

# Assignment report,

Machine Learning

MSc Applied Bioinformatics,

Marie Schmit

# Abstract

Data exploratory and analysis

1. A PCA was applied to bot datasets with the function pca, package MixOmics. Enose samples clusters were in adequation with their sensory scores. Sample 10F9 could be considered as an outlier. HPLC samples were less clearly clustered by samples, but three close groups still emerged. There were some potential outliers like 0F12 or 5F6.

Table PCA scatter plots of enose and HPLC grouped by sensors class

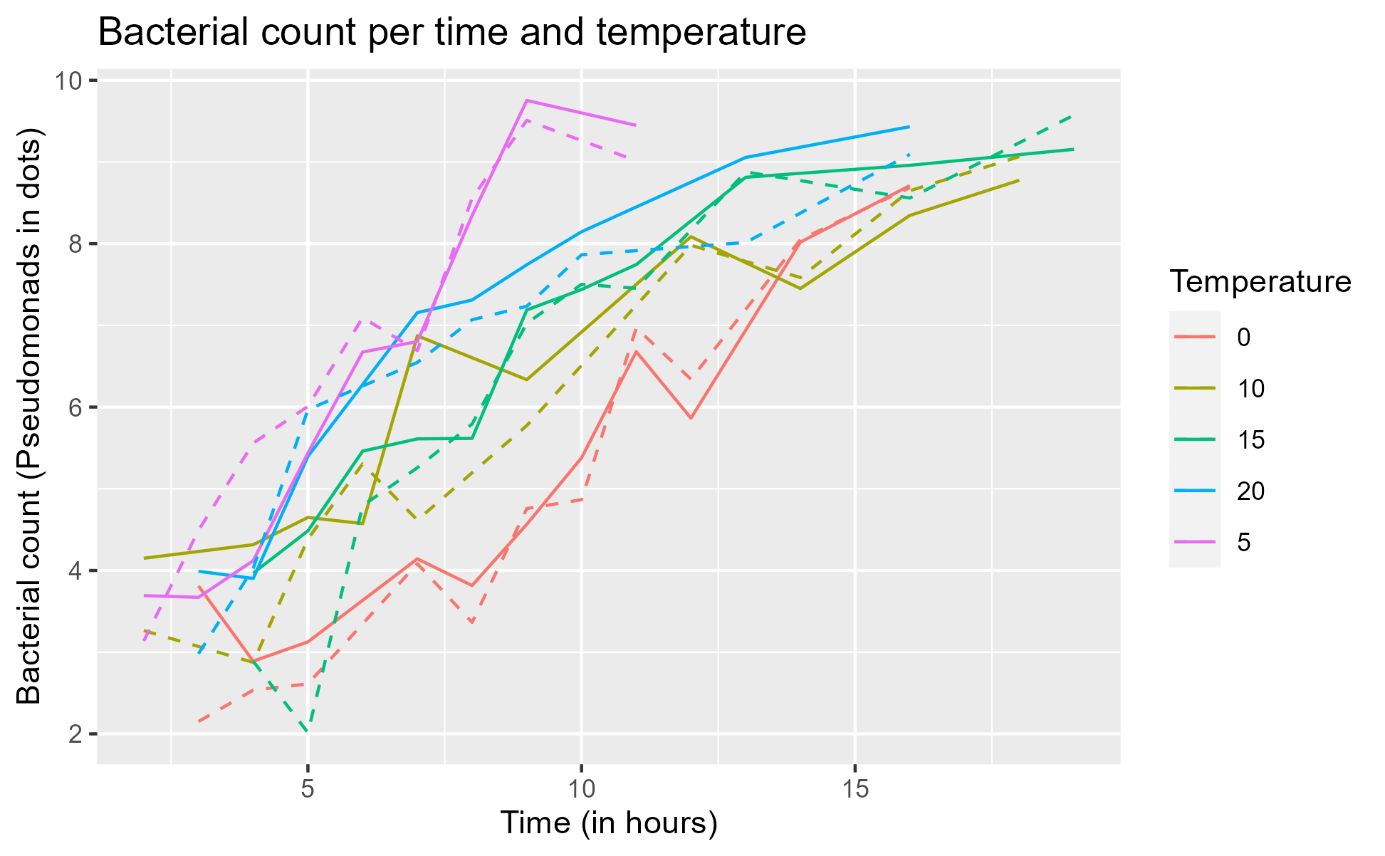
|  |  |
| --- | --- |
|  |  |

Analytical methods were tried to find better separation. First, PCA plots were displayd with other methods (in2D, 3D, in a biplot). Then, HCA was used with the function pheatmap, from the package pheatmap. With this method, three clusters emerged, but they did not necessarily correspond to their sensory value. The previous outlier for enose (10F9) was no longer clustered corresponding to its sensory value, but it was in the same cluster as F1a. The same goes for HPLC, for which the clusters 2 and 3 were often mixed. The clustering was better with PCA analysis method.

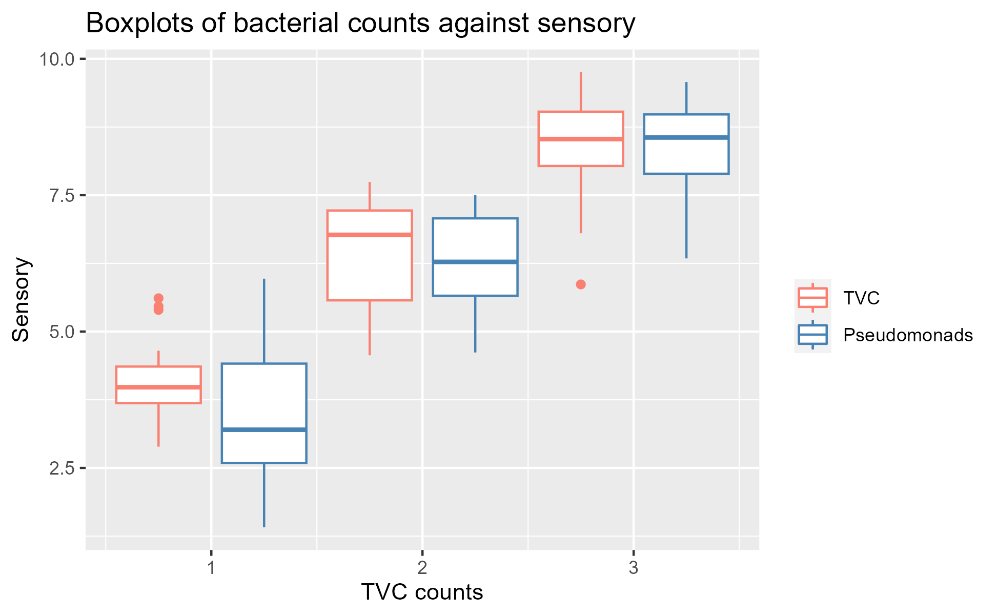
|  |  |
| --- | --- |
|  |  |

Since potential did not stand out enough to be remove (inducing a risk of errors and a loss of information), they were not removed.

1. The number of bacteria, for both type of bacteria, was increasing with time. They were more bacteria for higher temperatures, but the count of bacteria was very high for 5 degrees, temperature for which less values were provided. The number of bacteria seemed generally to grow with time and temperatures increase.



1. The number of bacteria was for both types TVC and Pseudomonias higher when the sensory score was higher. Rotted meat had thus more bacteria than fresh one. TVC bacteria were slightly more numerous than pseudomonas. Also, TVC had three outliers.



Time and temperature have both an influence on bacteria growth: as time passed and especially for temperature around 20C, TVC and pseudomonias bacteria grow. Those bacteria cause the meat rottenness.

## Classification

Three different classification methods (k nearest neighbours, random forest, support vector machine) were used to determine to which sensory class belonged enose and HPLC measures. Data was split into a training and a test set for cross validation. Each model was tunned to optimise its hyperparameters, and the results were compared, to established what model was the best for this classification.

Data was partitionned using the function createDataPartition from the package caret, that ensured a balanced representation of the train and test sets. This function creates an index of each set. The partition was made reproducible, with the seed set at 8.

Knn classification

A model using k nearest neighbour methods was first trained. The aim was to optimise it by finding the best fit and k parameter. Since knn is a distance-based algorithm, different methods of scaling (center, auto scale and range scale) were tested, to give the features the same weight in distance calculations. Different values of k, the number of nearest neighbours’ data taken in the same cluster, were tested, from 1 to 20. The model was trained using the function knn, from class package, that evaluate the Euclidian distance between nearest neighbours and decide the classification by majority vote (R documentation, knn {case}). The accuracy of the model was evaluated for each of those tested hyperparameter, using the function confusionMatrix from the caret package, that evaluates the cross-tabulation of predicted and observed data (R documentation, confusionMatrix {caret}). The parameters that were kept were the one leading to the best accuracy.

For HPLC data with one iteration, the results were the following. Without scaling, the best accuracy was 0.9 for k =7. With scaling, the best combination was no scaling with various k values (different k give the same accuracy, k=7 were chosen). Scaling was not necessary here. The selected tunned model was thus: no scaling, k=7.

Table Cross table for HPLC data, 1 iteration, k = 7, no scaling

Calendar

Description automatically generated with low confidence

This model was then trained on 100 iterations. For each iteration, the accuracy value was saved in a list that was later used to calculate the cumulative mean accuracy.

The same process was repeated for enose data. The best accuracy was 1 for non-scale data, k=3. This could indicate an overfitting: a model too much trained on specific values to be performant on new datasets. However, all the accuracies values were not equal to 1. Reducing the number of train with a 5:5 (50% of training data, 50% of test data) partition instead of a 7:3 reduced the accuracy to 0.83, which decreased the overfitting. The first partition of 7:3 was used to calculate the cumulative mean value for 100 iteration, k=3 and no scaling, to evaluate the overfitting issue with a larger number of trains. The mean accuracy for 100 iterations was 0.791. Even if the best accuracy value was still 1 (which could be due to the small number of trained values), this cumulative mean accuracy was not indicating a problematic overfitting, so the model was kept.

Svm classification

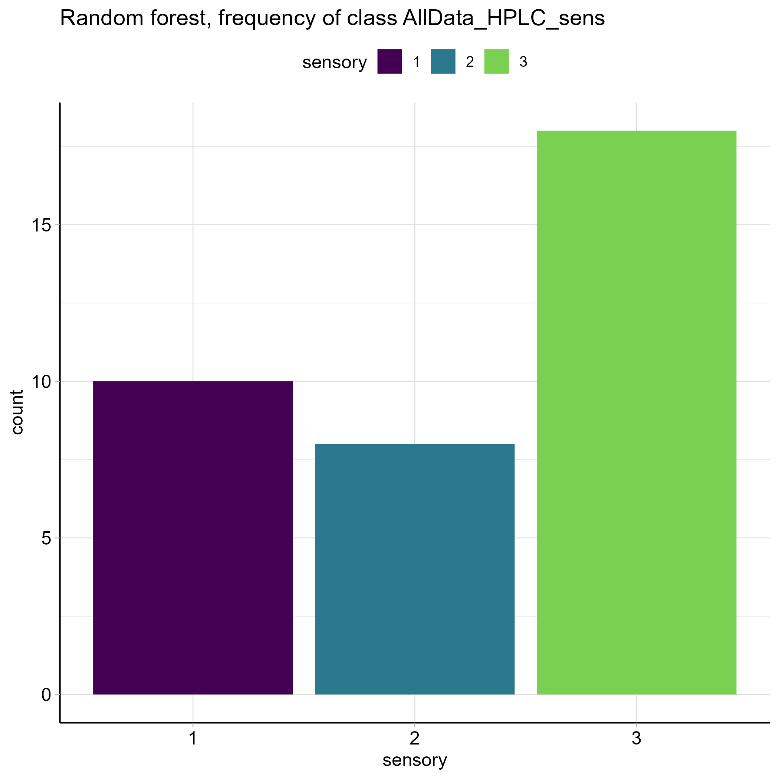
The svm-rd classification method was then applied to both datasets, using the partition of data previously created for knn. The function ksvm from package kernlab was used to train the model on the predictor sensory. To tun the model, different kernels were tested: 'vanilladot' the linear kernel, 'polydot' the polynomial kernel, 'rbfdot' the Gaussian kernel , 'tanhdot' the hyperbolic tangent kernel (R documentation, dots {kernlab}). The best model was the one leading to the best accuracy, calculated like before with the function confusionMatrix from caret package.

For HPLC data, the bast accuracy was 0.7 for the kernel “rbfdot” (the Gaussian kernel). This model was then train over 100 iterations, and its accuracy was saved in a vector for every iteration. The cumulative mean accuracy was 0.789. This approach was repeated for enose data. With the same kenerl, the best accuracy was 0.9, which was like for knn model a high accuracy, leading to a suspicious of overfitting. The cumulative accuracy for 100 iterations was 0.789.

Random forest

For random forest classification, a task was defined, with the predictor sensory, allowing to check the number of elements in each sensory class. A majority of the data belonged to class 3 (rotten meat) for HPLC.

Table Frequency plot, data HPLC, sensory classification



A learner was set for random forest with the function lrn from mlr3 package. The hyperparameters ntree which is the number of trees in the forest; mtry, the number of features to sample at each node; maxnodes, the maximal number of leaves; nodesize, the minimum number of cases authorised in a terminal node (leaf) were tuned. The lower and upper values of those parameters were defined: the number of features randomly sampled at every nodes had to be lesser than the total number of features, but still large enough to analyse trends of the data. The nodesie had to be small enough to avoid a too big tree but large enough to avoid underfitting. The methods of evaluation were resampling strategy cross validation (mlr3::rsmp) and performance measurement of classifier accuracy (mlr3::msr). Since it is too heavy to evaluate every hyperparameter, the tuning was stopped after 20 evaluations (bbotk::term). It was performed with grid search that evaluated each combination of hyperparameters (mlr3tuning::tnr).

For HPLC, the best random forest parameters were 200 trees, 2 feature sample at each node, 20 nodes per leave maximum, 2 allowed cases in a leaf minimum, with an accuracy of 0.84.

Table Confusion matrix for HPLC tuned model

Text

Description automatically generated with medium confidence

This model was tested for one iteration. The accuracy was 0.7, one sample of sensory 1 and one of sensory 2 were misclassified.

Chart, bar chart

Description automatically generated

The model was then trained and tested for 100 iterations, the mean accuracy for all iterations was 0.813.

The same steps were repeated for enose data set. The maximal number of features to sample for each node was set to 8 since enose has 8 sensors. The best model had 200 trees, 2 features to sample per node, 20 leaves maximum, 2 cases allowed in terminal node. The accuracy for this model was 0.731 for one iteration.

Table Confusion matrix for one iteration for tunned model, enose data

Text

Description automatically generated

Two elements were misclassified: one for sample one, the other for sample two.

Chart, bar chart

Description automatically generated

The model was trained and tested with the optimised hyperparameters for 100 iterations. The cumulative mean of all iterations was 0.745.

1. The cumulative mean accuracies were claculated for every dataset (enose and HPLC).
2. Graphical user interface, chart

   Description automatically generated

**For HPLC data, random forest had the higher cumulative mean**. Knn was very high for a few iterations, but it dropped rapidly to approximatively 0.7 accuracy.

|  |  |  |
| --- | --- | --- |
| Dataset HPLC | Method | Cumulative mean accuracy after 100 iterations |
|  | Knn | 0.714 |
|  | Svm | 0.789 |
|  | **Random forest** | **0.813** |

Chart, histogram

Description automatically generated

**For enose data, knn had the higher cumulative mean accuracy**. It was very high for a few iterations, then dropped and got higher again. Svm had a similar pattern. Random forest had a very poor mean accuracy at the beginning, but it got higher with more iterations.

|  |  |  |
| --- | --- | --- |
| Dataset enose | Method | Cumulative mean accuracy after 100 iterations |
|  | Knn | 0.791 |
|  | **Svm** | **0.789** |
|  | Random forest | 0.745 |

1. For each algorithm, we the number of misclassified (false positive or false negative number) samples was calculated for each iteration (gmodels::CrossTable), then summed up. Each of those sums was stored into a vector containing three element (one sum per class). The sum of the proportions was calculated for 100 iterations and the results were stored in a similar vector (see all the plots in folder “Plots”).

For enose, rf and knn had the smallest proportions of misclassification, with almost no misclassifications (2.5 in total proportion) for the class 1 for knn (the “true” class, corresponding to fresh meat). For HPLC, it was rf that had the smallest numbers for all the classes (with maximum 10% of missclassifications for class 1), but svm had the smallest number of misclassifications sum of proportion (approximatively 2).

|  |  |
| --- | --- |
| Chart, bar chart  Description automatically generated | Chart, bar chart  Description automatically generated |

1. In order to find the importance of every variable for the classification, a model was trained for knn (carety::train). The model was tuned using a grid search of k from 1 to 20. For knn, the importance (calculated with caret::varImp) of all the variables was positive. With HPLC data, RT7.9 and RT7.5 had both the highest importance, 100, for classes 1 and 3. RT6.8 was with 100 the maximum importance for class 2. RT18.7 had the smallest importance, approximatively 5, for fresh meat. RT8.4 had the smallest importance (approximatively 8) for both classes 1 and 3.

Chart, bar chart

Description automatically generated

For enose data, DF7 and DF1 had the largest importance of 100 for both classes 1 and 3. DF1 had also the maximum importance of approximatively 82. DF6 was the variable with the smallest importance for classes 2 and 3, with less than 5 importance. This variable was not important for class 1 (importance of 0).

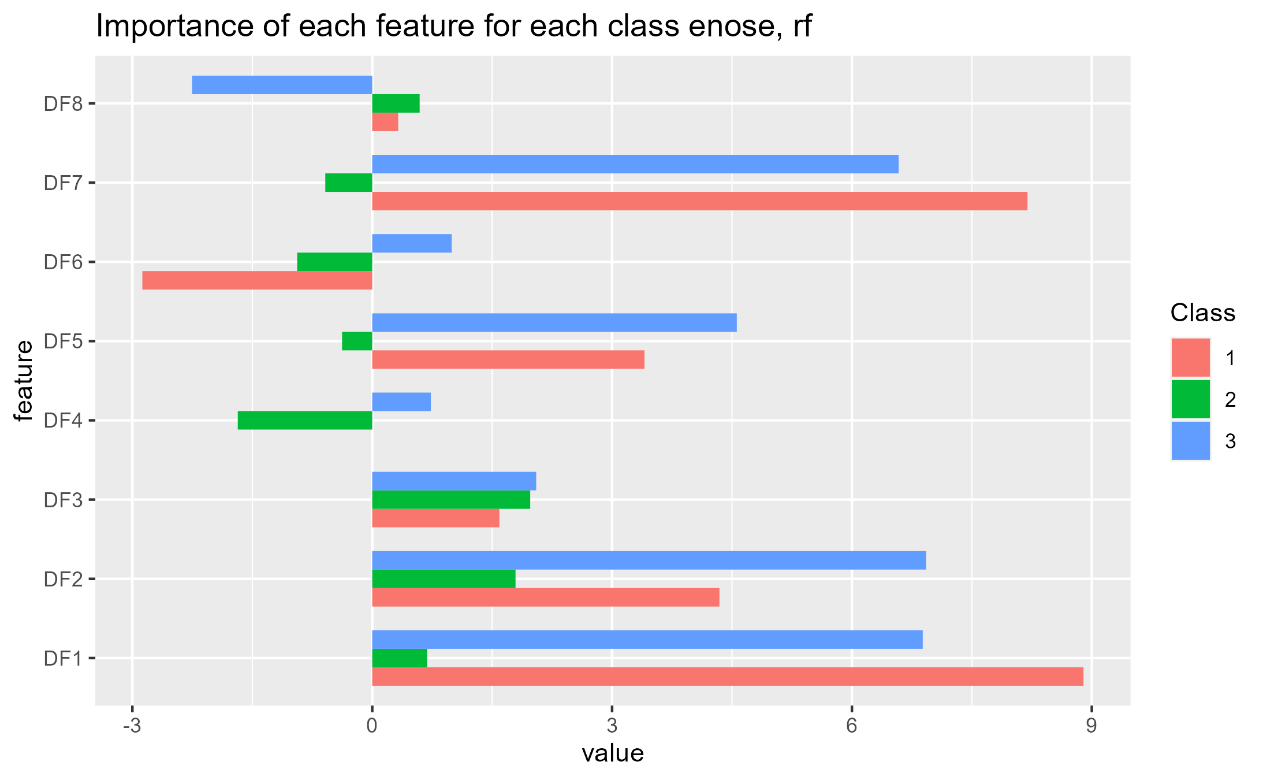
Chart, bar chart

Description automatically generated

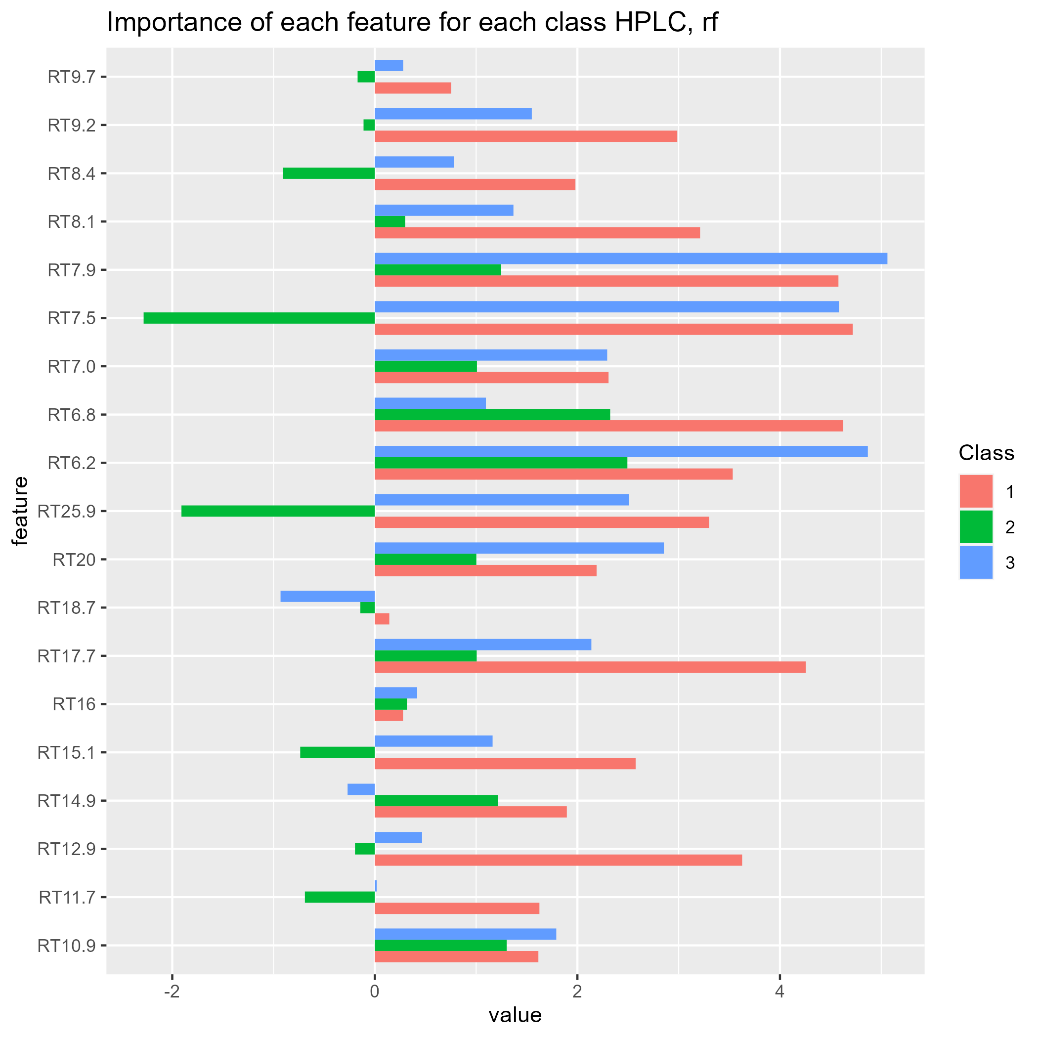
For svm, the model was again trained using the train function and the method which was the best method found with tunning (kernel rbfdot).

The model random forest was trained with randomForest::randomForest, the importances assessed with caret::varImp. For enose data and random forest, DF1 and DF7 had the major importance for both class 1 (importance between 8 and 9) and 3 (importance around 6.5). DF2 had the same importance for class 3.

The features did not consequently influence class 2, with some negative importances for DF4, DF5, DF6 and DF7 (the values were respectively: -2, -0.5, -1.5, -1). The higher values for class 3 were approximatively 2.5 for DF3 and DF2.



For HPLC, again with random forest, class 3 had again the lesser importance. Indeed, they ranged between -2 and 2.5. On the contrary, class 1 did not have negative values and had higher values in general, with RT7.9, RT7.5, RT6.8 and RT12.9 around 4.5. Class 2 was negatively influenced (value -1) by RT20, but highly influenced (respectively 5, 4.5, 4.9) by RT7.9, RT6.2 and RT20.



HPLC data importances summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Minimum or maximum importances | Class 1 | Class 2 | Class 3 |
| Knn | Max importance | RT79.5  RT79.8  RT6.8 | RT6.8  RT6.2 | RT7.9  RT7.5 |
| Min importance | RT18.7 | RT8.4 | RT8.4 |
| Svm | Max importance |  |  |  |
| Min importance |  |  |  |
| rf | Max importance | RT7.9  RT7.5  RT6.8 | RT6.2  RT6.8 | RT7.9  RT7.5  RT6.2 |
| Min importance | RT18.7  RT7.17 | RT25.9  RT7.5 | RT9.7  RT18.7 |

enose data importances summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Minimum or maximum importances | Class 1 | Class 2 | Class 3 |
| Knn | Max importance | DF7  DF1 | DF1 | DF7  DF1 |
| Min importance | DF6 | DF6 | DF6 |
| Svm | Max importance |  |  |  |
| Min importance |  |  |  |
| rf | Max importance | DF7  DF1 | DF3  DF2 | DF2  DF1 |
| Min importance | DF6 | DF4 | DF8 |

# Regression

Three regression methods were studied (knn, RF, PLS-R) to established a relationship between the predictor (the data from enose and HPLC) and a predicted number of bacteria depending on them. The number of bacteria in the meat was indeed an indicator of its freshness, as seen in the first part of thisreport. The aim was to determine the best regression method.

The first regression used was knn. The function caret::train was used for the model tuning, with the option tuneGrid that conducted a grid search to find optimal parameters. Here, the hyperparameter of interest was k, and its tested values went from 1 to 20.