UNIVERSITÀ DEGLI STUDI DI TRIESTE

Intelligenza Artificiale and Data Analytics

**INTRODUZIONE AL MACHINE LEARNING**

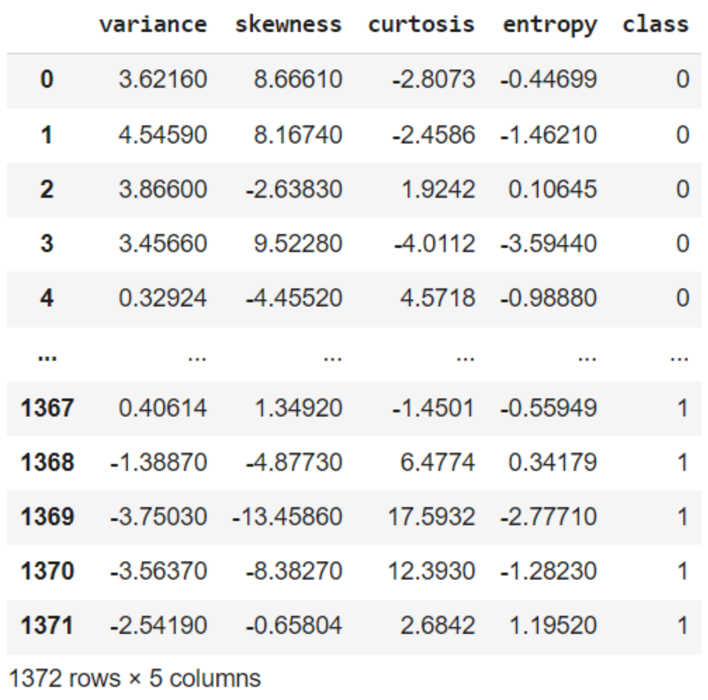
# Report challenge one

Alberto Facchin

Febbraio 2024

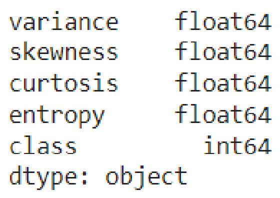
## 1. Data pretreatment

The dataset banknote authentication has 1372 rows and 5 columns.
It presents the following structure:



| | variance | skewness | curtosis | entropy | class |
|---|---|---|---|---|---|
| **0** | 3.62160 | 8.66610 | -2.8073 | -0.44699 | 0 |
| **1** | 4.54590 | 8.16740 | -2.4586 | -1.46210 | 0 |
| **2** | 3.86600 | -2.63830 | 1.9242 | 0.10645 | 0 |
| **3** | 3.45660 | 9.52280 | -4.0112 | -3.59440 | 0 |
| **4** | 0.32924 | -4.45520 | 4.5718 | -0.98880 | 0 |
| **...** | ... | ... | ... | ... | ... |
| **1367** | 0.40614 | 1.34920 | -1.4501 | -0.55949 | 1 |
| **1368** | -1.38870 | -4.87730 | 6.4774 | 0.34179 | 1 |
| **1369** | -3.75030 | -13.45860 | 17.5932 | -2.77710 | 1 |
| **1370** | -3.56370 | -8.38270 | 12.3930 | -1.28230 | 1 |
| **1371** | -2.54190 | -0.65804 | 2.6842 | 1.19520 | 1 |

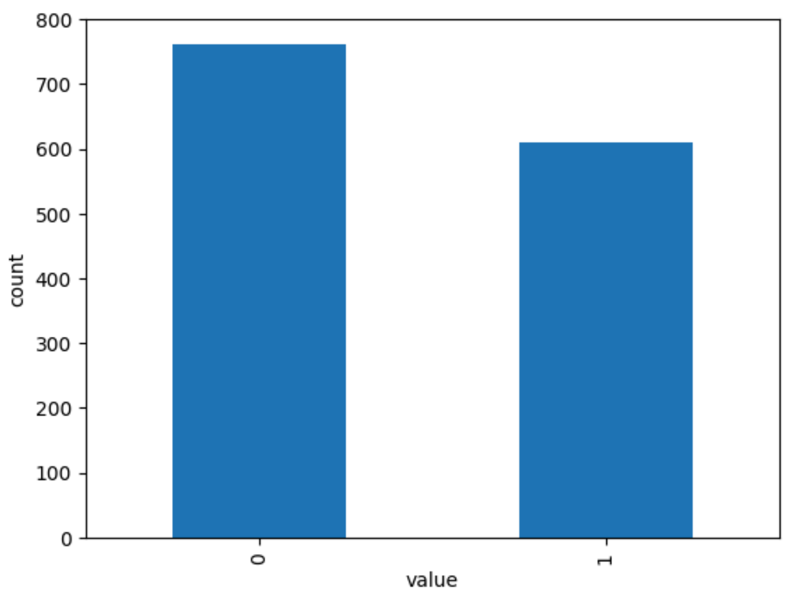
1372 rows × 5 columns



```
variance    float64
skewness    float64
curtosis    float64
entropy     float64
class         int64
dtype: object
```

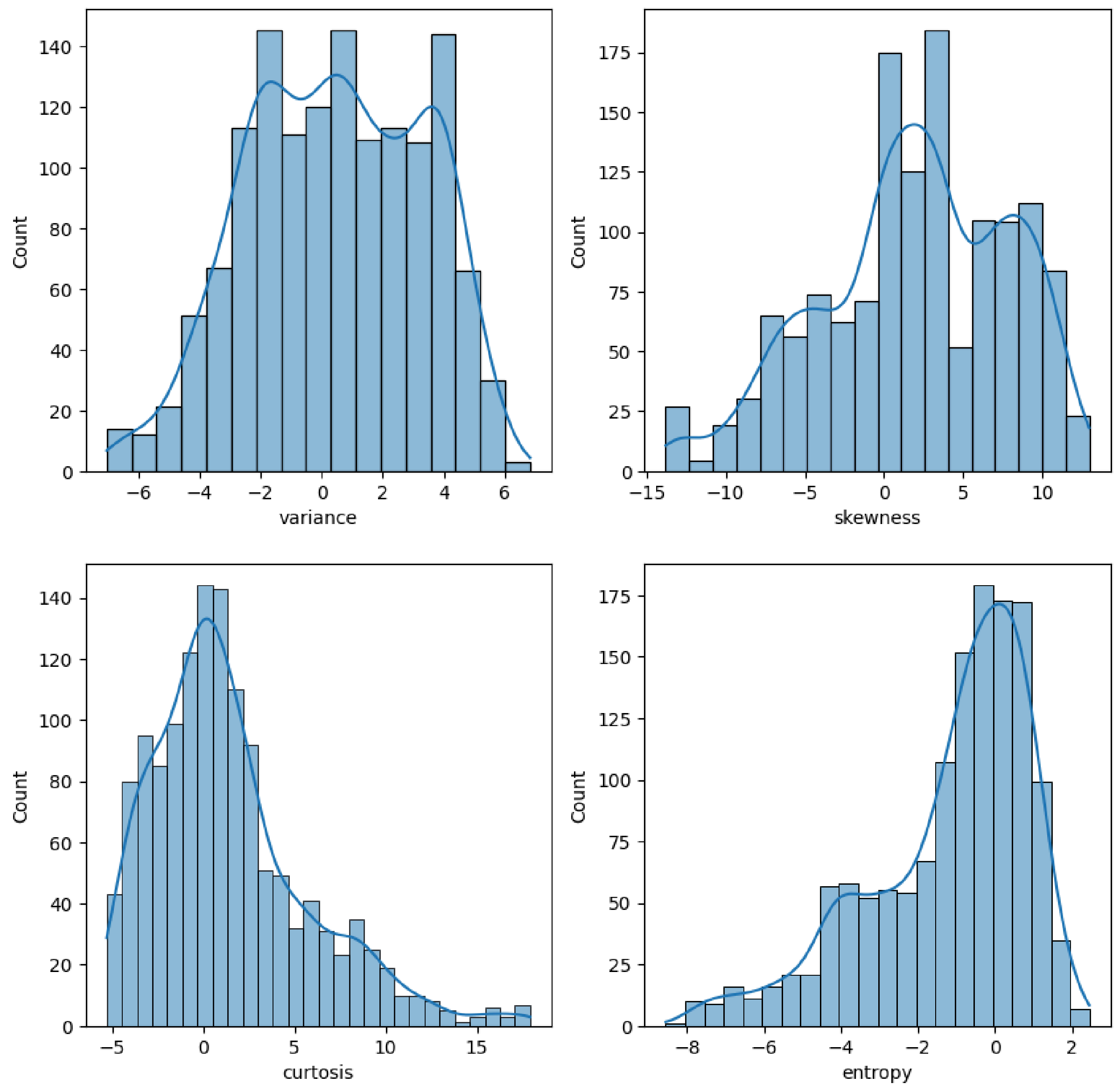
It seems that it is ordered, probably, in ascending by class, so I proceeded with a shuffling. Without shuffling data, the splitting into train set and test set could lead to wrong results because, for example, it can happen that the entire test set has the observations with class = 0 or class = 1, which means that the predictions won't be truthful.

Then I wanted to have an overview of the class values, so I used a barplot.
There are 610 observations for class = 1 and 762 for class = 0



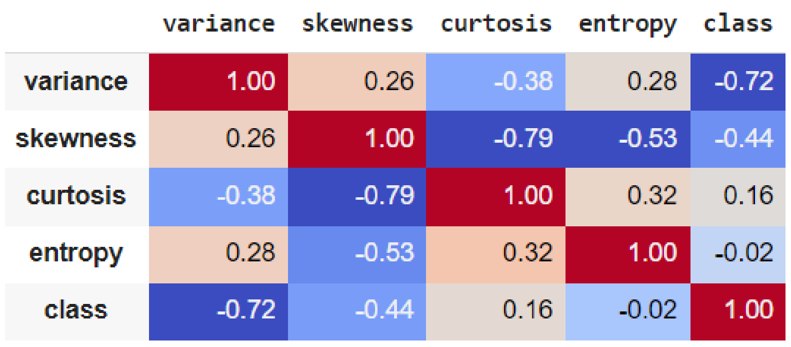
## Histograms with KDE

For the continuous quantitative variables I tried to see if they follow a Gaussian distribution by applying the kernel density estimation (KDE), and the plot below is the one I derived from:

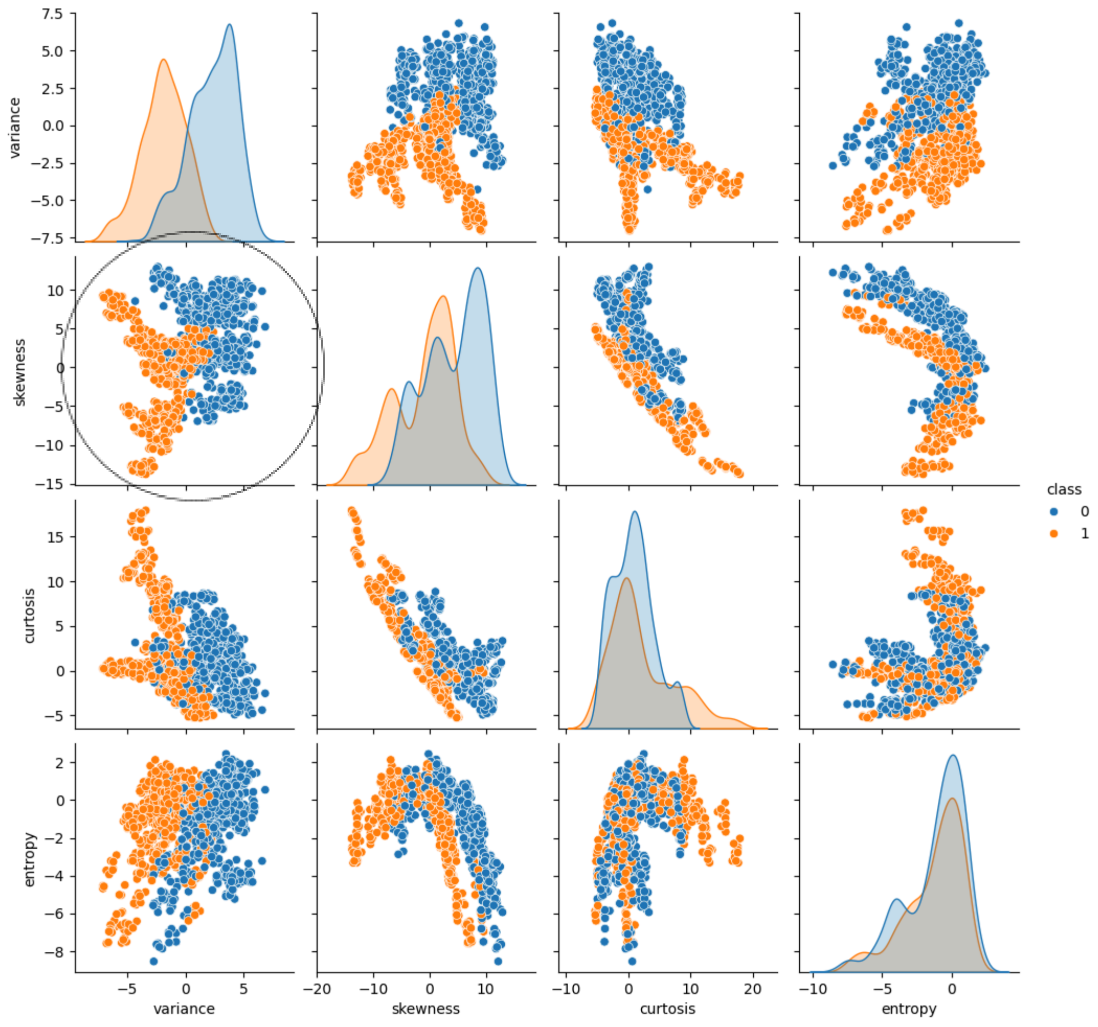


## Correlation matrix

In order to evaluate the dependency between the variables I used the correlation matrix and the pairplot.



| | variance | skewness | curtosis | entropy | class |
|---|---|---|---|---|---|
| variance | 1.00 | 0.26 | -0.38 | 0.28 | -0.72 |
| skewness | 0.26 | 1.00 | -0.79 | -0.53 | -0.44 |
| curtosis | -0.38 | -0.79 | 1.00 | 0.32 | 0.16 |
| entropy | 0.28 | -0.53 | 0.32 | 1.00 | -0.02 |
| class | -0.72 | -0.44 | 0.16 | -0.02 | 1.00 |



I took into consideration the variables skewness and variance, which has a Pearson correlation coefficient equal to 0.26.

## Scaling

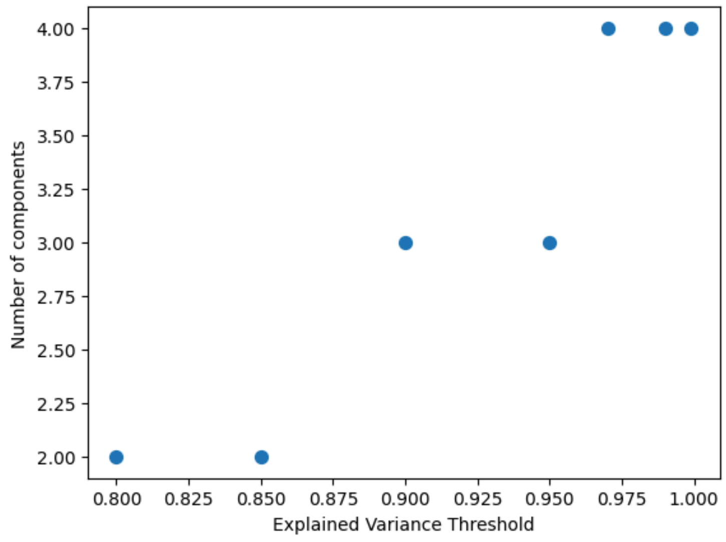
It is not so essential to rescale features since they are all in the same range and there are no problems with the units of measure, but there is no reason why the scaling (in this case) could lead to worse results, so I decided to use the standard scaler.

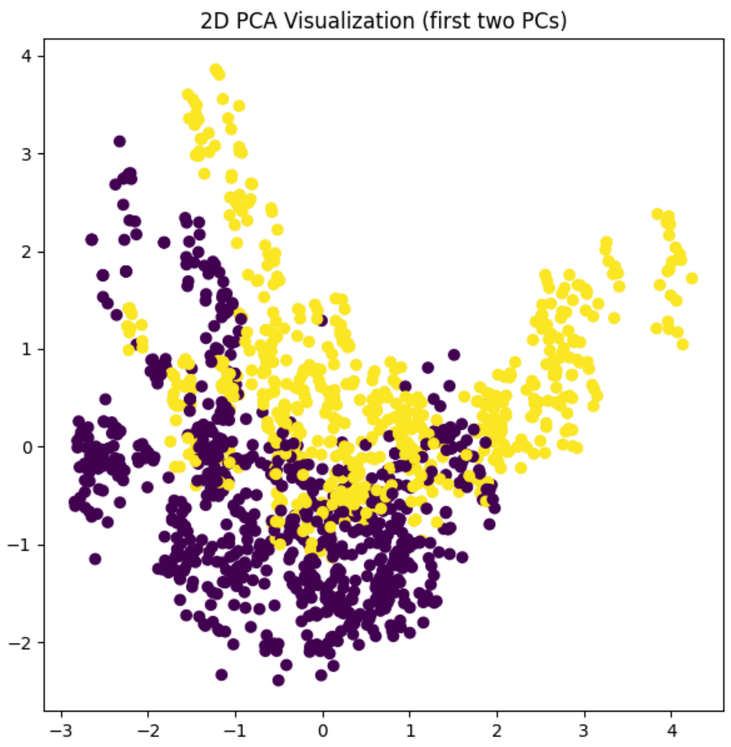
# 2. Unsupervised Learning

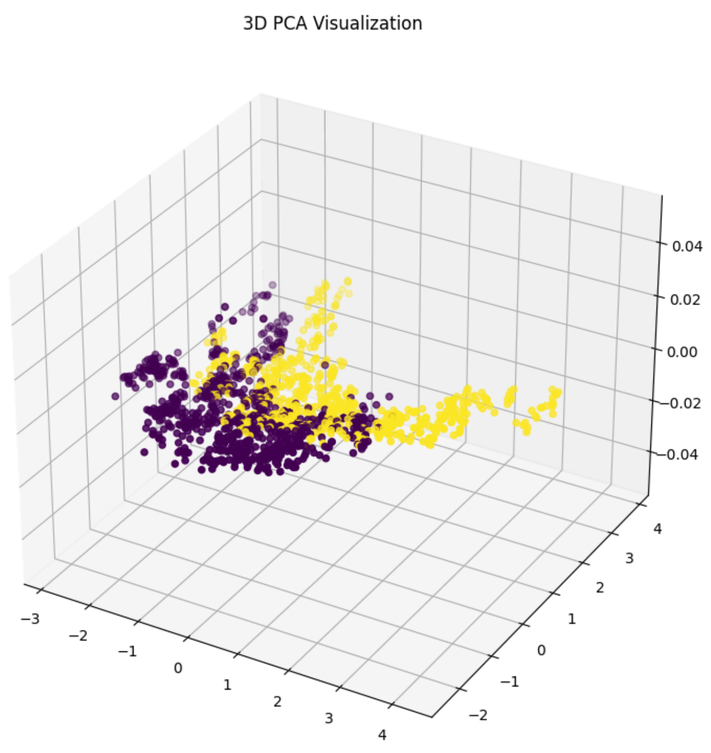
## PCA

After comparing the different values of explained variance based on the number of components, I used the first two Principal Components (PCs) and I obtained the following PCA.

As we can see from the plot below, with just two components we obtain a good explained variance, so it is sufficient to use these.

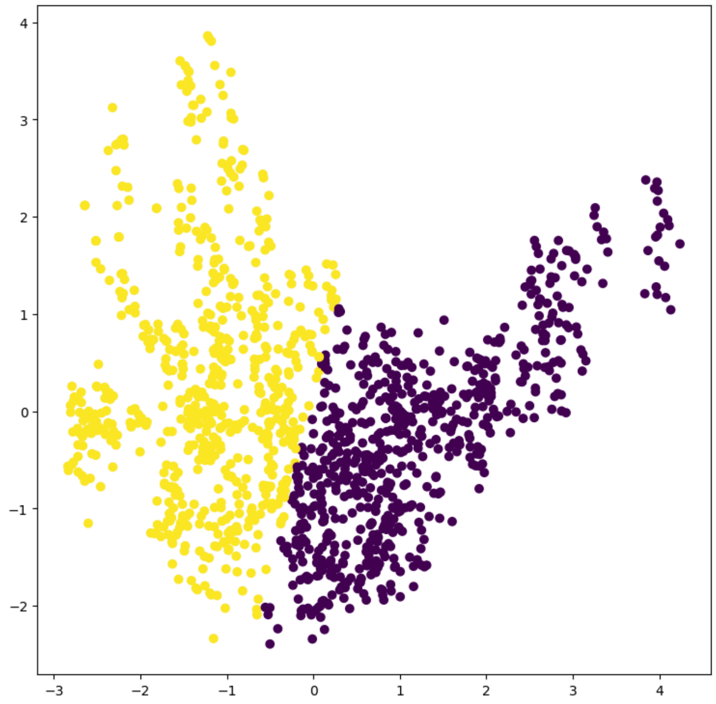


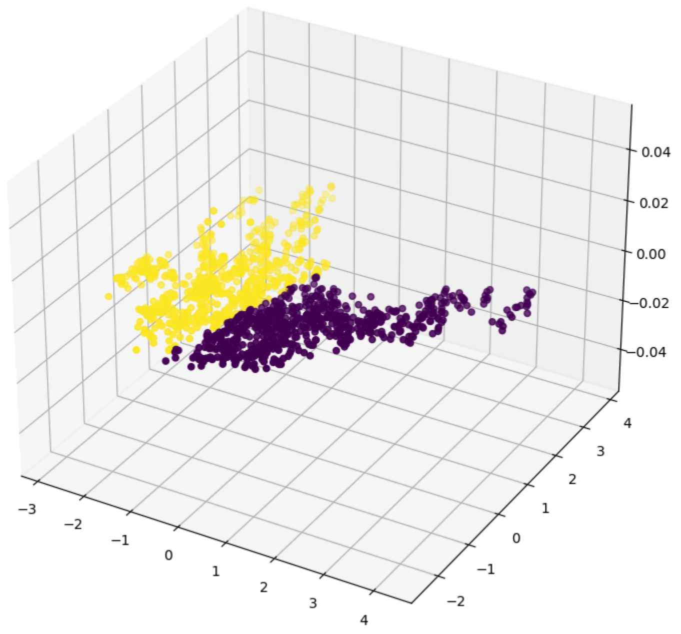




It seems that the classes are not linearly separable.
The explained variance with 2 PCs is equal to 0.85.

## k-means



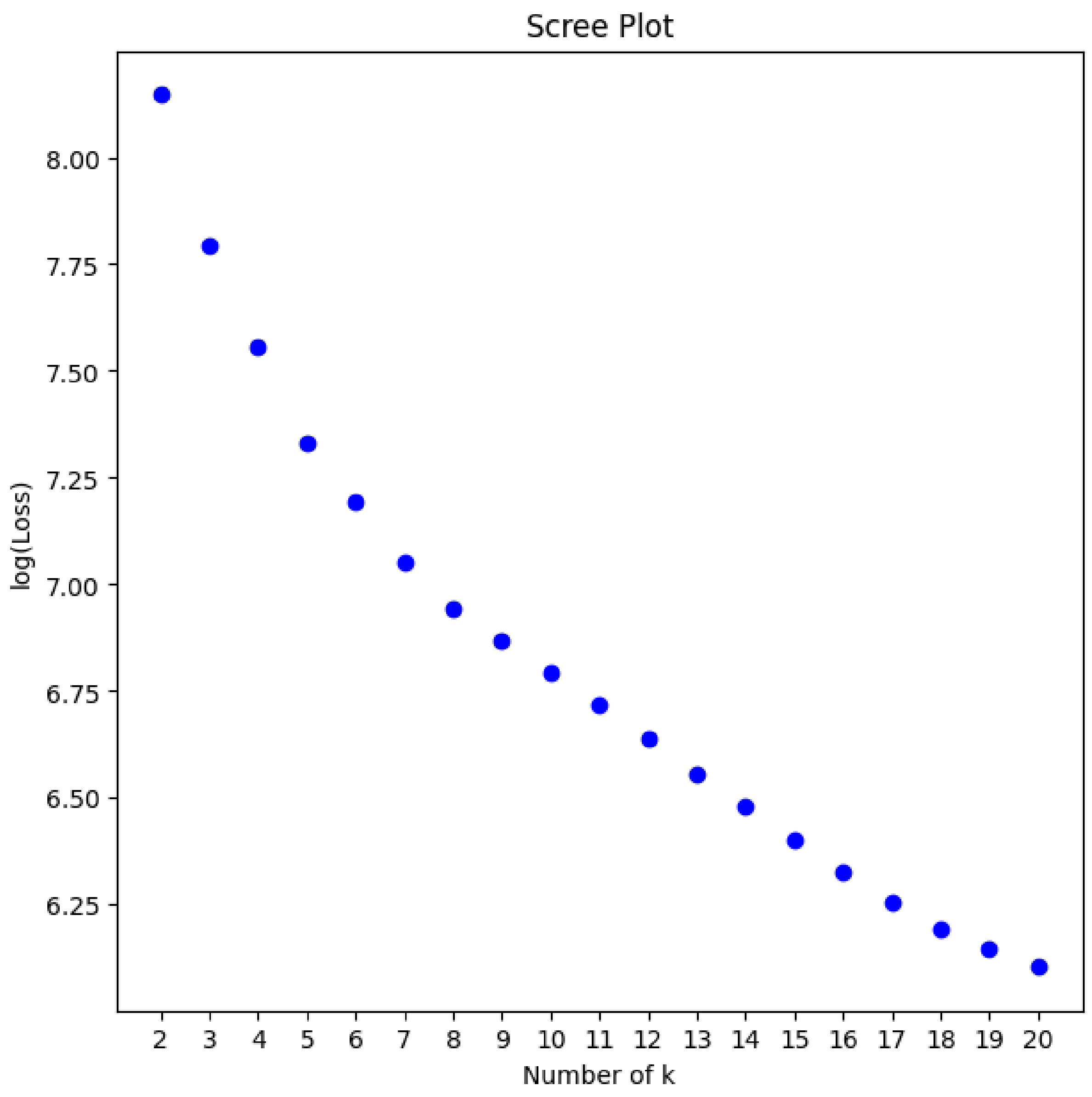


The minimum loss is 3453, with both k-means++ and the "classical" algorithm.
I used Normalized Mutual Information (NMI), a normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation).
In the notebook I also tried to use the scikit-learn kmeans algorithm as a test, and it returns me the same results, both graphically and numerically.

NMI for kmeans "handmade": 0.01099
NMI for kmeans made by scikit-learn: 0.01064

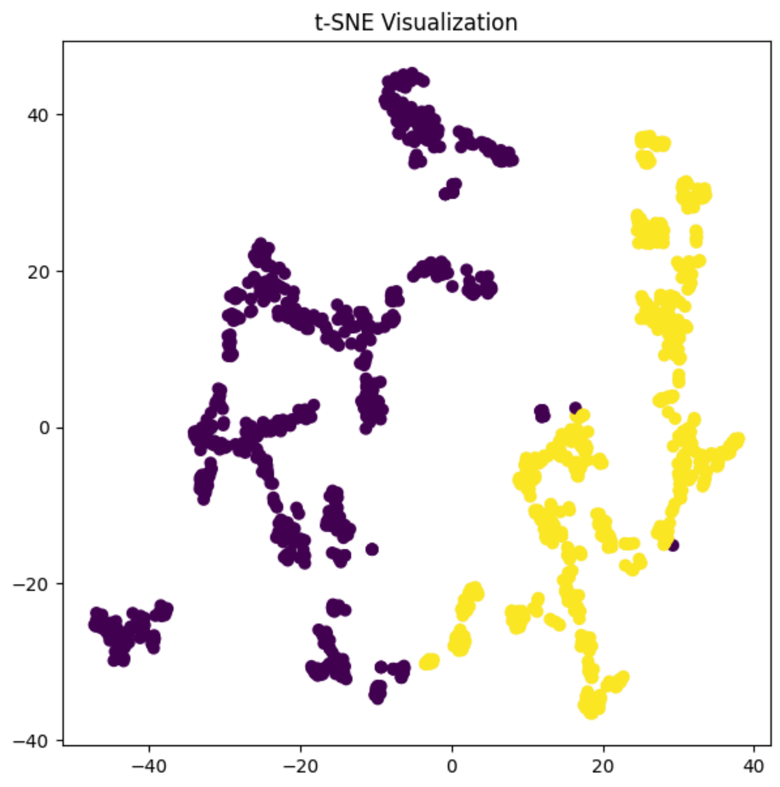
In our case, since NMI is equal to 0.01, we can assert that the agreement of the label assigned by k-means with the true labels is low.

## Scree plot



Here the scree plot provides useful information about the “optimal” value for the number of k (clusters) to use in kmeans. As we can notice from the plot above, using more than 2 or 3 clusters does not provide a substantial improvement under the aspect of the loss for kmeans.

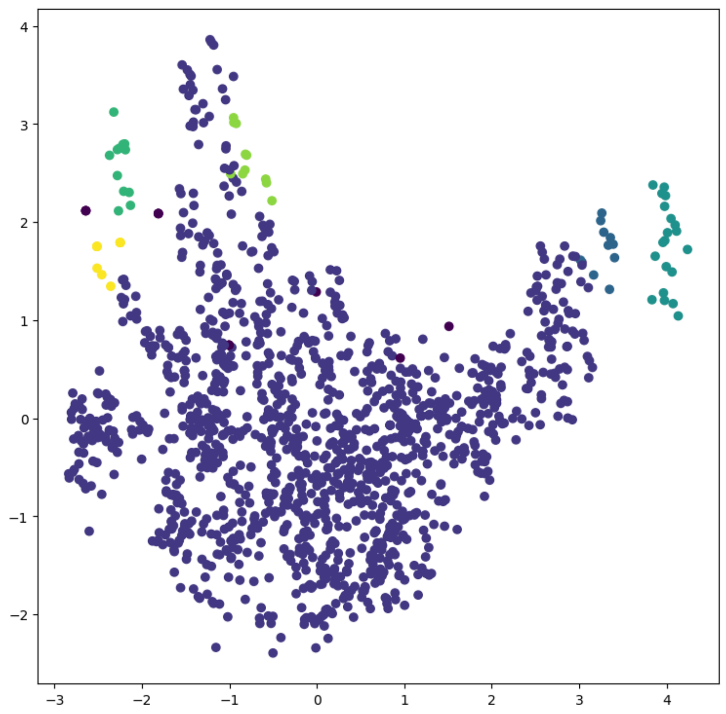
## t-SNE

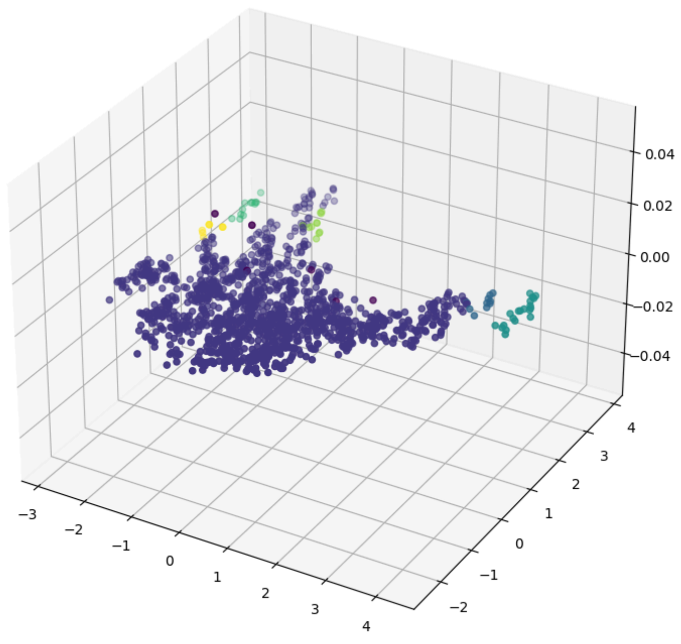


Then I applied t-SNE in the clustering process and also here I obtained two distinct clusters except for a small number of points, but overall the result provides, once again, a clue about the structure of the data.
Usually perplexity is set between 5 and 50, but after several trials with different values, the result obtained using perplexity = 30 is the one I chose.

## DBSCAN

I tried to use all the epsilon values in the range [0.1, 10.1] with a step of 0.5 and the value which returns the highest NMI is eps = 0.6.
Despite that I do not obtain just two clusters, but 7 of them.
The NMI (Normalized Mutual Information) generated for DBSCAN is **0.08**





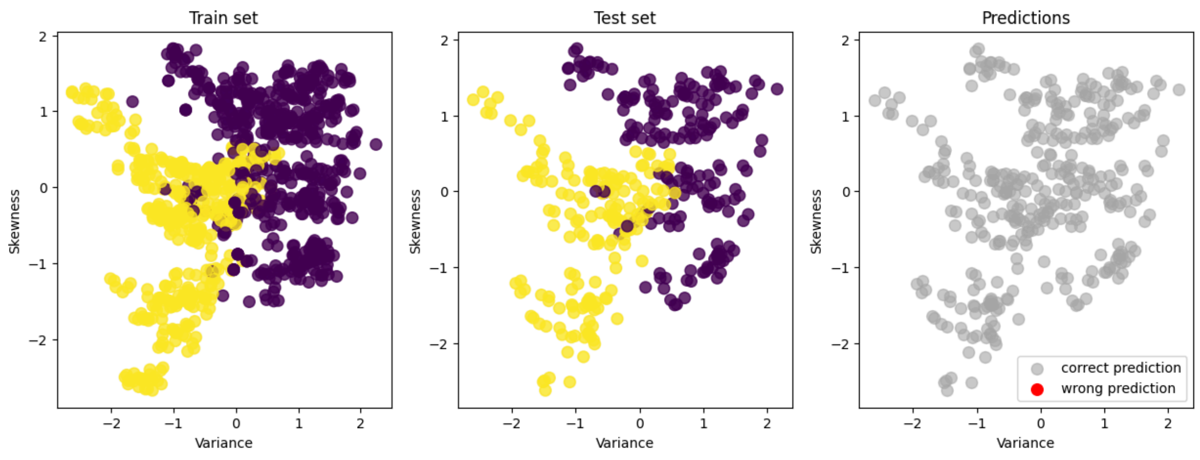
## 3. Supervised learning

The total number of observations is 1372, and following the instructions I split them into a train set and a test set, with 1000 and 372 observations respectively.

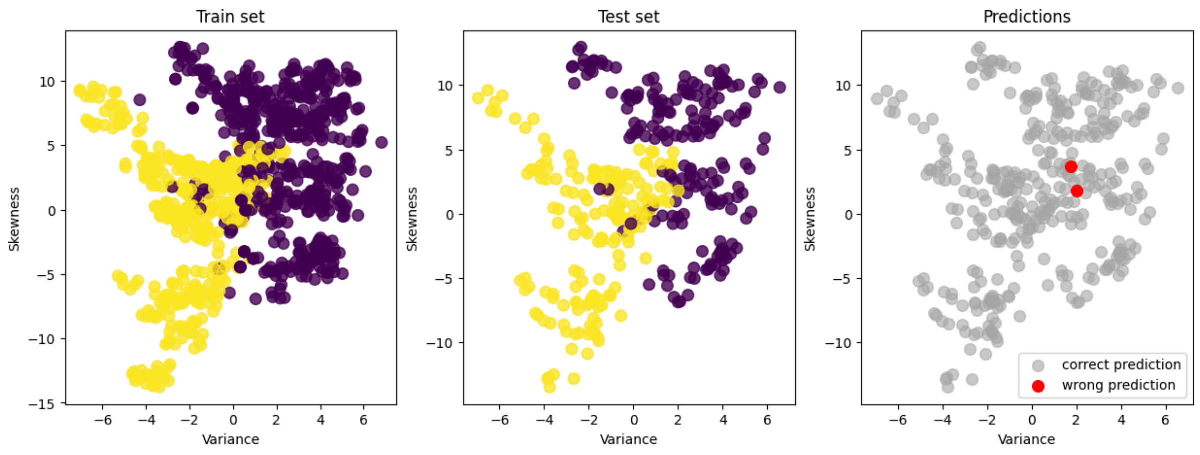
## Logistic regression

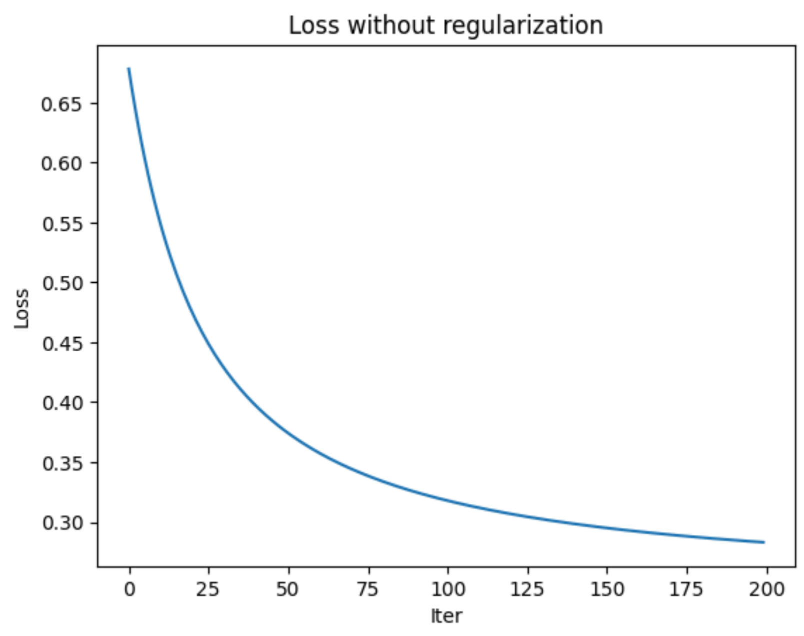
In the logistic regression part firstly I tried without regularization to see how many correct predictions the model supplies.
After that I repeated the process with L1 and L2 regularizations.

### Without regularization



As we can see we got 0 wrong predictions, compared to the ground truth labels.
Just for make a comparison, I also tried to print the same 3 plots without scaling the features, and this is what I obtained:



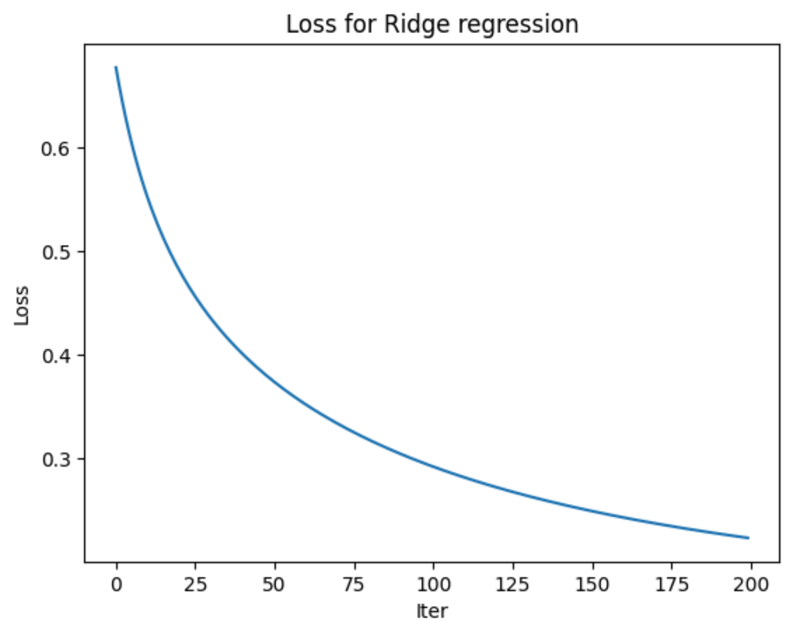


The loss is good because after 25 iterations it is equal to 0.45 approximately.

## K-fold Cross Validation

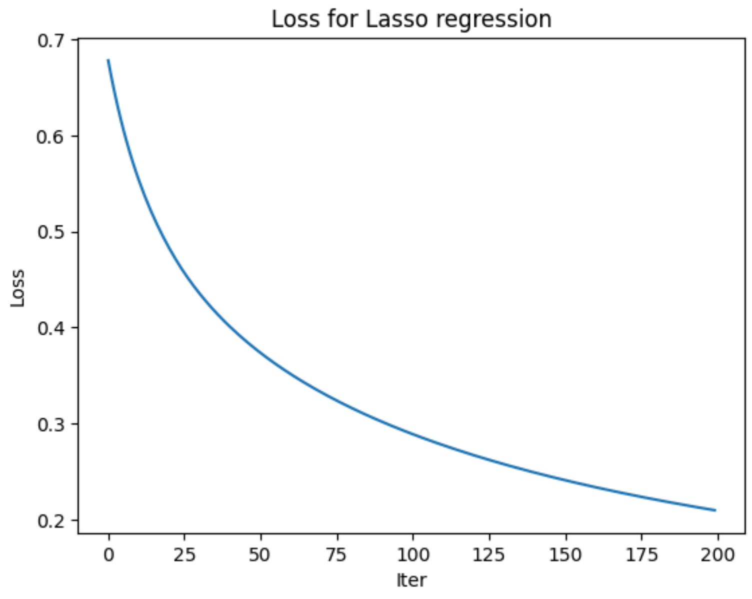
Using k = 5, k-fold cross validation provided the best lambda between 50 values generated in the interval [0, 1], but after some trials I decided to not use the result furnished and use smaller values instead.

## Ridge regression



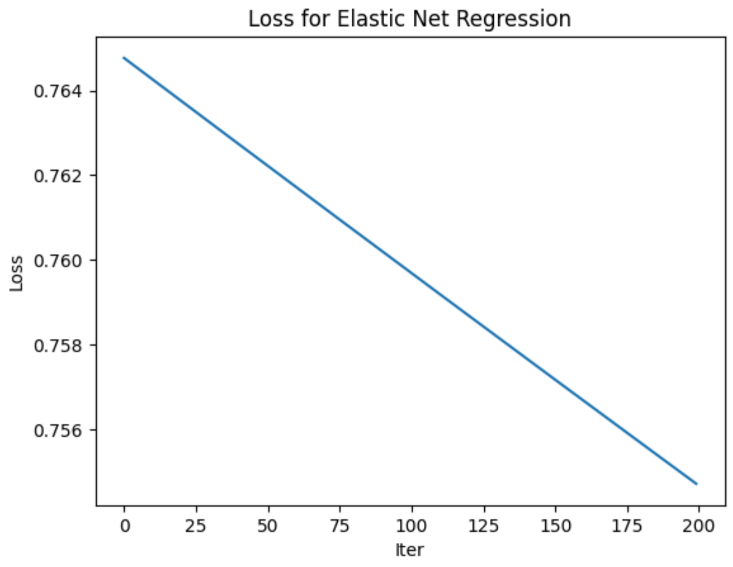
With the L2 regularization I do not get improvements under the aspect of the loss, indeed after 25 iterations the loss has not changed compared with the unregularized logistic regression.

## Lasso regression



The loss function with L1 regularization is exactly the same as the one I got previously.

## Elastic Net regression



Combining L1 and L2 regularization I had worse results for what the loss function concerns, and this can happen when variables have not a strong correlation.

## Accuracy

### Without regularization

Train set accuracy: 97.90%
Test set accuracy: 100%

### L2 regularization (Ridge)

Train set accuracy: 97.10%
Test set accuracy: 99.19%

### L1 regularization (Lasso)

Train set accuracy: 56.80%
Test set accuracy: 52.15%

### L1&L2 regularization (Elastic Net)

Train set accuracy: 56.80%
Test set accuracy: 52.15%

Regarding the accuracy, the model without regularization and the L2 regularized one are the two which produce the best outcomes in terms of correctness of the predictions compared to the ground truth labels.
Furthermore, both Lasso and Elastic Net produce only prediction values equal to zero, for this reason the accuracy is not as good as the one produced from the other two models.

## Model assessment

### With no regularization

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00       194
           0       1.00      1.00      1.00       178

    accuracy                           1.00       372
   macro avg       1.00      1.00      1.00       372
weighted avg       1.00      1.00      1.00       372
```

With L2 regularization (Ridge)

```
              precision    recall  f1-score   support

           1       1.00      0.98      0.99       194
           0       0.98      1.00      0.99       178

    accuracy                           0.99       372
   macro avg       0.99      0.99      0.99       372
weighted avg       0.99      0.99      0.99       372
```

With L1 regularization (Lasso)

```
              precision    recall  f1-score   support

           1       0.52      1.00      0.69       194
           0       0.00      0.00      0.00       178

    accuracy                           0.52       372
   macro avg       0.26      0.50      0.34       372
weighted avg       0.27      0.52      0.36       372
```

With L2 and L1 regularization (Elastic Net)

```
              precision    recall  f1-score   support

           1       0.52      1.00      0.69       194
           0       0.00      0.00      0.00       178

    accuracy                           0.52       372
   macro avg       0.26      0.50      0.34       372
weighted avg       0.27      0.52      0.36       372
```

The **precision** consists in the Positive Predict Value (PPV), in other words it is the number of True Positives (true value = 1 and predicted value = 1) divided by the total number of positive predictions, i.e. the sum of true positives and false positives (true value = 0 and predicted value = 1).
On the other hand the **recall**, also known as True Positive Rate (TPR), is the number of true positives divided by the sum of true positives and false negatives, i.e the sum of all the true values equal to one.
At last the **f1-score** is calculated by dividing the product of recall and precision by the sum of these two.

Again, the unregularized model and the L2 regularized one are the best two considering that they furnish values which are really close to the maximum possible for precision, recall and f1-score (the model without regularization gets the maximum value for all the three measures of quality, and this confirms what we saw in the plot of the predictions).

## Decision tree (using the ID3 algorithm)

Using this algorithm I get an accuracy equal to **0.954** and having a 95% of accuracy means that our decision tree is making predictions very precisely on the test set.

## Naive Bayes

The accuracy I get with the **Naive Bayes** is 0.5188, which means that one label every two is predicted correctly, but with continuous features it is also possible to try with the **Gaussian Naive Bayes**, and this is what I did.
The accuracy generated by applying that algorithm is 0.177, so substantially lower than the one obtained with the original algorithm.
This can happen when the continuous variables are not distributed like a Gaussian, and the kernel density estimation made in the data pretreatment confirms what has just been said.

## K-Nearest Neighbors

The accuracy achieved with the KNN is 1.0 (**100%**), which indicates that the K-Nearest Neighbors model correctly predicts all the instances of the target (class).
This is the best result obtained considering all the algorithms applied before.