Dataset Analysis using Classification and Unsupervised Learning

Analysis of two datasets concerning heart failure patients and toxic molecules using different methods.

ANDRÉ, JF, ARAÚJO

JOÃO, FES, PORTO

MIGUEL, SCF, NEVES

Instituto Superior Técnico

1 DATA PROFILING

1.1 Data Dimensionality

In the Heart Failure dataset there are 299 records and 13 variables (no dimensionality curse), of which 6 are boolean (*anaemia*, *diabetes*, *high_blood_pressure*, *sex*, where 0 stands for female and 1 for male, *smoking* and the target variable *DEATH_EVENT*), 4 integer (*creatinine_phosphokinase*, *ejection_fraction*, *serum_sodium*, *time*) and 3 float (*age*, *platelets*, *serum_creatinine*). There are no missing values. There are 203 records with *True DEATH_EVENT* and 96 records with *False DEATH_EVENT* (imbalanced classes).

In the Oral Toxicity dataset there are 8992 records and 1025 variables (no dimensionality curse), of which 1024 are boolean and 1 categorical, the target variable, with values *negative* and *positive*. There are no missing values. There are 8251 records with *negative* target variable and 741 with *positive* (imbalanced classes).

1.2 Data Distribution and Granularity

In the Heart Failure dataset, every numeric variable, except for *time*, has outliers (blue dots in Figure 1.1). 75 records have 1 outlier, 10 records have 2 outliers, and 0 records have 3 or more outliers. Because removing 75 records would be too many and removing 10 too few to make a difference, we decided not to remove outliers.

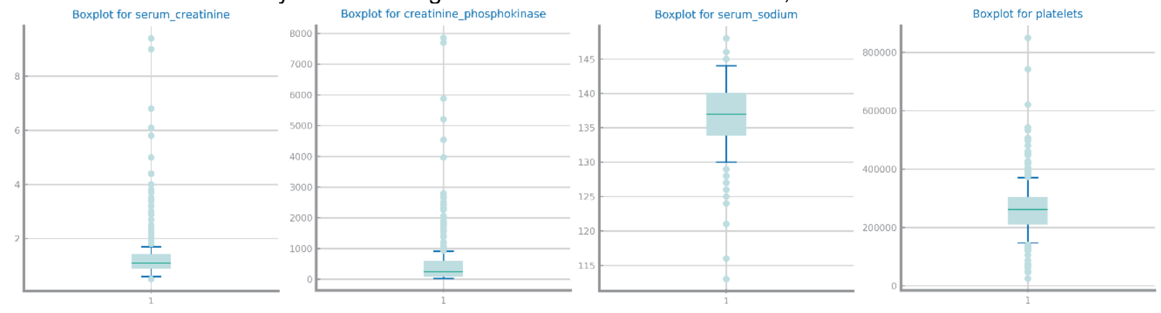


Figure 1.1: Boxplots for features with more outliers from Heart Failure dataset.

The variables *age* and *ejection_fraction* seem to follow a lognormal distribution, *creatinine_phosphokinase* and *serum_creatinine* an exponential distribution, *platelets* and *serum_sodium* a normal distribution and *time* doesn't seem to follow any known distribution (see Figure 1.2). To find these distributions we discretized the variable *age* in 10 bins, *creatinine_phosphokinase* in 20 bins, *ejection_fraction* in 8 bins, *platelets* in 10 bins, *serum_creatinine* in 14 bins*,* *serum_sodium* in 18 bins and *time* in 50 bins.

In the Oral Toxicity dataset there are no outliers because every variable is either boolean or categorical. There are 805 variables that are either *True* or *False* in more than 90% of the records.

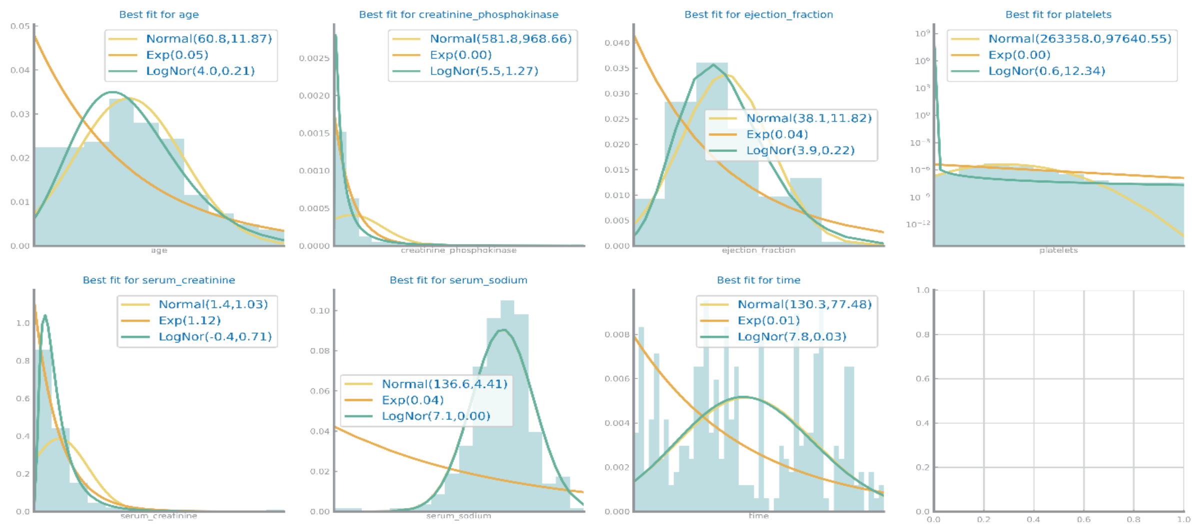


Figure 1.2: Histograms for Heart Failure Dataset

1.3 Correlation Analysis

In the Heart Failure dataset there are no high values (absolute value greater or equal than 0.9) of correlation between variables. That said, there is a positive correlation between the variables *sex* and *smoking* with value 0.45 that indicate that, in our dataset, males are more likely to be smokers than females. There is also a negative correlation between the variables *time* and *DEATH_EVENT* with value -0.53, that indicate that the longer the follow up period the higher the likelihood of the patient not dying of heart failure. The variables *age*, *ejection_fraction*, *serum_creatinine*, *serum_sodium* and *time* are the 5 most correlated variables with the target variable, ranging from absolute values between 0.20 and 0.53. In the Oral Toxicity dataset there are 72 high values of correlation between variables. There is just 1 variable highly correlated with the target variable. There are 854 variables with an absolute value of correlation with the target variable smaller than 0.05.

2 CLASSIFICATION

We tried several data preparation techniques: feature selection (using anova to select the k best features), data balancing (undersampling, oversampling, SMOTE and a mixed strategy, with half oversampling and half undersampling) and variable scaling (Z-Score and MinMax between 0 and 1). We split the Oral Toxicity dataset into train, validation and test sets using holdout strategy and the Heart Failure into train, validation and test sets using stratified k fold strategy with 5 splits. Every experiment was repeated 10 times and the averages computed to ensure we were not using a favorable split.

2.1 Naive Bayes

Of the data preparation techniques, the ones that may increase the performance of Naive Bayes models are data balancing and feature selection.

2.1.1 Heart Failure

We obtained the best results with an accuracy of 81.7% using a Gaussian classifier with oversampling of the training set. We managed to further increase the accuracy of the classifier selecting the 4 best features, achieving an accuracy of 83.0%.

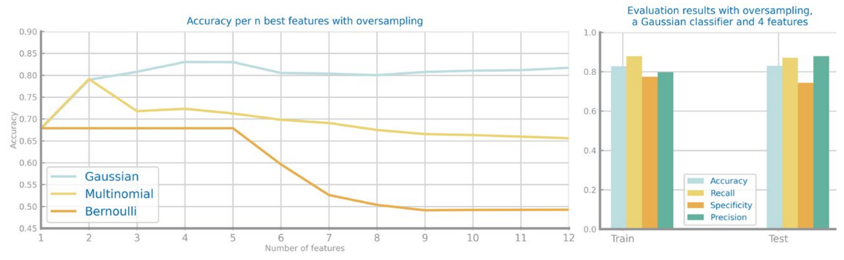


Figure 2.1: a) Model accuracy over different number of features b) Final evaluation results

2.1.2 *Oral Toxicity*

With the 40 best features the Multinomial classifier hits its peak of 92.1% accuracy, decreasing steadily until stabilizing with more than 70 features. We then applied balancing techniques to the training set and, even though the recall improved greatly, the accuracy decreased drastically.

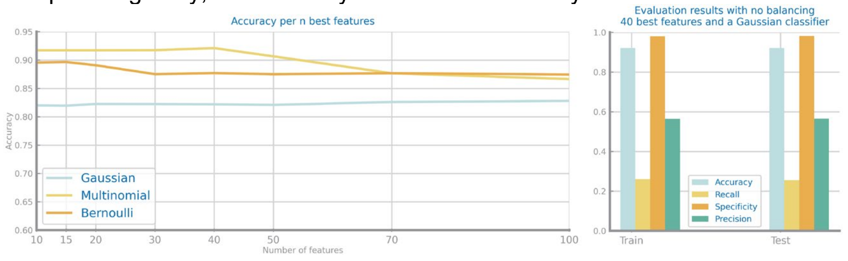


Figure 2.2: a) Model accuracy over different number of features b) Final evaluation results

2.2 Knn

From the data preparation techniques, the ones that may increase the performance of the Knn classifier are data balancing, feature selection and variable scaling. Variable scaling is not applicable to the second dataset because the variables are all boolean. We tested with Manhattan, Chebyshev, Euclidean and Jaccard distance.

2.2.1 *Heart Failure*

We started by testing the accuracy of the model without data preparation and obtained a peak accuracy of 70.6% with the Manhattan distance and 11 neighbours. We then tested balancing techniques to see if we could improve the accuracy further, but every technique decreased the model's accuracy to less than 60%. We managed to increase the accuracy to 87.4% by choosing the 5 best features, with the Manhattan distance and 5 neighbours. We also tried different scaling tecnhiques but none of them improved the model.

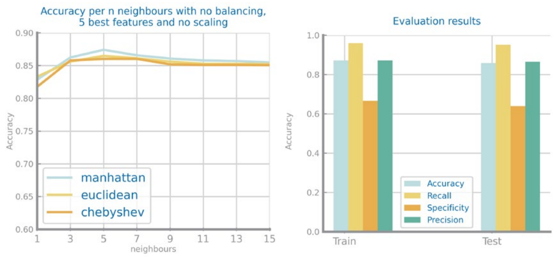


Figure 2.3: a) Model accuracy over different number of features b) Final evaluation results

2.2.2 *Oral Toxicity*

We started by testing how the model behaved with different number of features and distance measures with a default value of 5 neighbours and concluded that the accuracy peaked at 93.8% with the 200 best features and Jaccard distance. We also tried balancing the training set but, just like in Heart Failure, it did not increase the accuracy. We then varied the number of neighbours and obtained the best accuracy with 5.

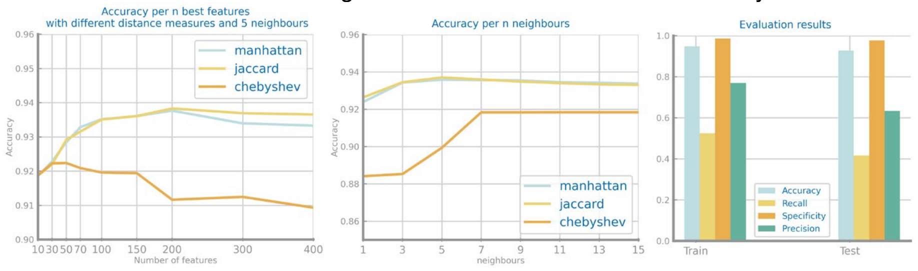


Figure 2.4: a) Model accuracy over different number of features b) Model accuracy over different number of neighbours c) Final evaluation results

2.3 Decision Trees

The parameters *maximum tree depth* and *minimum impurity decrease* were studied and manipulated to maximise accuracy. Data preparation techniques were studied with a Grid-Search approach to maximize performance metrics. As decision trees are impervious to scaling of the variables, we favoured studying feature selection and extraction which greatly affected the metrics, especially in the Oral Toxicity dataset.

2.3.1 *Heart Failure*

After testing, SMOTE proved to be the better option for balancing this dataset. When testing we noticed a plateau when increasing maximum depth beyond 10. This is explained by the size of dataset; the tree leaves have fewer and fewer records as depth increases until there are not enough records to split the leaf. Feature selection techniques show a slight increase in specificity score. Applying Principal Component Analysis worsens the model as the already low number of available features decreases dramatically and thus leaves very little options for the Decision Tree algorithm to decide how to split the records. The best tree's performance is described in figure 2.5.b, it achieves a specificity of 88.5% and an accuracy of 88.8%. It is the result of balancing the dataset with oversampling and selecting the top 10 features through anova.

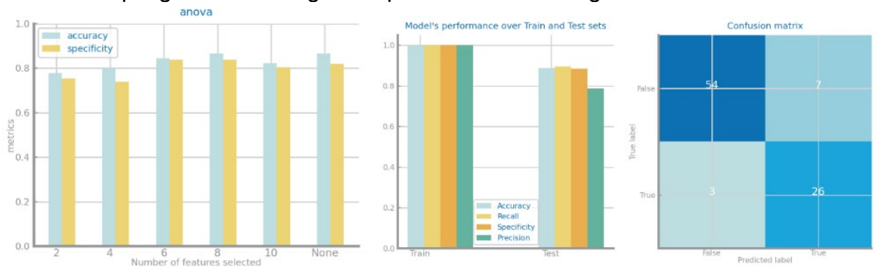


Figure 2.5: a) Decision Tree with different number of features b) Decision Tree results for the best performing estimator

2.3.2 *Oral Toxicity*

For the analysis of this dataset, oversampling proved to be the better sampling technique due to the heavy unbalance of the record classes. After this sampling, results showed a plateau in accuracy after the tree depth exceeds 30. A bigger dataset, with higher dimensionality, explains why the plateau in performance occurs much later than in the previous dataset as leaves now have enough records to continue dividing and the algorithm has more variables to divide them by. When applying feature selection, runtime was slightly improved and choosing 100 variables resulted in best accuracy. However, there was a significant decrease in accuracy, from 0.95 to 0.79. Principal Component Analysis greatly improved runtime and displayed the best results, outperforming in all metrics obtained so far. Higher *minimum maintained variance* values resulted in better metrics but also in a higher runtime, as seen in Figure 2.6.a. The best equilibrium was found with this value set at 0.9. The results from this tree are shown in Figure 2.6 b and c.

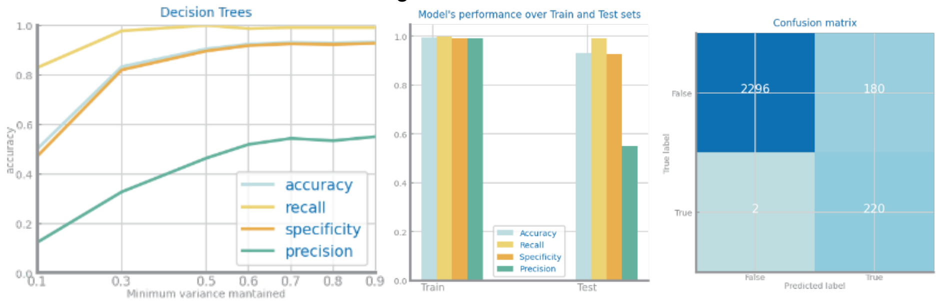


Figure 2.6: a) Plot displaying how *minimum maintained variance* affects performance b) Decision Tree evaluation results with Principal Component Analysis with *minimum maintained variance*=0.9 c) Confusion Matrix for best Decision Tree

2.4 Random Forests

Of the data preparation techniques, the ones that may increase the performance of the Random Forest classifier are data balancing and feature selection.

2.4.1 Heart Failure

We started by testing if balancing would improve the accuracy of the model with some default parameters and, even though marginally, oversampling did improve it. We then tried to see how the model behaved with the number of features and the accuracy was the highest with all 12, with a value of 87.4%. Even though we obtained the best accuracy with 12 features, after 5 the gain in accuracy was very negligible. We obtained the best accuracy with a value of 88.0% with depth 10, 30% of maximum features and 300 estimators.

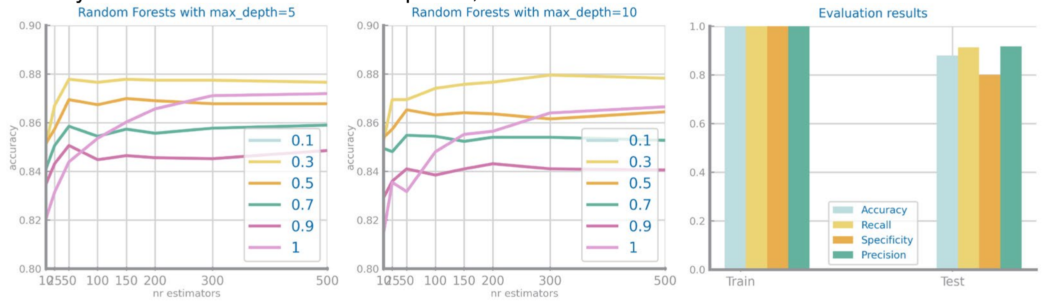


Figure 2.7: a) Results with different Random Forest hyperparameters b) Final evaluation results

2.4.2 Oral Toxicity

We started by testing if balancing would improve the accuracy of the model with some default parameters and, just like in Heart Failure, oversampling did marginally improve it. We then tried to see how the model behaved with the number of features and the accuracy stabilized after 300 features. We obtained the best accuracy with a value of 94.0% with depth 75, 100% of maximum features and 75 estimators.

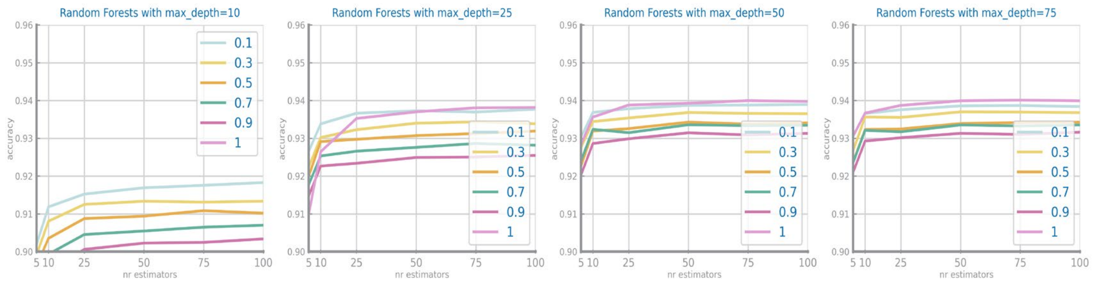


Figure 2.8: Results with different Random Forest hyperparameters

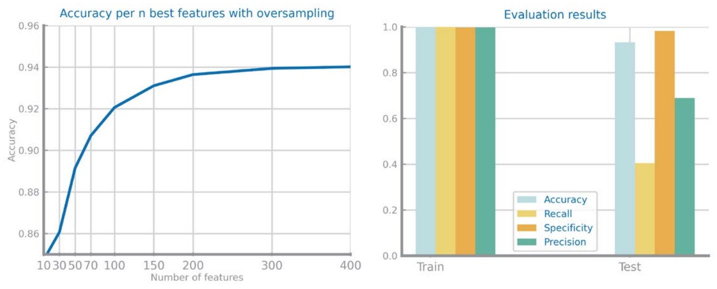


Figure 2.9: a) Model accuracy over different number of features b) Final evaluation results

2.5 XGBoost

The parameters *maximum tree depth, learning rate* and *number of estimators* suffered little changes throughout the analysis as data preparation techniques had the biggest impact on the performance of the estimator.

2.5.1 Heart Failure

Based on the information gathered in the Decision Tree chapter, the maximum depth of the trees should be lower or equal to 10. However, increasing *maximum depth* did not affect the performance very much as the best estimator was achieved with a *maximum depth* of 2. Instead of a deep tree, XGBoost appeared to distribute its decision through multiple shorter trees. This dataset was balanced with SMOTE throughout this analysis, other balancing techniques were studied but wielded worse results. Feature engineering had a clear effect on the metrics obtained and managed to slightly improve accuracy and specificity scores. In Figure 2.10 b and c we can see the results surpass the previous best, becoming the best estimator achieved with 10 best features.

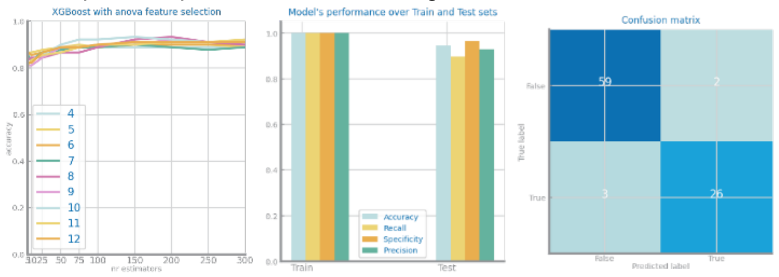


Figure 2.10: a) XGBoost metrics comparing different number of chosen variables b) evaluation results for the best performing estimator c) Confusion Matrix for best model

2.5.2 Oral Toxicity

The high number of records and dimensionality of this dataset makes it impossible to study apply this algorithm without feature engineering. Thus, two main approaches can be taken: PCA and Feature Selection. When applying feature selection, runtime was slightly improved over baseline but still slower than PCA. Although accuracy seemed high, a closer look into evaluation results showed a bias towards misclassifying non-toxic substances. Principal Component Analysis showed a plateau in performance after 50 estimators, with 100 estimators achieving only slightly better scores and higher estimator numbers not increasing any performance metric. The best estimator was chosen when *minimum_maintained_variance* was 0.9 and its evaluation results are seen in figure 24.b. The best estimator reached an accuracy of 98.44% and a specificity of 99.0%.

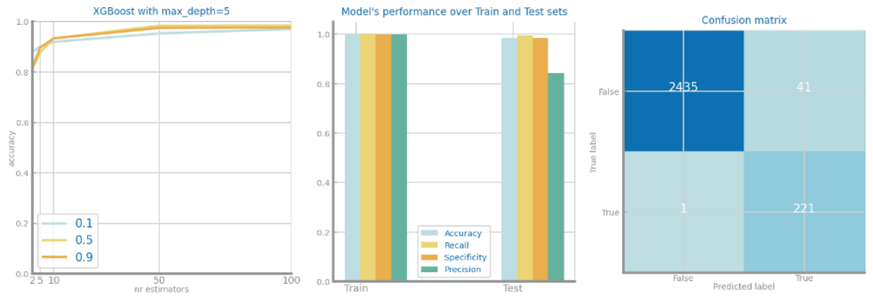


Figure 2.11: a) XGBoost accuracy results combining different learning rates and number of estimators b) Evaluation results using PCA with minimum_maintained_*variance*=0.9.

2.6 Conclusions

As we can see by the experiments, the accuracy of the models rarely increased after selecting a certain number of features. This happens because the features being selected by anova were, in the case of the Heart Failure dataset, the most correlated features with *DEATH_EVENT* and, in the case of the Oral Toxicity dataset, features with high variance, meaning they could provide more information about the target variable. We could also detect that even tough balancing rarely or just marginally improved the accuracy of the models, it did substantially improve other metrics, like specificity and recall. Finally, the accuracy results obtained for the Oral Toxicity dataset were always higher, because of the boolean nature of the variables and the significantly larger number of records.

3 UNSUPERVISED LEARNING

Unsupervised Learning consists of detecting patterns in a dataset without taking in consideration any target feature or pre-existing labels. The two processes we will consider here will be Association Rules and Clustering.

3.1 Association Rules

Association rule learning, or pattern mining, is used to discover interesting relationships between variables.

Although this is an unsupervised learning technique and these datasets contain target variables we decided to keep the target variable when applying pattern mining to observe any rules that may link other features to the target ones, this doesn't affect other rules that don't include the target variable, since they depend only on the features present.

To evaluate each rule we used support, confidence, and lift. For a rule to be useful it needs good support and good confidence. Support acts as a barrier for the rules we want, since after going under a certain value the rules increase exponentially, and the confidence is not the best measure for comparing rules since at lower support levels it tends to be a very high value. Given this, we use lift as our primary metric, so the most useful rules will be the ones with a good minimum support, close to 100% confidence and lift distant from one.

3.1.1 Heart Failure Dataset

Before applying pattern mining, we had to discretize the numeric variables, so we segmented these variables into the same bins used for finding their distribution (Section 1.2) and then dummified them.

As explained above, we see in Figure 3.1 that the number of rules increases with lower support, we have a lot of rules with high confidence (~7000 with 100%), and the rules with best lift also have high confidence.

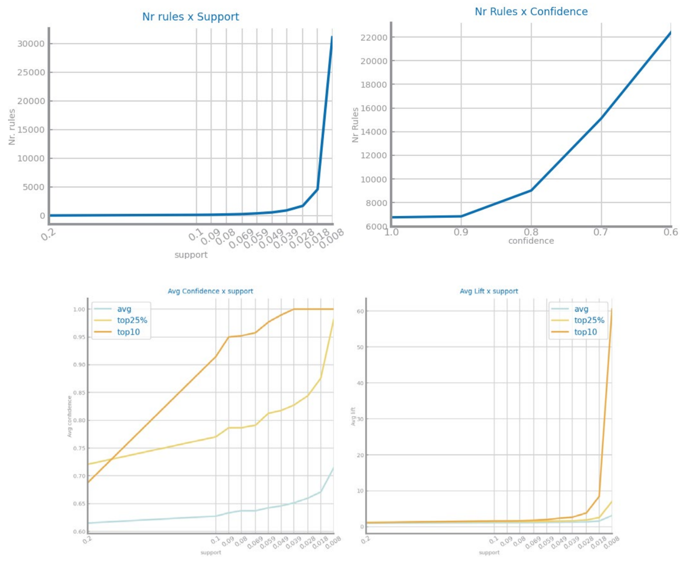
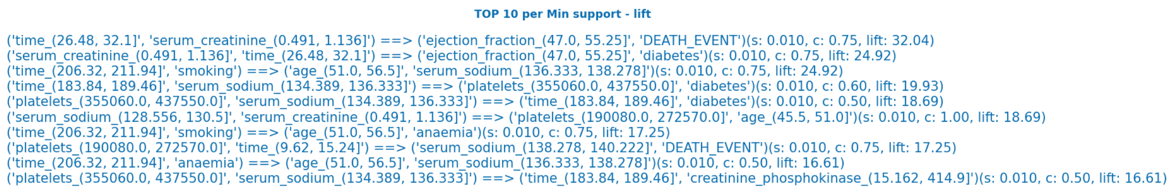


Figure 3.1: Metrics from pattern mining a) Number of Rules per Support b) Number of Rules per Confidence c) Average Confidence per Support d) Average Lift per Support

Looking at the graphs we can see 0.01 is a suitable minimum support value where we have a good number of rules, high confidence, and good lift. Using that minimum, we printed out the top 10 rules (according to lift) and obtained the rules in Figure 3.2, we also limited each rule to item sets with size 2 for better readability.

From these rules we can choose the ones we find relevant to describe the data, for example, the first rule has good lift and confidence and shows that a patient that spends 26 to 32 days in the study and shows serum creatinine between 0.491 and 1.136, likely exhibits ejection fraction between 47% and 55.25% and will die of heart failure.



TOP 10 per Min support - lift

```
('time_(26.48, 32.1]', 'serum_creatinine_(0.491, 1.136]') ==> ('ejection_fraction_(47.0, 55.25]', 'DEATH_EVENT')(s: 0.010, c: 0.75, lift: 32.04)
('serum_creatinine_(0.491, 1.136]', 'time_(26.48, 32.1]') ==> ('ejection_fraction_(47.0, 55.25]', 'diabetes')(s: 0.010, c: 0.75, lift: 24.92)
('time_(206.32, 211.94]', 'smoking') ==> ('age_(51.0, 56.5]', 'serum_sodium_(136.333, 138.278]')(s: 0.010, c: 0.75, lift: 24.92)
('time_(183.84, 189.46]', 'serum_sodium_(134.389, 136.333]') ==> ('platelets_(355060.0, 437550.0]', 'diabetes')(s: 0.010, c: 0.60, lift: 19.93)
('platelets_(355060.0, 437550.0]', 'serum_sodium_(134.389, 136.333]') ==> ('time_(183.84, 189.46]', 'diabetes')(s: 0.010, c: 0.50, lift: 18.69)
('serum_sodium_(128.556, 130.5]', 'serum_creatinine_(0.491, 1.136]') ==> ('platelets_(190080.0, 272570.0]', 'age_(45.5, 51.0]')(s: 0.010, c: 1.00, lift: 18.69)
('time_(206.32, 211.94]', 'smoking') ==> ('age_(51.0, 56.5]', 'anaemia')(s: 0.010, c: 0.75, lift: 17.25)
('platelets_(190080.0, 272570.0]', 'time_(9.62, 15.24]') ==> ('serum_sodium_(138.278, 140.222]', 'DEATH_EVENT')(s: 0.010, c: 0.75, lift: 17.25)
('time_(206.32, 211.94]', 'anaemia') ==> ('age_(51.0, 56.5]', 'serum_sodium_(136.333, 138.278]')(s: 0.010, c: 0.50, lift: 16.61)
('platelets_(355060.0, 437550.0]', 'serum_sodium_(134.389, 136.333]') ==> ('time_(183.84, 189.46]', 'creatinine_phosphokinase_(15.162, 414.9]')(s: 0.010, c: 0.50, lift: 16.61)
```

Figure 3.2: Top 10 Rules according to lift with minimum support 0.01

3.1.2 Oral Toxicity Dataset

This dataset contains a huge amount of records and variables, so we needed to reduce its dimensionality to be able to run the pattern mining algorithm. To do this we removed the features that had under 10% variance.

We can again verify, in Figure 3.3, that the number of rules increases with lower support, with a drop around 0.07 support, that there are a lot of rules with high confidence and that the rules with top lift have high confidence.

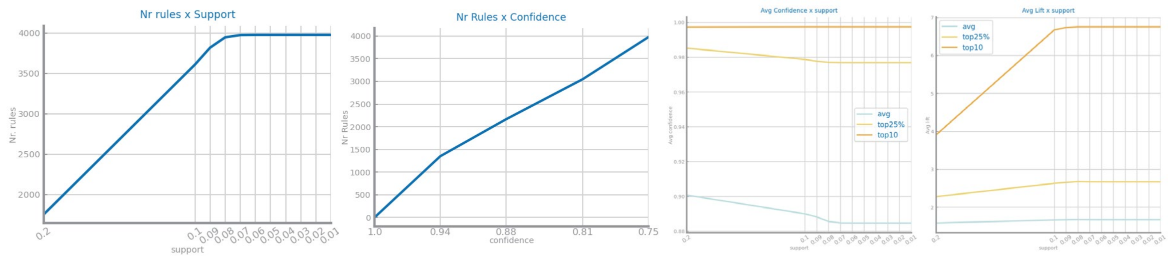
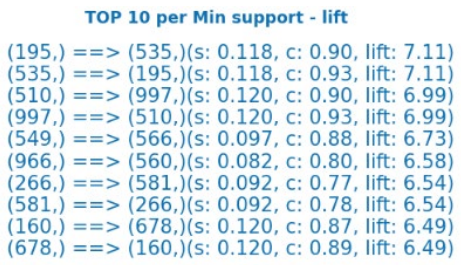


Figure 3.3: Metrics from pattern mining a) Number of Rules per Support b) Number of Rules per Confidence c) Average Confidence per Support d) Average Lift per Support

Based on Figure 3.3, we generated the top 10 rules in Figure 3.4 according to lift with 0.07 minimum support. Interestingly, we ran the algorithm with itemset up to size 3, but the top rules did not have more than size 1.



```
TOP 10 per Min support - lift

(195,) ==> (535,)(s: 0.118, c: 0.90, lift: 7.11)
(535,) ==> (195,)(s: 0.118, c: 0.93, lift: 7.11)
(510,) ==> (997,)(s: 0.120, c: 0.90, lift: 6.99)
(997,) ==> (510,)(s: 0.120, c: 0.93, lift: 6.99)
(549,) ==> (566,)(s: 0.097, c: 0.88, lift: 6.73)
(966,) ==> (560,)(s: 0.082, c: 0.80, lift: 6.58)
(266,) ==> (581,)(s: 0.092, c: 0.77, lift: 6.54)
(581,) ==> (266,)(s: 0.092, c: 0.78, lift: 6.54)
(160,) ==> (678,)(s: 0.120, c: 0.87, lift: 6.49)
(678,) ==> (160,)(s: 0.120, c: 0.89, lift: 6.49)
```

Figure 3.4: Top 10 Rules according to lift with minimum support 0.07

All the top 10 rules have very good support, confidence, and lift. These results could be interpreted as the molecular fingerprint on the left implies the one on the right, maybe because they describe the same molecule.

3.2 Clustering

The objective of Clustering is to group records in clusters according to their similarities, each cluster consists of rules that separate similar records from others. We will evaluate each method used based on the cohesion of the formed clusters, using the Mean Squared Error (MSE) and their separation, using Silhouette (SC).

To generate the clusters, we used a plethora of different methods. Those were Partition-Based (KMeans), Model Based (Expectation-Maximization), Density-Based (DBSCAN) and Hierarchical (Agglomerative Clustering). We also ran these models with multiple different settings, for example different *eps* for DBSCAN and different distance measures (Euclidean, Chebyshev, etc.) for DBSCAN and Agglomerative Clustering.

3.2.1 Heart Failure Dataset

To display the generated clusters, we plotted the records according to *serum_creatinine* and *ejection_fraction* because these two features where shown to be highly correlated to the target variable (Section 1.3).

The methods that showed clusters with the best quality, that is, the lowest MSE and highest SC, were KMeans with 7 clusters and Agglomerative Clustering with 9 clusters, complete linkage and either Euclidean or Chebyshev distance. For the later we chose the best number of clusters with “The Elbow Method” in Figure 3.5.a and chose the best distance metric from Figure 3.5.b obtaining the clusters in Figure 3.5.c.

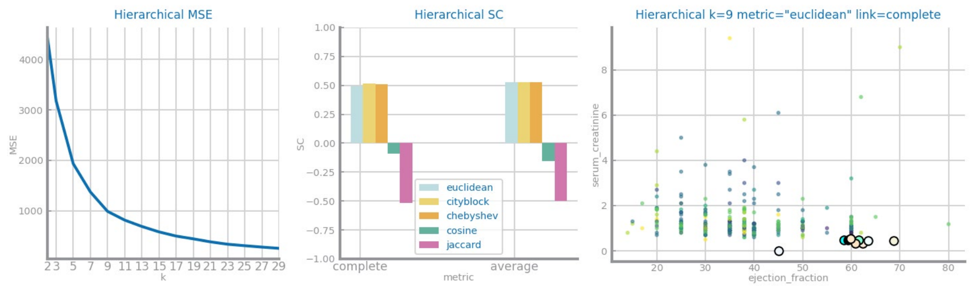


Figure 3.5: Hierarchical Clustering a) MSE per k b) SC per metric c) Clusters from k=9, metric=Euclidean, link=complete

Using the methods with good results we could generate clusters to group patients into clinically meaningful communities to provide the best care possible based on the specific profile of a patient.

We also tried to generate clusters after applying feature generation to the dataset, but the variance could be explained entirely by a single principle component. To generate the clusters, we needed to add a second component, with little impact on the variance, resulting in very inaccurate clusters.

3.2.2 Oral Toxicity Dataset

Since this dataset contains exclusively boolean variables, we cannot compute clusters directly, we must first apply feature extraction to obtain numeric variables. We used Principal Component Analysis to generate the principal components that explained at least 85% of the variance.

For most models, the cohesion increased with the number of clusters, as expected, but the Silhouette was consistently around zero, meaning most records are very close to the boundary between the cluster they are placed in and the neighboring ones, which makes sense given how densely packed the records are.

The only models that presented decent separation were Density-Based Clustering with the Chebyshev metric and Hierarchical Clustering with either Euclidean or Cityblock distance metrics, as shown in Figure 3.6.a and Figure 3.6.b, but by plotting the clusters we can see those methods group most records in one main cluster and the remaining ones contain only outliers, as seen in Figure 3.6 c and d. This result is the same even when running with different *eps* and number of clusters.

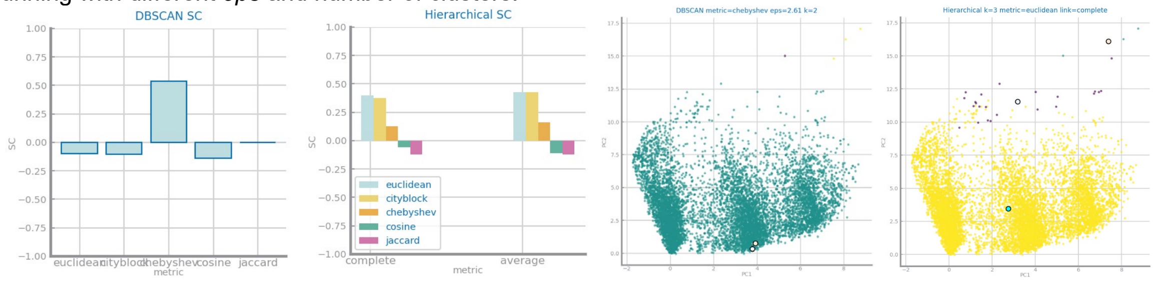


Figure 3.6: a) SC for Density-Based Clustering b) SC for Hierarchical Clustering c) Clusters from Density-Based Clustering with Chebyshev distance d) Clusters from Hierarchical Clustering with Euclidean distance

Although looking at graph with only the first two principal components we may spot 3 distinct clusters, it's clear that when taking into consideration all the features, and by looking at the generated clusters, all the records are very similar to each other and so clustering may not be very well suited to distinguish between them. The resulting clusters may instead be used to detect possible outliers from records added in the future.