Politecnico di Torino

DRUG CONSUMPTION TENDENCY CLASSIFICATION

SCORCA FRANCESCO

SUMMARY

1 INTRODUCTION

The following work consists of an in-depth analysis of the "Drug consumption dataset": the objective is to infer an individual tendency to assume LSD, basing on psychological, social, individual, environmental, and economic factors.

The study envisioned the development of a preprocessing pipeline consisting of

1. one-hot encoding
2. possible outliers removal
3. possible transformation of the dataset through PCA or FDA
4. SMOTE resampling
5. k-fold cross-validation for hyper-parameters selection.

We tested 4 machine learning algorithms, namely KNN, Decision Tree, Random Forest, and SVM.

The evaluation tools employed are

- confusion matrices
- f1-score, recall, precision, accuracy
- ROC curves.

The best performing model (SVM) reached an f1-score of 0.70, and results showed that, for the proposed classification task, outliers removal and data transformation were secondary with respect to hyper-parameters tuning,

2 DATASET EXPLORATION

2.1 DESCRIPTION

The data set was collected by an anonymous online survey methodology by Elaine Fehrman yielding 2051 respondents. It contains information on the consumption of 18 central nervous system psychoactive drugs including alcohol, amphetamines, amyl nitrite, benzodiazepines, cannabis, chocolate, cocaine, caffeine, crack, ecstasy, heroin, ketamine, legal highs, LSD, methadone, magic mushrooms, nicotine, and Volatile Substance Abuse (VSA), and one fictitious drug (Semeron) which was introduced to identify over-claimers. Participants wrote for each drug if they never used this drug (CL0), used it over a decade ago (CL1), or in the last decade (CL2), year (CL3), month (CL4), week(CL5), day (CL 6).

The total number of instances is 1885, characterized by 12 attributes, with no missing values to handle.

There are 7 numeric features obtained using the Revised NEO Five-Factor Inventory (NEO-FFI-R), the Barratt Impulsiveness Scale version 11 (BIS-11), and the Impulsivity Sensation-Seeking scale (ImpSS):

- **Neuroticism** (N), representing an individual tendency of feeling psychological distress;
- **Extroversion** (E), a dimension underlying a broad group of traits, such as sociability, activity, and the tendency to feel good emotions;

- **Openness to Experience** (O), accounting for different tendencies, like intellectual curiosity, sensitivity to arts, emotional complexity, behavioral flexibility;
- **Agreeableness** (A), another relational trait. High-A individuals are generally cooperative, trusting, sympathetic, while low-A individuals tend to be cynical and antagonistic;
- **Conscientiousness** (C), accounting for organizational traits;
- **Barratt Impulsiveness Scale** (BIS-11), measuring the behavioral construct of impulsiveness, is based on three subscales: "motor impulsiveness", "attentional impulsiveness", and "non-planning". The "motor" aspect reflects acting without thinking, the "attentional" component poor concentration and thought intrusions, and the "non-planning" a lack of consideration for consequences. A unique scalar indicator is obtained by aggregating the results;
- **Impulsiveness Sensation-Seeking** (ImpSS), combining the traits of impulsivity and sensation-seeking, but generally regarded as a measure of a general sensation-seeking trait.

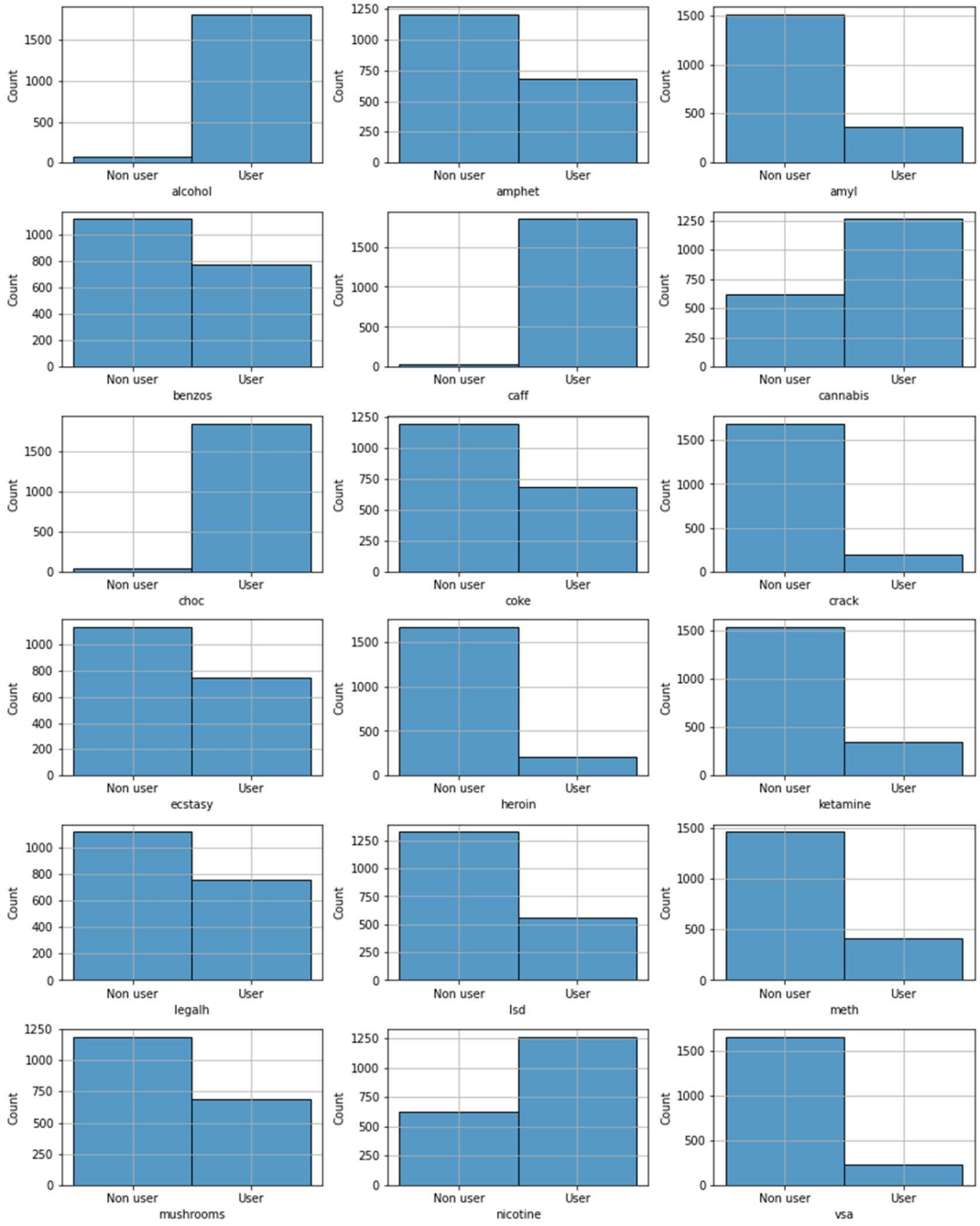
The remaining 5 attributes are categorical:

- **Age**, optimizing anonymity, persons reported their age band, rather than their exact age: 18-24 years, 25-34, 35-44, 45-54, 55-64, and over 65;
- **Gender**;
- **Level of education**;
- **Ethnicity**;
- **Country of residence**.

The distributions of the number of users for each drug are presented in the following figure. Most of them have an exponential-like shape but several have bimodal-like distributions. The distributions of the number of users for the three legal drugs have a maximum at "Used in last day" or "Used in last week". The distribution of the number of nicotine users (smokers) has three maxima: "Used in last day" for smokers, "Used in last decade" for smokers who broke smoking, and "Never used". All illegal drug users distributions have a maximum in the category "Never used". However, the distribution of cannabis users has two maxima: the main in the category "Used in last day" and the second in the category "Never used".

An in-depth study can be conducted by choosing as target variable one of the drugs or a group of them, referred to as “pleiads”. Furthermore, data does not contain a definition of users and non-users groups. Formally only a participant in the class “Never used” can be called a non-user, but it is not a significant definition because a participant who used a drug more than a decade ago cannot be considered a drug user for most applications. Consequently, also these individuals have been assigned to the new class “Non-user”, assigning the remaining labels to the “User” class. This approach transforms the problem into a binary classification one, named in the referring literature as “decade-based”.

The new resulting distributions are the following.



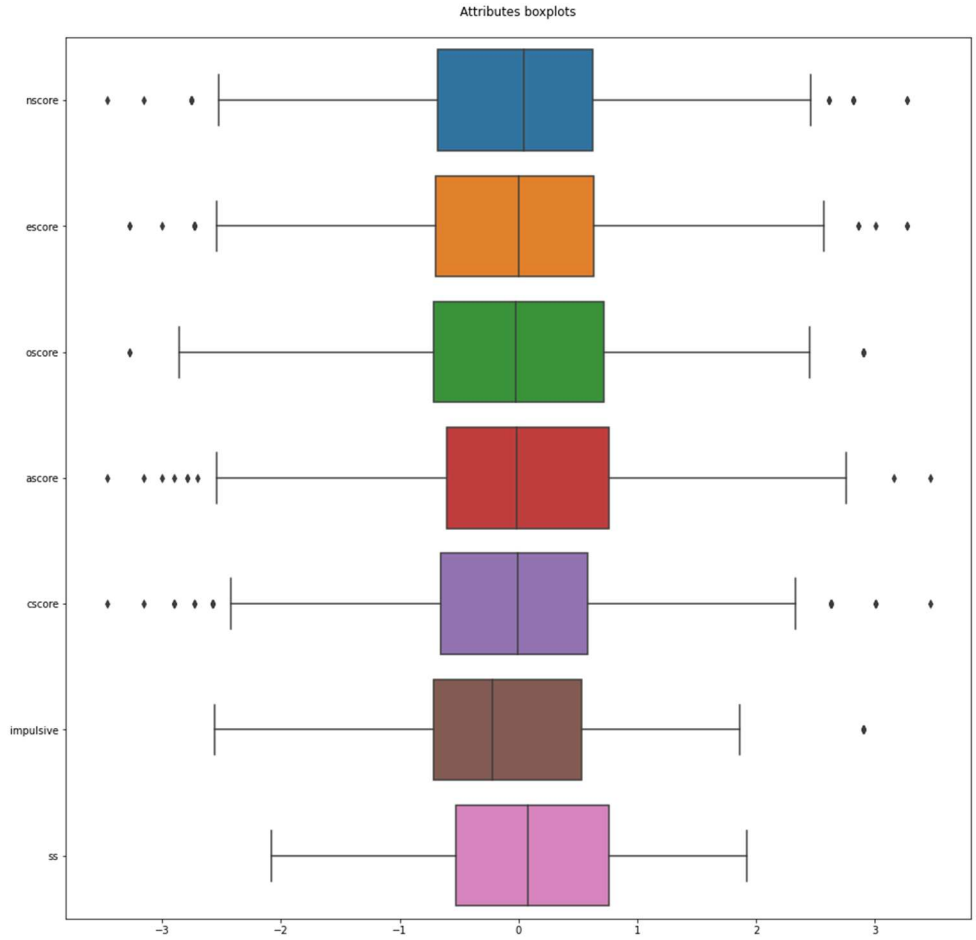
It is also possible to visualize how the different attributes are distributed:

The following study will focus on the consumption of only one drug, namely "LSD". This allows for the visualization of the distributions of the attributes, with respect to the target variable.

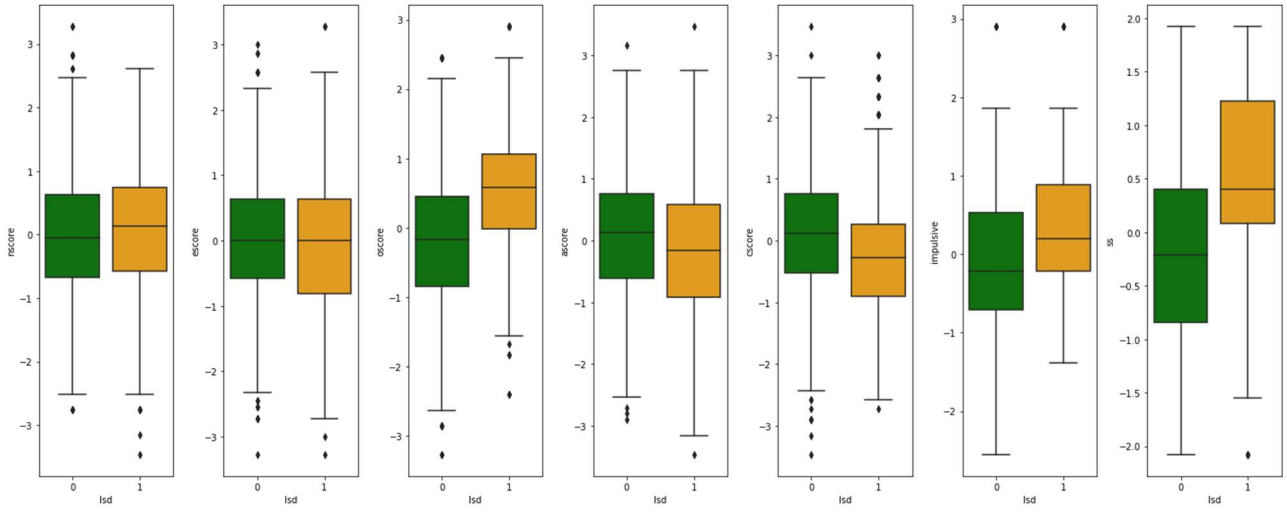
2.2 BOXPLOTS

This kind of plot shows the values from the first to the third quartile of the distribution along with extreme values. The "whiskers" extend to points that lie within 1.5 IQRs (Inter-Quartile Range, defined as the difference between 75th and 25th percentile) of the lower and upper quartile, and then observations that fall outside this range are displayed independently.

The boxplots of the quantitive features are the following.



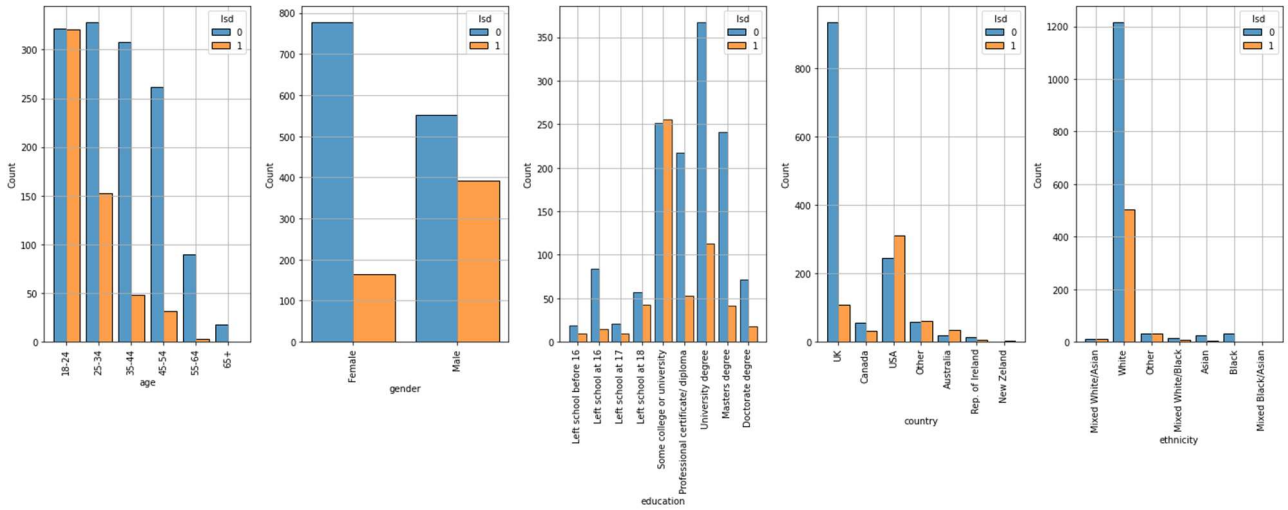
It is possible to infer that numerical predictors have been normalized, as it was possible to notice also from their feature counts in the previous paragraph. Furthermore, having chosen the target variable, it is possible to show the boxplots for users (in the definition we gave in the previous paragraph) and non-users.



We can observe that in the collected data drug users present on average a higher openness to experiences, impulsiveness and sensation-seeking tendency. Regarding these last two attributes, their distribution is also more concentrated in the case of users.

2.3 OCCURRENCES COUNTS

For the categorical features, the counts of the instances (with respect to the target) are plotted.



We can observe how the frequency ratio between users and non-users tends to decrease with age, and that it loosely increases with education level until “some college or university”, and decreases after such a value.

2.4 FEATURES CORRELATION

The "Pearson's correlation coefficient" is the most common measure of dependence, suggesting the degree to which a pair of variables are linearly related. It is obtained by taking the ratio of the covariance of two numerical variables, normalized by the product of their standard deviations:

$$Corr(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

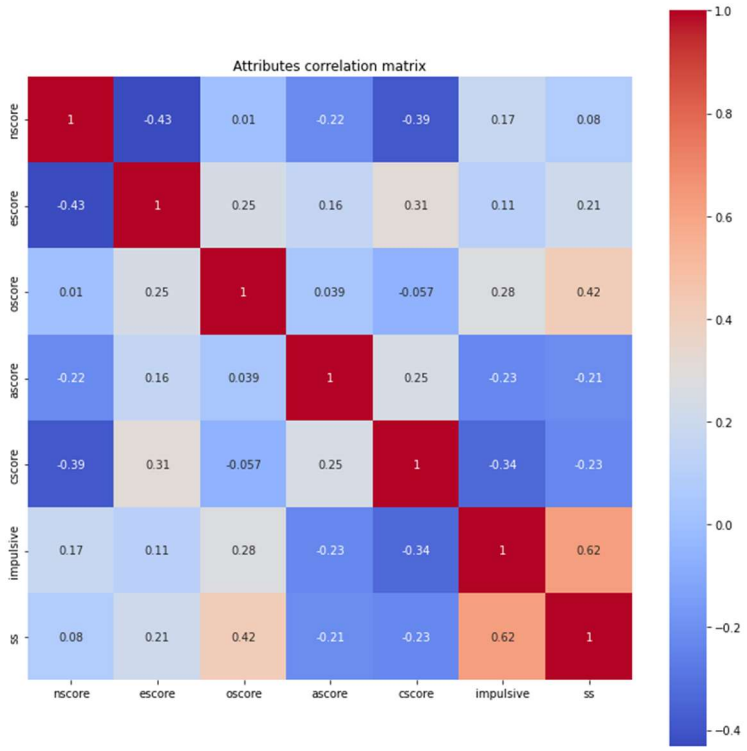
Having n instances the sample estimates of the parameters involved are the following:

$$Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_X)(y_i - \mu_Y)$$

$$\sigma_X = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_X)^2}$$

$$\sigma_Y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \mu_Y)^2}$$

The following is the feature correlation matrix, presenting the pairwise correlation of the numerical columns.



We can highlight a strong positive correlation among impulsiveness, sensation-seeking tendency and openness to experience, as well as a strong negative correlation between neuroticism and the two attributes extroversion and consciousness.

3 DATASET MANAGEMENT

Priorly to any analysis, it is mandatory to split the dataset into training and testing sets, to address the risk of overfitting. In this case, 20% of the original database is used for testing, adopting a stratified split, ensuring that the ratio of positive (and negative) instances is the same in both the splits.

Having done this, the techniques shown in this chapter are applied separately in two stages of the study:

1. on the training sets created with the stratified k-fold technique (Paragraph 3.7), to find the optimal combination of hyper-parameters for the algorithm considered;
2. on the overall training set, before providing it to the tuned algorithms.

3.1 ONE-HOT ENCODING

To provide categorical features to the majority of algorithms it is necessary to convert them into numeric. Although it is possible to assign an arbitrary integer to each value of a feature, the algorithm could infer a natural ordering even when not existing. Because of this, one-hot encoding is exploited in the presented work, consisting in adding a new binary variable for each unique categorical value.

3.2 DATA CLEANING

Outlier detection is a statistical procedure that aims at finding suspicious events or items that are different from the normal form of a dataset. It has drawn considerable interest in the field of data mining and machine learning. There are two general types of outlier detection: global and local. Global outliers fall outside the normal range for an entire dataset, whereas local outliers may fall within the normal range for the entire dataset, but outside the normal range for the surrounding data

points. If the outliers cannot be detected and removed, a machine learning technique applied to the data is likely to be misled and lead to inaccurate outputs.

3.2.1 Isolation Forest

In a data-induced random tree, partitioning of instances is repeated recursively until all instances are isolated. This random partitioning produces noticeable shorter paths for anomalies since (a) the fewer instances of anomalies result in a smaller number of partitions, so shorter paths in a tree structure, and (b) instances with distinguishable attribute values are more likely to be separated in early partitioning. Hence, when a forest of random trees collectively produces shorter path lengths for some particular points they are likely to be anomalies. Thus, one way to detect anomalies is to sort data points according to their path lengths, where the path length *h(x)* of a point *x* is measured by the number of edges *x* traverses in an isolation tree (iTree) from the root node until the traversal is terminated at an external node. Since each partition is randomly generated, individual trees are generated with different sets of partitions, and path lengths are averaged over a number of trees to find the expected path length *E(h(x))*. Since iTrees have an equivalent structure to Binary Search Tree (BST), the estimation of average *h(x)* for external node terminations is the same as the unsuccessful search in BST:

$$c(n) = 2H(n-1) - (2(n-1)/n),$$

whit *n* number of instances, *H(i)* is the harmonic number and it can be estimated by *ln(i)* + 0.5772156649 (Euler's constant). As *c(n)* is the average of *h(x)* given *n*, it is used to normalize *h(x)*. The anomaly score *s* of an instance *x* is defined as

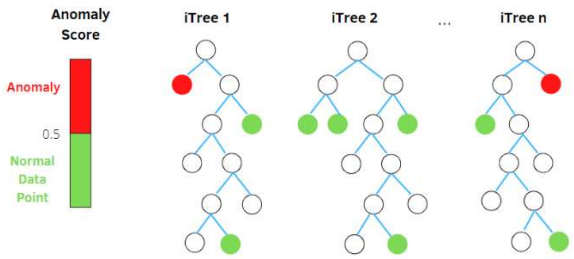
$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

s is monotonic to h(x) and it results that

- when E(h(x)) → c(n), s → 0.5;
- when E(h(x)) → 0, s → 1;
- when E(h(x)) → n − 1, s → 0.

Using the anomaly score *s*, we can make the following assessment:

a) if instances return *s* very close to 1, then they are anomalies;
b) if instances have *s* much smaller than 0.5, then they are quite safe to be regarded as normal instances;
c) if all the instances return *s* ≈ 0.5, then the entire sample does not have any distinct anomaly.



For the implementation, we leave the default parameters, consisting of the exploitation of 100 trees and in the decision of marking as outliers all those with anomaly scores greater than 0.5[1]. After the application of this outliers removal technique, the training set passes from 1508 to 1422 samples.

3.2.2 Local outlier factor (LOF)

The local outlier factor (LOF) is an algorithm that captures the relative degree of isolation; it is outlined here, along with its required definitions.

k-distance of an object p. For any positive integer k, the k-distance of object p denoted as *k-distance(p)* is defined as the distance d(p,o) between p and an object o ∈ D (D data-set) such that

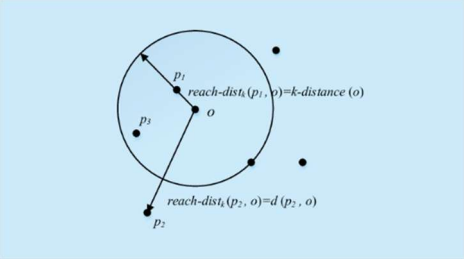
1. for at least k objects o'∈ D \ {p} it holds that d(p, o') ≤ d(p,o);
2. for at most k-1 objects o'∈ D \ {p} it holds that d(p,o') < d(p,o).

k-distance neighborhood of an object p. Given the k-distance of p, the k-distance neighborhood of p contains every object whose distance from p is not greater than the k-distance, i.e. $N_k(p) = \{ q \in D\backslash\{p\} \mid d(p, q) \leq k\text{-}distance(p) \}$. These objects q are called the k-nearest neighbors of p.

Reachability distance. Let k be a natural number. The reachability distance of object p with respect to object o is defined as

$$reachdist_k(p, o) = max\{kdistance(o), d(p, o)\}$$

Thus, if object p is far away from o, then the reachability distance between the two is simply their actual distance. However, if they are "sufficiently" close, the actual distance is replaced by the k-distance of o.



Local reachability density. The local reachability density of p is defined as

$$lrd_k(p) = \frac{1}{\left(\frac{\sum_{o \in N_k(p)} reachdist_k(p, o)}{|N_k(p)|}\right)}$$

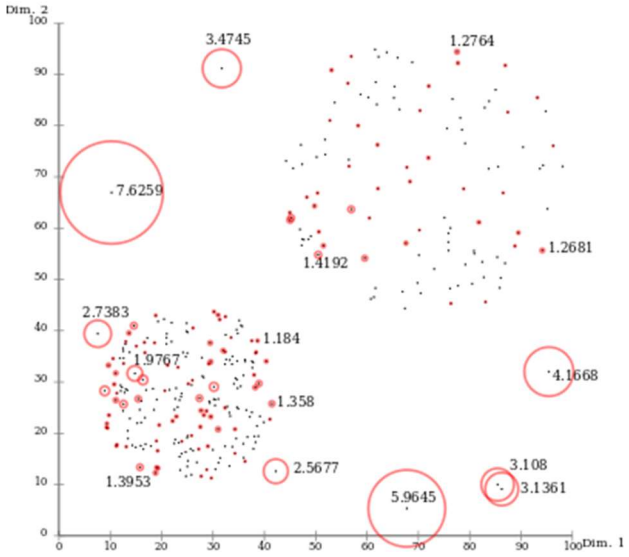
The local reachability density of an object p is the inverse of the average reachability distance based on the k neighbors of p.

Local outlier factor. The (local) outlier factor of p is defined as

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|}$$

[1]The actual default implementation consists in taking the opposite of the anomaly score, subtracting offset = -0.5 and marking as outliers the samples with such a value lower than 0.

The outlier factor of object p captures the degree to which we call p an outlier. It is the average of the ratio of the local reachability density of p's k-nearest neighbors and the one of p. It is easy to see that the lower p's local reachability density is, and the higher the local reachability densities of p's k-nearest neighbors are, the higher is the LOF value of p. Intuitively, this means that if a point is characterized by a low-density neighborhood, but the points in the neighborhood do not, then it will be an outlier (otherwise it won't and we will be facing just low-density data). Conversely, objects deep inside a cluster have LOFs close to 1, and should not be labeled as local outliers.



One of the most important hyperparameters to set is k, and in the literature, it is highlighted how the minimum value to look for is 10, since values below this could label some samples as outliers even if these are generated from a uniform distribution.

After the application of this removal technique with k=10, the training set passes from 1508 to 1504 samples, consequently, we do not test greater values since this would imply having even fewer outliers.

3.2.3 One-Class SVM (ν-SVM)

This method is based on SVM, a binary classifier described in section 4.2.5. In this version, the objective is to capture regions in the input space where the probability density of the data lives. To do this it maps the data into the feature space corresponding to the chosen kernel and separates them from the origin with maximum margin. Thus, the algorithm returns +1 in a "small" region capturing most of the data points and −1 elsewhere. Formally, this results in a quadratic programming minimization function (similar to the original SVM one):

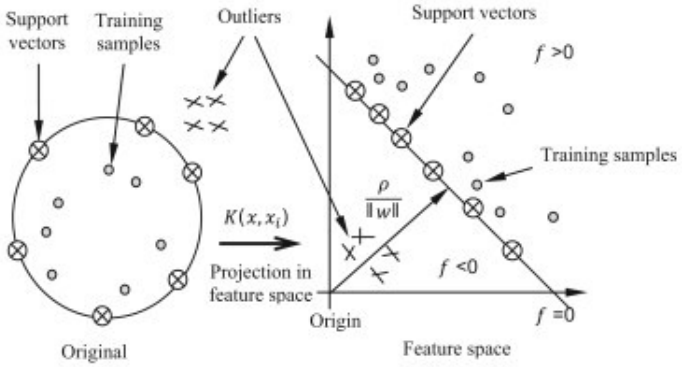
$$\min_{\boldsymbol{w},\rho,\boldsymbol{\xi}} \frac{\|\boldsymbol{w}\|^2}{2} + \frac{1}{\nu m}\sum_{i=1}^{m}\xi_i - \rho,$$

$$\text{s.t.} \quad \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x_i}) \rangle \geq \rho - \xi_i \;\; \forall i \;\in 1, \ldots, m$$

The parameter ν characterizes the solution, by setting an upper bound on the fraction of outliers and being a lower bound on the number of training examples used as Support Vector.

Finally, by using Lagrange techniques and using a kernel function for the dot-product calculations, the decision function becomes the following.

$$f(\boldsymbol{x}) = sgn(\langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle - \rho) = sgn\left(\sum_{i=1}^{m} \alpha_i K(\boldsymbol{x}, \boldsymbol{x_i}) - \rho\right)$$



The default parameter ν in the python implementation is 0.5, leading to a possible loss of half the dataset, hence we fix ν=0.1, more in line with the other outliers detection techniques. After its application to the dataset, it passes from 1508 to 1357 samples.

3.3 STANDARDIZATION

Standardization of datasets is a common requirement for many machine learning estimators. In particular, features are standardized by removing the mean and scaling to unit variance.

$$z = \frac{x - \mu}{\sigma}$$

3.4 PRINCIPAL COMPONENTS ANALYSIS (PCA)

Principal components analysis (PCA) refers to the process of computing principal components, a smaller number of representative variables that collectively explain most of the variability in the original set.

The first principal component of a set of p features is the linear combination of the features that has the largest variance:

$$Z_1 = a_{11}X_1 + \cdots + a_{p1}X_p$$

$$s.t. \sum_{i=1}^{p} a_{i1}^2 = 1$$

These constrained scalars are named "loadings" of the first principal loading component, and together they make up the principal component loading vector $\mathbf{a_1} = (a_{11}\ldots, a_{p1})^T$. They are constrained to have the sum of squares equal to one since otherwise setting these elements to be arbitrarily large in absolute value would result in an arbitrarily large variance.

Let us consider an n×p data set X, with n instances and p features, and let us assume that each of the variables in X has been centered to have mean zero (that is, the column means of X are zero). The problem results in looking for the linear combination of the sample feature values of the form

$$z_{i1} = a_{11}x_{i1} + \cdots + a_{p1}x_{ip}$$

that has the largest sample variance, given the before mentioned constraint.

Given that $\mathbf{A}=\mathbf{X}^{\mathbf{T}}\mathbf{X}$ is the scatter matrix, analogous to the variance-covariance matrix except for a scaling factor[2], the underlying optimization is

$$\max_{\boldsymbol{a_1}} \boldsymbol{a_1^T A a_1}$$

$$s.t.\, \boldsymbol{a_1^T a_1} = 1$$

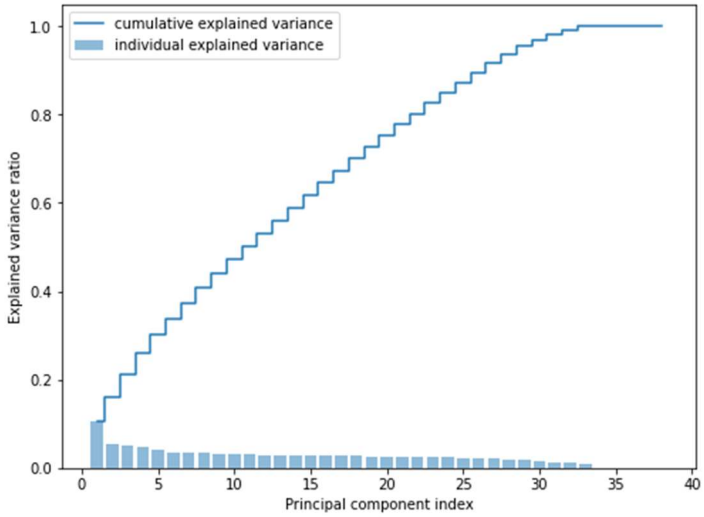
Then, writing its Lagrangian formulation and setting the derivates to 0 we have that the $\mathbf{a_1}$ is an eigenvector of **A**.

$$\boldsymbol{A a_1} = \lambda_1 \boldsymbol{a_1}$$

Substituting this result in the objective function we have

$$\max_{\boldsymbol{a_1}} \lambda_1,$$

Thus we have to take the highest eigenvalue, and we understand that this represents the proportion of variance explained. After this, it is possible to find further principal components. The second principal component is the linear combination of X columns that has maximal variance out of all linear combinations that are uncorrelated with the first principal component. This constraint is obtained by constraining $\mathbf{a_2}$ to be orthogonal to $\mathbf{a_1}$. Thus, for further principal components, it is necessary to solve an analogous optimization problem, constraining orthogonality to the previous components. Eventually, to apply PCA with m components we take the m eigenvectors of **A** associated with the m greatest eigenvalues and the proportion of variance explained (PVE) will be the ratio between their sum and the total variance[3].



[2] $\frac{1}{n}$ or $\frac{1}{n-1}$

[3] Consisting in the sum of all the eigenvalues.

The previous figure shows the variance explained by each of the 38 components[4] and the cumulative one (PVE). In general, to decide the number of components to use, we look for a knee in such a curve, but in the data-set under analysis the increase in PVE was loosely linear. Consequently, the chosen approach consisted in looking at the "individual explained variance" and observing that there are sequences of PCs with the same value: we choose the number of PC after which there is a drop in this value, namely at the 8th and 18th, where the PVE corresponds respectively to 0.40 and 0.70.

3.5 Fisher Linear Discriminant Analysis (FDA)

FDA attempts at finding the vector that maximizes the separation between classes of the projected data. Maximizing "separation" can be ambiguous. The criterion that Fisher's linear discriminant follows to do this is to maximize the distance of the projected means and to minimize the within-class scatter of the projected samples. Scatter, like variance, measures the spread of data around the mean, with the only difference being that it is multiplied by the number of samples.

For each class c, having n_c samples we have:

$$\mu'_c = \frac{\sum_{i=1}^{n_c} \boldsymbol{v^t x_i}}{n_c} = \boldsymbol{v^t \mu_c}$$

$$\boldsymbol{S_B} = (\boldsymbol{\mu_1} - \boldsymbol{\mu_2})(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})^t$$

$$\boldsymbol{S_c} = \sum_{i=1}^{n_c} (\boldsymbol{x_i} - \boldsymbol{\mu_c})(\boldsymbol{x_i} - \boldsymbol{\mu_c})^t$$

$$\boldsymbol{S_W} = \boldsymbol{S_1} + \boldsymbol{S_2}$$

$$s'^2_c = \sum_{i=1}^{n_c} (\boldsymbol{v^t x_i} - \mu'_c)^2 = \boldsymbol{v^t S_c v}$$

Where u'_c is the projected mean for class c, $\boldsymbol{v}$ is the optimal vector onto we will project, $\boldsymbol{S_c}$ is the scatter matrix for class c [5], s'^2_c is the scatter for project samples of class c, $\boldsymbol{S_B}$ is the "between scatter matrix", measuring the separation between the means of two classes before projection. Thus the objective is the maximization of the following function of v.

$$J(\boldsymbol{v}) = \frac{(\mu'_1 - \mu'_2)^2}{s'^2_1 + s'^2_2} = \frac{\boldsymbol{v^t S_B v}}{\boldsymbol{v^t S_W v}}$$

Setting to 0 the derivative of such a function we derive a generalized eigenvalue problem:

$$\boldsymbol{S_B v} = \lambda \boldsymbol{S_W v}$$

Which is immediately solvable if $\boldsymbol{S_W}$ is invertible, since $\boldsymbol{S_B v}$ lies in the direction of the differences of the means:

$$\lambda \boldsymbol{v} = \boldsymbol{S_W^{-1} S_B v} = \boldsymbol{S_W^{-1}} \alpha(\boldsymbol{\mu_1} - \boldsymbol{\mu_2}), \text{ then } \boldsymbol{v} = \boldsymbol{S_W^{-1}}(\boldsymbol{\mu_1} - \boldsymbol{\mu_2}).$$

[4] 38 is the number of features after one hot encoding.
[5] In the sum we have products between a "column" vector and a "row" one so the result is a matrix, not a scalar as if they were inverted.

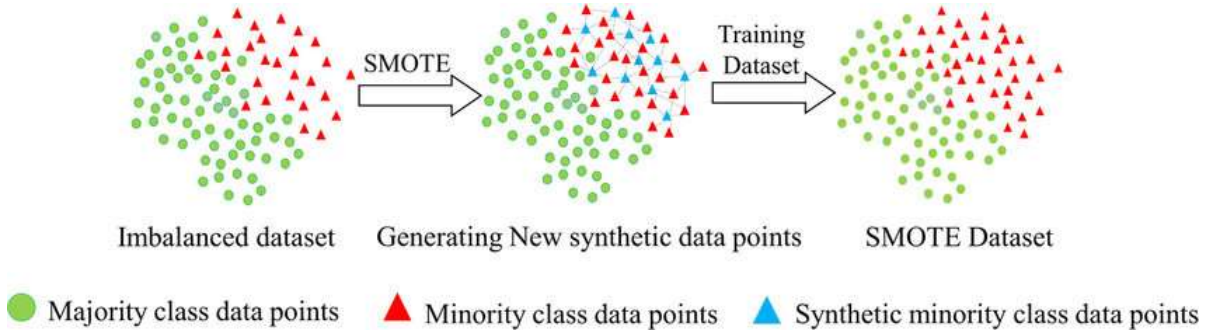
The study will involve four versions of the dataset, two in which we apply PCA with 8 and 18 components, one in which we apply FLD, and one in which we apply neither of the two.

3.6 SMOTE: Synthetic Minority Over-sampling TEchnique

The dataset under analysis presents an imbalance towards the negative instances, an issue addressable by under-sampling the majority class, or over-sampling the minority one. The technique adopted in this work is "Smote", an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples rather than by over-sampling with replacement. Synthetic samples are generated in the following way:

- take the difference between the feature vector (sample) under consideration and its nearest neighbor;
- multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration.

This causes the selection of a random point along the line segment between two minority class samples.

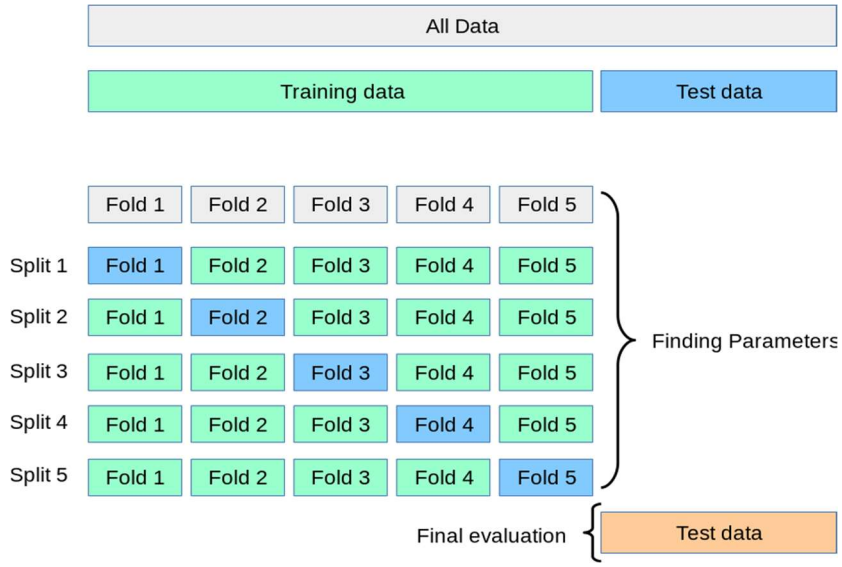


The actual implementation envisioned a slight modification, due to the use of one-hot encoding (Paragraph 3.1) on the categorical features. These derived attributes need to assume only two levels[6] to not lose their semantics, hence we have to exclude these columns from interpolation. This variant is named SMOTE-NC[7] and it is applied only on the dataset where no PCA nor FDA was performed.

3.7 K-fold cross-validation

An estimation of the true risk can be obtained by using some of the training data as a validation set, over which one can evaluate the success of the algorithm's output predictor. In k-fold cross-validation, the original training set is partitioned into k subsets (folds) and, on each fold, we compute the error of the output of the algorithm trained on the union of the other folds. Finally, the average of all these errors is the estimate of the true error. In the proposed study k-Fold cross-validation is used for parameters tuning with k=5 and, once the best parameters-set is chosen, the algorithm is retrained on the entire training set. Also in this case we implemented the stratified variation, ensuring that the original ratio between positive and negative samples is preserved in each split.

[6] Initially the levels are 0 and 1, but after standardization each column will have different values for the two levels.
[7] Synthetic Minority Over-sampling Technique for Nominal and Continuous



4 EVALUATION

4.1 METRICS

Given the following notation for a binary classification problem:

- TP: true positives
- TN: false negatives
- FP: false positives
- FN: false negatives

the metrics under analysis are the following.

- Accuracy: the ratio of the number of correctly predicted observations and the total number of observations.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- Precision: the ratio of the number of correctly predicted positive observations and the total number of predicted positive observations.

$$Precision = \frac{TP}{TP + FP}$$

- Recall: the ratio of the number of correctly predicted positive observations and the total number of positive observations. Intuitively this value suggests how good the model is at spotting the elements belonging to the class of interest.

$$Recall = \frac{TP}{TP + FN}$$

- F1 score: The harmonic mean of Precision and Recall. Balancing the ability of the model in recognizing the class of interest and the precision in doing so, will be the reference metric in the evaluation phases[8].

[8] To select the best model after K-fold training, and during the actual testing.

$$F1 = 2\frac{Recall \times Precision}{Recall + Precision}$$

4.2 MODELS

In this section, the models employed for this classification task are described, along with the results they achieved.

4.2.1 The Bayes Classifier

The Bayes classifier produces the lowest possible test error rate, called the "Bayes error rate". The classification rule consists in assigning each observation x_0 to the most likely class *c*, given its predictor values:

$$c = argmax_j \Pr(Y = j | x = x_0)$$

In a binary classification problem where there are only two possible response values, say class 1 or class 2, the Bayes classifier corresponds to predicting class 1 if $Pr(Y = 1|X = x0) > 0.5$, and class 2 otherwise. Although we would always like to predict qualitative responses using the Bayes classifier, for real data we do not know the conditional distribution of Y given X, and so computing the Bayes classifier is impossible. Therefore, the Bayes classifier serves as an unattainable gold standard against which to compare other methods, which have actually been employed in the proposed study.

4.2.2 K-Nearest Neighbours (KNN)

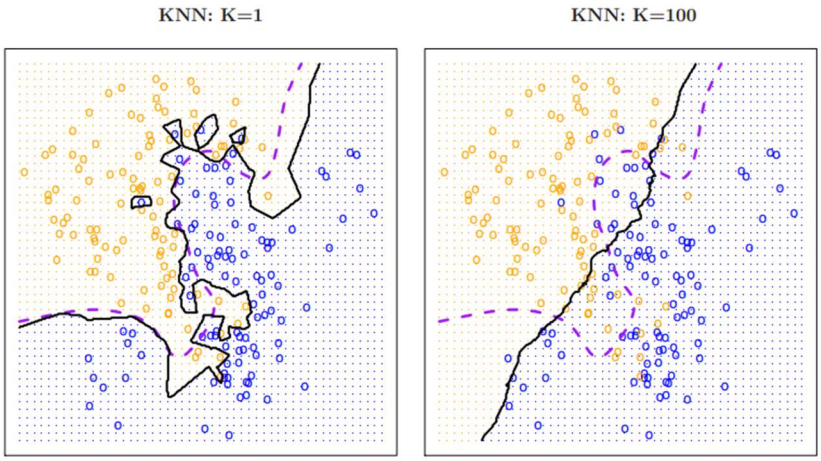
4.2.2.1 Overview

Nearest Neighbor algorithms are among the simplest of all machine learning algorithms. The idea is to memorize the training set and then predict the label of any new instance based on the labels of its closest neighbors in the training set, figuring out a label on any test point without searching for a predictor within some predefined class of functions. The rationale behind such a method is based on the assumption that the features that are used to describe the domain points are relevant to their labelings in a way that makes close-by points likely to have the same label. Specifically, this method attempts to estimate the conditional distribution of Y given X, and then classify a given observation to the class with the highest estimated probability. Given a positive K, a distance metric, and a test observation x_0, the KNN classifier first identifies the K points in the training data that are closest to x_0, represented by N_0. It then estimates the conditional probability for class j as the fraction of points in N_0 whose response values equal j:

$$\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j)$$

Finally, the test sample is assigned to the class with the highest probability.

The choice of K has a drastic effect on the KNN classifier obtained: as K grows, the method becomes less flexible and produces a decision boundary that is smoother and smoother, getting closer to linear. In general, as we use more flexible classification methods, the training error rate will decline but the test error rate could not: the method could become excessively flexible and overfit.

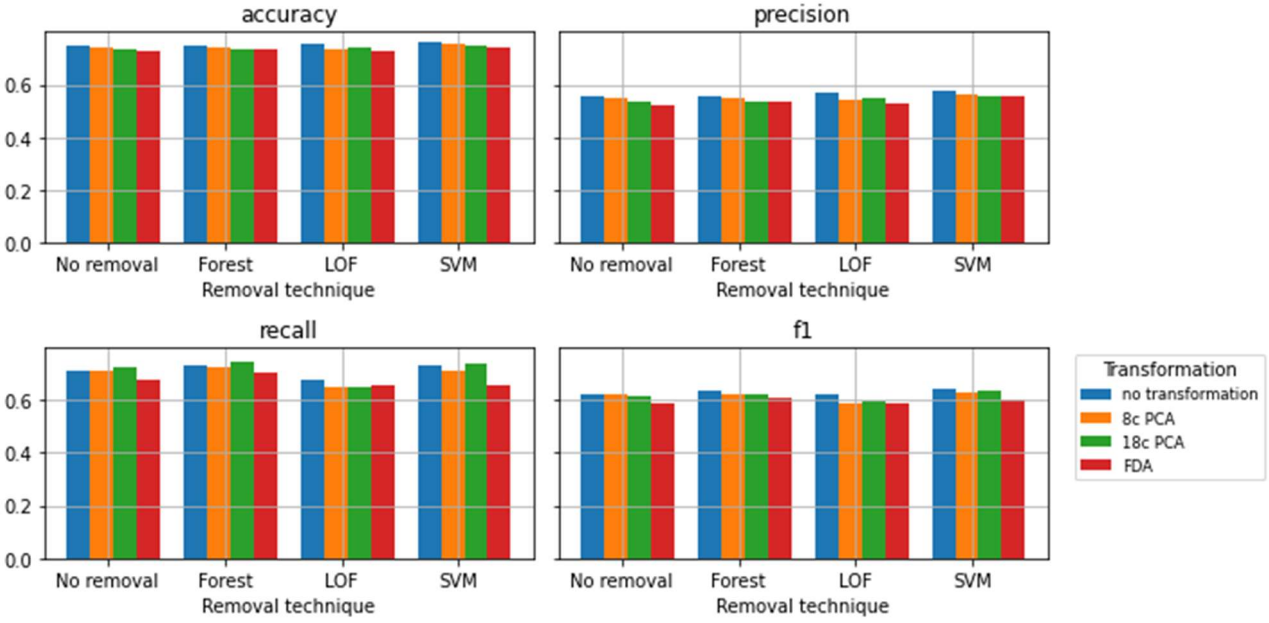


A comparison of the KNN decision boundaries (solid black curves) obtained using K = 1 and K = 100. With K = 1, the decision boundary is overly flexible, while with K = 100 it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.

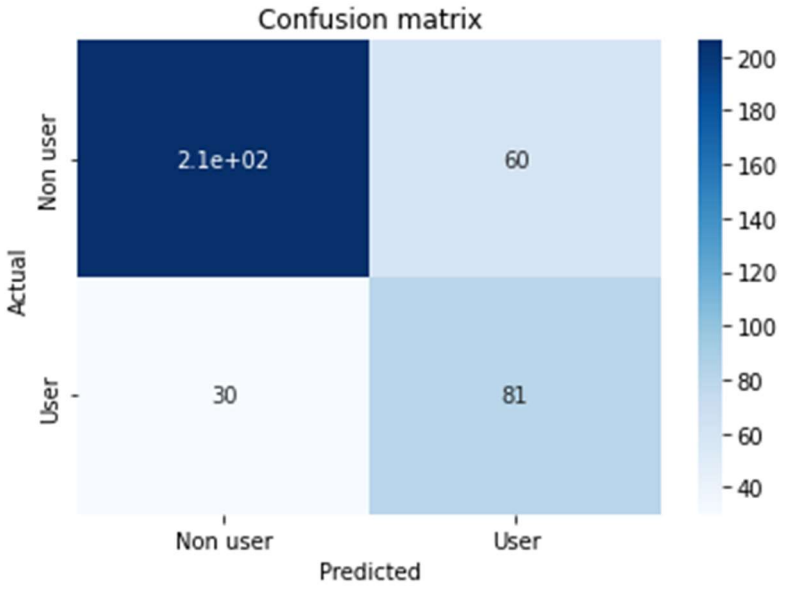
4.2.2.2 Results

We perform a grid search on the following parameters.

- Number of neighbors: {3, 5, 7, 10, 15, 20, 25, 30, 40}
- Weight function used in prediction: "uniform" to give the same weights in voting, "distance" to weight points by the inverse of their distance
- The distance metric, namely "Manhattan" or "Euclidean"



The best f1-score is 0.64, envisioning a preprocessing pipeline only consisting of the One-Class SVM technique for outliers detection. The model used k=7, the Manhattan distance to choose neighbors and a uniform weighting scheme.

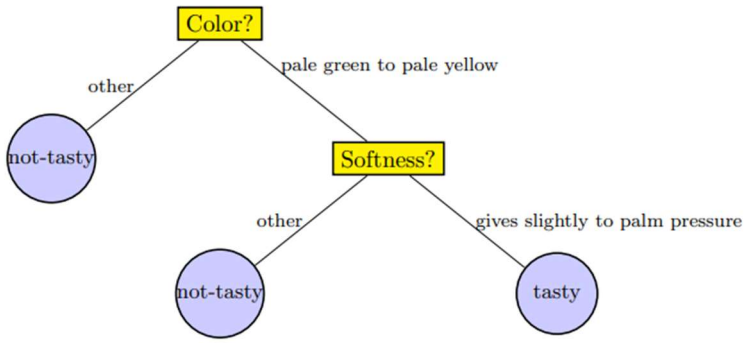


The figure shows the confusion matrix of the model, namely from top-left to bottom right the portion of TN, FP, FN, and TP. The model finds a considerable portion of positive instances, presenting recall=0.73, but presents a high FP value, resulting in a precision of 0.57. The global accuracy is 0.76.

4.2.3 Decision tree

4.2.3.1 Overview

A decision tree is a classifier that predicts the label associated with an instance *x* by traveling from a root node of a tree to a leaf. At each node on the root-to-leaf path, the successor child is chosen based on a splitting of the input space. Usually, the splitting is based on one of the features of *x* or a predefined set of splitting rules. A leaf contains a specific label and because of this one of the main advantages of decision trees is given by its understandability and interpretability.



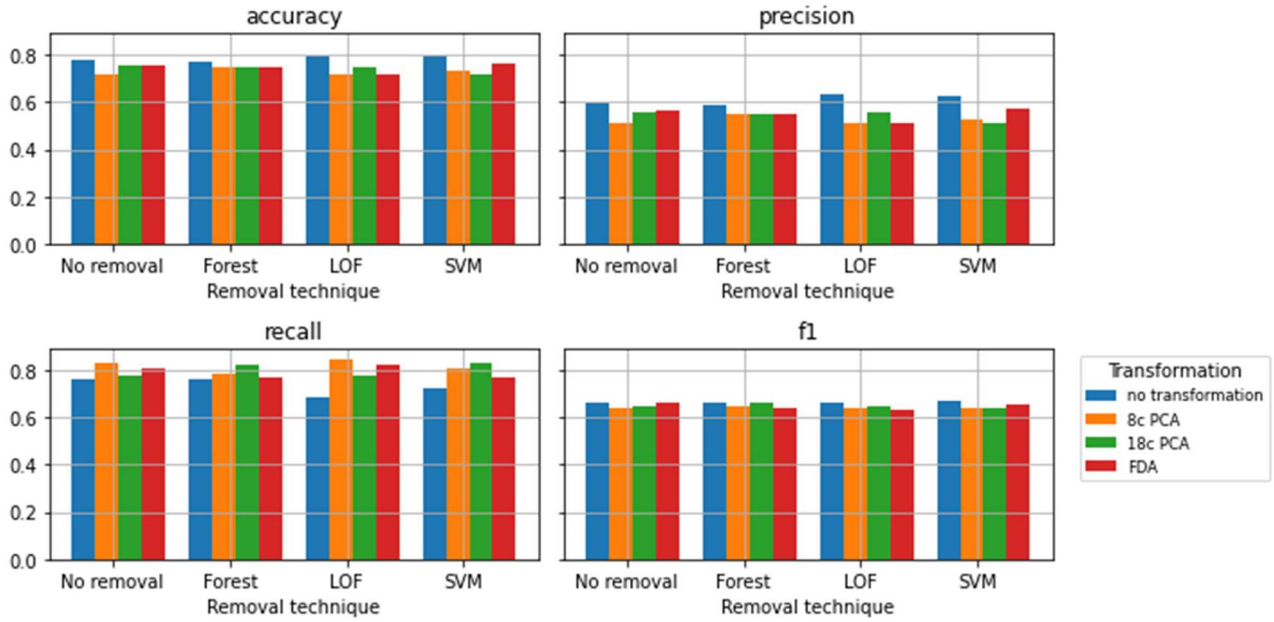
Decision tree learning algorithms are based on heuristics such as greedy approaches, where the tree is constructed gradually, and locally optimal decisions are made at the construction of each node. Such algorithms cannot guarantee to return the globally optimal decision tree but tend to work reasonably well in practice. A general framework for growing a decision tree is as follows. We start with a tree with a single leaf (the root) and assign this leaf a label according to a majority vote among all labels over the training set. We now perform a series of iterations. On each iteration, we examine the effect of splitting a single leaf. We define a loss measure and among all possible splits, we either choose the one that minimizes it or we choose not to split the leaf at all. The loss functions evaluated in the study are the following.

- **Gini index:** $Gini(t) = 1 - \sum_j p(j|t)^2$, where $p(j|t)$ is the relative frequency of class j at node t. Its maximum value is (1- $1/n_{classes}$) when records are equally distributed among all classes, implying the highest impurity degree; its minimum value is 0 when all records belong to one class, implying the lowest impurity degree, after which it would have no sense to further split. The qualities of the possible splits are then computed as
$$Gini_{split} = \sum_{t=1}^{k} \frac{n_i}{n} Gini(t),$$
where k is the number of partitions, n_i is the number of instances of t children node, n is the number of instances in the node to split.
- **Entropy impurity measure:** $Entropy(t) = -\sum_j p(j|t) \log_2 p(j|t)$, with maximum $\log_2(n_{classes})$, and minimum 0, for which analogous considerations to the Gini index hold. In this case, we want to maximize the "information gain":
$$GAIN_{split} = Entropy(p) - \sum_{t=1}^{k} \frac{n_i}{n} Entropy(t)$$
measuring the reduction in entropy achieved because of the split.

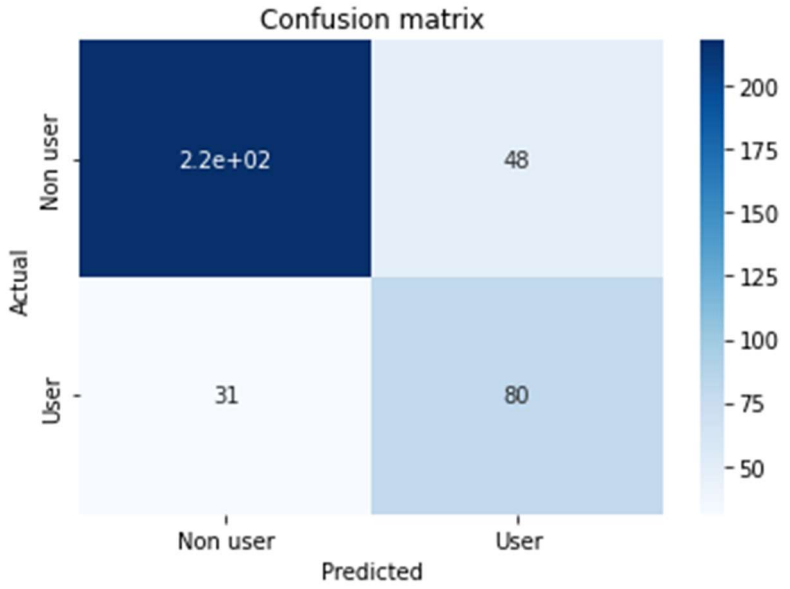
4.2.3.2 Results

We perform a grid search on the following parameters.

- The maximum depth of the tree: {4, 6, 10, 12, 14} or not limiting it.
- The function for measuring the quality of a split, between Gini and Entropy, as mentioned.



The best f1-score is 0.67, preprocessing the data-set with One-Class SVM for outliers detection, but not applying any transformation. The model used the "Entropy" as quality measure and imposed a maximum depth of 4, and it is worth noticing that this parameters-set was the same independently from the removal technique and transformation adopted.



This method returns recall=0.72, comparable with the KNN's one, but a considerably increased precision=0.63. The overall accuracy is 0.79.

4.2.4 Random forest

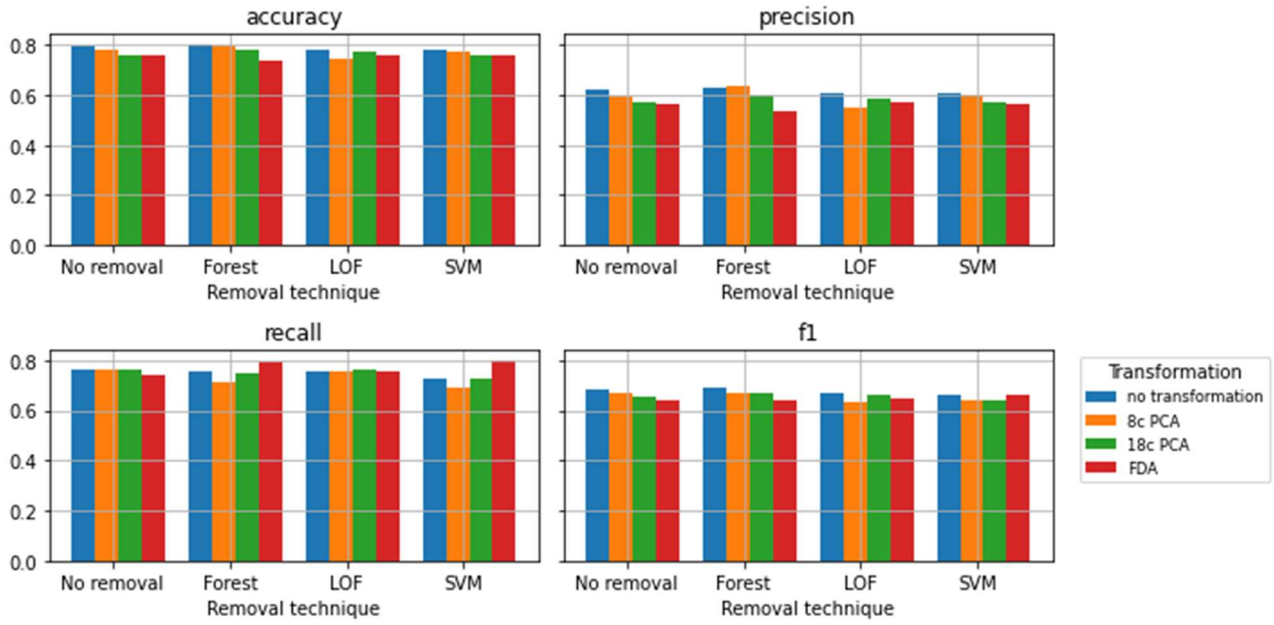
4.2.4.1 Overview

This is an ensemble method, namely an approach that combines many simple models to obtain a single and potentially very powerful model. Indeed, decision trees suffer from high variance, namely fitting a decision tree on different splits of the training data could provide considerably different results. Bootstrap aggregation, or bagging, is a procedure for reducing the variance of a statistical learning method, based on the fact that given a set of n independent random variables $Z_1,..., Z_n$, each with variance σ^2, the variance of the mean Z_{avg} is σ^2/n. In fact, bagging consists of taking many training sets from the population, building separate prediction models using each training set, and averaging the resulting predictions. Since we generally do not have access to multiple training sets we can bootstrap, by taking repeated samples from the (single) training data set. Then, a number of decision trees are built on bootstrapped training samples. Random forest goes further bagging: each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors, where generally $m \approx \sqrt{p}$. The rationale behind this is that if we allow choosing among all the features, most or all of the trees will use mainly the strong ones and will look quite similar to each other, providing highly correlated predictions. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. Random forests overcome this problem by forcing each split to consider only a subset of the predictors, hence decorrelating the trees, thereby making the average of the resulting trees less variable and thus more reliable.

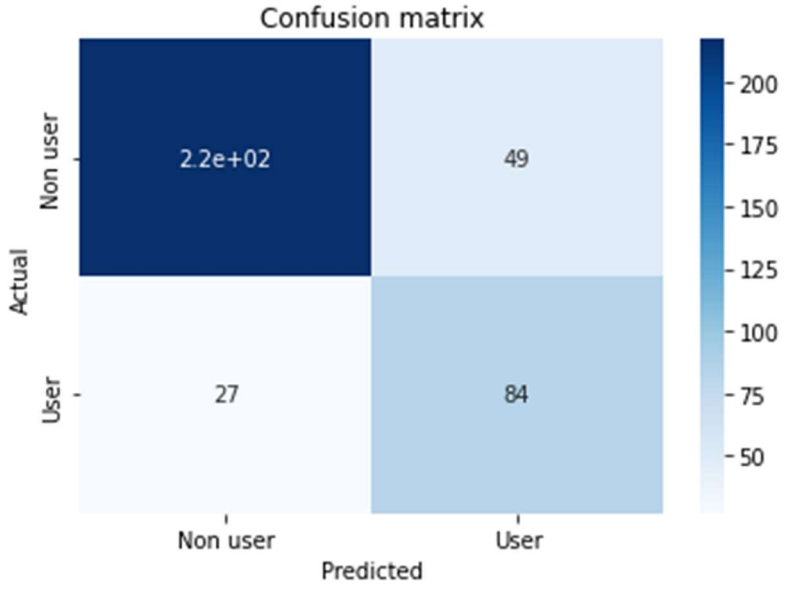
4.2.4.2 Results

We perform a grid search on the following parameters.

- The maximum depth of the trees: {4, 6, 10, 12, 14} or not limiting it
- The number of trees: {50, 100, 200}



The best f1-score is 0.69, employing Isolation Forest for outlier detections, but not transforming the data (as the Decision Tree classifier did). The model imposed a maximum depth of 6 and made use of 100 estimators.



The remaining metrics are recall=0.76, precision=0.63, and accuracy=0.80: the ensemble method was able to increase the capability of finding drug users, keeping the same precision of the DT.

4.2.5 Support Vector Machines (SVM)

4.2.5.1 Overview

Let $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$ be a training set of examples, with $\boldsymbol{x}_i \in R^d$ and $y_i \in \{\pm 1\}$. We say that this training set is linearly separable if there exists a halfspace $(\boldsymbol{w}, b)$, such that

$$y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) > 0, \qquad \forall i \in \{1, \ldots, m\}.$$

All halfspaces *(**w**, b)* that satisfy this condition are Empirical Risk Minimization ERM hypotheses, then it is necessary to add another criterion to choose the final learner. In particular, Hard-SVM is the learning rule in which we return an ERM hyperplane that separates the training set with the largest possible "margin", defined as the minimum distance between a point in the training set and the hyperplane. The rationale for this choice is that if a hyperplane has a large margin, then it will still separate the training set even if we slightly perturb each instance. Formally, SVM is defined as follows:

$$argmax_{(\boldsymbol{w},b):\,\|\boldsymbol{w}\|^2=1} min_i |\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle + b|,$$

$$\text{s.t.} \quad y_i(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle + b) > 0$$

It is possible to provide another equivalent formulation of the Hard-SVM rule as a quadratic optimization problem:

$$(\boldsymbol{w_0}, b_0) = argmin_{(\boldsymbol{w},b)} \|\boldsymbol{w}\|^2,$$

$$\text{s.t.} \quad y_i(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle + b) \geq 1$$

$$\text{output: } \boldsymbol{w'} = \frac{\boldsymbol{w_0}}{\|\boldsymbol{w_0}\|}, \boldsymbol{b'} = \frac{\boldsymbol{b_0}}{\|\boldsymbol{b_0}\|}$$

To simplify this problem we can reduce the problem of learning non-homogenous halfspaces to the problem of learning homogenous halfspaces by adding one more feature to each instance of $\boldsymbol{x_i}$, thus increasing the dimension to d + 1.

However, the Hard-SVM formulation is based on the strong assumption that the training set is linearly separable, which can be hard to satisfy. Because of this, it is possible to introduce Soft-SVM as a relaxation of the Hard-SVM rule that can be applied even if the training set is not linearly separable. The new algorithm is the following:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \lambda\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \xi_i,$$

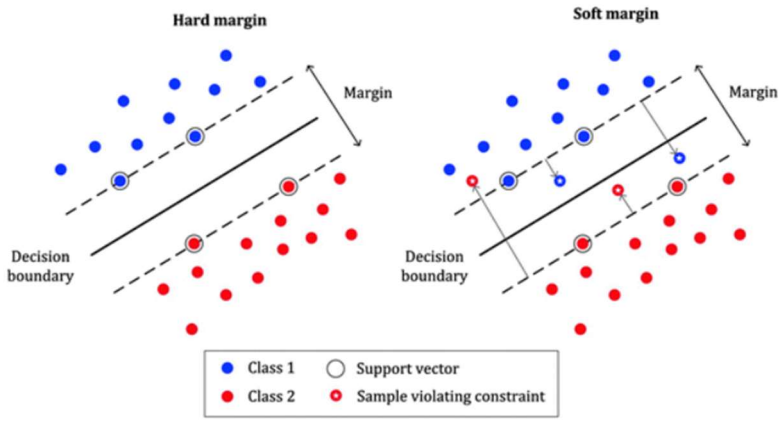
$$\text{s.t.} \quad y_i(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle + b) \geq 1 - \xi_i$$

The variables ξ_i are referred to as slack variables, and allow to relax the linear separability constraint, also quantifying by how much this constraint is violated. Then Soft-SVM jointly minimizes the norm of $\boldsymbol{w}$ (margin) and the average of ξ_i (violations of the constraints), with the trade-off between the two objectives controlled by a positive parameter λ.

However, the main characteristic of this algorithm is the fact that the solution $\boldsymbol{w_0}$ is in the linear span of the examples that are exactly on the margin:

$$\boldsymbol{w_0} = \sum_{i \in I = \{j:\, |\langle \boldsymbol{w}, \boldsymbol{x_j}\rangle| = 1\}} \alpha_i \boldsymbol{x_i}$$

The vectors on the margin are therefore called "support vectors" and from this stems the name of the algorithm.



Further considerations can be drawn recurring to the dual formulation of the previous problem:

$$\min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha}\in\mathbb{R}^m:\boldsymbol{\alpha}\geq 0} \frac{\|\boldsymbol{w}\|^2}{2} + \sum_{i=1}^{m} \alpha_i(1 - y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle)$$

It is possible to flip the order of min and max operations, at most decreasing the objective of the function, referring to this as "weak duality" (in this case also "strong duality" holds, namely the inequality holds with equality). Thus, fixing $\boldsymbol{\alpha}$, the problem is unconstrained in $\boldsymbol{w}$, and the equation can be differentiated. Thus, at the optimum, the gradient equals zero and it results

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x_i}$$

This highlights that the solution must be in the linear span of the examples, and substituting this in the objective function after some computation we obtain

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^m:\boldsymbol{\alpha}\geq 0} \sum_{i=1}^{m} \alpha_i - \sum_{j=1}^{m}\sum_{i=1}^{m} \alpha_i\alpha_j \; y_i y_j \langle \boldsymbol{x_i}, \boldsymbol{x_j}\rangle$$

It is possible to notice that the dual problem only involves inner products between instances and does not require direct access to specific elements within an instance, a key property allowing the implementation of SVM with "kernels".

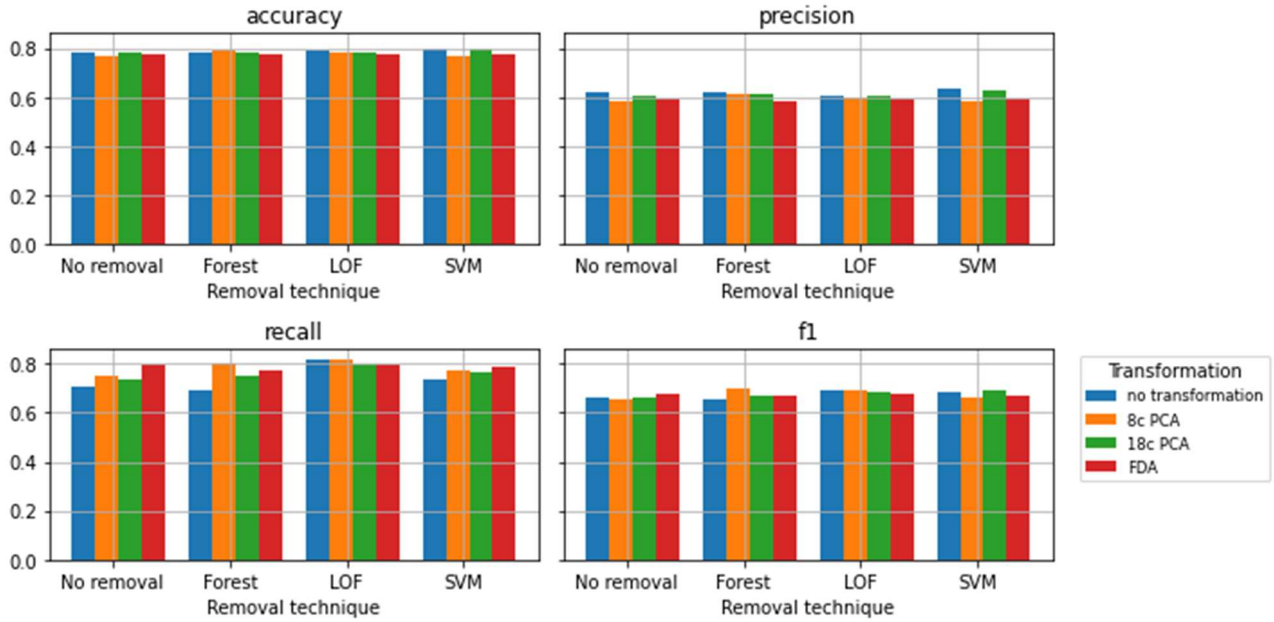
It is also possible to enrich the class of halfspaces by first applying a nonlinear mapping, ψ, that maps the instances into some feature space and then learning a halfspace in that feature space. However, computing linear separators over very high dimensional data may be computationally expensive: the common solution to this concern is kernel-based learning. Given an embedding ψ of some domain space X into some Hilbert space, we define the kernel function $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \psi(\boldsymbol{x}), \psi(\boldsymbol{x}')\rangle$. The main advantage of such a paradigm is that it allows learning linear separators in high dimensional feature spaces without having to specify points in that space or expressing the embedding ψ explicitly. Finally, considering a similarity function $K : X \times X \rightarrow R$, it is possible to understand if it represents an inner product for some feature mapping ψ, namely if it is a valid kernel function. This happens if the following condition is satisfied:

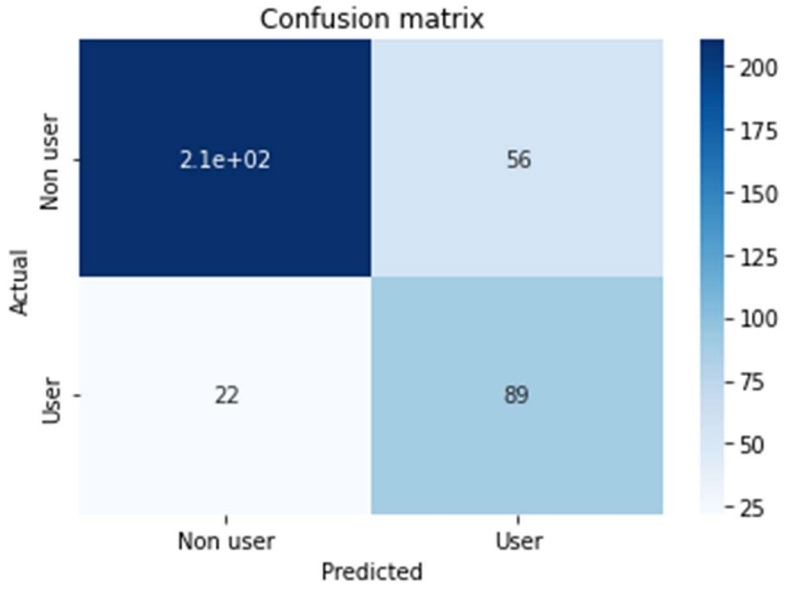
a symmetric function $K : X \times X \rightarrow R$ implements an inner product in some Hilbert space if and only if it is positive semidefinite; namely, for all $x_1, \ldots, x_m$, the Gram matrix, $G_{i,j} = K(x_i, x_j)$, is a positive semidefinite matrix.

4.2.5.2 Results

We perform a grid search on the following parameters.

- The kernel type: Polynomial with degree 3, Gaussian, or Linear
- The regularization parameter C: {0.5, 1, 1.5, 2}





The best f1-score is 0.70, envisioning Isolation Forest for outliers removal and PCA with 8 components. The best hyper-parameter configuration was C=0.5 and Gaussian kernel. Furthermore, this model has the highest tendency to label the samples as positive, reaching recall = 0.80 and precision=0.61. The global accuracy is 0.79.

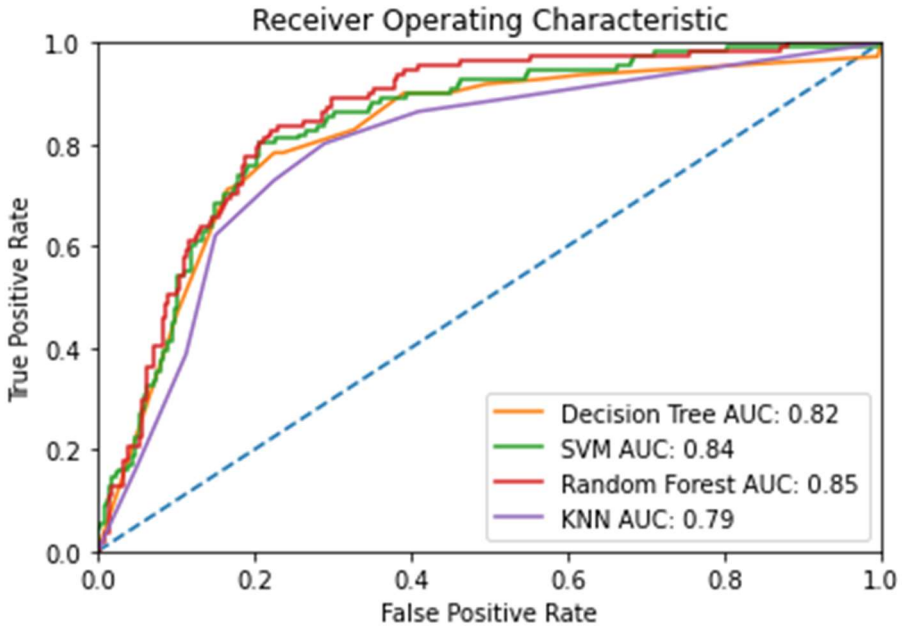
4.3 ROC CURVES

Further analysis can make use of the "receiver operating characteristic" (ROC) curves of the models, namely a graphical plot that illustrates the diagnostic ability of a binary classifier system as its

discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. However, the tested models directly output a label, thus their ROC curves would be just a point in the graph (overall FPR and TPR). Thus, we have to output probabilities (for the sample being 0 or 1) in model-specific ways:

- KNN: the predicted class probability is the fraction of points in the neighborhood belonging to that class (Paragraph 4.2.2.1)
- Decision Tree: the predicted class probability is the fraction of samples of the same class in a leaf.
- Random Forest: the predicted class probabilities of an input sample are computed as the mean predicted class probabilities of the trees in the forest.
- SVM: in this case, we do not output probabilities, but a confidence score, proportional to the signed distance of a sample from the hyperplane.

Consequently, the curves will represent how confident are our models of their prediction. An aggregate measure of performance across all possible classification thresholds is the "area under the curve" (AUC), interpretable as the probability that the model ranks a random positive example more highly than a random negative example.



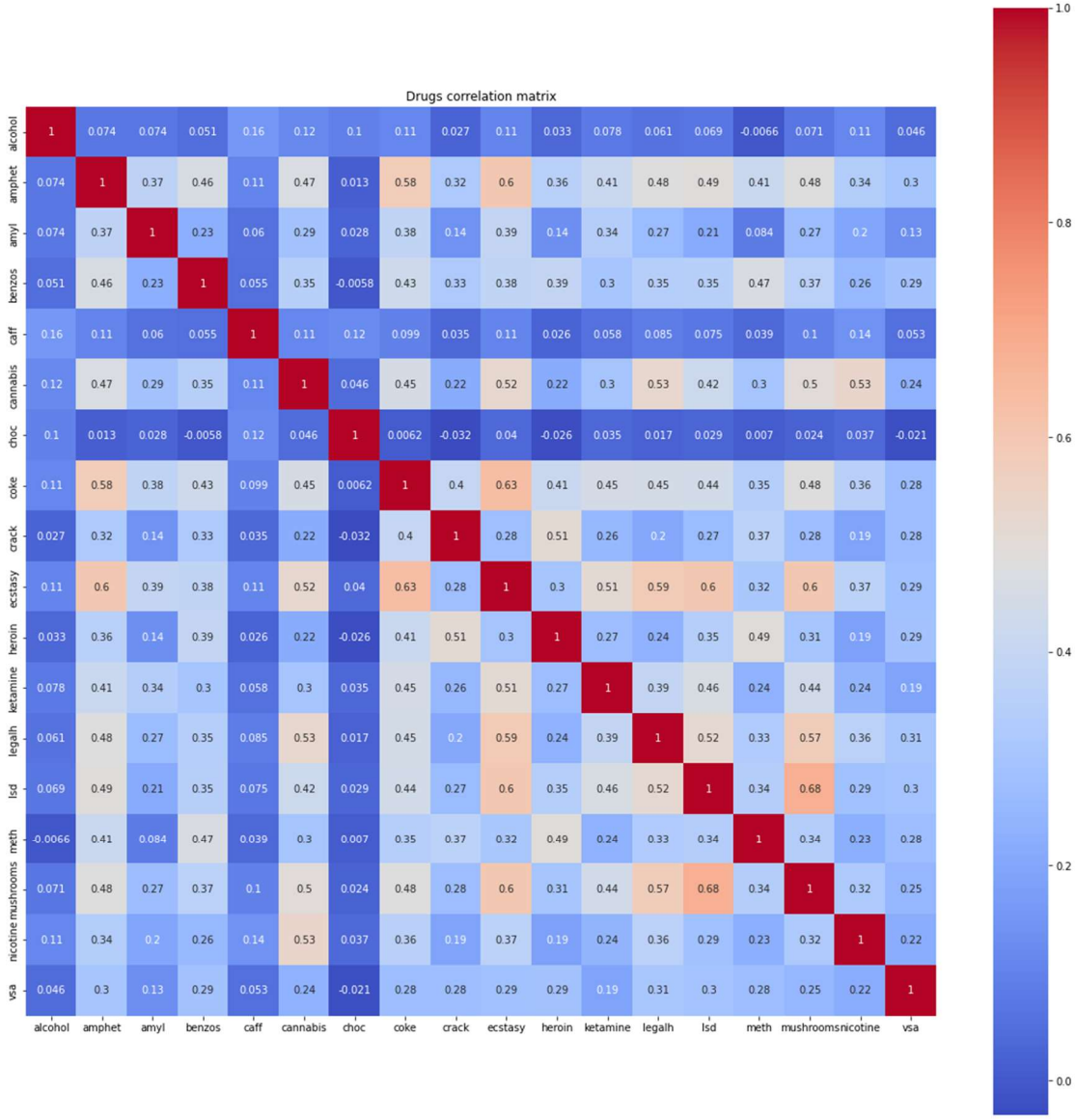
The ROC curves are almost overlapping with similar AUC values, coherent with their f1-scores. The only exception is that the Random Forest Classifier presents a higher AUC value than the SVM's one. Although this second model was more successful in the prediction task, the RF is more confident in its prediction, it better separates the samples: it could be more robust, even though the AUC difference is minimal, hence this remains just a conjecture. Finally, we can observe the improvements of RF with respect to DT: although they had similar performances the considerably higher AUC values suggest that RF better separates the instances.

5 CONCLUSIONS AND PROPOSALS

The study allowed the development of a preprocessing pipeline consisting of one-hot encoding, possible outliers removal, possible transformation of the dataset, and SMOTE resampling, followed by stratified k-fold cross-validation for hyper-parameters selection.

The f1-score difference between the best performing model (SVM f1=0.70) and the least performing one (KNN f1=0.64) was 0.06, and the f1-score differences between models of the same type were often negligible. This highlights that for this classification task the outliers removal and the application of PCA or FDA were secondary with respect to the hyper-parameters tuning, as long as it is done separately on datasets with different pre-processing steps. This could allow for an extreme reduction of dimensionality if required. To verify such a hypothesis we propose to apply grid-search also on the parameters of the algorithms used for outliers detection, rather than rely on conjectures.

Furthermore, it is possible to repeat the same study on drugs other than LSD, or even on "pleiads", if we perform an aggregation on different drugs. A suggested way to do this is by considering correlations among them to define a cluster, then labeling as positive a subject that made use of at least one of the drugs in it.



6 BIBLIOGRAPHY AND REFERENCES

E. Fehrman, A. K. Muhammad, E. M. Mirkes, V. Egan and A. N. Gorban, "The Five-Factor Model of personality and evaluation of drug consumption risk."

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique”

Fei Tony Liu, Kai Ming Ting, Zhi-Hua Zhou, “Isolation Forest”

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, Jörg Sander, “LOF: Identifying Density-Based Local Outliers”

Amer, Mennatallah & Goldstein, Markus & Abdennadher, Slim. (2013). Enhancing one-class Support Vector Machines for unsupervised anomaly detection. Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, ODD 2013. 8-15. 10.1145/2500853.2500857.

Bernhard Scholkopf, Robert Williamson, Alex Smola, John Shawe-Taylort, John Platt, “Support Vector Method for Novelty Detection”

Shalev-Shwartz, Shai, and Shai Ben-David, “Understanding machine learning: From theory to algorithms.”

James, G., Witten, D., Hastie, T., Tibshirani, R., “An Introduction to Statistical Learning with Applications in R.”

Dirk P. Kroese, Zdravko I. Botev, Thomas Taimre, Radislav Vaisman, “Data Science and Machine Learning: Mathematical and Statistical Methods”

Scikit-learn: https://scikit-learn.org/stable/index.html