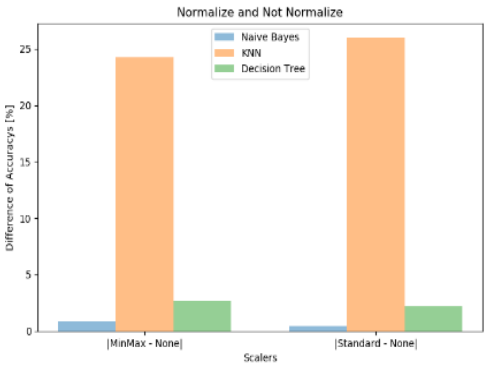
Statistical analysis

- The 1st dataset has 756 records, 754 attributes. The target class is binary. All attributes are quantitative.
- The 2nd dataset has 581012 records and 54 attributes. The target class has 7 possible values. Types of attributes:
    - 44 binary
    - 10 quantitative
- There aren't missing values in any of the datasets.

Preprocessing

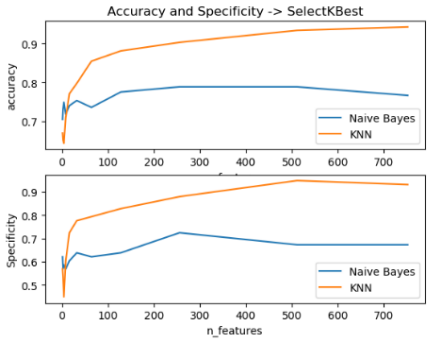
Scaling:

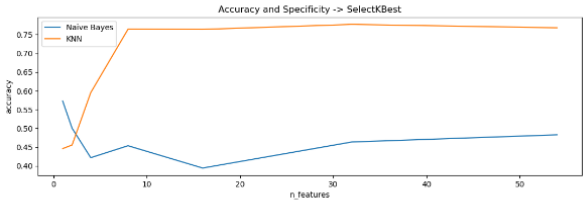
- We suspect that scaling will improve classification results in the classifiers that may be affected by scaling, due to the different scales of the features in the dataset.
- NB: should not be very affected because the model's priors determined by the count in each class and not by the actual value.
- **Distance-based** such as KNN methods are affected by scaling.
- **Tree-based** models are not distance-based and so can handle varying ranges of features. Thus, scaling did not affect much the quality of the trees model.
- In the Cover Type dataset, we reduced the records per category to 1000, by taking random samples from the original dataset.



Feature Selection

- We applied both filter and wrapper models to both datasets. However, given that the wrapper model is more computationally expensive, we only used one algorithm (SelectFromModel) instead of two as for the filter (SelectKBest, SelectPercentile). For the filter model, we use a sequential forward selection approach to hypothesize the best number of features.
- Moreover, the learning algorithms that were used to analyze the performance for a specific number of features were Naïve Bayes and KNN.



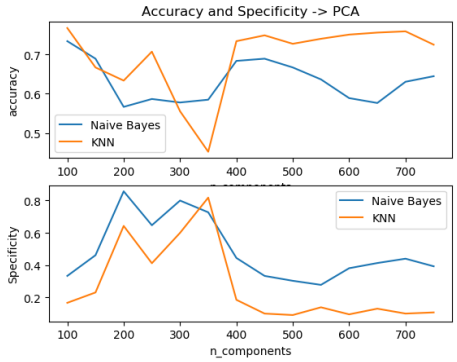


Results

- Naïve Bayes doesn't change much the accuracy with the change in the number of features. However, it gets the best specificity with the number of features equal to 256.
- KNN has a better specificity when we reduce the number of features to 500, which makes sense given the curse of dimensionality. However, if we keep decreasing the number of features the accuracy will decrease.
- As we can see, feature selection depends on the classifier that we are using. So, it's very difficult to say exactly what is the best number of features. Nevertheless, we conclude that for the PD dataset the number of features should be between 256 and 512, and for the CT dataset between 25 to 35 features.

PCA

- Our approach to PCA analysis was similar to feature selection. We vary the number of components and analyzed its performance using Naïve Bayes and KNN. Let's take a closer look at the results (PD dataset)



Results

- We can immediately observe that, for both algorithms, the increase number of components the performance in terms of specificity is lower, but the accuracy its higher overall. We can also see that there is a peak on specificity and a valley in the accuracy for the same number of components.
- Unfortunately, we were unable to run PCA with the CT dataset. However, we suspect that outcome would be similar, since there is a lot of records in the original, selecting the best components would improve the classification report.

Classification

The first thing we tried in classification was to fix the preprocessing and see how tuning the parameters for each classifier would affect the accuracy (and sensitivity in Parkinson dataset) of the classification.

Fixed preprocessing for PD and CT:

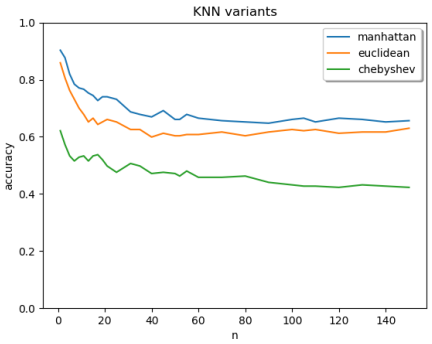
- Balanced with SMOTE
- Scaled with StandardScaler

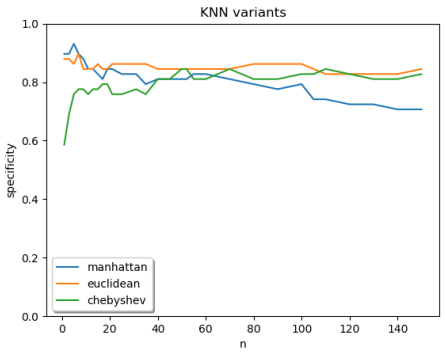
1. Naive Bayes

- Multinomial and Bernoulli distributions are discrete probabilities distributions and are appropriate for counts. Since the 1st dataset has mostly real values, the MultinomialNB and BernoulliNB are not appropriate for this kind of classification, so we only use the GaussianNB.
- We did not change any parameters in GaussianNB.

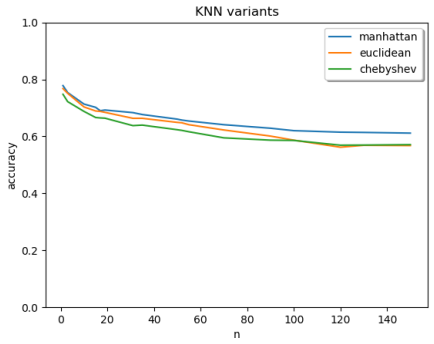
2. KNN

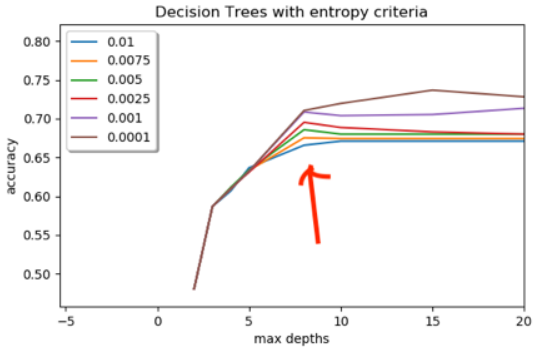
- In KNN, we changed the 2 following parameters:
    - Number of neighbors
    - Distance calculation method
- The PD dataset has approximately as many features as records. And so, it makes sense that the best number of neighbors is close to 1, because it is unlikely to have more than 1 record similar to a test instance, if the number of features is not reduced. As we can see in Figure x, the accuracy decreases instantly when the number of neighbors increases. As we get good accuracy and specificity for k=1, we can conclude that the dataset doesn't have much noise. We also thought that due to the curse of dimensionality and the lack of observations, the KNN classifier for k = 1 would have a bad performance in the testing set. This may not happen because of a possible similarity between training set and testing set.





- In the CT dataset, we have a similar behavior to the PD dataset in the value of accuracy as we increase the number of neighbors. This dataset has many records and few features and so it makes sense that k=1 classifier gives good accuracy as shown in Figure w.

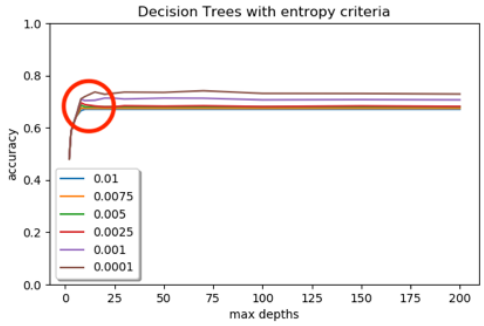




3. Decision Trees

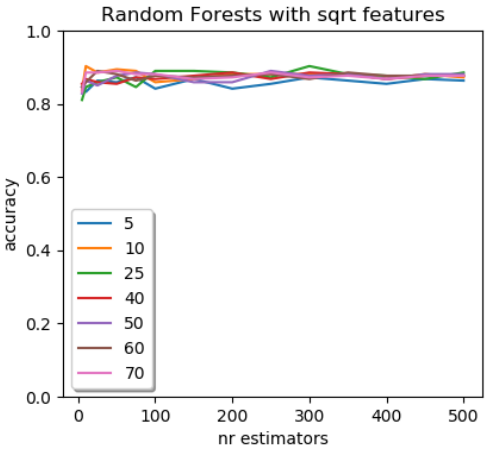
- In decision trees we changed the 3 following parameters:
    - Criteria: gini or entropy
    - Maximum tree depth
    - Minimum number of samples to be a leaf node

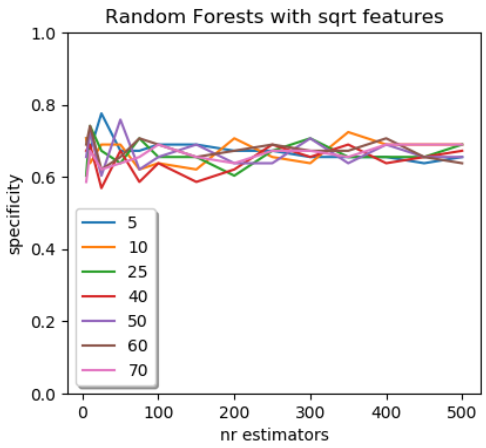
- Gini and entropy criterion give approximately the same results in both datasets.
- In the Parkinson dataset, the best maximum depths are 3 and 4. The specificity decreases as we increase the maximum depth for the tree, but at most 5%. As we increased the maximum depth of the tree, we expected the accuracy to decrease a lot due to overfitting. We think this is not happening because of low sampling error in the given dataset. This may also happen due to high similarity between training and testing set.
- In the Cover Type dataset, as we have 7000 records, it makes sense that the accuracy will increase as the minimum samples leaf parameter is decreases. The tree gets to a depth around 20 and stops increasing, and so from this point increasing the maximum depth of the tree parameter will not improve the accuracy.



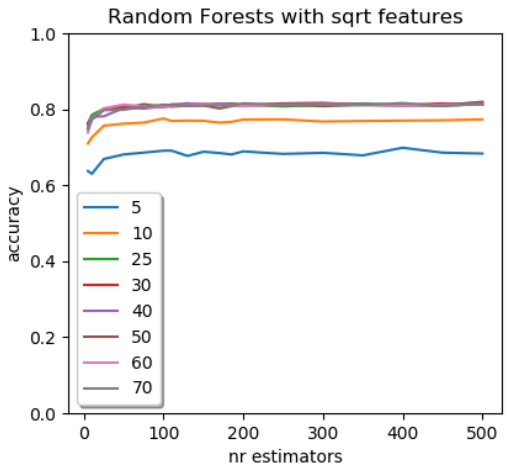
4. Random Forests

- In random forests we changed the 3 following parameters:
    - Number of trees in the forest
    - Maximum tree depth
    - Number of features to consider for the best split
- The accuracy doesn't change with the method (sqrt or log2) we use to choose the number of features to consider for the best split.
- Random forests can combat the increase in variance by averaging over multiple trees and so they will lower the overfitting. There seems to exist no overfitting in both datasets, because by increasing the maximum depth or the number of estimators, the accuracy remains approximately the same, as we can see in Figure y.
- Knowing that Random Forests is an improvement of decision trees, it should have better performance in classification, and it does, as shown in the following figures in comparison to the 2 previous ones.



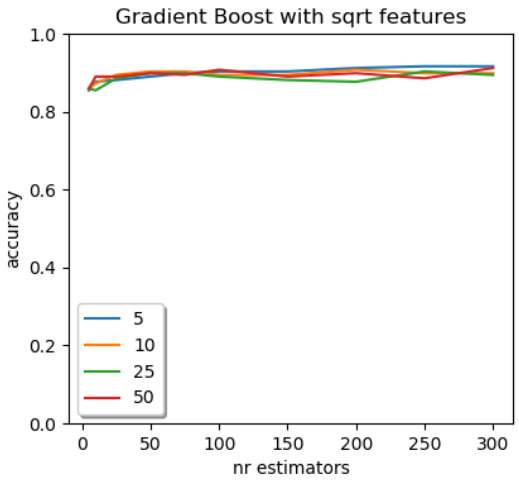


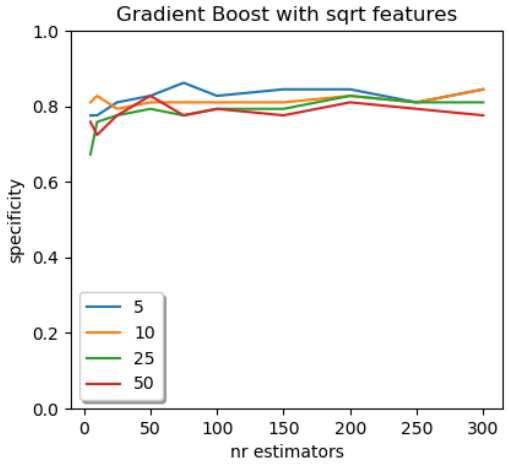
- In the 2nd dataset, we can see that the accuracy increases as we increase the maximum depth of the tree until it reaches a maximum depth of 25. An increase in the number of estimators is relevant until the number of estimators is close to 100 and then the accuracy stabilizes.

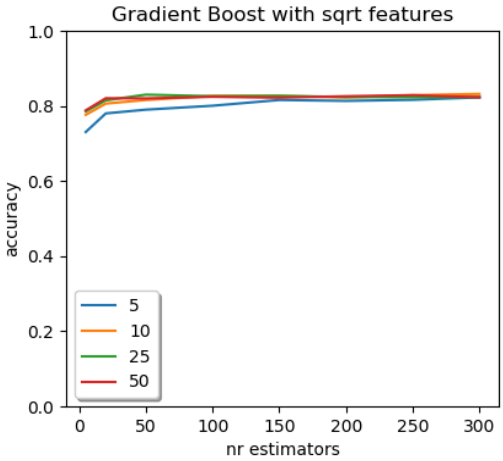


5. Gradient Boosting

- In gradient boosting we changed the 4 following parameters:
  - Learning rate
  - Number of trees in the forest
  - Maximum tree depth
  - Number of features to consider for the best split
- We previously thought that gradient boosting algorithm would have a better performance than the previous classifiers that did used a single model to classify. The observed results confirmed it, as we can see the specificity for the Parkinson dataset has values over 80% which is better than what we have observed so far.
- One of the reasons for good performance of this classifier is that the datasets analyzed don't have significant noise.

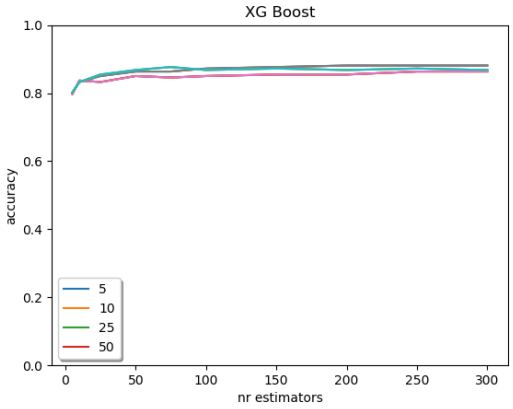


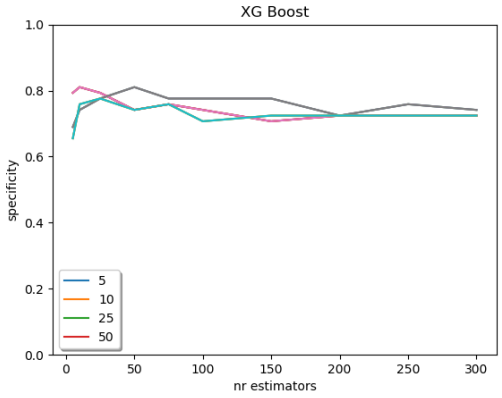




6. XGBoost

- In xgboost we changed the 2 following parameters:
  - Number of trees in the forest
  - Maximum tree depth
- The accuracy increases slightly with the increase of the number of estimators





Unsupervised

• Clustering

To determinate the best number of clusters for each dataset, we used the Elbow Method. The **elbow method** runs k-means clustering on the dataset for a range of values for k (1-12) and then for each value of k computes an average score for all clusters. Our average score, the distortion score is computed, with the sum of square distances from each point to its assigned center.

Then we calculated the measures sillhoutte and rand-index for the k-means with k fixed (result from elbow method), to understand the quality of the resulting clustering. Results for dataset1:

- Number of clusters: 4
- Average Silhouette: -0.444
- Rand Index: 0.012

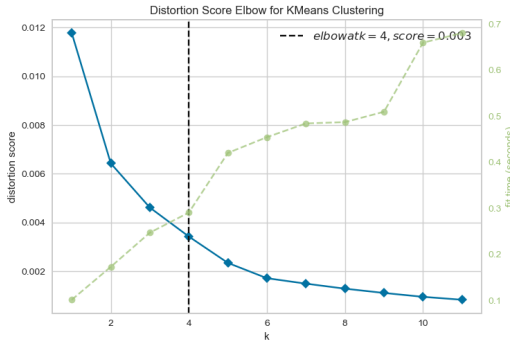
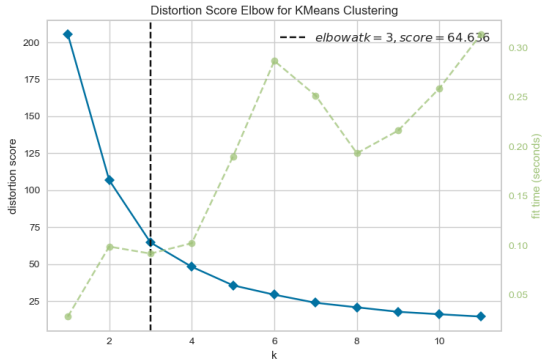


Figura 1 - Elbow method (PD dataset)

For dataset2:

- Number of clusters: 3
- Average Silhouette: 0.106
- Rand Index: 0.068



To visualize the clustering solution we thought that was a good idea to plot with the 3 best features for each dataset so we used SelectKBest to extract the features.

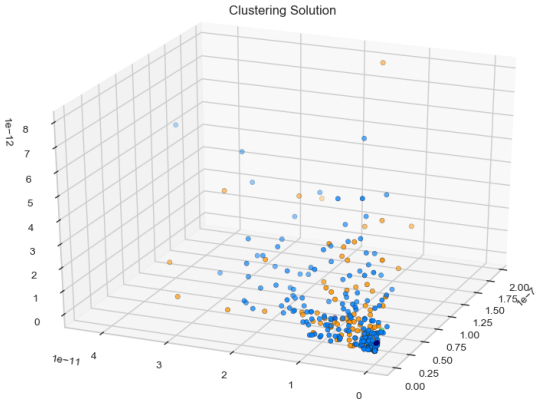


Figure 3 – Clustering(dataset1)

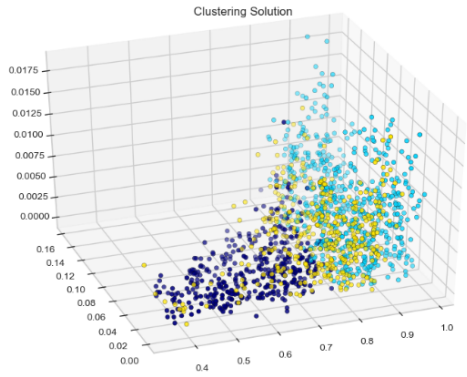


Figure 4 – Clustering(dataset2)

- ## Pattern Mining

To perform pattern mining in the datasets we used the apriori algorithm to find the frequent itemsets with different minimum supports. Then we compute association rules with the given frequent itemsets. We discretize the two datasets with the frequency technique(qcut).

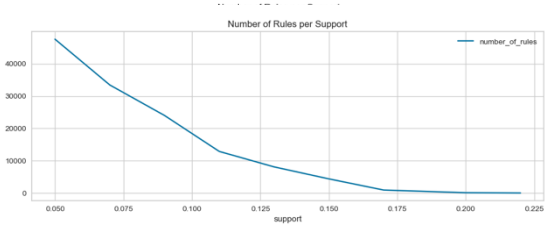


Figure 5 – Number of rules per support (dataset1)

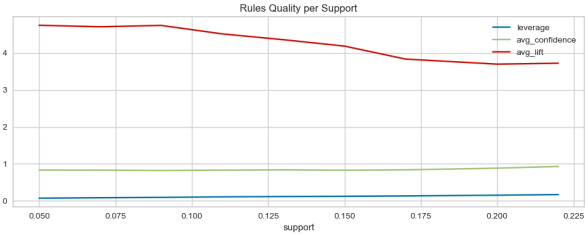


Figure 6 – Rules Quality per support (dataset1)

For the second dataset we discretize just the first 10 variables from dataset because the rest of them are binary. We choose to discretize with 6 bins in dataset1 because with less bins the algorithm was finding repeated patterns and the top rules were repeating themselves so we think that this number of bins it´s a good balance between the number of rules and number of variables. With the same reasoning we used 5 bins in dataset2.

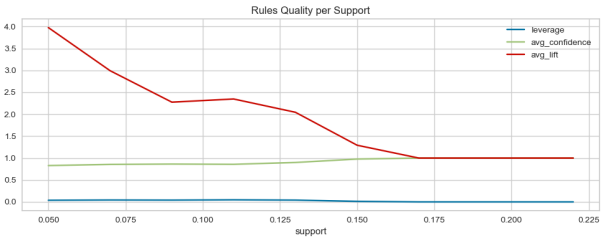


Figure 7 – Rules Quality per support (dataset2)

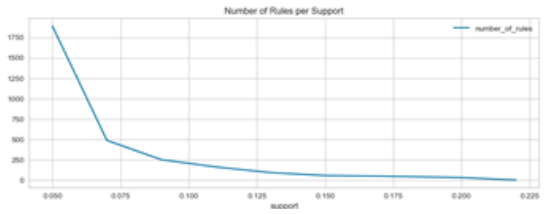


Figure 8 – Number of rules per support (dataset2)

To find the best rules we used the measure confidence to compare different rules:

-Top 5 Rules in dataset1:
('6_3', '5_3') -> ('9_0')
('6_3', '8_0', '5_3') -> ('9_0')
('9_0', '8_0') -> ('6_3')
('9_0', '8_0', '5_3') -> ('6_3')
('6_3', '9_0', '5_3') -> ('8_0')

-Top 5 rules in dataset2:
('Aspect_4','Hillshade_3pm_4') -> ('Hillshade_9am_0')
('Hillshade_9am_0', 'Hillshade_3pm_4') -> ('Aspect_4')
('Slope_4', 'Hillshade_3pm_0') -> ('Hillshade_Noon_0')
('Hillshade_Noon_0', 'Hillshade_3pm_0') -> ('Slope_4')
('Hillshade_3pm_4') -> ('Hillshade_9am_0')