Themes on machine learning (TOML)

Third and Fourth project

Contents

[1. Third Project 2](#_Toc107058475)

[1.1. Introduction 2](#_Toc107058476)

[1.2. Data observation and first graphs 2](#_Toc107058477)

[1.3. Modeling and calibration 5](#_Toc107058478)

[1.3.1. Forward Subset Selection (MLR- FSS) 5](#_Toc107058479)

[1.3.2. Multiple Linear Regression (MLR) with regularization 6](#_Toc107058480)

[1.3.2.1. Ridge Regression (RR) 6](#_Toc107058481)

[1.3.2.2. Lasso Regression (LR) 8](#_Toc107058482)

[1.3.3. K-nearest neighbor (KNN) 10](#_Toc107058483)

[1.3.4. Kernel Ridge Regression with RBF kernel (KRR-RBF) 11](#_Toc107058484)

[1.3.5. Random Forest (RF) 13](#_Toc107058485)

[1.3.6. Support Vector Regression (SVR) 14](#_Toc107058486)

[1.4. Comparison between the method results 15](#_Toc107058487)

[2. Fourth Project 16](#_Toc107058488)

# 1. Third Project

## 1.1. Introduction

The main goal of this project is to calibrate an air pollution sensor in an air pollution monitoring sensor network. To do this the data obtained by this network is modeled using different methods. The results will be compared using graphs and metrics and one of them will be selected as the best method in the conclusions part of the report.

The distribution of the report is as follows: first the sensor data files and data is explained and analyzed using graphs, then several modeling techniques are applied to this data and, finally, the conclusion of which is the best modeling technique is made using graphs and metrics to compare in the last section.

## 1.2. Data observation and first graphs

At the beginning various data files must be merged into a single data file to enable the modeling in the next steps. The final merged file has the following columns containing data from different sensors:

* **date:** This column has the timestamp of the data.
* **RefSt:** This column has the O3 value of the reference station. The value is in µg/m³.
* **Sensor\_O3:** This column has the data of the O3 sensor in KOhms.
* **Temp:** This column has the temperature data in ºC.
* **RelHum:** This column has the relative humidity data in %.
* **Sensor\_NO:** NO sensor data.
* **Sensor\_NO2:** NO2 sensor data.
* **Sensor\_SO2:** SO2 sensor data.

First, let us compare the plots of the reference station data and the O3 sensor data. The magnitude should be different, but the shape should be the same since the data is the same and is taken in the same periods/intervals of time.

Gráfico, Gráfico de líneas

Descripción generada automáticamenteGráfico, Gráfico de líneas

Descripción generada automáticamente

The shape of both graphs is the same. It has slight differences but overall, it has the same shape. The next step is to plot the relationship between both parameters. As mentioned before, both parameters are the same kind of data but taken by different sensors at the same time and with different units, thus, the relationship should give us a scatter plot with a 45 degrees shape approximately.

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

The graph confirms that the data of both sensors is related and has the same kind of variations. Next scatter plots show the relationship between the Sensor\_O3 data end all the other features to be in the model.

Gráfico, Gráfico de dispersión

Descripción generada automáticamenteGráfico, Gráfico de dispersión

Descripción generada automáticamenteGráfico, Gráfico de dispersión

Descripción generada automáticamenteGráfico, Gráfico de dispersión

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

The same is done for the RefSt data. Next plots show the relationship of the reference station data with all the other parameters in the data file.

Gráfico, Gráfico de dispersión

Descripción generada automáticamenteGráfico, Gráfico de dispersión

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamenteGráfico, Gráfico de dispersión

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

In both groups of plots (for the O3 sensor and for the reference station sensor), the temperature data is the one that has the most visible pattern. The next feature that shows a relationship pattern is the RelHum data. All the other variables do not have a distinctive pattern with the dependent variables of the model (Sensor\_O3 and RefSt). This indicates that these parameters should be the most important in terms of the construction of a model. As we will see in the next sections, this is true.

For the plotting of the former graphs, the data is not normalized since the analysis done in this section is more focused in the shape rather than in the magnitude or the value of the variables. For the modeling part, all the data is normalized due that is a mandatory previous step to get correct models.

## 1.3. Modeling and calibration

This section has the explanation of the different methods used to calibrate the data of the sensors and the different graphs obtained from the implementations in python. The first two sections are done using all the data features, then from KNN and on, the features used are the ones obtained as the best ones in the subset selection (the ones that are the most significative in the plots of the regularization methods).

### 1.3.1. Forward Subset Selection (MLR- FSS)

The first part of the project is picking which variables of the data in the datasets model the reference station data with the best metrics (the most significative amongst all the features). This selection is done by implementing Forward Subsect Selection in python. Forward subset selection has two phases:

1. First, it fits all the possible models that come from the combinations of k features (independent variables of the model) out of a total of P. For example, if the data has P = 3 features it will create a model with k = 1 (3 models with a single feature), k = 2 (3 combinations of two features) and k = 3 (one model involving all features). From each k group (k = 1, k = 2 and k = 3) the best model is selected using metrics as RSS, CV error or Adjusted R-squared. The k is given to the function and is the hyper parameter of the method. Then the criteria to choose the model is also an input of the python function.
2. The last step consists of picking the best model of all the models gotten in the first step. The selection in this phase is done using the same metric as in the first step.

The next graphs show the predictions done by a model with just k = 1 and a model done with k = 6 (all the features).

Gráfico, Histograma

Descripción generada automáticamenteGráfico

Descripción generada automáticamente

As expected, the predictions done by the simple model (k = 1) are bad and the predictions with all the features in the model are the best ones. The problem is that the last model is complex, and this can end up causing overfitting. The model can be made of significative features avoiding more features to be in. This can be seen in the next plot:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

The changes in the metrics when 2 main features are included in the model are not so significative. Therefore, the model of the sections KNN and forward is going to have just 3 features ['Sensor\_O3', 'Temp', 'RelHum'].

### 1.3.2. Multiple Linear Regression (MLR) with regularization

Multiple linear regression is a method to create models using two or more independent variables to predict the outcome of a dependent variable. The model is fitted using training data which adjusts the coefficients of the model to get to a point where the cost function between the model line and the training data is at its minimum. The cost function usually applied is least squares error (LS).

One of the problems with the models is that they can get complex whenever they have a lot of features (as in our case). When a model gets too complex it can suffer from overfitting, i.e., it fits the training data perfectly, but it has a huge bias with test data (hence, it has huge variance between the train and the test data). A way of solving this problem is to use a regularizing method. These methods shrink the coefficient estimates of the model to zero, so the model gets less complex (some of its less significative features disappear as they have coefficients close to 0 or equal to 0) and it avoids the risk of overfitting. Another way of saying the same is that whenever the model is penalized it becomes less sensitive to the changes in the features it has, so the ones that already had low impact on the model become inexistent (or almost inexistent) for the model. This also reduces the variance of the model without substantially increasing its bias. Ridge regression and Lasso regression are two examples of MLR with regularization methods.

#### 1.3.2.1. Ridge Regression (RR)

Ridge regression, also called L2 regularization, has the following formulation for N data points and P features:



The goal is to find the lambda that gets the minimum of the former formulation where RSS stands for Residual Sum of Squares. A lambda equal to zero corresponds to a simple linear regression (the penalizing term goes to zero). With greater lambda is, the less flexible the model gets, and it reduces model complexity and multicollinearity. Since great values of coefficients mean more flexibility (complexity), increasing the lambda of our model should give coefficients that get asymptotically near zero (but never equal to zero). This can be seen in the next graph where the coefficients of the model are plotted against the value of lambda (alpha in python).

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Also, we should have better fits for the data with alpha values in the lower range of the 0 to 1000 interval of the plot, because higher alphas give a hipper simplified model (insensible to almost all changes in data). Next plots show the reference station data in blue and the prediction in orange for different values of alpha.

Gráfico

Descripción generada automáticamente Gráfico

Descripción generada automáticamenteGráfico

Descripción generada automáticamente

The first graph corresponds to a simple linear regression prediction. The next two graphs to a low penalization (left) and a high penalization (right). The low penalization model gets closer to a good prediction as the simple linear model. The high penalization model gets worse predictions, the amplitude of the data is smaller in comparison with the other two plots.

Next plot shows the impact of the value of lambda in the metrics of the model. What we should see is a loose of relevance of the model (R squared should go low) and an increase in the error metrics since the less flexible a model is (less features the model has), less accuracy it has when predicting data.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

#### 1.3.2.2. Lasso Regression (LR)

LASSO (Least Absolute Shrinkage and Selection Operator), also called L2 regularization, has the following formulation for N data points and P features:



As it can be seen the LASSO regression is like the Ridge Regression, in fact they only differ in the penalization and what they achieve with it. In the case of LASSO, the penalization is the module of squares of the coefficients. This makes the penalization more aggressive towards bigger coefficients. This also enables the model to get rid of some of the coefficients because they can be 0. The demonstration can be explained as follows. If we think Ridge and LASSO as a problem to solve. For example, for Ridge of 2 parameters would be where s is a constant that exists for each value of lambda. Graphically:

Diagrama

Descripción generada automáticamente

The representation of the left is the LASSO problem, on the right is the Ridge regression problem. The red ellipses are the contours of RSS. Points of these ellipses are the RSS values. Regressions with point on the green zones have the lowest RSS value. When an ellipse has a huge s, its center lies in the green zone, this means that we are in a case of simple linear regression. For points of the ellipse that just touch the green zone we have the coefficients of LASSO/Ridge regressions. In the case of Ridge, since it is a circle, no point will touch a null value in any of the axes. In LASSO this can happen as it can be seen in the image. This makes 0 coefficients possible. In out model the coefficients of lasso are zero for maximum values of lambda:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

The most significative variables are the last ones turning 0 and the less significative ones are the ones that get to 0 earlier. In LASSO this also implies that the model will have more error and worse estimations. Next plots show four predictions two predictions with low alpha and two with high alpha (lambda).

Gráfico

Descripción generada automáticamenteGráfico

Descripción generada automáticamente

Gráfico

Descripción generada automáticamenteGráfico, Histograma

Descripción generada automáticamente

The greater the penalization, the worse the model gets, and the estimation is worse.

Next plot shows the evolution of the metrics of the model with respect to the value of the lambda.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

As we can see, with the penalization, the R square, which evaluates the fit and the sensibility of the model, decreases. Also, the errors increase because the model gets worse in the predictions when its less sensible (more rigid/inflexible).

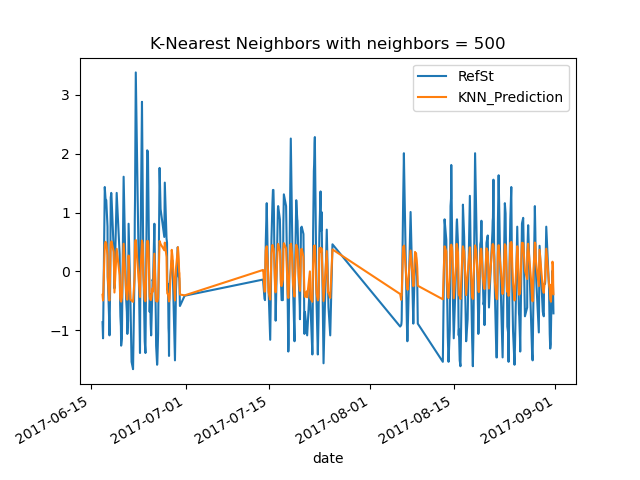
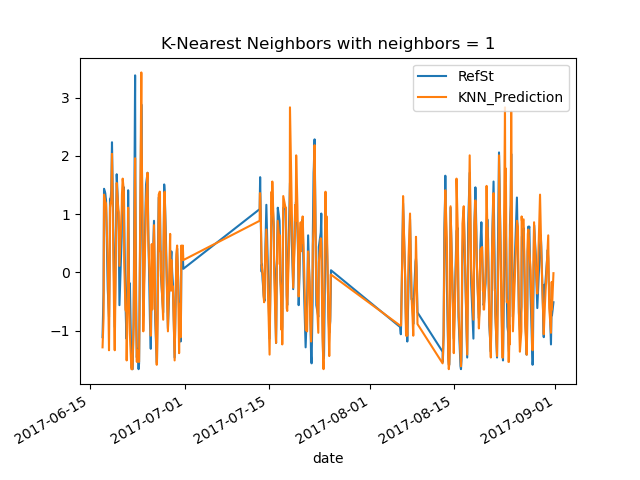
### 1.3.3. K-nearest neighbor (KNN)

KNN is a non-parametric method that approximates the association of independent variables and the outcomes by averaging the observations in the same neighborhood. It uses feature similarity to predict the values of the new data points, i.e., new points are assigned values depending on how close they resemble the training data. The size of the neighborhood is a hyperparameter. This method becomes impractical with huge dimensionality (great number of independent variables).

The steps of the algorithm are the following:

1. First, the distances (Euclidean distances) of the test data points and the training data points are calculated.
2. The closest k (number of neighbors hyperparameter) neighbors of each test data point are selected based on the distance calculated in 1.
3. The average of these k neighbors values is calculated for each test data point. This average is the prediction of the algorithm for a datapoint.

Knowing how the algorithm works we should expect bad predictions for small neighborhoods because the model becomes too simple and bad predictions for big values of k. For huge values of k, the algorithm selects neighbors with big distances to the test point that is being evaluated, and thus, the average value is perverted with wrong data (huge error). The next graph shows a model with a low number of neighbors and one with too many neighbors.



For a small number of neighbors, the result is an estimation with more amplitude that the real data in some places (it has error). If the number of neighbors is too big (close to the full size of the training set) the model is even worse. This can be seen in the next metrics graph.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

It gets to a point where the model is not fitting anything. The correct value of neighbors can be found in the k = 1 to k = 15.

Gráfico

Descripción generada automáticamente

Looking at the metrics tables, we can see that the best k value is 10.

### 1.3.4. Kernel Ridge Regression with RBF kernel (KRR-RBF)

Kernel Ridge regression consists of the kernelization of the Ridge Regression method. The method can be mathematically explained in the following (simplified) way:

1. Express using real-valued function , this is:

  with 

1. Define an empiric risk function  to assess how “good” a candidate function  (from the training set ) is. Typically, it is the average of a loss function.

 where n is the size of the training set .

The penalization in this method is applied on the functions of the feature space. This regularization also prevent overfitting.

1. Define a positive definite kernel on Z and solve  where the function can be expressed as:



Also, we have:  Where k is the Gram matrix.

1. Then the KRR problem is equivalent to:



This is a convex quadratic minimization problem.

The kernel measures the difference between the training set and the test set, in our case the kernel is RBF (Radial Basis Function) which has the following formulation:

where 

The results of increasing lambda should be the same as in the ridge regression method. The next plots demonstrate that this is true (and that a much smaller lambda is needed to penalize at the same rate).

**Gráfico, Gráfico de líneas

Descripción generada automáticamenteGráfico, Histograma

Descripción generada automáticamente**

The predictions get worse with higher lambda get worse results. This also is seen in the errors graph with respect to the lambda value:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Another parameter that is important for the RBF kernel is the which plays with the width of the gaussians of each dot of the reining set. If the gaussian is wider, it can cause overfitting since more points will enter in the range of significative weights (more points will be identified as similar, as it happens in k-nearest neighbors). This parameter has been left as default.

### 1.3.5. Random Forest (RF)

Random forest algorithm uses decision trees as base. The algorithm works in the following way:

1. N number of random records are taken from the training set. N is a hyper parameter and corresponds to the number of trees in the forest.
2. A decision tree is created from each sample of data.
3. Each decision tree generates an output.
4. The final output of the forest will be the mean of all the values of the leaves of the trees.

This means that more trees will have better results since the mean will consider more information from all the trees. Less trees will have the opposite effect. As in the other methods, the trick is to get the best number of trees which is the one that minimizes the error in the predictions of the model. Also, there is a point where including more trees makes no difference in the result of the forest since the mean is not changed. This can be seen in the next metric plot:

Gráfico

Descripción generada automáticamente

In this method, the difference from estimations of 1 tree and estimations of, for example, 12 trees are huge. This is shown in the next two plots.

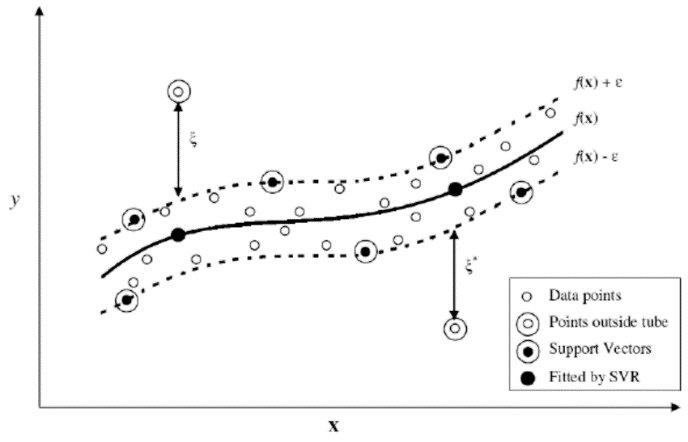
Gráfico

Descripción generada automáticamenteGráfico, Gráfico de líneas

Descripción generada automáticamente

### 1.3.6. Support Vector Regression (SVR)

SVR is a generalization of the classification in the Support Vector Machines method. In this case the output of the method is continuous instead of labeled. Next image shows how does the method work:



SVR, as SVM, is based on the construction of a tube (called epsilon insensitive tube). All the points are called vectors. This method only considers the vectors that are outside of the tube. The vectors outside of the tube are called support vectors since the regression line of the model is based on the value of the slack variables of these support vectors. The line in the middle of the tube is the regression line. We aim to create a tube that fits the best regression line with the data that we have. For this, the epsilon can be modified to set the tube width. In our case this parameter has been left to default (0.1). The wider the tube, the worse SVR is going to fit since the errors taken ion consideration will be fewer (all point will be inside the insensitive tube). The formulation for the Support Vector Regression is the following:



We aim to find a tube that minimizes the former expression. Notice that the last formula has a penalization term that comes from the summation of the distances of the support vectors and the tube (summation of slack variables) multiplied by a constant C. In the python code, this constant C can be tuned, and it is the hyperparameter used to create the graphs of this section. C can be seen as a regularization parameter.

The kernel used by the SVR method is the same as in KRR, the RBF kernel. The formulation of this kernel has been seen before and is:

where 

In our case, Gamma is set to auto. The results of the method for lower C and higher C are the following (in python, C works inversely, more C means less penalization):

Gráfico

Descripción generada automáticamenteGráfico

Descripción generada automáticamente

As we can see both models look good but the one that fits better is the one with less penalization (C = 500). Next plot shows the evolution of the metric with respect to the C value and confirms that less penalization gets better results (remember that C it is inverted in python).

Gráfico

Descripción generada automáticamente con confianza media

## 1.4. Comparison between the method results

After selecting the hyperparameters that give the best results for each one of the regression methods, the resulting table for all the metrics is the following:

Tabla

Descripción generada automáticamente

The plot that corresponds to the results of this table is the following:

Gráfico, Gráfico de barras

Descripción generada automáticamente

The conclusions of these results are the following:

* The best method is SVR it has the highest R squared, and the lowest RMSE and MAE. Its complexity makes it a good estimation method which can achieve better fits than the other methods explained during the report.
* KRR has similar results as SVR due that the basis of both algorithms is the same. Both use RBF kernel to get the results.
* The simple linear regression and the regularized regression methods have the overall worse results. This happens because those methods are simple compared to the other ones in the project.
* The best method amongst the MLR and MLR with regularization is simple MLR with the subsect selection. This is because all RR and LASSO have a penalization factor that makes them worse in the final model but better in the variance between the estimations. Also, LASSO has a more aggressive penalization and that is why it is the one with the worst results.
* Also, RF and KNN have similar results the best of them is RF because in a more randomized way so is able to learn more significant differences between the data points.

# 2. Fourth Project

Introduction