Data Science Project

Instituto Superior Técnico - Data Science - 1st Semester - Group 75

António Fernandes (84370)*
antonio.m.fernandes@tecnico.ul.pt

João Lousada (84274)*
joao.b.lousada@tecnico.ulisboa.pt

Mariana Mota (84410)*
mariana.mota@tecnico.ulisboa.pt

ABSTRACT

Data science techniques have become essential when it comes to analyse and retract relevant information from large amounts of complex data, or big data. In this project some of these techniques are explored in order to retrieve information from two specific data sets: Parkinson's Disease Classification Data Set and Covertype Data Set. For each one, it is presented the results obtained for some methods of description, preprocessing, classification and unsupervised learning of the data, as well as some relevant comments about them.

KEYWORDS

Data Science; Data Preparation; Classification; Unsupervised Learning; Python

1 INTRODUCTION

For this project, we intend to build a framework, through data science methods and algorithms, that would allow us to analyse more efficiently a given data set, by retrieving as much knowledge as possible from it. In specific, we study two data sets with distinct number of variables and number of instances, which will bring different challenges to overpass for each one.

In order to do such analysis, one shall first explore the data, in order to find its shape and predict what kind of study one should do. For further analysis, it is then important to prepare the data, i.e., to do some data preprocessing such as normalization, data balancing and variable dummification. This will allow us to build a more efficient framework that ensures an easier analysis when it comes to applying some data classification and unsupervised learning methods.

Finally, we will have enough conditions to proceed to a more deep and meaningful study of the data, and thus retrieve more knowledge about the data. Classification and unsupervised learning will, then, lead us to take major conclusions about both data sets.

2 DATA EXPLORATION

This variables vs instances measure, mentioned earlier, is called **data shape**. For any x instances and y variables, we will have a certain point (x, y) in a 2D plane, which we will define as **Data Shape Plane**. In this plane, the position of a certain data set tells us which kind of data science methods will be more suitably applied to it. For instance, a low instances number data set probably won't require any undersampling treatment, unlike a high instances number one. For practical reasons, we define n_{inst} and n_{vars} as the numbers of variables and instances, respectively. In the Figure bellow, it is presented a logarithmic representation of the Data Shape Plane.

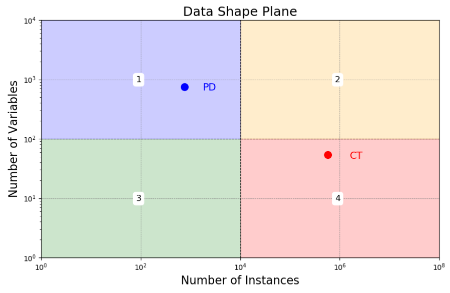


Figure 1: Data Shape lane representation. Both working data sets are presented: PD (Parkinson's Disease) data set with shape (756, 754), and CT (Covertype) data set with shape (581012, 54).

The plane can be divided in 4 major regions, whose limits go as a function of n_{vars} and n_{inst}[1]. As previously mentioned, for each region some data science methods will be applied more suitably than others, as we will show along this report. In specific, we will be studying the 1 (with PD) and 4 (with CT) regions.

2.1 Statistical Description

Both study data sets have a classification associated task. For the first one, the classification problem comes from the 'class' variable, with values 0 or 1, while for the second data set the classification problem comes from the 'Cover_type' variable, which takes values from 1 to 7.

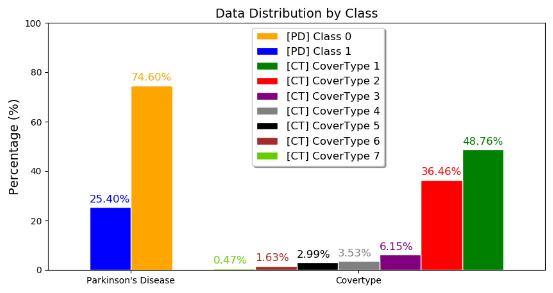


Figure 2: Data classification task, for both PD and CT data sets.

*All authors contributed equally to this project.

[1]Note that the regions delimitation is very subjective, so they should be taken as dynamic regions and not strict limits.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 2019, Montreal, Canada

https://doi.org/doi

It is clear that, for both cases, the data is not balanced, and thus it requires preprocessing methods, such as oversampling. For this specific project, we will be using the Synthetic Minority Oversampling TEchnique (**SMOTE**), which over-samples the minority class by creating synthetic minority instances, based on the nearest neighbors judged by Euclidian distance between data points. We will discuss some results and the importance of SMOTE application along this report.

3 PREPROCESSING

Before applying any learning model, it is important to make sure that the raw data is transformed into an understandable format. Data preprocessing techniques make sure that a data set is balanced, reduced, complete, centered and consistent, potentiating the quality of the learning models used.

The techniques chosen are in accordance with the properties of the input data set and the behavior of the target learning algorithm. All of them were applied, firstly, only to the training set and afterwards to training and testing set, because testing points represent real-world data. If one normalizes the whole data set, future information is taken into account in the training process. This methodology ensures there is no bias introduced.ya One of the preprocessing techniques used in the datasets was the normalization technique. This involves re-scaling the original data without changing its behaviour, where the new boundaries are (0,1), and it is an helpful tool when features have different ranges. This technique is particularly useful in distance-based algorithms, such as KNN or K-Means. Here the methods used to centre and reshape the data were *StandarScaler()* and *MinMaxScaler()*.

3.1 Correlation Matrix

The analysis of the correlation between variables provides us a very important information when it comes to variable suppression: the more correlated two variables are, more likely it is to drop one of them without affecting major results. Hence, depending on the requested precision, we can define a **threshold** T such that variables with correlation number (absolute value) above $|T|$ may be suppressed. In Figure 8 it is shown the variables correlation values distribution for both data sets.

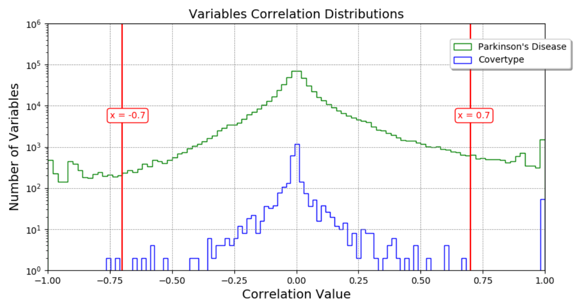


Figure 3: Variables Correlation Distributions for both date sets. It is also presented the line for $|T| = 0.7$: by applying variable drop methods, all the variables whose correlation is above $T^+ = 0.7$ or below $T^- = -0.7$ may be suppressed.

For the PD, we have many variables highly correlated, and thus they can be suppressed, dropping drastically the number of variables (as we can see on Figure 4). On the other hand, for the CT data set we'll need a much lower threshold in order to reduce the number of variables, as they are not well correlated.

Note that, the lower the threshold is, the higher the probability the data suppression affects the whole data analysis, so one should be careful when choosing a good value of T. Nevertheless, applying a certain threshold will always be helpful for analysis, specially when it comes to a region 1 or 2 data set (high variable number).

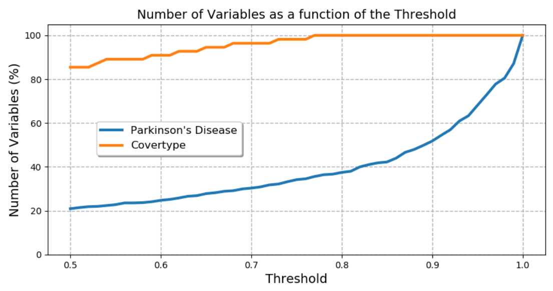


Figure 4: Number of variables as a function of the T, for both PD and CT data sets. The n° of variables is shown in comparison with the original one. PD is clearly more sensitive to the threshold.

In terms of computational work, having a much lower number of variables to compute leads to an drastic reduction of the computing time. Throughout this report we will see the impact of the threshold on the global data evaluation.

3.2 SMOTE and Random Under-Sampling

In order to compensate an imbalanced data set, and adjust its class distribution, oversampling and undersampling techniques are used. The oversampling technique that was applied to the datasets was SMOTE, as meantioned previously. The chosen undersampling technique was Random Under-Sampling, where samples are randomly removed from the majority class. Throughout the report, it is possible to see the repercussions on the accuracy of the models.

3.3 Outliers Analysis

Outliers represent observations that diverge from the overall pattern on a sample. These extreme values ought to be treated with sensitivity, they may correspond to 'mistakes' in the data or they may reflect a particular behaviour that is not there by chance, a **novelty**. One could easily identify them with a boxplot or clustering techniques, the hard part is to decide where to go next: the most straightforward strategy would be to, simply, discard these values, although it can lead to loss of information; other way is to substitute these values by the mean value of the distribution, which would maintain the same shape, but introduce bias right away. Two methods were used to detect outliers: using **Z-Score**, that should work best applied to Gaussian distributions; and using **IQR-Score**, based on the information given by quantiles, as can be seen by boxplots.

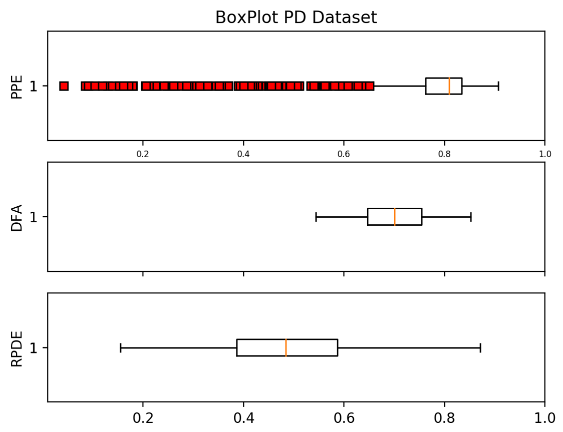


Figure 5: Boxplot - PD

Some features have a lot of outliers, which might mean that this is not a good statistical variable to study, while others are well distributed and centered.

3.4 Feature Selection

In order to reduce dimensionality and try to smooth the learning function, feature selection is applied to the data, while preserving most of the relevant information and its structure. This way, only a subset of features will be analyzed, while irrelevant or redundant features will be removed without loss of information. (Which should impact significantly clustering learning models)

Varying k in SelectKBest(), applying Naive-Bayes model and comparing accuracies:

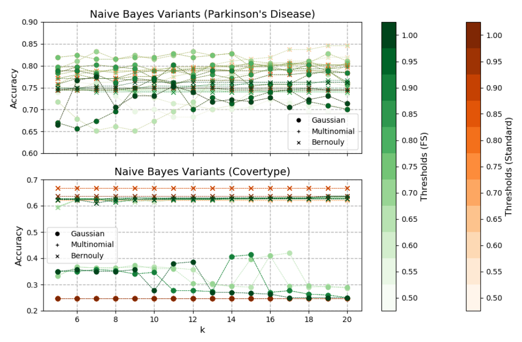


Figure 6: Naive-Bayes Accuracy vs k (Feature Selection), as a function of the threshold. It is also shown the same results for Stantard train test split.

3.5 PCA & LDA

Other algorithms to reduce dimensionality are PCA and LDA, which are linear transformation techniques. The first one can be interpreted as a technique that finds the directions of maximal variance. Varying the number of components in PCA and applying Naive-Bayes model, we were able to produce the following:

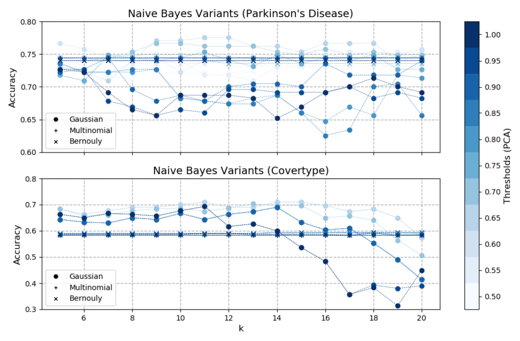


Figure 7: Naive-Bayes Accuracy vs k components (PCA), as a function of the threshold.

In contrast to PCA, LDA attempts to find a feature subspace that maximizes class separability. Applying the same as for PCA, we were able to obtain the following:

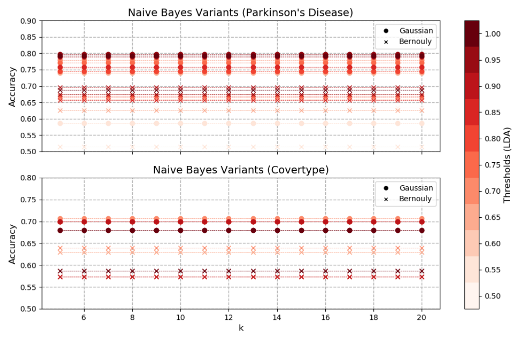


Figure 8: Naive-Bayes Accuracy vs k components (LDA), as a function of the threshold.

During Classification and Unsupervised Learning, all of the techniques developed above and many combinations of them were applied. To avoid cluttering, only the more relevant ones were chosen for display.

4 CLASSIFICATION

In this section, we will only focus on discussing the results of the application of the models with the best tuned parameters, consequence of the preprocessing analysis made above.

4.1 KNN

In pattern recognition, the k-nearest neighbors (**KNN**) algorithm is a non-parametric method used for classification. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

The distance function to use to choose the neighbors may be obtained by different metrics. In specific, we'll be using manhattan, euclidean and chebyshev metrics.

The results of the KNN are sensitive to the preprocessing method used. In Figure 10 it is shown a comparison of the KNN convergence, as a function of the thresholds, for two different preprocessing methods: standard train test split and SMOTE application. It is clear that, for the PD data set, SMOTE shows a worse overall accuracy than the standard preprocessing. This may happen because of the following: by applying SMOTE, the minority class will become balanced with the majority one (see Figure 2), and so the KNN computing process may assume both classes are actually the same, leading to misleading accuracy.

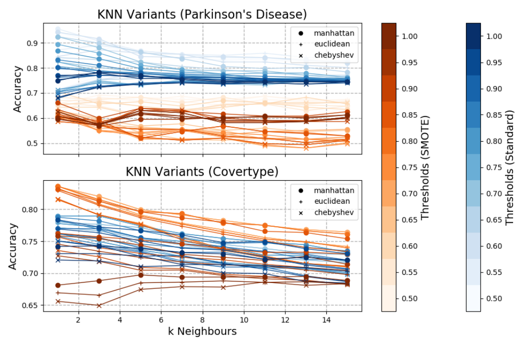


Figure 9: KNN variants convergence for both data sets. It is shown a comparison between the standard train test split algorithm and the application of SMOTE, as a function of the threshold.

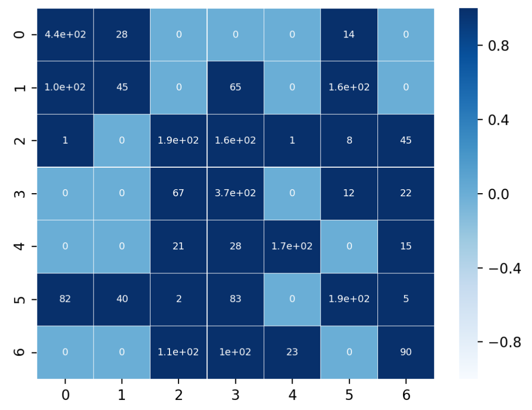
In the case of CT, SMOTE and standard technique overall accuracy are overlapping, so the SMOTE application impact is not much visible. As we have 5 minor and 2 major classes for the CT, the application of SMOTE may be ambiguous: it may help on data balancing and clusters evaluation, but it may also cause classes to overlap, and mislead the algorithm to interpret them as the same class, thus leading to inaccuracies. Regardless of this problematic, it is clear that for lower thresholds we'll have much better metric scores (with a visible maximum for $|T| \approx 0.7$).

In this analysis we find that SMOTE may not be very appropriate in the KNN application for the raw values. However, we see that lowering the threshold value causes the accuracy to increase, and so, by varying both parameters one can find the best possible scenario.

4.2 Naïve-Bayes

Naïve-Bayes algorithm assumes strong independence between the features, so regarding the Parkinson Disease dataset, because we have many correlated variables, a good strategy would to handle them: correlation matrix and threshold application, PCA/LDA or FeatureSelection, which will not only reduce drastically computation time but will improve this classifiers scores. (as was seen in preprocessing section). It is expected that the success of the algorithm does not depend on class balancing, so SMOTE and UnderSampling should produce insignificant variations. GaussianNB() assumes features are distributed as a gaussian distribution , so combining this with StandarScaler() normalization should produce better results, MultinomialNB() gives the frequencies with which data has been generated by a multinomial.

By looking at the confusion matrix we can see that, overall, the values in the main diagonal are higher than most of the rest. However, a few of these values outside the diagonal are high enough to say that the accuracy of this classifier is far from ideal.



	0	1	2	3	4	5	6
0	4.4e+02	28	0	0	0	14	0
1	1.0e+02	45	0	65	0	1.6e+02	0
2	1	0	1.9e+02	1.6e+02	1	8	45
3	0	0	67	3.7e+02	0	12	22
4	0	0	21	28	1.7e+02	0	15
5	82	40	2	83	0	1.9e+02	5
6	0	0	1.1e+02	1e+02	23	0	90

Figure 10: Confusion Matrix for CT dataset

In conclusion, this algorithm despite starting on the imprecise assumption of strong independence of features, it is very practical to implement and as long as the correct class is more probable than the others, the classifier will make correct predictions.

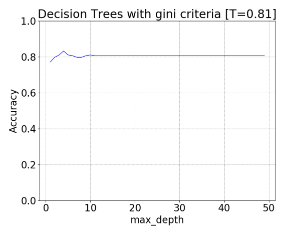
Results for the accuracy of the learning model can be found in Table 1, for both sets of data, corroborating the statements above.

4.3 Decision Trees and Random Forests

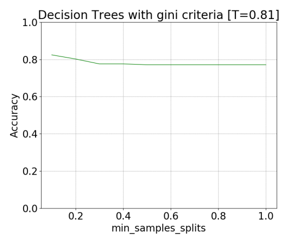
Decision trees are essentially a series of questions designed to assign classification, each question involving one feature and one split point. When studying the accuracy of decision trees, a few parameters can be taken into account, such as the *min_samples_split*, which is the minimum number of samples required to split an internal node, and *max_depth*, which indicates how deep the tree can be. This second one should be controlled since a high value for it results in a very deep classification tree with many nodes, which often leads to overfitting on the training dataset. Since both gini and entropy criterion were found to give around the same results, gini will be the one used.

Random forests are essentially sets of multiple decision trees, with the advantage that they don't suffer from overfitting like deep decision trees do. For both datasets, the $max\_features$ parameter was set to *sqrt*.

4.3.1 PD. For the PD dataset, the PCA preprocessing technique was found to give the best results for the Decision Trees accuracy, so this was applied to the data. After examining which threshold value would give the best results and locking it to 0.81, the accuracy was first studied based on the *max_depth* value. Looking at the results below, as expected, it can be seen that the accuracy stays at a fixed value starting at a certain value for the maximum depth. Keeping in mind that a very high value for this can lead to overfitting, and a low value could lead to loss of information, the ideal depth was locked to 15. For this value, the accuracy study was done based on the *min_samples_split*, and the results can be seen below:



(a) vs *max_depth*

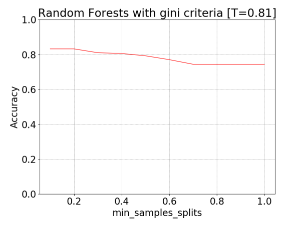


(b) vs *min_samples_split*

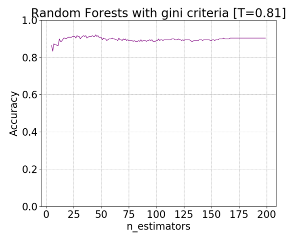
Figure 11: Classification trees accuracy for gini criteria.

As we can see, the accuracy for the *min_samples_split* study gives an all around good result, so the ideal fraction for the total amount of samples could be 0.2

For Random Forests, the threshold value used was also 0.81 because it was found that this gave the best results (even though there was not a big fluctuation between thresholds). As in the Decision Trees, the ideal *max_depth* was also obtained in the same way, with a found ideal value of around 20, and locked for the following tests. Then, the *min_samples_split* was also studied, as well as the *n_estimators*, which corresponds to the number of trees in a forest. The results can be seen below:



(a) vs *min_samples_split*

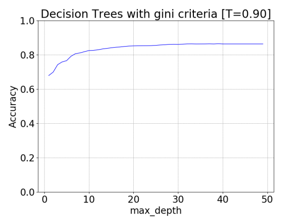


(b) vs *n_estimators*

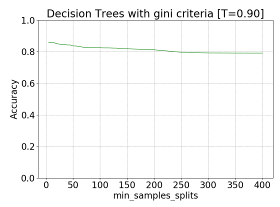
Figure 12: Random Forests accuracy for gini criteria.

By observation of the results, we can conclude that a good value for the *min_samples_split* is also 0.2, which was the same result obtained for the Decision Trees. As for the ideal number of trees, this does not seem to make a big impact in the accuracy, and the value where it appears to hit a maximum is at $n\_estimators = 45$.

4.3.2 CT. The analysis for the CoverType dataset was the same as before. For this dataset, the threshold value was found not to impact the accuracy results in the slightest, so a value of 0.9 was set for it. Like before, the accuracy was studied first based on the *max_depth* values, and the ideal one was set to 30 for the reasons mentioned before. Then, the accuracy study was done based on the *min_samples_split* value. The results can be seen below:



(a) vs *max_depth*

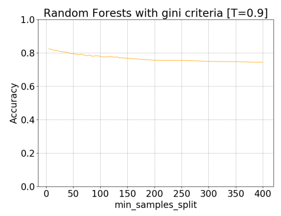


(b) vs *min_samples_split*

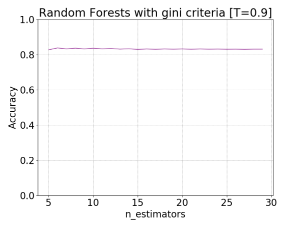
Figure 13: Classification trees accuracy for gini criteria.

For this dataset, the ideal value for *min_samples_split* would be essentially as low as possible, around 5 or 10.

For Random Forests, the ideal *max_depth* was again found and set to 30. The *min_samples_split* and $n_estimators$ parameters were then again studied:



(a) vs *min_samples_split*



(b) vs n_estim

Figure 14: Random Forests accuracy for gini criteria.

Just as before, the *n_estim* parameter does not seem to make a difference in accuracy. The ideal *min_samples_split* is, just like for the decision trees, the lowest possible value, so around 5 or 10.

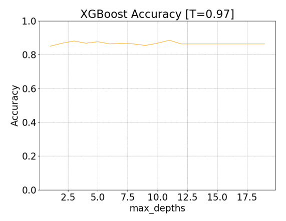
It can be concluded that for both datasets, the best way to achieve the best results for the Decision Trees and Random Forests classifiers is to find the best parameters by varying them individually, and establishing the ideal ones.

4.4 Gradient Boosting

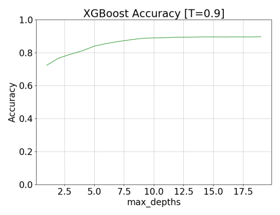
Gradient boosting is the last step on the evolution of tree-based algorithms, and XGBoost is the optimized gradient boosting algorithm which will be now studied. This algorithm is designed to be highly effective, so it is expected to give better accuracy results than the last two classification techniques.

For both datasets, the accuracy was measured as a function of the *max_depth* parameter.

For this algorithm, the threshold was fixed to 0.97 for the PD dataset because this was found to give the best results. For the CT dataset, as it was mentioned above, the threshold does not make an impact so it was set to 0.9.



(a) PD



(b) CT

Figure 15: XGBoost Accuracy vs *max_depth* parameter.

By analyzing the results, the best values for the *max_depth* parameter for the PD and CT datasets are ≈ 11 and 10, respectively.

The best results were found to be given by raw data, without any preprocessing techniques applied, just like for the previous two tree-based models (except Decision Trees in the PD dataset, where PCA was applied). This can be explained by the fact that tree-based models are not influenced by feature scaling, since the best split positions are searched along a feature's data axis and so the relative positioning of data points is not important.

5 UNSUPERVISED LEARNING

5.1 Association Rule Mining

Pattern mining concentrates on identifying rules that describe specific patterns within the data. For this subject, we'll be focusing on two specific methods: Association Rule Mining (ARM) and Sequential Pattern Mining (SPM). The first one is a rule-based machine learning method, that allow the discovery of interesting relations between variables, while the second one address the problem of discovering the existing maximal frequent sequences in a given database.

To apply ARM, we first need to discretize real-valued attributes into binary. This can be done by using a width based algorithm ('cut') or a depth based algorithm ('qcut'). On Figure 16, we show how the 'qcut' application leads to a very balanced counting on variable discretization, while the 'cut' gives us a better perspective on the values overall distribution.

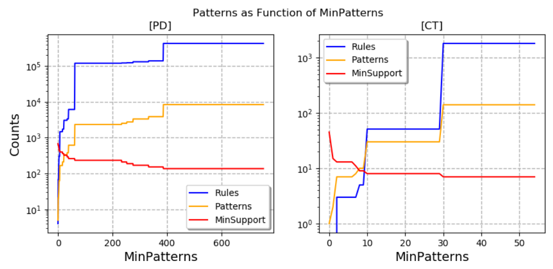


Figure 17: Pattern mining parameters counting as a function of minimum support.

Unfortunately, due to a lack of computational power, we were not able to perform the apriori arm algorithm using the previously obtained dummified data. Instead, we'll be using an alternative Recursive Elimination (RElim) method. It has the advantage that it does its work without prefix trees or any other complicated data structures, hence it is faster, but it can only find frequent item sets, not association rules. Nevertheless we will explore it.

On Figure 17 we can see how the number of patterns, the number of rules and the minimum support evolves as we increase the minimum number of patterns. It comes to be obvious that the number rules goes along with the number of patterns found. Besides that, we can also see that the minimum support decreases as the number of patterns and rules increase.

As support is an indication of how frequently the itemset appears in the dataset, it comes to be obvious that, by increasing the number of patterns, the required minimum support decreases.

This RElim method has a fast response and seems to be a very 1st order pattern "miner", but for a more detailed pattern discovery, one should consider using an ARM model using the dummified data. We presented a 3 bins discretized data on Figure 16, but to obtained a more detailed data distribution, one shall discretize the data into more bins.

5.2 Clustering

In this section, several variants of clustering techniques were applied in order to try to find the model which has the highest metric scores. To do this, kmeans, DBSCAN and Agglomerative Clustering were experimented, all of them with different combinations of normalizations. For the PD dataset, clustering was applied to the 7 different subsets and for the CT dataset, due to being very large, clustering was applied to several subsets. Other important aspect is how the number of clusters for each subset is selected.

This choice is based on the parameter **Sum Squared Error**-SSE , that is a measure of how well our model fits to the data. In principle, the lower this value is the better, and as the number of clusters increases the more division we have in the data, resulting in a more homogeneous model. However, there is a trade-off between precision and interpretability/computing power, so the strategy should be trying to minimize the SSE while trying to keep the minimum number of clusters. If we plot SSE vs nr. clusters, a kind of 'elbow' line is shown, and from there the optimal value for k is selected (as shown below). To cross-check this reasoning, a dendogram will also provide information on how to choose the number of clusters.

NOTE: These considerations above are based on experiment and comparing the metrics(i.e silhouette and adjusted rand index) of each case scenario, selecting the most favorable ones. Below, the most relevant results that corroborate our assumptions are shown.

5.2.1 PD. Relative to the PD dataset, a separation by index was made, to try to find specific subgroups. During the analysis, different indexes would have different numbers of clusters.

Applying preprocessing.Normalizer(), SelectKBest() and PCA(), to the first subset of PD data we were able to obtain Figure 18.

As we can see, the 'elbow' gives optimal $k = 2$ and this example is particularly interesting, because selectKbest and PCA, actually give a worst result than just normalizing the data. By looking at the silhouette coefficient and the adjusted rand index (ideally 1 and 0, respectively). With just normalization, we can say we have 'instant' classification. Without supervision, we have two well compact,

	MinMax (1)	StandardScaler (2)	(1) + (2)	SMOTE (3)	Feature Selection (4) + (1)	PCA 14 components (5) + (1)	LDA (6)	(2) + (4)	(2) + (5)	(5) + (3)
KNN	0.8166	0.8816	0.8816	0.8639	0.7869	0.9290	0.7159	0.7869	0.9289	0.9231
NB	0.8757	0.6390	0.6390	0.4970	0.6804	0.662	0.7159	0.6923	0.7160	0.497
DT	0.7396	0.7633	0.7633	0.8284	0.8520	0.8875	0.7278	0.8520	0.8875	0.5976
RF	0.9763	0.9526	0.9526	0.8224	0.9171	0.9881	0.7278	0.9171	0.9882	0.7455
XGB	0.9467	0.952	0.9526	0.9112	0.8816	0.8816	0.7278	0.8817	0.8816	0.8106

Table 1: Accuracy Scores for Classification - Parkinson's Disease

	MinMax (1)	StandardScaler (2)	SMOTE	RandomUnderSampler (3)	Feature Selection (4) + (3)	PCA 14 components (5) + (3)	LDA (6)	Outlier Removal (7) + (3)
KNN	0.6167	0.6091	0.8133	0.795	0.7693	0.8133	0.779	0.7708
NB	0.1967	0.3921	0.5746	0.5636	0.5076	0.55	0.603	0.6563
DT	0.6294	0.6294	0.617	0.583	0.6113	0.53	0.655	0.6568
RF	0.6269	0.6269	0.5763	0.588	0.61	0.542	0.7278	0.6563
XGB	0.7170	0.7170	0.86	0.817	0.8363	0.853	0.792	0.6917

Table 2: Accuracy Scores for Classification - Cover Type Dataset

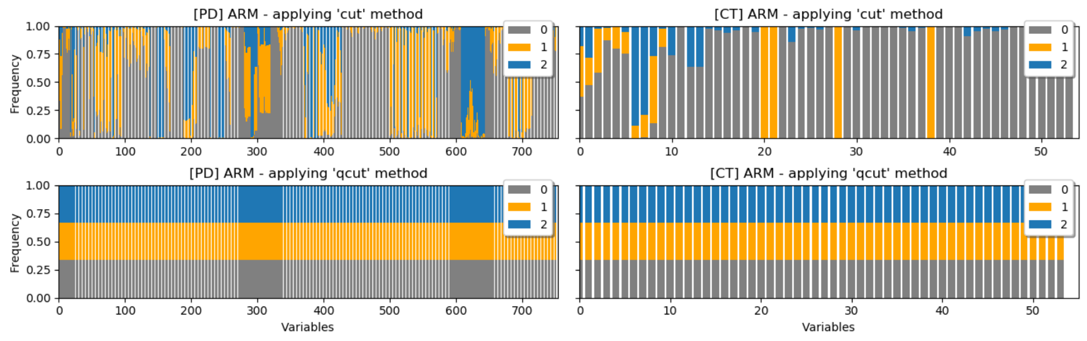


Figure 16: Association Rule Mining (ARM) discretization frequencies along the data sets variables. Frequency values are given in comparison with the total number of instances.

separated and homogeneous clusters, representing the sick and healthy population. This is important, because now we know that if we explore these features during classification, the metrics will be better.

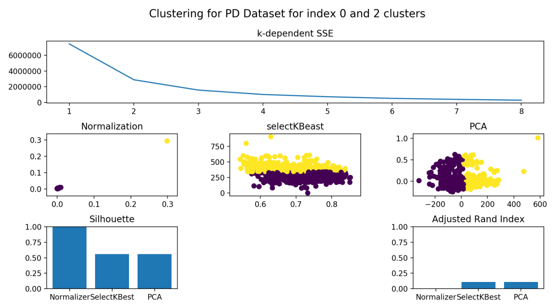


Figure 18: KMeans algorithm - PD dataset - index: 0, 2 clusters

To finish KMeans analysis, we take a look at a subset with a higher number of clusters.

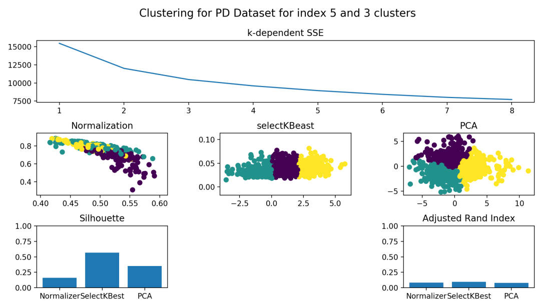


Figure 19: KMeans algorithm - PD dataset - index: 3, 3 clusters

In this case, it is not so obvious the number of clusters to select. The 2 possibilities were tested and the one with better metrics was chosen, which is 3 clusters. Here, normalizing the data does not provide good results. SelectKBest retrieves the best metrics and we can see, these features form 3 different subgroups and the silhouette coefficient is only 0.5, although the adjusted rand index is not very pronounced. In addition, it is perfectly noticeable that we are in the presence of outliers, that belong to a subgroup, but have a considerable distance to the centre of the cluster. Ideally these could be removed.

Finally, we test how the removal of outliers affects the clustering, to the whole dataset.

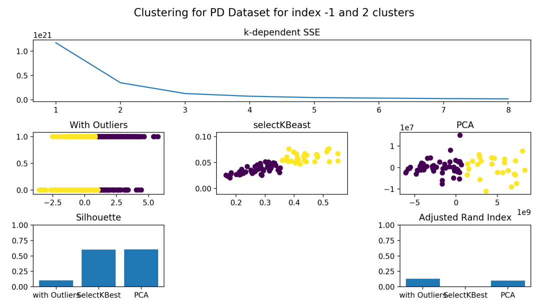


Figure 20: KMeans algorithm - PD dataset - 2 clusters

Where we compare, how the clustering would be like if the outlier analysis was not made and with outlier removal combined with SelectKbest and PCA.

5.2.2 CT. Due to the fact that the CT dataset is very populated, firstly we try to implement density based and fast cluster algorithms like Agglomerative Clustering and DBSCAN. The main difficulties rely on choosing the appropriate parameters to maximize the scores. After analysing the optimal k, with the sum squared error plot:

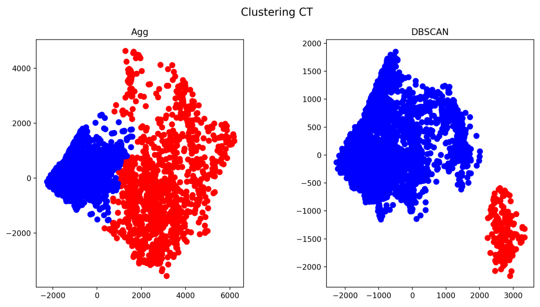


Figure 21: Agg and DBSCAN algorithm - CT dataset - 2 clusters

Output Dendrogram from Hierarchical Clustering, that shows one way of grouping features into clusters, drawing horizontal lines and seeing how many branches it intercets. From there we see that mainly we can form 2 or 3 distinct clusters, whose separablity might be optimal.

Applying KMeans is much more difficult given the amount instances, the algorithm is much slower to run. Applying Random under sampling techniques, and after that centering the data with MinMaxScaler, for SelectKBest() - 2 clusters and PCA() - 4 clusters, we were able to obtain Figure 23.

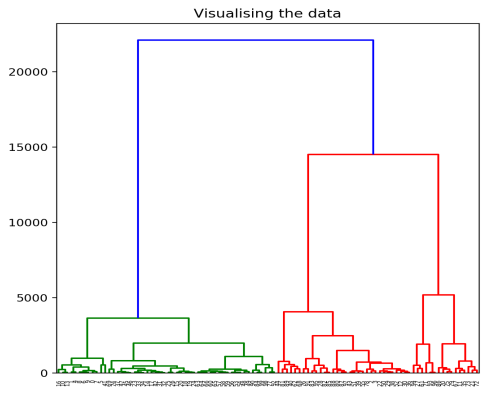


Figure 22: Dendrogram CT dataset

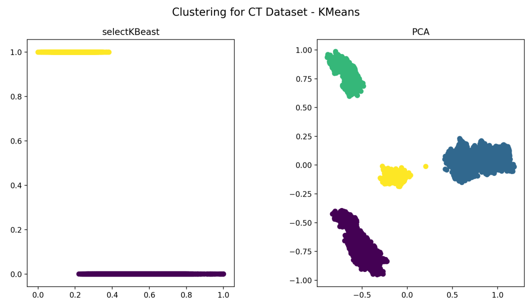


Figure 23: KMeans algorithm - CT dataset

	KMeans (PCA)	KMeans(SelectKBest)	DBSCAN	AggClust
Silhouette	0.7954	0.5856	0.5013	0.6312
Adj. Rand Index	0.3191	0.3106	0.1001	0.1210
Homegeneity	0.3225	0.4010	-	-

Table 3: Clustering metrics for CT Dataset

From the table, KMeans with PCA revealed to be the technique with better metrics, with silhouette scores close to 1, meaning the elements inside each cluster are alike one another, when compared to other clusters, making the clusters consistent. Also it is worth noticing that these clusters are not homogeneous, which is expectable when there are 7 classes in this dataset, and only 4 clusters. A good value for the adjusted rand index should be close to 0, however this measure that translates into the similarity between clusters, is 20% higher than the other clustering methods applied. So concluding, the most advantageous point of choosing density-based clustering is its flexibility to handle large dataset with large number of clusters, with the trade-off of cohesion in each one of them.