Università degli studi di Trieste

Introduzione al Machine Learning

"Challenge one"

Martinelli Davide

March 10, 2024

0: Introduzione

Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images. These features are:

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer)

We have a binary classification problem. The assignment can be divided in several parts:

1. Load the data and pretreatment.
2. Data exploring by Unsupervised Learning techniques.
3. Construction of several models of Supervised Learning.

1: Data pretreatment

For the first part of the challenge, I focused on the bivariate analysis of the dataset. But before doing anything, I made sure that the dataset did not contain any missing values.

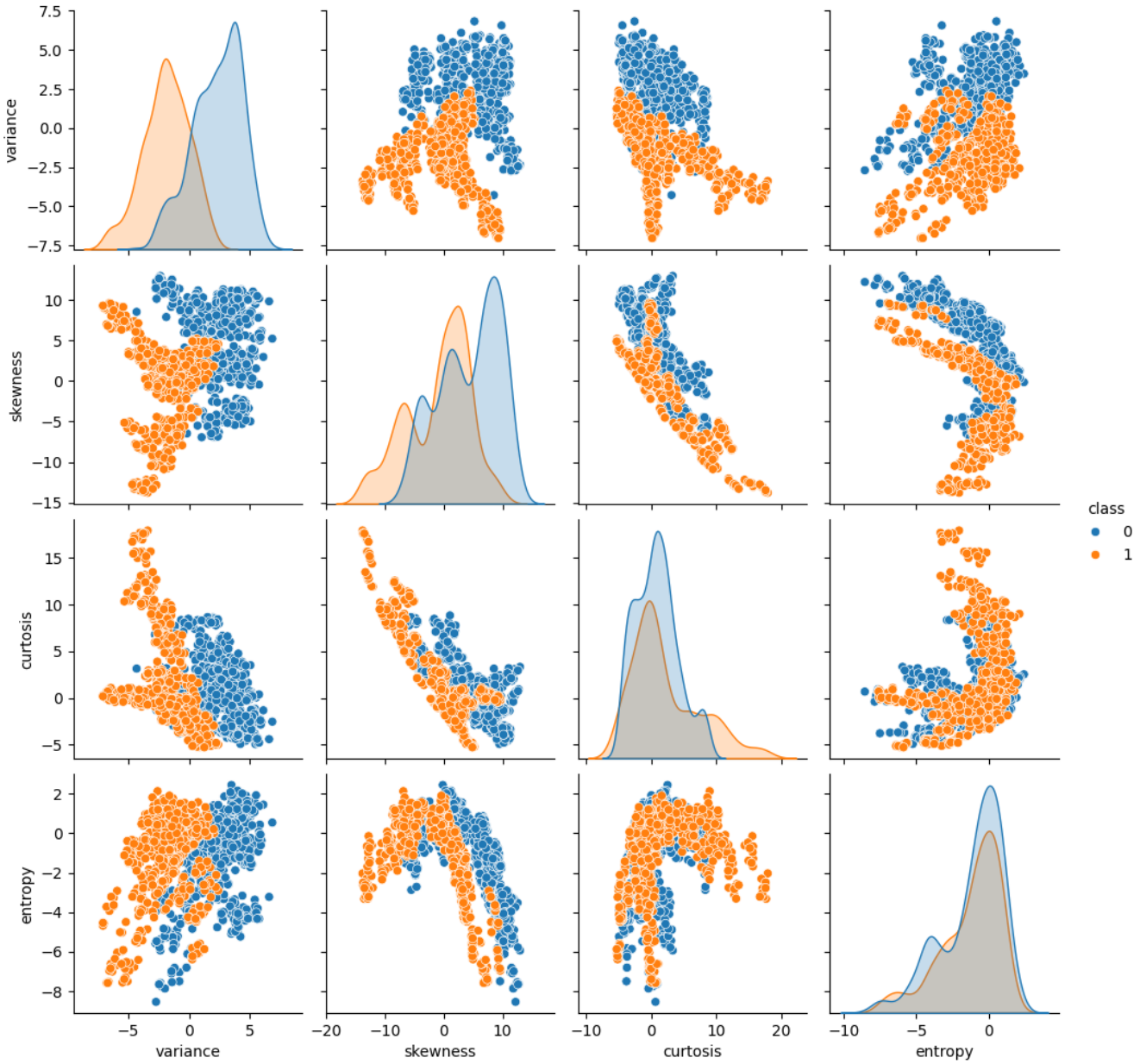
	variance	skewness	curtosis	entropy	class
count	1372.000000	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.433735	1.922353	1.397627	-1.191657	0.444606
std	2.842763	5.869047	4.310030	2.101013	0.497103
min	-7.042100	-13.773100	-5.286100	-8.548200	0.000000
25%	-1.773000	-1.708200	-1.574975	-2.413450	0.000000
50%	0.496180	2.319650	0.616630	-0.586650	0.000000
75%	2.821475	6.814625	3.179250	0.394810	1.000000
max	6.824800	12.951600	17.927400	2.449500	1.000000

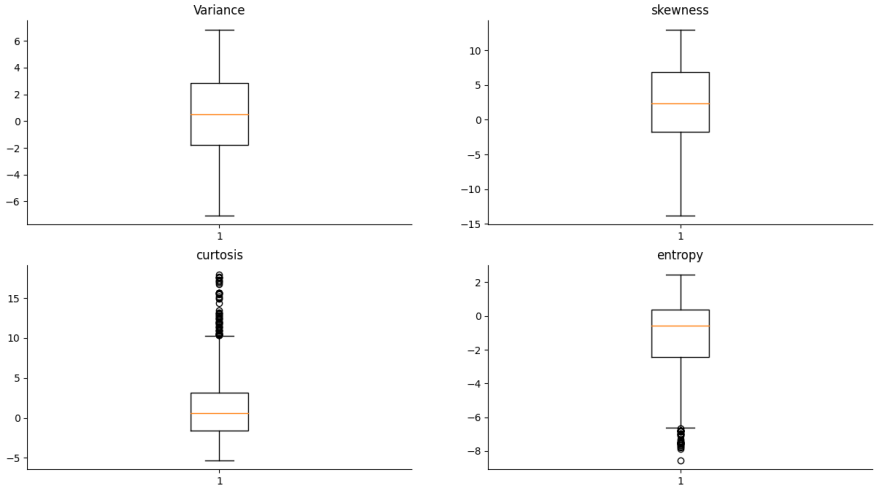
Before proceeding to plot the features, I used the describe() function, which automatically calculates some of the most important values for preliminary analysis. In particular: mean, standard deviation, minimum, maximum, first, second, and third quartiles. It is observed that the features have values very close to each other; in particular, the 'skewness' feature has a very wide range of values, from -12 to +12, unlike all the other features.

Therefore, it will be necessary to standardize the data to avoid 'skewness' to carry more weight than the others during model creation.

After that, I used the pairplot function from the seaborn library. What it does is create a grid of scatter plots and histograms to examine the bivariate relationships between variables in a dataset. This visualization is useful to get an overview of the relationships between all pairs of variables. But above all, it significantly reduces the time I would have spent on this part of the analysis.
From the graphs, it is immediately evident that there is no linear correlation between any of the

features, except for 'curtosis' - 'skewness'; while between 'skewness' - 'entropy' and 'curtosis' - 'entropy,' it can be observed that the points approach each other on a parabola with concavity facing downwards. Regarding the rest of the points, they tend to accumulate in a central area that distinguishes genuine banknotes from false ones, with the rest of the values approaching the outer zones.





I decided to use boxplots as well to allow a graphical visualization of the values obtained from describe(), making it more immediately understandable than simple values. It can be observed that 'curtosis' and 'entropy' have many extreme values compared to the central mass, which is not immediately apparent from the raw data.

As for the labels, the number of false banknotes is greater than the genuine ones (55% vs 45%).

Here concludes the bivariate analysis of the dataset. What remains is to split the data into training and testing sets and standardize the labels. I won't go into describing the code, as it is all explained in the attached notebook. The only thing I emphasize is that by standardizing the data and reusing pairplot, the cluster of points becomes more concentrated, as the range in which the points were located has narrowed.

2: Unsupervised learning

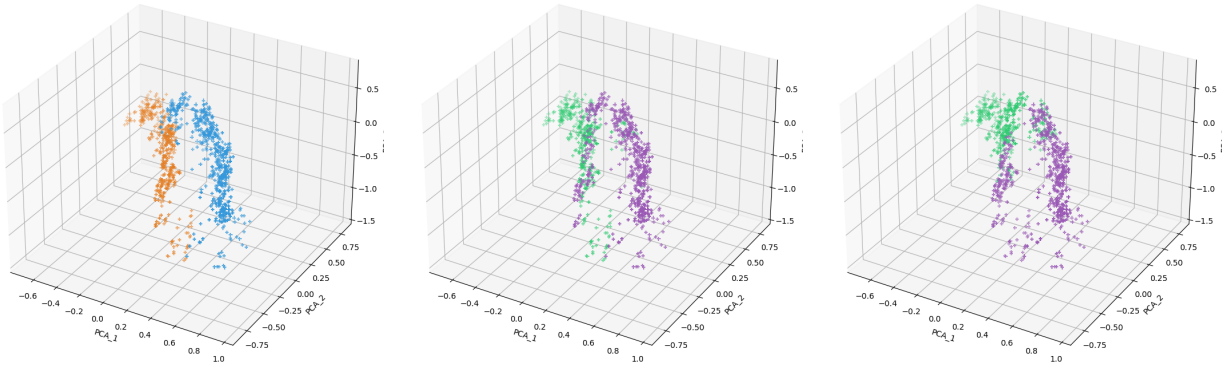
The second part of the project is dedicated to unsupervised learning. In particular, it will be divided into 4 parts: PCA, K-means, t-SNE, and DBSCAN.

Since the number of features is greater than the number of dimensions we can comprehend (¿3), we will use PCA to perform dimensionality reduction (in this case to 2 or 3) to allow better visualization and understanding of the entire dataset. How is this done? By identifying the most relevant variables. The first variable explains 55% of the data, the second explains 31%, the third 10%, and the fourth only 4%. In other words, the first two variables can explain 86% of the dataset, while a third dimension brings us to 96%. This way, representing a 3D plot will allow us a full understanding of the data.

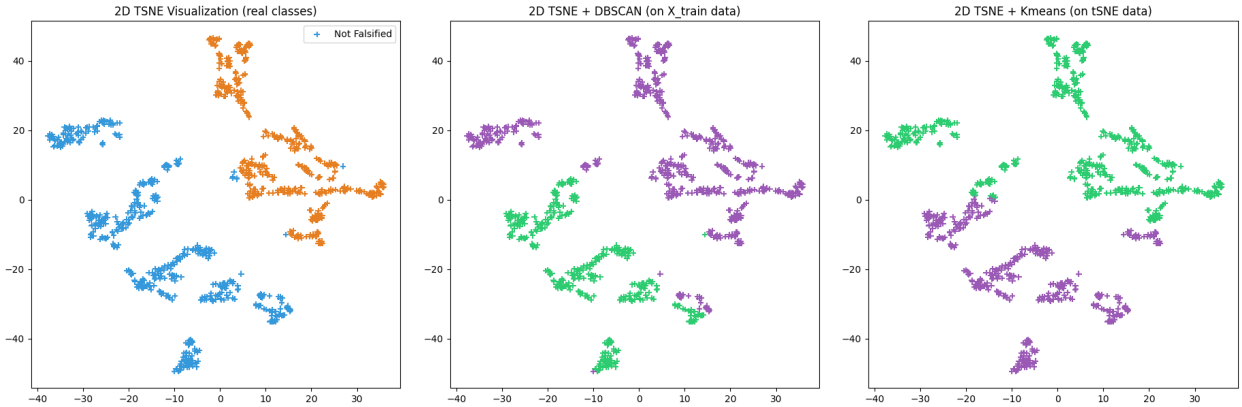
After calculating the new dataset reduced to two (or three) dimensions, we use it to fit k-means. The K-Means algorithm is an unsupervised clustering method that divides a dataset into groups (clusters) based on the similarity between examples. The main goal is to group objects so that objects within the same cluster are more similar to each other than to those in other clusters. In our case, we want to obtain 2 clusters of data. We apply it to both the data extracted from PCA and the data from X_train_std (standardized).

Whether we use PCA data or X_train_std data, k-means has some difficulty identifying the clusters. Since the algorithm is based on the distance from the points, it is not surprising that it struggles to identify points in such a dense data cloud.

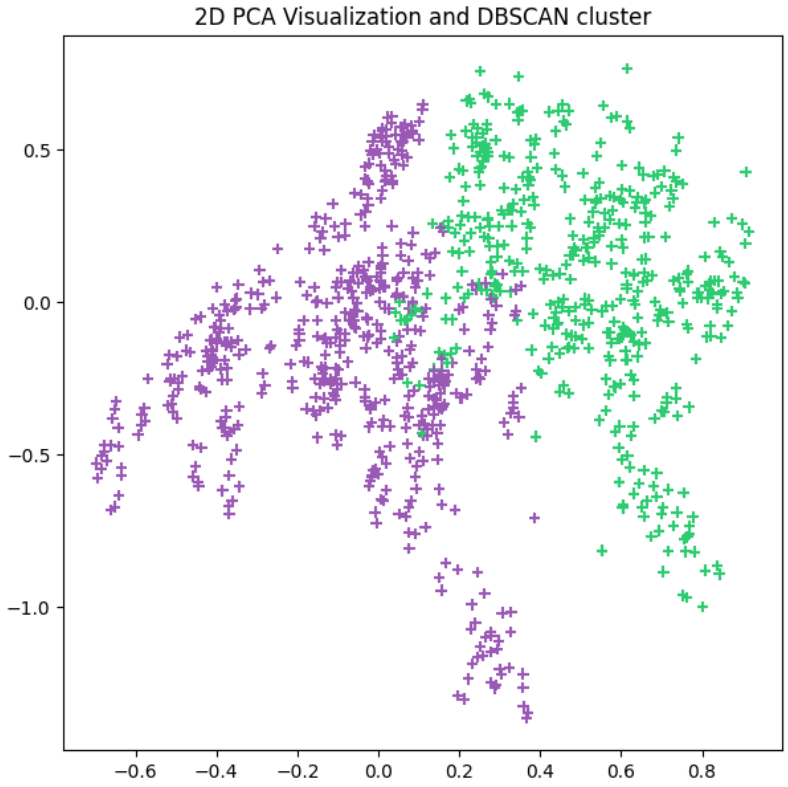
We also try to plot in 3D to obtain an additional visualization of the data.



Now let's focus on t-SNE, always for the visualization of multi-dimensional datasets. In this case, we use the DBSCAN clustering method.



In this case, the use of the k-means clustering method does not respect the actual groupings. On the other hand, DBSCAN performs better, although not optimal, on this representation of the data.

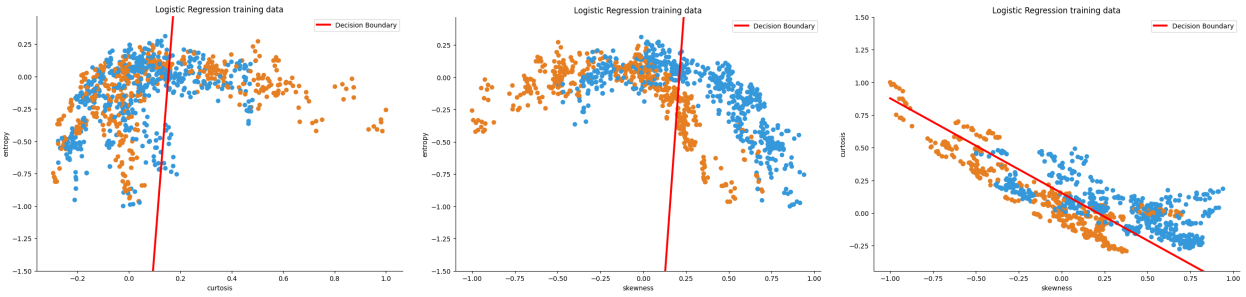


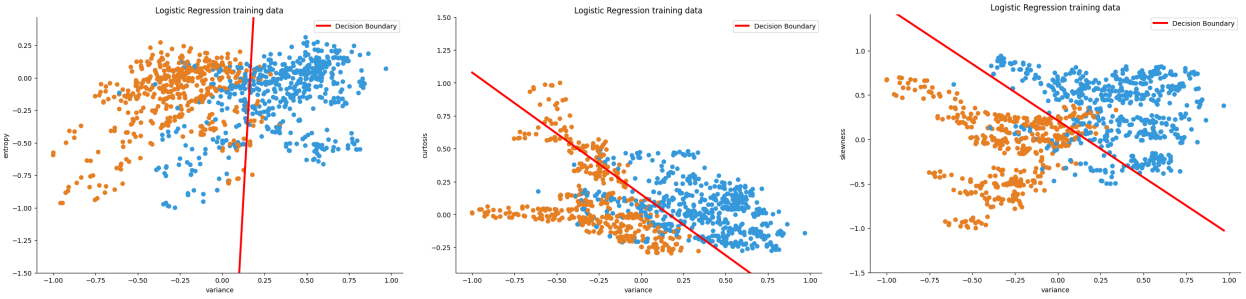
The result obtained through PCA Projection and Coloring/clustering via the DBSCAN method is undoubtedly the best, due to the remarkable similarity with the PCA graph with real classes. The clusters that are identified are indeed very similar to the real groups, unlike before.

3: Supervised Learning

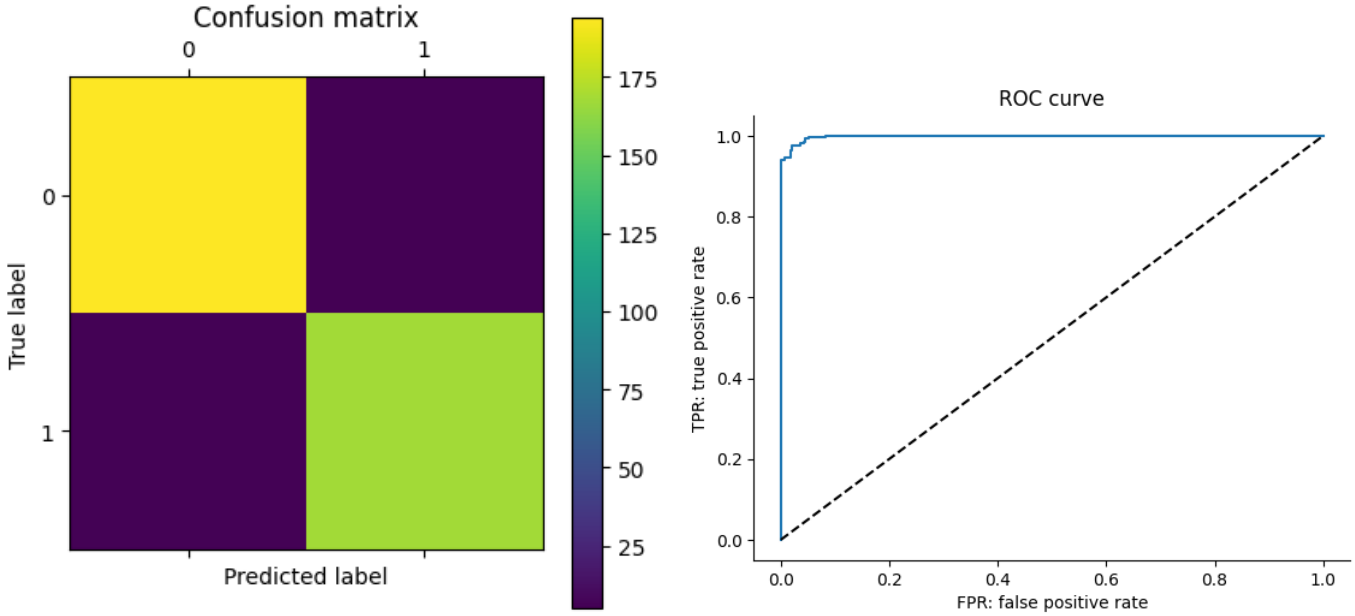
The third part of the project is also divided into 4 sub-parts. Logistic regression (with various regularizations), decision tree (with ID3), naive Bayesian, and K-NN will be used. Again, as in the first two parts, I won't delve too much into the code explanation but will go straight to the results obtained.

Starting with logistic regression, I used the function provided by the sklearn library. The model achieved excellent results, with an accuracy almost at 100%.
Below are all the features of the dataset with the separation line calculated using the model parameters.

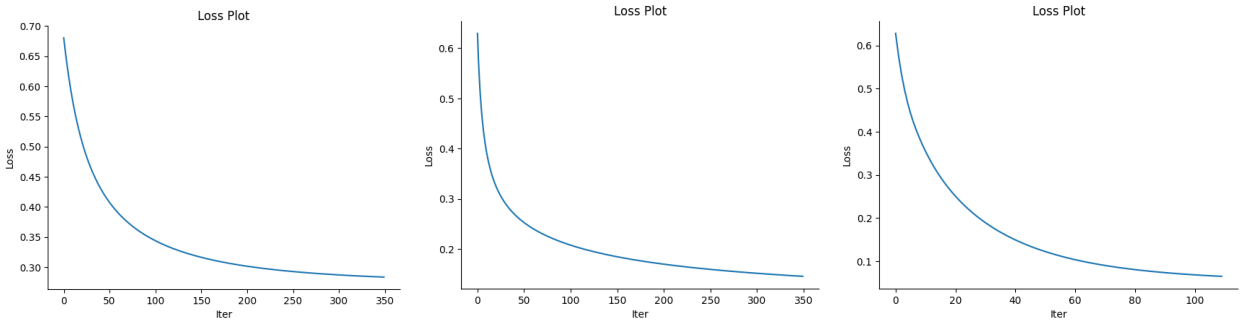




Additionally, the plot of the ratio between positive or negative values and the ROC curve.



Regarding regularizations, I implemented Ridge, Lasso, and Elastic Net by hand, then calculating the loss for each model and accuracy. Concerning the loss, Elastic Net is the one that achieves better results, with LASSO and RIDGE following suit.



Below is the table with the accuracy of each model.

1. Logistic Regression (no regularization): 0.968
2. Logistic Regression (with Ridge): 0.873
3. Logistic Regression (with Lasso): 0.924
4. Logistic Regression (with Elastic Net): 0.876

I won't go into much detail about the next three algorithms, both because there's little to explain and because the implementation is explained in the attached notebook. The three algorithms are

Decision Tree, Naive Bayes, and K-NN.

The only one of these three that performs optimally is Decision Tree with an accuracy of 0.978, while Naive Bayes approaches a good 0.839 and K-NN with a precision of 0.53 (with k ranging from 1 to 15).

Here the tables with accuracy, precision, recall and F1 score of all models.

Model	Accuracy	Precision	Recall	F1 score
Sklearn LR no Reg.	0.98656	0.98656	0.98656	0.98656
LR Ridge	0.97581	0.99394	0.95349	0.97329
LR Lasso	0.97312	0.99394	0.94798	0.97041
LR ElasticNet	0.97312	0.99394	0.94798	0.97041
Tree	0.97855	0.93939	0.85165	0.89337
Naive Bayes	0.83952	0.84263	0.85923	0.84238
K-NN (1 to 15)	0.53763	0.46061	0.47799	0.46914

In conclusion, we can consider classification models based on logistic regression and decision trees to have good accuracy, precision, recall, and F1-score, which are also confirmed graphically through two-dimensional and three-dimensional plots. The subsequently considered models, however, show more disappointing results. As considered earlier, especially in the case of k-NN, this may be closely related to the metric used to measure distances between statistical units.