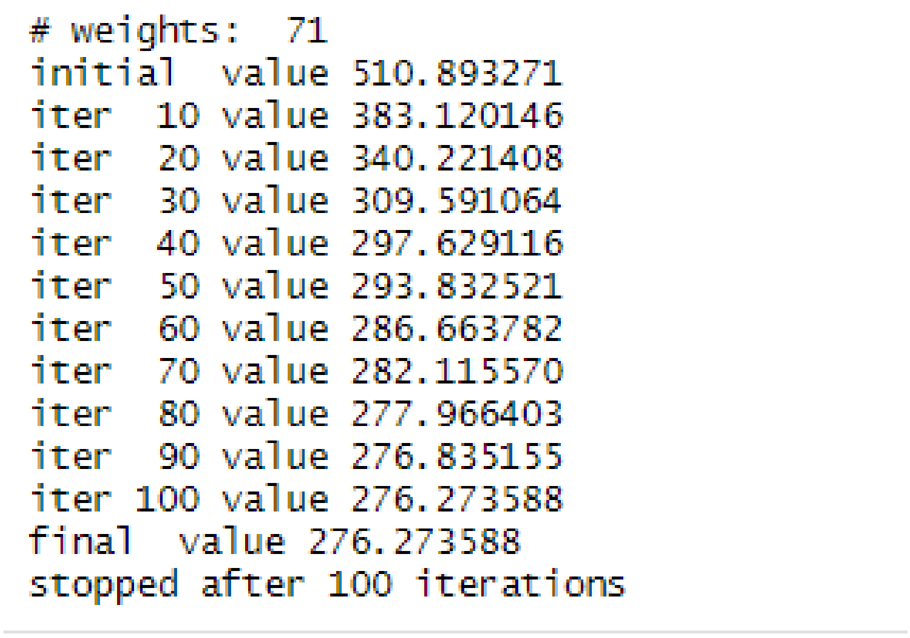


Figure 42: nn performance



```
# weights:  71
initial  value 510.893271
iter  10 value 383.120146
iter  20 value 340.221408
iter  30 value 309.591064
iter  40 value 297.629116
iter  50 value 293.832521
iter  60 value 286.663782
iter  70 value 282.115570
iter  80 value 277.966403
iter  90 value 276.835155
iter 100 value 276.273588
final  value 276.273588
stopped after 100 iterations
```

-Regression error: MSE=643.8371 -Classification error:

	FALSE	**TRUE**
FALSE	171	24
TRUE	3	11

Table 18: Confusion Matrix

3.8.4 ROC curve:

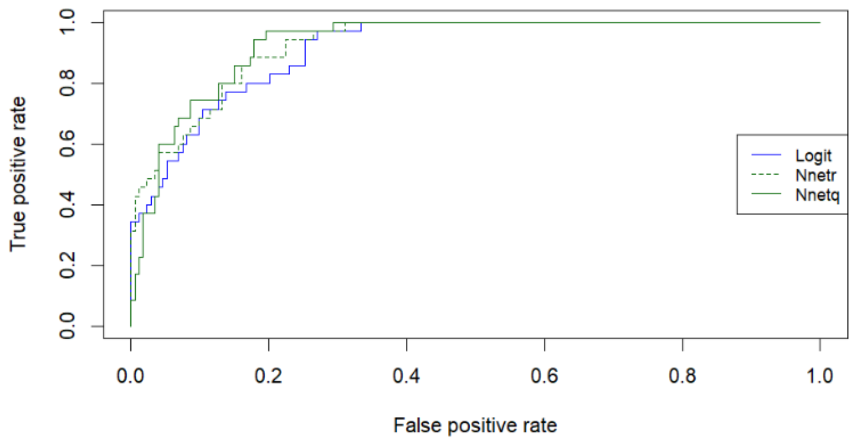


Figure 43: ROC curve

Based on the ROC curve, none of the methods (Logit, Nnetr, Nnetq) seems significantly better than the others. All three methods show very similar performance, with their ROC curves closely following each other.

In ROC curves, a model is considered better if its curve is closer to the top left corner, indicating higher sensitivity (true positive rate) for a lower false positive rate. Since the curves for Logit, Nnetr, and Nnetq overlap and show nearly identical trends, we can infer that their performances are quite comparable.