CDados - Report

DIOGO SILVA, Instituto Superior Técnico
TIAGO MELO, Instituto Superior Técnico
PEDRO PIRES, Instituto Superior Técnico

There exists a plethora of Data Science techniques applicable to almost any type of presentable problem related to the scouring and interpolation of information from a given record of data. From simple treatment of data to prediction of results, clustering of data and even the mining of patterns there seems to be no end to the applicability and usefulness of Data Science techniques.

CCS Concepts: • **Computing methodologies → Classification and regression trees**; **Feature selection**; *Boosting*; Cross-validation; • **Information systems → Clustering**; Association rules.

Additional Key Words and Phrases: datasets, data profiling, data preparation, unsupervised learning, pattern mining, classification

ACM Reference Format:
Diogo Silva, Tiago Melo, and Pedro Pires. 2023. CDados - Report. 1, 1 (January 2023), 11 pages.

1 DATA PROFILING

In this section we will be looking at each of our two datasets separately and analyzing factors about their records, such as *dimensionality distribution*, *granularity* and *sparsity*.

Firstly the Heart Failure Prediction Dataset contains data pertaining to the prediction of the lethality of a cardiac arrest given several factors[1]. Within this dataset we have **299** different records and **13** variables, out of which the last - **DEATH_EVENT** - is our *Target Variable*. Disregarding the target, we are presented with a **24.92** records to variable ratio which is acceptable. Furthermore we observed that all of our variables are **numeric**, being either *float64* or *int64*. Additionally we also verified that none of these variables had any missing values. Looking at our variable's values we noticed that some of them only assumed two unique values - 0 and 1 - and as such should be considered *binary* rather than numeric. Figure 1 shows us each of our variable's distributions of values and, as can be seen, all of our numeric variables follow a Gaussian distribution, except for *time*, which presents a more complex multi-model distribution. We also analyzed each of our variables mean, standard deviation, minimum and max values, which led us to some conclusions in regards to the presence of outliers in variables like *creatinine* which has a max value that is much larger than the mean. In terms of feature variance, we noted that there were 4 in particular that varied very little in comparison to the others - *creatinine*, *platelets*, *serum_creatinine* and *serum_sodium*. This comparative analysis required us to first scale our data, a step that will be further explained in the next section. Figure 2 allows us to visualize the presence of outliers in our non-binary variables.

Authors' addresses: Diogo Silva, diogo.goncalves.silva@tecnico.ulisboa.pt, Instituto Superior Técnico; Tiago Melo, tiago.melo@tecnico.ulisboa.pt, Instituto Superior Técnico; Pedro Pires, ptpires@tecnico.ulisboa.pt, Instituto Superior Técnico.

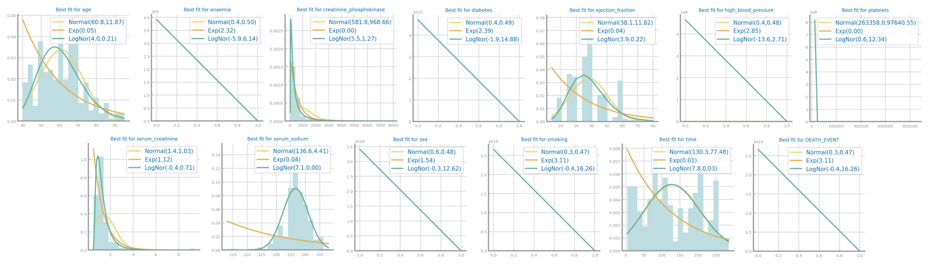


Fig. 1. Data distribution per variable.

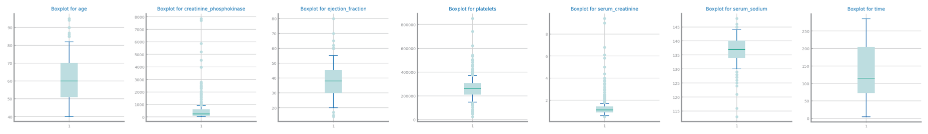


Fig. 2. Boxplots of the distribution of each numeric (non-binary) variable.

Finally, in terms of correlation between variables we noted that no two variables were ever too highly dependent, as seen in Figure 3. The highest relation we found was of **0.45** between **sex** and **smoking**, but even that was too low a value to be considerable. Taking into account we do not have any correlation coefficients equal to 0 we can assume that our dataset is sufficient to cover our domain. Interestingly we also noted that some variables correlate very little to our target variable.

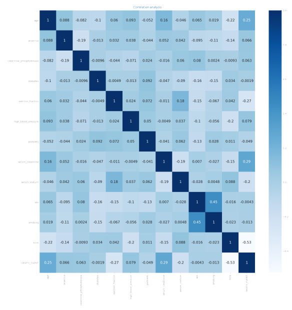


Fig. 3. Correlation Matrix showing us the sample correlation coefficient between any two variables.

The second dataset - QSAR Oral Toxicity - consists in an agglomerate of molecular fingerprints used to classify chemicals as either toxic or nontoxic. This dataset consists in **8992** different records, each with **1025** variables (out of which the last one is the *Target Variable*). All variables are numeric, but possess only 2 different unique values and as such, can be considered *binary*, in exception to the last one which is a *string* either *"positive"* or *"negative"*.Our ratio between number of records and number of variables is well over 1 (**8.78**) and as such, if our data were to not cover the domain, it would not be due to a lack of records. We also verified that there were no missing values present. This amount of data made the dataset hard to deal with, both computationally and in terms of analysis and as such, we devised a way to make our data easier to handle, without losing any information. Considering we had 1024 different variables, plus a target variable, what we did was concatenate each group of **32** variables, which we called **bits**, into a new variable, which we called **word**. This method generated **32 words** of **32 bits** each. The last 33rd variable became our target variable and it's value remained unchanged. With this we were able to more easily analyze factors such as **sparsity** simply by generating the correlation matrix between each of our 32 words and our target variable. We then decided that if any of the words had a correlation over 0.95 we would analyze the correlation between those two words' bits individually but this was never the case. Since no correlation coefficients were equal to 0 we can also assume we have sufficient coverage of our domain. With our variables being binary we obviously have no outliers, and in terms of variance we noted that certain variables varied very little.

2 DATA PREPARATION

Several steps of data processing were conducted on both of our datasets. Figure 4 showcases the basic pipeline we put our datasets through, but note that not all of those steps were applied to both datasets (for example, applying scaling to our *QSAR Dataset*, which is composed exclusively of binary values wouldn't make sense) and we have a plethora of versions of our datasets with several possible combinations of each of these steps.

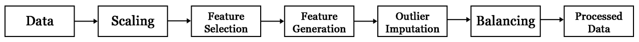
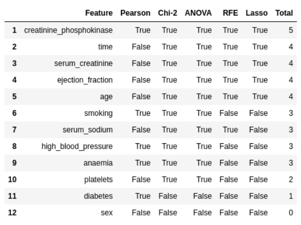


Fig. 4. The Data Processing/Preparation pipeline.

Firstly, **Scaling** was only applied to our *Heart Failure Dataset* for the reasons previously stated. We generated both **normalized** and **standardized** versions. In general *standardization* produces a more versatile dataset, but some of the techniques that we wanted to apply would not allow for negative values, and as such, it was convenient for us to store both versions.

Then we tried reducing the complexity of both our datasets through **Feature Selection**. We first started by analyzing the variance of each variable to see if any variable could be removed due to low variance and indeed found out that in the QSAR Dataset quite a lot of the 1024 variables had variance rates under **0.025** so immediately we created a new dataset without them. Note that with every feature selection technique applied to our base or scaled datasets, a new one was stored. We also analyzed the covariance between each of the dataset's variables (using the *Reduced32* dataset as a stand-in for the QSAR) not only to see if we had any redundancy (of which we found none since no two variables ever presented a high enough correlation factor) but also to observe how each variable correlated to the target. With this we noted that some variables in the Heart Failure dataset correlated very little (under 0.0025 absolute) with the target and as such, we stored a new dataset without them. Another form of feature selection utilized was *Univariate Selection.* We tested a wide range of filters such as *ANOVA* (appropriate for the HF Dataset since it has both categorical and numeric variables and a categorical target), *CHI-Squared*, *Mutual Information* (both good for our fully binary-categorical QSAR Dataset) and Pearson, as well as a Wrapper method in the form of *Recursive Feature Elimination* (using a *Linear SVR*) and an embedded method (methods that use algorithms with built-in feature selection methods) called *Lasso* (which is allegedly really good for when you have both categorical and numeric features [3]). Additionally we also performed Feature Selection via **mixing**, i.e, we took the results of all of these univariate methods and kept only the features that got selected by 4 or more of them, as seen in Figure 5. Finally we performed feature selection through **Feature Importance**, an inbuilt class that comes with Tree Based Classifiers that gives each of our features a score that is higher the more relevant the feature is to our target [4].

In terms of **Feature Generation**, we had technically already done it for the *QSAR Dataset* when we created our **Reduced32 Dataset**. As for our *Heart Failure Dataset* we noticed, when doing the previous feature selections, that



	Feature	Pearson	Chi-2	ANOVA	RFE	Lasso	Total
1	creatinine_phosphokinase	True	True	True	True	True	5
2	time	False	True	True	True	True	4
3	serum_creatinine	False	True	True	True	True	4
4	ejection_fraction	False	True	True	True	True	4
5	age	False	True	True	True	True	4
6	smoking	True	True	True	False	False	3
7	serum_sodium	False	True	True	True	False	3
8	high_blood_pressure	True	True	True	False	False	3
9	anaemia	True	True	True	False	False	3
10	platelets	False	True	True	False	False	2
11	diabetes	True	False	False	False	False	1
12	sex	False	False	False	False	False	0

Fig. 5. Which variables each of the different Univariate Selection techniques decided to keep.

most of our methodologies would discard the variables *diabetes*, *high_blood_pressure*, *anaemia* and *smoking*. These variables on their own seemed to have little impact on the mortality of a heart attack, however they still do correspond to an elevated risk of mortality and simply dismissing them might lead us to lose important information. As such we decided to create a new variable - **elevated_risk**. This categorical variable is set to True when a minimum of 3 out of 4 of those factors are true. We hope that with this new variable we can still weigh in the influence of these variables, not individually, but with their accumulated risk.

Since none of our datasets had missing values we did not perform missing value imputation. We did still, however, have to **treat the outliers** present in the *HF Dataset*. We accomplished this with two different techniques. The first was **Trimming** which simply removes records from the dataset that are identified as being outliers, which were identified by computing either the z-score - the signed number of standard deviations by which the record's variable's value is above its mean - or IQR - a measure of statistical dispersion, being equal to the difference between the upper and lower quartiles of value distribution. The second, **Winsorization**, does not actually remove records, and instead changes the values of outliers with the largest or smallest value observed that are not considered outliers.

Finally, we applied **Balancing** to our datasets. We tested balancing with **undersampling**, **oversampling** and **SMOTE** applied to both our datasets plus the *Reduced32*. We decided to store only the values of **SMOTE** since it allows us to synthesize new values without merely duplicating (oversampling) therefore balancing the data without having to remove records (undersampling). Figure 6 shows us the result of this transformation on our two datasets.

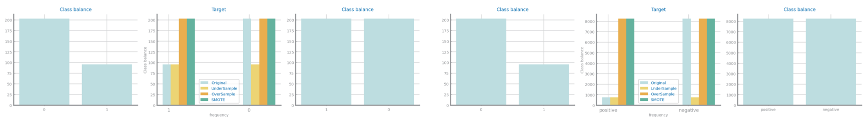


Fig. 6. Graphs showing the balancing of our HF dataset (left) and QSAR dataset (right), including a graph showing the balanced number of records of each of the target variables values using several balancing techniques.

Lastly, we performed **Feature Extraction** through Principal Component Analysis. This was done using three variants of PCA: **Linear Dimensionality Reduction using Singular Value Decomposition** (SVD), **Kernel PCA**, which consists of non-linear dimensionality reduction through the use of Kernels and **Incremental PCA**, which conceptually is very similar to PCA using SVD, but is more memory efficient and allows sparse input. By following three different approaches we have a baseline model we can use as reference later and can have a better notion of what type of Feature Extraction works best for our dataset. In order to select the number of components to keep, we opted to select the minimum number of vectors that resulted in a cumulative sum of Explained Variance of at least 95% [5], as can be seen in Figure 7.

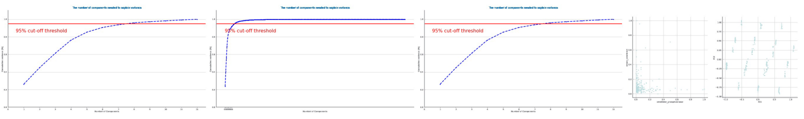


Fig. 7. Explained variance ratio cumulative sum in function of principal components. Scatter plot that shows the correlation between two variables before PCA and correlation between two Principal Components after PCA.

3 CLASSIFICATION

3.1 Naïve Bayes

For the **Naïve Bayes** model, we compared the performances of three event models: *Gaussian*, *Multinomial* and *Bernoulli*. We performed holdout (70% training) and K-fold (10 shuffled splits) for each of the different models, which yielded very similar results. This process was followed for the original, unprocessed dataset and for the processed ones. Given the high volume of resulting datasets we opted to consider only the best performing event model for comparison of each dataset, in order to understand what type of pre-processing resulted in the most performance gain. For most datasets, the Gaussian Naïve Bayes model results in the best accuracy, with the exception of the Standardized data. This is not a surprising result, considering our data follows a Gaussian distribution. Looking at the precision, recall, and confusion matrices, it can easily be inferred that the ability to recognize both true positives and negatives greatly benefits from balancing; especially considering the difficulties the model presented in accurately predicting positive labels, as we can see from the low Recall scores prior to this transformation. Lastly, from the histogram regarding feature engineering, we can conclude that the pre-processing that yields the best results for this model consists of scaling, balancing, outlier trimming with IQS and mixed feature selection. Considering the simplicity of this model, an average of 80% accuracy was pretty acceptable. As for the *QSAR Dataset*, we followed a similar approach. The studied datasets for this model were the original, balanced and both balanced and reduced. Unlike the HF dataset, for this data we have varying performances for different models, i. e., there is not a best model for all the datasets. From the histogram we infer that regardless of the model, there is an average of **80%** accuracy for the original and balanced datasets, however, the charts show all three models struggled to accurately label negative records, as we can see from the low precision scores. Looking at the confusion matrices as well as precision and recall scores, we can conclude that balancing the dataset prior to training the model has a dramatic impact on its performance, as was to be expected. When using feature engineering, we can spot a significant increase in accuracy for the *Reduced32 dataset*, with over 90%. Besides this dataset, our best performance was with feature selection using Extra Tree Classifier.

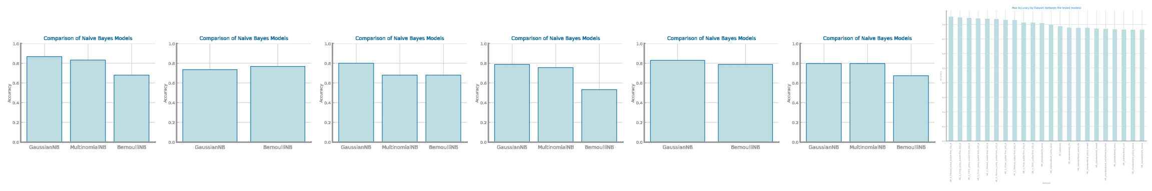


Fig. 8. Side-by-side comparison of the accuracy obtained using the most relevant datasets regarding feature engineering and further data processing for the Heart Failure dataset.

3.2 KNN

Our approach evaluated the performances of the **K-Nearest Neighbors** using the *Manhattan*, *Euclidean*, *Chebyshev* and *Jaccard* distances. The followed procedure was similar for both datasets. We measured the accuracy for different combinations of hyper-parameters. Afterwards, we selected the best-performing model and computed the aforementioned metrics. Provided that KNN uses distances, we can expect that a scaled dataset performs significantly better than a non-processed one. Hence, for a first approach we validated the model for the original, standardized and normalized datasets. Unlike in the previous model, we did not use our SMOTE balanced datasets nor our Windsorized datasets since we are computing distances between values and we should maintain the proportional relationships between them. Generally, every model's performance seems to increase as the number of neighbors increases, regardless of the distance metric used, but it seems to rapidly reach a *plateau*. As expected, processing the data *a priori* to training the model has a great impact on its performance: both scaled datasets present a 10% accuracy increase in comparison to the original. Additionally, we also noted that this pre-processing also benefitted the model's ability to accurately predict positive records, as we can see from the improved Precision scores in the second and third bar charts 9. Nevertheless, the model still struggles in labelling such cases for all the datasets, as can be seen through the confusion matrices, and in none of the cases does it display a great accuracy. Similarly to previous results, engineered datasets displayed far better overall performance, with the highest accuracy reaching over **90%**. Such was achieved using the dataset with standardization, trimming outliers using IQS and feature selection using importance. For the original *Oral Toxicity Dataset*, the obtained results varied slightly: instead of a rapid increase towards a *plateau* in function of the number of neighbors, the chart suggests the performance does not depend too much on it, as the accuracy is stable for every distance metric. On the other hand, the feature engineered datasets show a steady decrease in performance as the number of neighbors increases, for some cases. In others, the behavior seen in the original dataset is replicated.

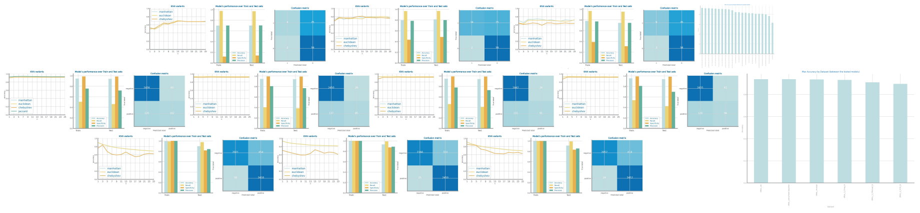


Fig. 9. Accuracy evolution using different metrics in function of the number of neighbors, along with performance details on the best-performing model for each dataset: original, standardized and normalized, respectively (Top). Accuracy evolution using different metrics in function of the number of neighbors, for the original and feature engineered datasets (Bottom).

3.3 Decision Trees

For Decision Trees we basically did the same as for the algorithms above. For the *HF dataset*, first, we ran the algorithm with the original dataset in order to obtain baseline results so that we could compare and assess how our data preparation techniques would impact them. And even though the models did not seem to overfit, we still tested how pruning would affect them. For the hyperparameter search we experimented with both **gini** and **entropy** and then performed combinations of **max_depth** and **min_impurity_decrease** parameters. For the *HF original dataset*, we obtained an accuracy of **87%** with both recall and precision also above **80%** but the specificity only around **65%**. When pruned, the tree achieves an accuracy of **89%** and all other metrics were also above **80%** even though recall dropped by a bit. This comes from the fact that with pruning the tree does not give as many false negatives but increases the number of false positives. Then, we tested on some of our processed datasets. The results can be seen on Fig 10. From looking at the top 5, all those datasets had balancing, scaling and some type of outlier treatment applied to them.

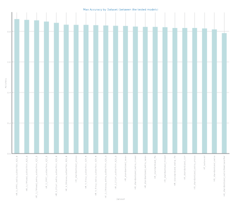


Fig. 10. Side-by-side comparison of the accuracy obtained using the most relevant datasets regarding feature engineering and further data processing for the Heart Failure dataset and Decision trees models.

For the second dataset we followed the same process. With the original, results were not the best, with an accuracy around **93%** and an ever higher specificity it would seem good, but this is due to the fact that classes are really unbalanced, since precision is only around **70%** and recall even lower at **53%**. When testing with pruning, the model simply classified everything as negative. With the *Reduced32 Dataset*, the model trained would classify almost every point as negative, leaving us with a recall smaller than **20%**, so not a good option. For the processed datasets, balancing is the way to go. When we applied any type of feature selection, the results were similar to the baseline with high accuracy but low recall, even lower than with the original. But when we applied balancing to the datasets, the results were much better, with every one of our 4 evaluation metrics being around **90%**.

3.4 Random Forests

We used both **Random Forests** and **Extremely Randomized Trees**, a variant of the former that proposes lower variance [6]. We experimented every combination of three specific hyper-parameters: the **Number of Estimators** *id est* the number of trees in the generated forest, the **Maximum Depth** of each tree, and the **Maximum Number of Features** to be kept, passed along as percentages. In Figure 11 we can visualize the accuracy evolution in function of the number of estimators for different numbers of features and different maximum depths. Once again, this was done using both a series of *Holdouts* (with 70% training size) and *K-fold* using 10 splits. From the results, we infer there is no clear best combination of parameters for the former model. As for Extremely Randomized Trees, it is interesting to note how the different performances stabilized around different values according to the number of features kept, suggesting this model is more sensitive to such changes. However, much like **Random Forests**, the number of estimators does not seem to have a great impact on overall model performance. Much like in previous approaches, our models were fine-tuned, the best performing set of hyper-parameters were picked and then both variants were compared. For the *HF Dataset*, we verified the best results using the *standardized*, *RFE feature selection*, *IQS outlier trimming*, *balanced* version, resulting in a test accuracy of **92%**, using a maximum tree depth of 10, 10% of the original features and 75 estimators in the **Extremely Randomized Trees** variant. When tested against a wider range of pre-processed datasets, we notice that balancing and scaling the data beforehand are possibly the most important steps for such models. We can infer this since most of the datasets that have been through those techniques display a higher accuracy overall. As for QSAR, we see even less variance in function of the number of estimators, as both models seem to have a stable accuracy regardless of the combination of hyper-parameters.

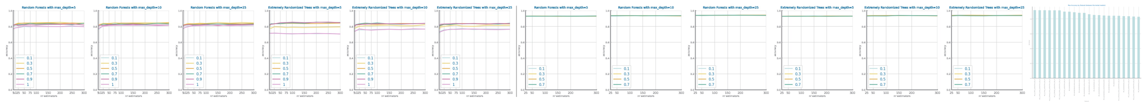


Fig. 11. Accuracy evolution in function of number of estimators for different maximum tree depths. First three correspond to Random Forests and the latter correspond to Extremely Randomized Trees. Implemented for the Heart Failure original dataset (left), for the QSAR Oral Toxicity original dataset (middle), and maximum accuracy obtained between both fine-tuned models (right)

3.5 Gradient Boosting

Finally, we also tried **Gradient Boosting**. For implementation we utilized a multi-stage additive model in which each stage regression trees are fit on the negative gradient of the given deviance loss function. As main parameters we have the **Learning Rate** - which shrinks the contribution of each tree in the classifier's ensemble - the **Number of Estimators** - The number of boosting stages to perform which should help with reducing overfitting - and the **Max Depth** - Limits the number of nodes in each individual regression estimator tree. Originally, our approach was to iteratively tweak each of these values. We started off by checking the accuracy of our estimator for a range of learning rates, then picked the learning rate that obtained the best accuracy on our Test set and repeated the procedure for Number of Estimators and the same for Max Depth. A possibly better approach however, rather than iteratively picking the parameters would be to generate every possible combination between them, therefore assuring that we are getting the best combination of Learning Rate, Number of Estimators and Max Depth rather than the best Learning Rate, best Number of Estimators in function of the previously obtained best Learning Rate, and so on. One downside to this, however, would be that we would have to reduce the number of values we were testing for each of our parameters, since we would be generating many more combinations. We ran this algorithm for both our main datasets, with combinations of pre-processing steps, having computed the best parameters for each of them using both *Holdout* (with 70% train size) and *K-Fold* (with 10 shuffled splits). For our *Heart Failure Dataset* the best results obtained were for the *standardized* version with *importance feature selection*, *feature generation*, *outlier imputation* and *smote balancing* as we managed to obtain over **90%** test accuracy (for both Holdout and K-Fold) with a max depth of 2, 100 estimators and a surprising learning rate of 1. As for the *QSAR Dataset* our best results were with *Mutual-Filter Feature Selection* and *SMOTE balancing* which garnered us an astonishing accuracy of over **96%** for both *holdout* and *K-fold* for a learning rate of 1, 200 estimators and max depth of 6. We also ran the algorithm for our *Reduced32* dataset with *Chi-Filter Feature Selection* which also achieved over **95%** accuracy with a max depth of 6, 300 estimators and a learning rate of 1. It should be stated that not only was the accuracy values obtained astonishingly high, both the precision and recall associated with those results were also high, as can be seen in Figure 12. Gradient Boosting managed to finely predict both positives and negatives equally well. Oddly enough all of our best estimators had a learning rate close to or equal to 1 meaning the contribution of each tree was minimal.

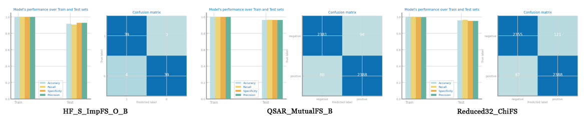


Fig. 12. Evaluation Metrics obtained from running GBC on a prepared Heart Failure Dataset (Left), prepared QSAR Dataset (Middle) and prepared Reduced32 Dataset (Right)

4 UNSUPERVISED LEARNING

4.1 Clustering

We applied 4 different clustering methodologies - **K-Means**, **Hierarchical**, **Density-Based** and **Expectation Maximization**.

Starting off with **K-Means**, we ran our algorithm for an increasing value of k (*id est*, number of centroids), drawing for each K the scatter plot of our two first variables (originally we were drawing the scatter plots between each two pairs of variables but this was unfeasible due to computation limitations). For each K we also calculated the *Mean Square Error*, and two validation metrics - *Silhouette Coefficient* and *Davies-Bouldin Scores*. We ran K-Means for a plethora of our datasets but taking into account that K-Means uses an euclidean distance to assign values to a cluster we had some insight that the values obtained would not be too great for either datasets since the *HF Dataset* contains both numerical and binary data and the *QSAR Dataset* is purely binary. We also had to be careful to not use any data preparation that led to the alteration of our record's proportions, meaning we could not use our balanced datasets (since they used *SMOTE*) nor our datasets with outlier imputation through *winsorization*. This concern extended to all of our Clustering methods. Scaling was still fine, however, since all values were scaled proportionally to each other so the relation between values was not altered. As expected the results were not too great. For our *HF Dataset* we never managed to break an *SC* score of 0.4 and while applying feature extraction techniques to our data, the best we could ever achieve was for our standardized dataset with RFE feature selection, outlier trimming and PCA feature extraction as seen in Figure 13, but even still whilst our *MSE* was noticeably decreasing with the addition of centroids, our *SC* and *DB* scores seemed to contradict each other since one peaked at a K of 25 (SC), the other presented better results for lower K's (DB). The optimal K in this situation would probably be around 17 since it seems to strike the best balance between our metrics. This was even worse for our *QSAR Dataset* which presented even poorer results as seen in the same figure. Although the DB score was pretty high, the SC never surpassed 0.15. All in all, K-Means is not too applicable to our datasets.

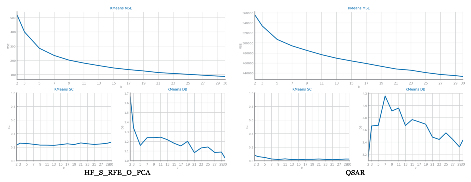


Fig. 13. MSE, SC and DB for Heart Failure (Left) and QSAR (Right)

When it comes to **Density based** we ran the algorithm, for *HF* on the standardized one and then with the standardized with feature selection and feature generation and outliers treatment, and then the same after PCA. For several distance measures we did a small study to see what **EPS** we should use. The distances were **euclidean**, **cityblock**, **chebyshev**, **cosine**, **jaccard**, **hamming** and **gower**. Gower was a test as it seems to be a really good distance to use when we do not only have numerical variables [7]. For the standardized all of the metrics had an MSE around **0.16** and a DB score between **1.4** for cosine and **2.5** for gower, with jaccard being the worst at **3.5**. For the SC Coefficient only euclidean chebyshev and gower had positive values, but all around **0.3**, so not very good results. For the processed dataset, none of the metrics had a positive SC Coefficient so results were not good. After PCA results were very similar. For the QSAR dataset we did the same testing, expecting that distances for binary variables would perform better. And indeed, while all metrics had a similar MSE around **0.9**. When looking at the SC Coefficient, only the Hamming distance has a positive value around **0.3** which is still not very good.

For **Hierarchical** we applied it to the same datasets mentioned above. Then, first we do a small study to choose the best K then test it with the aforementioned distance measures minus gower, and with complete or average linkage. For the standardized one, with a K value of 3, the MSE was around **0.15** for complete and slightly higher for average. Euclidean and chebyshev distances had a SB coefficient higher than **0.4** and cityblock at **0.5** for average linkage. When testing the other two datasets, results were worse. While MSE did not really change, SC was smaller never reaching **0.3** and DB also increased. For the *QSAR dataset* the chosen K was also 3. While all distances have a MSE just above **0.08** only euclidean, cityblock and hamming have positive values for SC Coefficient. These three also have the lowest DB scores at **0.5** with average linkage. In both datasets, average linkage yielded better results.

Finally for **Expectation Maximization** we performed a search for the optimal K value. For the Standardized dataset as K increases, both MSE and DB decrease which seems promising but the SC coefficient never even reaches **0.2**. When applied to processed datasets, results were basically the same. For QSAR we did the same search for K values. In this case MSE barely changed when K increased becoming only slightly smaller, also with SC lower than **0.1** and DB varying between 3 and 4, results were rather bad.

4.2 Pattern Mining

Finally we proceeded to look for patterns in our datasets. Firstly, however, we had to deal with the fact that our *HF Dataset* had numerical data, hence the need for **discretization**. We applied 3 types of discretization - **Dummification**, **Equal-Width Binning** (with a number of bins equal to the number of unique values of the variable with the least amount of unique values), and an extreme case of **Equal-Frequency Binning** (in which each variable had a different number of bins equal to the number of unique values). After this was done, we tried to find patterns in our data and then performed quality evaluation using several metrics (such as *confidence*, *support* and *lift*). We found a comparable amount of patterns for all of our discretized versions of our *HF Dataset*, with all of them rounding around 19750 patterns and 2976800 rules using a minimum confidence of 50%. For the *QSAR Dataset* we found only 56 patterns, with the only metrics capable of finding rules being lift and conviction (of which we used a minimum of 0.5) with 96 rules found. We also ran pattern mining on our *Reduced32 Dataset* which, unfortunately, was not able to find any patterns. These values are showcased in Figure 14. All in all we verified that the number of patterns seemed to increase exponentially in function of the given support threshold on all of our datasets. Additionally, for our *HF Dataset*, considering confidence as the quality measure, average quality seems to be very similar to the best rules', but as the Confidence threshold increases, the average confidence drops immensely and most rules appear to be symmetric, in the sense that if A implies B, then B most likely implies A for every A, B class in the dataset.

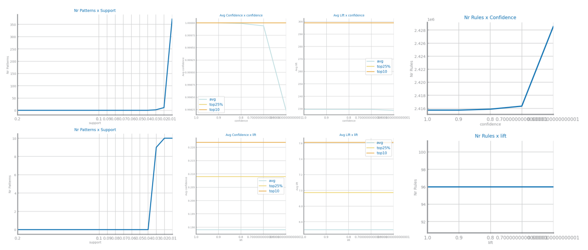


Fig. 14. Number of Patterns found per support and association rules quality metrics for HF (Top) and QSAR (Bottom)

5 CONCLUSION

Throughout this project we have studied a wide range of modern techniques in the scope of Data Science and Machine Learning. Moreover, the conducted procedure also allows us to infer what data preparation techniques yield the best results for what models. This analysis cannot be without also considering the simplicity of model, both conceptual and implementation-wise.

The performance of the **Naïve Bayes** model is extremely reliant on the assumed event model distribution, *id est*, Gaussian, Multinomial or Bernoulli, each one having varying performances for distinct datasets. Additionally, this model suggests it offers best results when the data is balanced and scaled prior to its training. Regarding **KNN**, as expected, normalizing the data displays a better overall accuracy regardless of the distance metric used. All the best performing datasets have also been scaled, leading us to believe scaling is also fundamental when employing **KNN**. **Decision Trees** as well as **Random Forests** on the other hand, are far more sensitive to prior data balancing, which can be inferred not only from the improved accuracy, but also far better recall and specificity scores. This sort of similarity between the two models makes sense, given the latter is an ensemble of the former. Lastly, **Gradient Boosting** seems to yield the best results, making stable predictions for both positive and negative records. As for *Unsupervised Learning*, although our data does not lend itself too well to **Clustering**, we still managed to achieve some interesting results. Moreover, in **Pattern Mining** we infer that the support threshold seems to be correlated with the number of patterns found in an exponential manner and that symmetry could be found in all of the rules that displayed the highest quality.

6 REFERENCES

REFERENCES

[1] Davide Chicco and Giuseppe Jurmann. *Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone*. BMC Medical Informatics and Decision Making, 2020.

[2] Davide Ballabio, Francesca Grisoni, Viviana Consonni, Roberto Todeschini. *Integrated QSAR models to predict acute oral systemic toxicity, Molecular Informatics*. Milano Chemometrics and QSAR Research Group, 2019.

[3] Rahul Agarwal. *The 5 Feature Selection Algorithms every Data Scientist should know*. towardsdatascience, 2019.

[4] Raheel Shaikh. *Feature Selection Techniques in Machine Learning with Python*. towardsdatascience, 2018.

[5] Bartosz Mikulski. *PCA - How to choose the number of components?*. mikulskibartosz.name, 2019.

[6] Ashish Anand. *kNN and Unbalanced Classes*. Cross Validated, 2019.

[7] J. C. Gower. *A General Coefficient of Similarity and Some of Its Properties* . Biometrics 27, 857–874, 2019.

A IMAGES

All image resources, graphs and code created can be found in the repository at https://github.com/HerouFenix/cdados-project