Tesina
Mathematics in Machine Learning
Drug Consumption Dataset

Luca Benfenati
s286582

October 2022

Abstract

The problem of evaluating an individual's risk of drug consumption and misuse is highly important. An online survey methodology was employed to collect data including Big Five personality traits (NEO-FFI-R), impulsivity (BIS-11), sensation seeking (ImpSS), and demographic information. The data set [1] contained information on the consumption of 18 central nervous system psychoactive drugs. Correlation analysis demonstrated the existence of groups of drugs with strongly correlated consumption patterns. Among these different kind of drugs, one was selected on which we carried out the analysis. A number of classification methods were employed (decision tree, random forest, k-nearest neighbors, logistic regression and support vector machine). The best results achieved over 75% in accuracy. Code is provided at [2].

Contents

1 Introduction

The following report shows results obtained with a thorough analysis on the *Drug Consumption Dataset*, which was created in 2016 as result of a collaboration between the University of Leicester, the University of Nottingham and the Rampton Hospital (Retford,Nottinghamshire). Exploiting machine learning and supervised algorithms, we will try to address the problem of evaluating an individual's risk of drug consumption and misuse, considering different kind of drugs and different types of users. Data have been collected through an online survey methodology, including the Big Five personality traits (NEO-FFI-R), impulsivity (BIS-11), sensation seeking (ImpSS), and demographic information.

2 Dataset Exploration

The dataset contained information on the consumption of 18 central nervous system psychoactive drugs: alcohol, amphetamines, amyl nitrite, benzodiazepine, cannabis, chocolate, cocaine, caffeine, crack, ecstasy, heroin, ketamine, legal highs, LSD, methadone, mushrooms, nicotine and volatile substance abuse and one fictitious drug (Semeron) which was introduced to identify over-claimers. For each drug they have to select one of the answers: never used the drug, used it over a decade ago, or in the last decade, year, month, week, or day.

Database contains records for 1885 respondents. For each respondent 12 attributes are known, which can be divided in:

1. Demographic Information,
    - **age**: 18-24, 25-34, 35-44, 45-54, 55-64, 65+
    - **gender**: Female, Male
    - **country of residence**: Australia, Canada, New Zealand, Republic of Ireland, UK, USA, Other
    - **ethnicity**: Asian, Black, White, Mixed-Black/Asian, Mixed-White/Asian, Mixed-White/Black, Other
    - **level of education**;
2. Personality measurements which include Five Personality Traits:
    - **neuroticism** (Nscore),
    - **extraversion** (Escore),
    - **openness to experience** (Oscore),
    - **agreeableness** (Ascore),
    - **conscientiousness** (Cscore);
3. **Impulsivity** (Impulsive), measured by the Barratt Impulsiveness Scale (BIS-11),
4. **Sensation seeking** (SS), measured by ImpSS Scale.

2.1 Problem definition

Database contains 18 classification problems, one for each drug. Each of independent label variables contains seven classes: Never Used (CL0), Used over a Decade Ago (CL1), Used in Last Decade (CL2), Used in Last Year (CL3), Used in Last Month (CL4), Used in Last Week (CL5), and Used in Last Day (CL6). Among all the possible alternatives, we need to define the classification problem we want to solve. For example:

- we may solve a binary classification problem for each type of drug, considering the two classes as *User* vs *Non-User*. In this case, we have 18 possible binary classification problem;
- we may solve a 7-class classification problem for each type of drug, considering the seven classes (from CL0 to CL6), thus predicting the type of user, not only whether an individual is a user or not.

Since the focus of this elaborate is to carry out a thorough analysis, considering the mathematical side of the methods and algorithms we are going to use, we decided to focus on a single central nervous system psychoactive drug. Once we selected the drug of interest, we decided to solve a binary classification problem, distinguishing between drug *user* and *non-user*. Thus:

1. the user class includes Used in Last Decade (CL2), Used in Last Year (CL3), Used in Last Month (CL4), Used in Last Week (CL5), and Used in Last Day (CL6);
2. the Non-User class includes Never Used (CL0), Used over a Decade Ago (CL1).

In this way, a 7-classes classification problem has just been translated into a binary one. However, we still need to decide how the 18 different drugs must be treated. In Fig. 1, we report an example of the distribution of the users between the different classes in both cases: indeed, we have the 7-classes case and the 2-classes case.

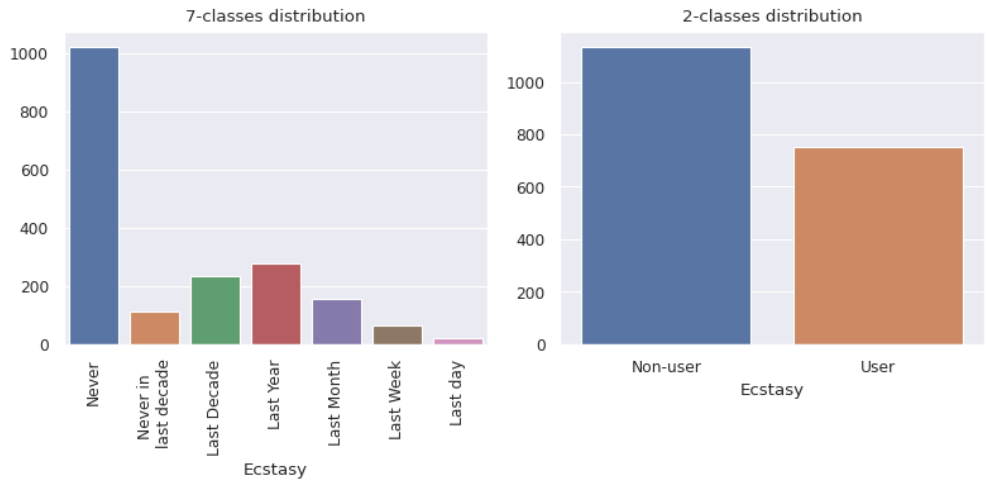


Figure 1: Distribution of Users considering the *Ecstasy* case

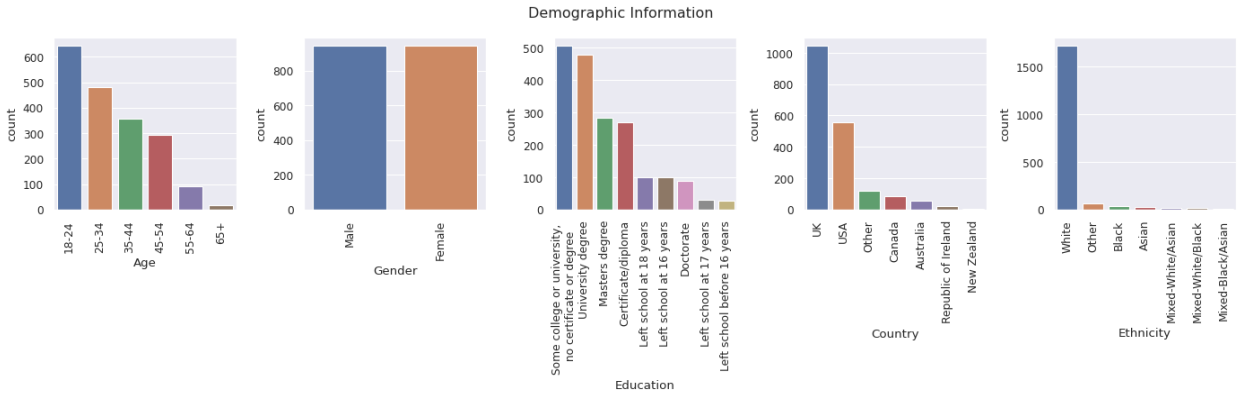


Figure 2: Demographic information

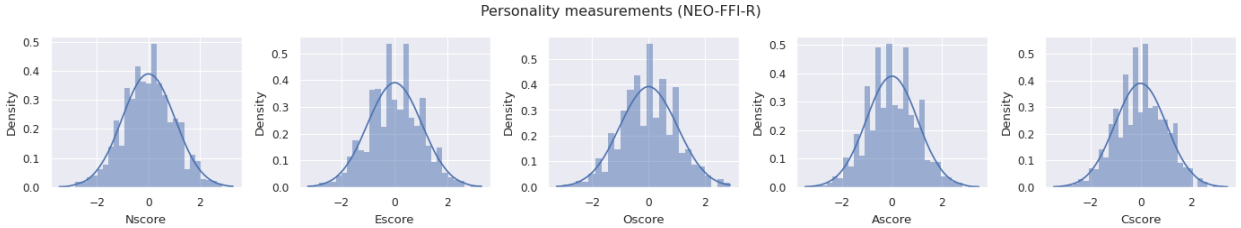


Figure 3: Personality measurements

2.2 Feature distribution

In this section, we focus on analyzing the distribution of the different features. Specifically, we can see that all of the features have been already grouped, quantized and encoded in a previous pre-processing step. In order to provide better visualization, we decided to map the encoded values with the original ones for the categorical features whose original meaning is interesting to show. In particular, demographic information such as age, gender, education, country and ethnicity have been re-mapped with the original value and are represented with, as common for categorical features. On the other hand, features that were originally real numbers and that have been simply normalized, such as personality measurements, impulsivity and sensation seeking, we represent their probability distribution.

From Figure 2, we can easily see the characteristics of individuals who have responded

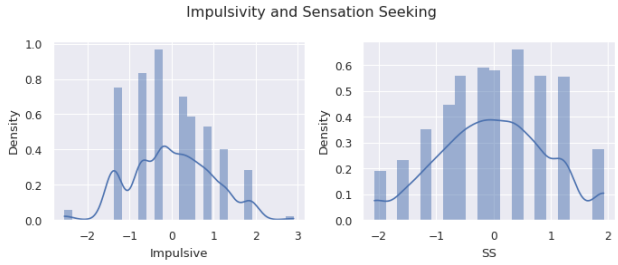


Figure 4: Impulsivity and Sensation Seeking

to the online survey methodology: it is worth noticing that most of subjects are between 18-24 years old, they have not yet completed their studies, they are still frequenting some college or some university. Additionally they are mostly UK citizens (more than 50%), of white ethnicity with no prevalence of either men or women. As we can easily see from Figure 3, the features regarding Personality measurements have been already standardized, as the bell-shaped values are centered around mean with a unit standard deviation. The same observation can not be grasp for the Impulsivity and Sensation Seeking feature, which, however, have been already pre-processed and scaled in a previous unknown step.

2.3 Feature correlation

In this section we present an analysis of the correlation between the features exploiting two different measures. Specifically, the first one is the Spearman correlation coefficient: the Spearman's rank-order correlation is the non-parametric version of the Pearson product-moment correlation. Spearman's correlation coefficient, ρ, measures the strength and direction of association between two ranked variables. Spearman's correlation determines the strength and direction of the monotonic relationship between your two variables rather than the strength and direction of the linear relationship between your two variables, which is what Pearson's correlation determines. The correlation coefficient is calculated considering the following formula:

$$\rho = 1 - \frac{6 \sum_i d_i^2}{n(n-1)} \tag{1}$$

where $d_i = x_i - y_i$ is the difference between the ranks of the two features of the $i-th$ instance and n is the number of samples. We also explore the Kendall Tau-b Correlation Coefficient, which is considered to be more robust and generally the preferred method between the two explored. The Kendall Tau-b correlation coefficient,τ_b , is a non-parametric measure of association based on the number of concordances and discordances in paired observations. Two observations $(X_i, Y_i), (X_j, Y_j)$ are:

- *concordant*, if they are in the same order with respect to each variable;
- *discordant*, if they are in the reverse ordering for X and Y, or the values are arranged in opposite directions;
- *tied*, if $X_i = X_j$ and/or $Y_i = Y_j$.

The total number of pairs that can be constructed for a sample size of n is:

$$N = \binom{n}{2} = \frac{1}{2} n(n-1) = P + Q + X_0 + Y_0 + (XY)_0 \tag{2}$$

where P is the number of concordant pairs, Q is the number of discordant pairs, X_0 is the number of pairs tied only on the X variable, Y_0 is the number of pairs tied only on the Y variable, $(XY)_0$ and is the number of pairs tied on both X and Y. The Kendall tau-b for measuring order association between variables X and Y is given by the following formula:

$$\tau_b = \frac{P - Q}{\sqrt{(P + Q + X_0)(P + Q + Y_0)}} \tag{3}$$

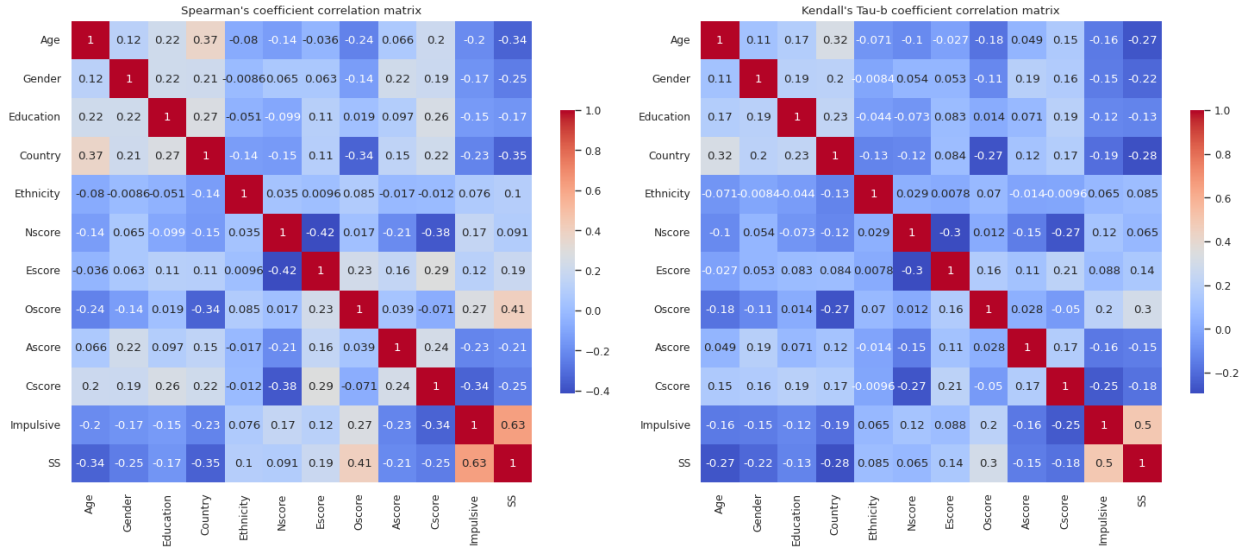


Figure 5: Feature correlation matrices

Notice that both the Spearman and the Kendall Tau-b coefficients are limited between -1 and 1, thus $-1 \leq \rho \leq 1$ and $-1 \leq \tau_b \leq 1$. The difference between these two coefficients can be seen in the two matrices: Kendall's Tau-b results in less strong correlations overall, probably because of its independence of the monotonic behavior of the variables considered. It is also worth noticing that Kendall's Tau-b works generally better with ordinal values. However, since we have not highly correlated features that may compromise the performance of a model, no feature will be discarded as predictor.

2.4 Drug selection

Now that correlation coefficients have been defined, we can focus on choice of the drug to be used as label. In order to do that, we compute the Kendall's Tau-b correlation coefficients of each drug with respect to the others. Then, we compute the mean of these coefficient: results are reported in table 1.

Drug	Alchol	Amphet	Amyl	Benzos	Caff	Cannabis	Choc	Coke	Crak	**Ecstasy**
Mean Correlation	0.12	0.38	0.24	0.32	0.12	0.35	0.07	0.38	0.27	**0.40**

Heroin	Ketamine	Legalh	LSD	Meth	Mushrooms	Nicotine	Semer	VSA
0.29	0.32	0.36	0.37	0.30	0.38	0.29	0.08	0.25

Table 1: Mean Kendall's Tau-b coefficient correlation between drugs

The rationale behind is that usage of some drugs are significantly correlated between each other. We found three groups of drugs with highly correlated use. The central element is clearly identified for each group. These centres are: heroin, ecstasy, and benzodiazepines. It means that the drug consumption has a 'modular structure'. The modular structure has

clear reflection in the correlation graph. The idea to merge the correlated attributes into 'modules' called as correlation pleiades is popular in biology, as explained in [3]. Among these three modules, we select the one with the highest mean correlation, thus the **Ecstasy**, whose distribution has been already showed in Figure 1.

Now that we define the label to be predicted (i.e. User vs Non-User in the Ecstasy case), in Figure 6, we observe the relationships between two features at a time. Given a data point, the value of its first variable is represented on the X axis, the second one on the Y axis.

3 Dataset Preparation

In this section, we address the preparation step, analysing different methods to deal with the pre-processing phase of our dataset. Specifically, we present methods to normalize data, to manage the presence of outliers, to deal with unbalanced data and dimensionality reduction.

3.1 Feature scaling

A feature scaling step is necessary since features have different range scales. In this context, we explore two different solutions, the Standard Scaler and the MinMax Scaler. The Standard Scaler applies the following formula to a given value x:

$$z = \frac{x - \mu}{\sigma} \tag{4}$$

where z is the standardized value, $\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$ and $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$, and N is the number of samples. On the other hand, for the MinMax Scaler, given x:

$$x_{sc} = \frac{x - \min_i(x_i)}{\max_i(x_i) - \min_i(x_i)} \tag{5}$$

where x_{sc} is the scaled value with MinMax Scaler and $i \in [1, N]$. In this case, we have that $0 \leq x_{sc} \leq 1$.

Once the features have been scaled, a boxplot can be drawn to compare them: specifically, in Figure 7, we distinguish between User and Non-User of Ecstasy, highlighting also the presence of outliers.

3.2 Outlier detection and management

The second pre-processing step which is taken into consideration concerns outliers and their management or, eventually, their removal. Outliers are data that largely deviate from the other observations. Outliers can be generated by novelties in the data, but also by experimental errors or human mistakes. There are many methods that are able to detect outliers in different ways. In the boxplot representation, it is important to define the Z-score method: Z-score indicates how many standard deviations a point is from the sample means. If the value is bigger then a certain threshold is discarded. Interquartile range consists in computing the difference between the first and the third quantile. Outliers here are defined as

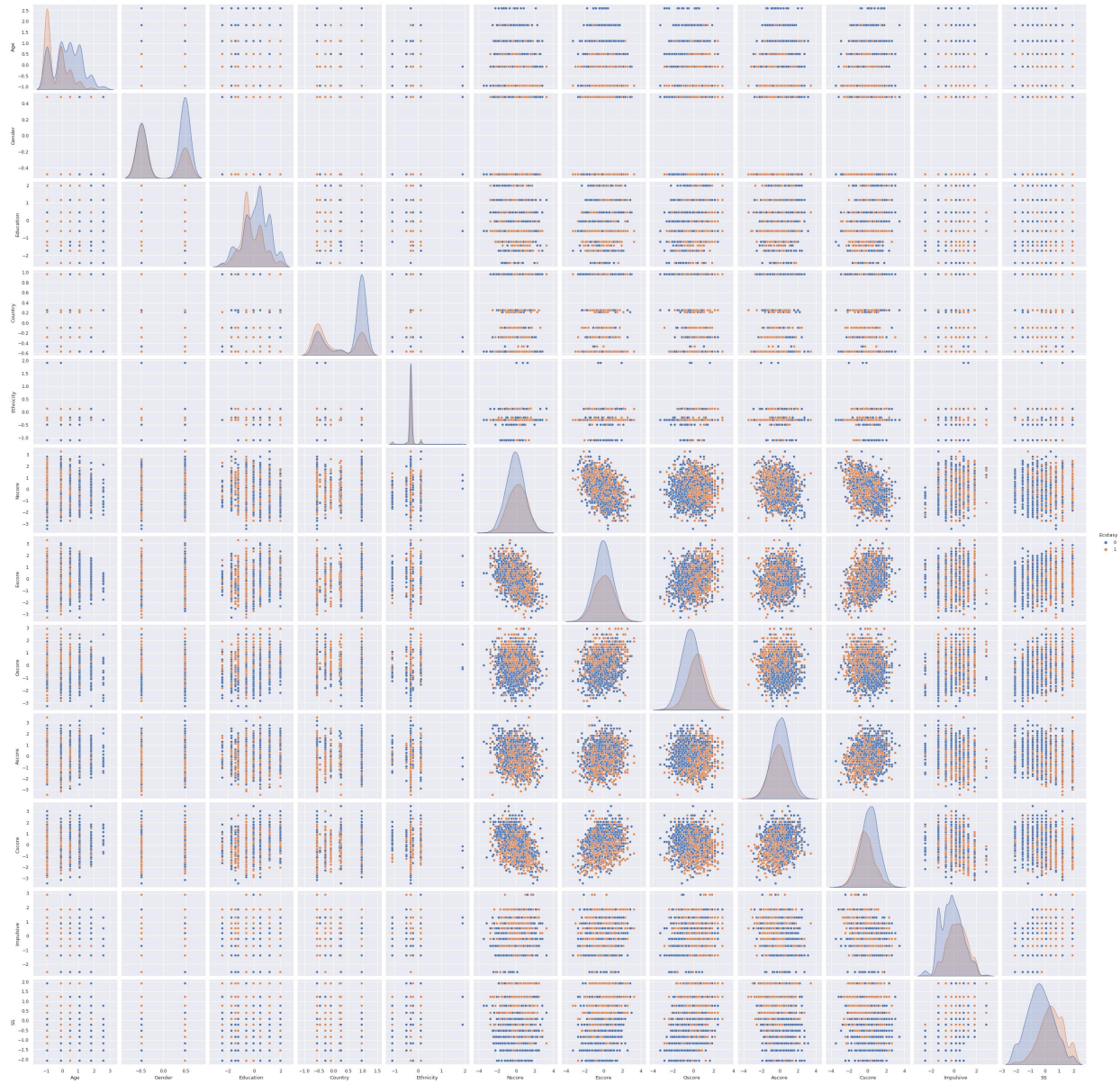


Figure 6: Pair plot of data with respect to Ecstasy label

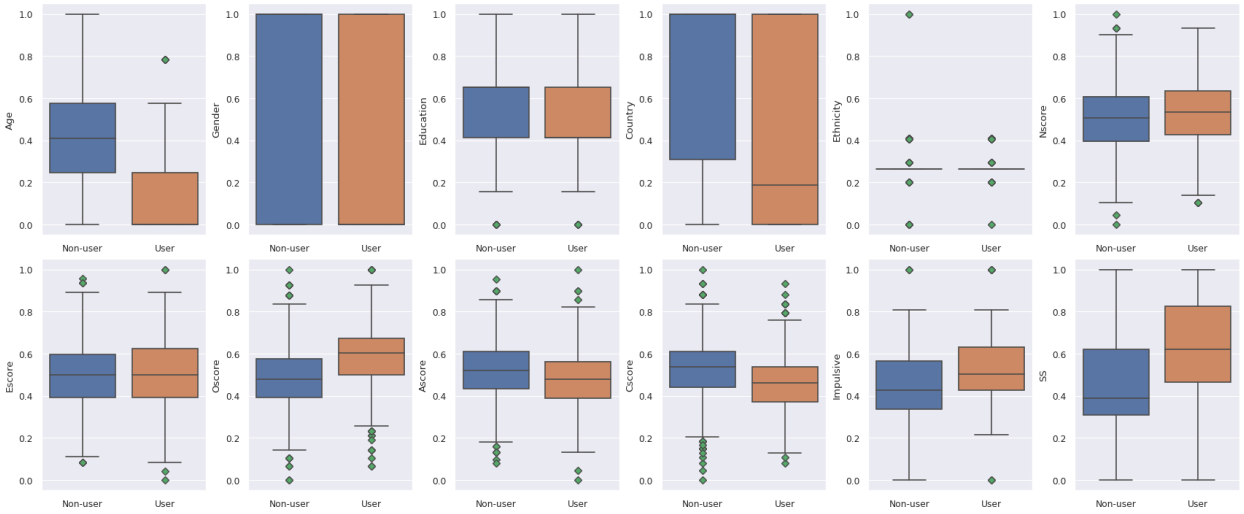


Figure 7: Boxplot of features distinguishing User and Non-User after scaling

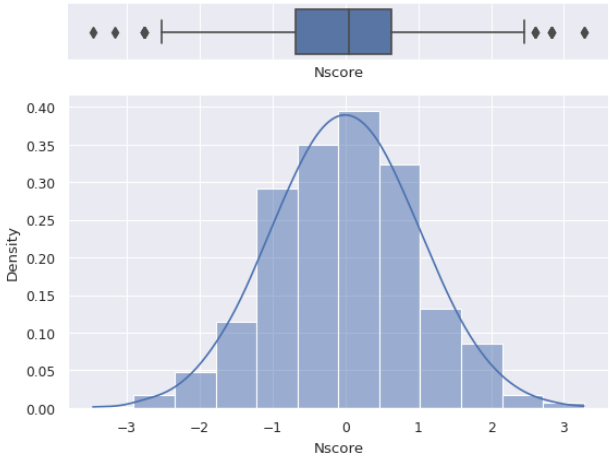


Figure 8: Nscore distribution and boxplot

observations that fall below Q1 1.5 IQR or above Q3 + 1.5 IQR. In the boxplot they are indicated as extreme individual points. The Z-score method has been used as one of the methods to deal with outlier. In Figure 8 we show the distribution and the boxplot for the *Nscore* feature.

Several methods to deal with outliers have been explored during the analysis.

3.2.1 Isolation Forest

Isolation Forest isolates anomalies by creating decision trees over random attributes. This random partitioning produces noticeable shorter paths for anomalies due to the fact that few instances are in smaller partitions and distinguishable attribute values are more likely to be separated in early partitioning. This means that when randomly partitioning the domain space, the anomaly will be detected in smaller number of partitions than a normal point. In

order to do that, we need to define an anomaly score for the decision making process:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{6}$$

where $h(x)$ is the path length of the observation x, $c(n)$ is the average path length of unsuccessful search in a Binary Search Tree (i.e. $c(n) = 2H(n-1) - 2(n-1)/n$, where H is the harmonic number) and n is the number of external nodes. Each observation is given an anomaly score and the following decision can be made on its basis:

- when $E(h(x)) \to c(n) \implies s \to 0.5$;
- when $E(h(x)) \to 0 \implies s \to 1$;
- when $E(h(x)) \to n-1 \implies s \to 0$;

Each observation is given an anomaly score and the following decision can be made considering that a score close to 1 indicates an anomaly, while scores much smaller than 0.5 indicates normal observations; if all scores are close to 0.5 then the entire sample does not seem to have clearly distinct anomalies.

3.2.2 Local Outlier Factor

The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers as outliers the samples that have a substantially lower density than their neighbors. In order to understand this algorithm we need to introduce:

- **K-distance** is the distance between the point, and it's K^{th} nearest neighbor. K-neighbors denoted by $N_k(A)$ includes a set of points that lie in or on the circle of radius K-distance;
- **Reachability Density (RD)**: $RD(X_i, X_j) = max(K\text{−}distance(X_j), distance(X_i, X_j))$, defined as the maximum of K-distance of X_j and the distance between X_i and X_j;
- **Local Reachability Density (LRD)**:

$$LRD_k(A) = 1/\sum_{X_j \in N_k(A)} \frac{RD(A, X_j)}{||N_k(A)||} \tag{7}$$

  which is the inverse of the average reachability distance of A from its neighbors. Intuitively according to LRD formula, more the average reachability distance (i.e., neighbors are far from the point), less density of points are present around a particular point. This tells how far a point is from the nearest cluster of points. Low values of LRD implies that the closest cluster is far from the point.

Then, we can introduce the **Local Outlier Factor (LOF)** is computed as:

$$LOF_k(A) = \frac{\sum_{X_j \in N_k(A)} LRD_k(X_j)}{||N_k(A)||} x \frac{1}{LRD_k(A)} \tag{8}$$

The LRD of each point is used to compare with the average LRD of its K neighbors. LOF is the ratio of the average LRD of the K neighbors of A to the LRD of A. Intuitively, if the point is not an outlier (inlier), the ratio of average LRD of neighbors is approximately equal to the LRD of a point (because the density of a point and its neighbors are roughly equal). In that case, LOF is nearly equal to 1. On the other hand, if the point is an outlier, the LRD of a point is less than the average LRD of neighbors. Then LOF value will be high.

Generally, if $LOF > 1$, it is considered as an outlier, but that is not always true. Let's say we know that we only have one outlier in the data, then we take the maximum LOF value among all the LOF values, and the point corresponding to the maximum LOF value will be considered as an outlier.

3.2.3 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a a density-based and unsupervised machine learning algorithm. It groups "densely grouped" data points into a single cluster, identifying clusters in large spatial datasets based on local density of the data points. DBSCAN clustering is able to identify outliers: indeed, data points that are not grouped within any cluster are data points that do not fall under any density agglomerate; those points are labeled with -1, and we consider them to be outliers.

DBSCAN requires only two parameters: *epsilon* and *minPoints*. Epsilon is the radius of the circle to be created around each data point to check the density and minPoints is the minimum number of data points required inside that circle for that data point to be classified as a Core point. In fact, to understand the clustering algorithms, we need to define three different kind of points:

- Core point: if the circle around it contains at least minPoints number of points;
- Border point: if the number of points is less than minPoints;
- Noise point: if there are no other data points around any data point within epsilon radius.

In Figure 9 we show a simple interpretation of these different kind of definitions for a point.

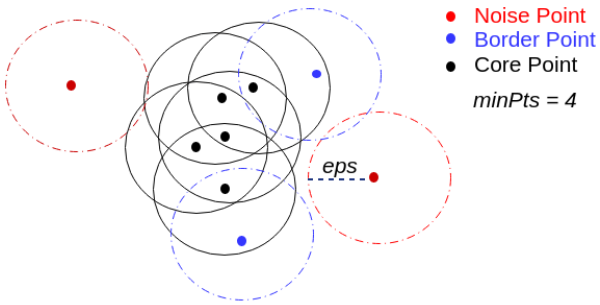


Figure 9: DBSCAN: Core, Border and Noise point

DBSCAN is very sensitive to the values of epsilon and minPoints. Therefore, it is very important to understand how to select the values of epsilon and minPoints. The value of minPoints should be at least one greater than the number of dimensions of the dataset. However, as a rule of thumb, usually it is twice the number of dimensions. The value of epsilon can be decided from the K-distance graph. The point of maximum curvature in this graph is representative of the value of epsilon.

3.3 Principal Component Analysis

PCA is an unsupervised technique commonly used for dimensionality reduction. The objective of this procedure is to find a data representation in a lower dimensional space. Since we want to lose as little information as possible, we project the data to n vectors such that the first vector (first principal component) has the direction of the largest variance of the data, while each subsequent principal vector is orthogonal to the previous one and goes in the direction of largest variance of the residual subspace. As a consequence, the principal components form an orthonormal basis in which the dimensions are uncorrelated. PCA is highly effective if deployed in feature spaces with higher dimensionality than the one we are considering. However, we decided to explore it in our pipeline, understanding if it is worth applying it.

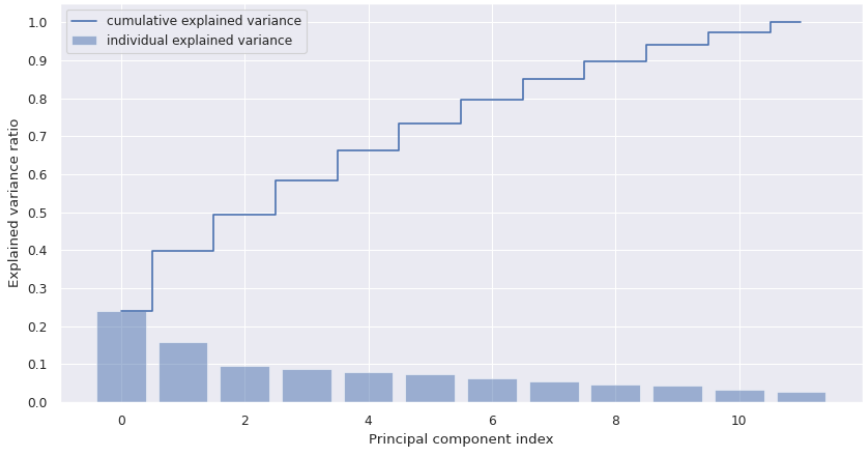


Figure 10: PCA's component and Variance Explanation

In Figure 10, we show the proportion of variance explained by each component. Already with 6 components 80% of the variance is explained, while with 8 components we reach more than 90% of explained variance. During our testing pipeline, PCA have been explored with different number of components.

3.4 Sampling techniques

As we have already seen in Figure 1, the dataset is unbalanced because the number of Ecstasy Non-User instances is higher than the User ones, as we could expect. This can affect the performances of the classifier increasing the misclassification of the less represented

class. To deal with this possible threat to the model, different techniques can be considered and deployed. In this case, we perform sampling, a procedure to balance the instances of each class in the training set. Specifically, four different techniques will be explored: two undersampling techniques and two oversampling techniques. Undersampling is a technique to balance uneven datasets by keeping all of the data in the minority class and decreasing the size of the majority class. On the other hand, oversampling artificially increase the samples of the minority class to balance the number of instances for the different target class values.

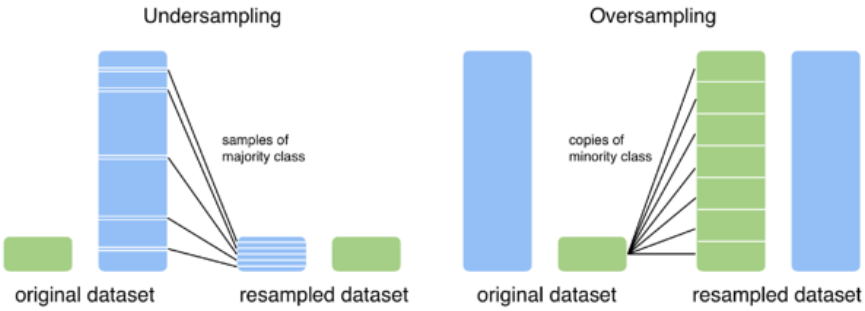


Figure 11: Undersampling vs Oversampling techniques

3.4.1 Undersampling

The **random undersampling method** pick n samples randomly from the class data that is more represented (Non-User class in our case), where n is the number of instances of the less represented one (the User class). In contrast, the data related to the less represented class are taken without changing.

The second explored undersampling method is called **Clustered Centroids**. This technique is based on computing for the majority class the centroid, obtained as the average of the points in the feature space. Therefore, the points that are far from the centroid are considered less important and are removed up until the sample is balanced.

3.4.2 Oversampling

The most widely used approach to synthesizing new examples is called the Synthetic Minority Oversampling TEchnique (SMOTE), which basically works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. Standard SMOTE works by utilizing a k-nearest neighbour algorithm to create synthetic data. Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found. A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

Borderline-SMOTE is a variation of SMOTE, which only makes synthetic data along the decision boundary between the two classes. In particular, there are two kinds of Borderline-

SMOTE: Borderline-SMOTE1 and Borderline-SMOTE2. Borderline-SMOTE1 also oversampled the majority class where the majority data are causing misclassification in the decision boundary, while Borderline-SMOTE2 only oversampled the minority classes.

Another variation of Borderline-SMOTE is Borderline-SMOTE SVM, or we could just call it **SVM-SMOTE**. The main differences between SVM-SMOTE and the other SMOTE are that instead of using K-nearest neighbors to identify the misclassification in the Borderline-SMOTE, the technique exploits the SVM algorithm. In the SVM-SMOTE, the borderline area is approximated by the support vectors after training SVMs classifier on the original training set. Synthetic data will be randomly created along the lines joining each minority class support vector with a number of its nearest neighbors.

K-means-SMOTE is another extension of SMOTE. K-Means SMOTE aids classification by generating minority class samples in safe and crucial areas of the input space. The method avoids the generation of noise and effectively overcomes imbalances between and within classes. This method works in three different step:

1. cluster the entire input space using k-means algorithm;
2. distribute the number of samples to generate across clusters:
   (a) filter out clusters which have a high number of majority class samples;
   (b) assign more synthetic samples to clusters where minority class samples are sparsely distributed;
3. oversample each filtered cluster using standard SMOTE.

ADASYN is another variation from SMOTE. ADASYN takes a more different approach compared to the Borderline-SMOTE. While Borderline-SMOTE tries to synthesize the data near the data decision boundary, ADASYN creates synthetic data according to the data density. The synthetic data generation would be inversely proportional to the density of the minority class. It means more synthetic data are created in regions of the feature space where the density of minority examples is low, and fewer or none where the density is high. In simpler terms, in an area where the minority class is less dense, the synthetic data are created more. Otherwise, the synthetic data is not made so much.

4 Methods

In this section we present first of all the flow we follow to carry out our analysis, then we define the metrics we used to evaluate the performance of the tested model, and finally we present the classification model we explored in our analysis.

4.1 Training pipeline

The training pipeline is showed in Figure 12, below. Notice that, this flow of work has been carried out considering different type of:

- outlier removal methods;

- sampling methods, both oversampling and undersampling;
- normalization and standardization methods;
- binary classifiers.

A thorough search of the best combination has been carried out considering all these different alternatives, where we define as "best" the one with the best metric values. Metrics considered will be described in the next section (section 4.2), while the K-Fold Cross validation will be define after that (section 4.3)

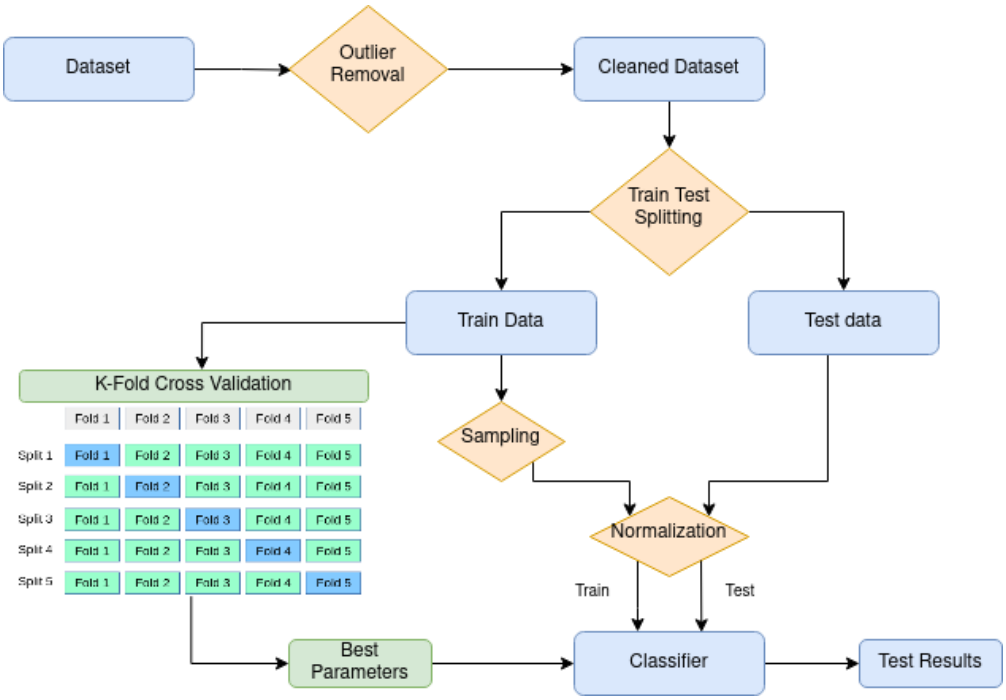


Figure 12: Diagram pipeline of the analysis

4.2 Metrics

To evaluate and compare results of different algorithms we take into consideration some metrics, which gave us an overview on the overall performance among different classifier. Specifically, in the following section, we will deal with Accuracy, Confusion Matrix, F1 Score and Receiver Operator Characteristic

4.2.1 Accuracy

Accuracy is one of the most common and intuitive metric of evaluation. It is the ratio of number of correct predictions to the total number of input samples, or:

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total number of Predictions Made}} \tag{9}$$

This metric works well when the dataset is balanced. Indeed, if we have similar number of samples for each class, accuracy is a great measure to look at. In our case, the dataset is unbalanced, but we deploy oversampling to deal with this problematic.

4.2.2 Confusion Matrix

The Confusion Matrix is a technique for summarizing the performance of a classification algorithm. Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset. Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.

The number of correct and incorrect predictions are summarized with count values and broken down by each class. In a two-class problem as ours, we are often looking to discriminate between observations with a specific outcome, from normal observations, such as User and Non-User. In this way, we can assign the event row as "positive" and the no-event row as "negative". We can then assign the event column of predictions as "true" and the no-event as "false". Thus, we define:

- True Positive (TP), for correctly predicted event values;
- False Positive (FP), for incorrectly predicted event values;
- True Negative (TN), for correctly predicted no-event values;
- False Negative (FN), for incorrectly predicted no-event values.

In the table 2, we show the Confusion Matrix scheme.

		Predicted Class	
		P	N
Actual Class	P	True Positive (TP)	False Negative (FN)
	N	False Positive (FP)	True Negative (TN)

Table 2: Confusion Matrix

4.2.3 F1 Score

F1 Score is an appropriate metric to be considering when dealing with an unbalanced dataset. F1 Score is the Harmonic Mean between precision and recall. In order to define this metric, we need to introduced two additional metrics: Precision and Recall. The Precision is a measure of the correctly identified as positive in the all positive predicted:

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

The recall is a measure of the correctly identified as positive in the all the really positive:

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (i.e. how many instances it classifies correctly), as well as how robust it is (i.e. it does not miss a significant number of instances). High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model. Mathematically, it can be expressed as:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{12}$$

F1 Score tries to find the balance between precision and recall. We will be used in our hyperparameters tuning and best algorithm search to evaluate which perform better.

4.2.4 Receiver Operator Characteristic

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds and it is one of the most widely used metrics for evaluation of binary classification problem. The ROC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. Before defining ROC, we need to define two terms:

- **True Positive Rate** (Sensitivity), that corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points:

$$TPR = \frac{TP}{FN + TP} \tag{13}$$

- **False Positive Rate**, that corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points:

$$FPR = \frac{FP}{TN + FP} \tag{14}$$

These terms both have values in the range $[0, 1]$. FPR and TPR both are computed at varying threshold values such as $(0.00, 0.02, 0.04, \ldots, 1.00)$ and a graph is drawn. AUC is the area under the curve of plot FPR vs TPR at different points in $[0, 1]$. The ROC curve is a good way to compare classifiers when the accuracy parameters is not enough. Generally, bigger is the AUC better are the performances.

4.3 K-Fold Cross Validation

The evaluation of a machine learning algorithm can be very difficult because the metric that is used to evaluate the algorithm can be affected by biases related to the data that are in

the validation set. Particularly, the validation is a common operation made to tune the hyperparameters and get improved and more robust results. This technique removes the aleatory aspect of the train-test choice in the splitting phase, which may lead to data which are not representative of the whole distribution. The main strategy of this approach is to divide the training set in k splits. The training and the validation operations are performed k time using a different split as validation and the others as training. Each time is obtained a score and the final result of the cross validation is the mean of the k scores obtained.

The scheme of a 5-fold cross validation has been already reported in figure 12. As in our case, the dataset is split in 5 fold, trained on 4, and validated on the remaining one, rotating the test fold at each iteration. The metrics are then computed as mean of the metrics across the different splits.

4.4 Classification models

In this section we present and define the different algorithms considered in our pipeline in order to predict the User vs Non-User case in the Ecstasy drug. Let us underline, once again, that we are dealing with a binary classification problem, with an unbalanced dataset that has been already pre-processed, sampled and normalized.

4.4.1 Decision Tree

Decision tree is a simple surpervised algorithm that models a set of sequential and hierarchical decision rules: This method divides the predictor space in non-overlapping regions that are high dimensional rectangles. The label assigned is the most commonly occurring class in that region. The algorithm uses a greedy approach and at each step chooses the best predictor to split according to some criteria. The criteria that have been explored in our analysis are:

- GINI index $= 1 - \sum_{k=1}^{N} p_{mk}$, which represents a measure of impurity of the node. It calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly.
- Cross Entropy $= \sum_{k=1}^{N} p_{mk} \cdot \log(p_{mk})$, which represents the measurements of the randomness in the data points.
- LogLoss $= -\frac{1}{N} \sum_{k=1}^{N} [y_k ln(p_k) + (1 - y_k) ln(1 - p_k)]$, which is indicative of how close the prediction probability is to the corresponding actual. The more the predicted probability diverges from the actual value, the higher is the log-loss value.

Another important parameters in the Decision Tree algorithms is the maximum depth the tree can reach. The higher the depth, the more the tree is able to define more specific split and leaf nodes, with, however, higher risk of overfitting. In Figure 13, we report a Decision Tree and its corresponding division in the Bidimensional space. Generally, Decision Trees are highly interpretable but often tends to overfit, compromising its performance.

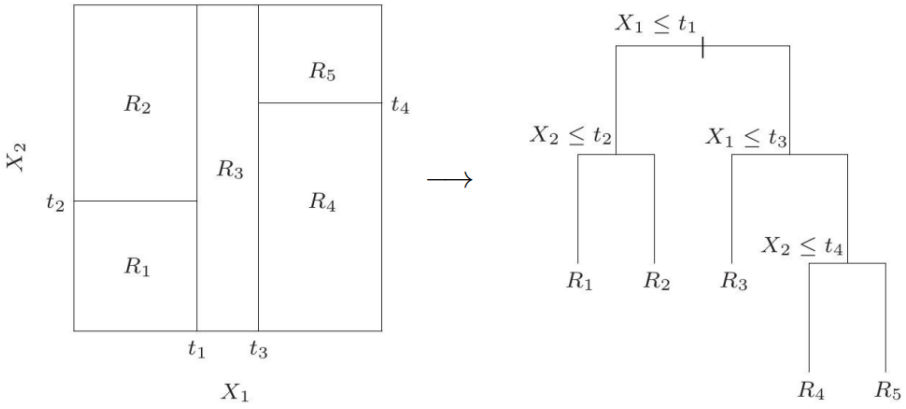


Figure 13: Decision Tree in Bidimensional space and corresponding Decision Tree

4.4.2 Random Forest

A popular way to overcome the limitation of decision trees, is building an ensemble of trees, called Random Forest. Random Forest is a supervised algorithm that is based on the bagging of decision trees. The bagging method consists in creating N training set starting from the original one with bootstrap. On each of these datasets a decision tree is trained using only a subset of the features (typically $\sqrt{N}$). The results given by each decision tree are aggregated to obtain the overall result. More in details, each decision tree produces a response and the overall results is the most common one between the all trees. The result of this method is much more robust and stable. In Figure 14, we show a scheme of the Random Forest method. For what concerns hyperparameters, we consider the maximum depth of the single trees of the forests (as for the Decision Tree before) and the number of estimators, which represents the number of trees in the forest.

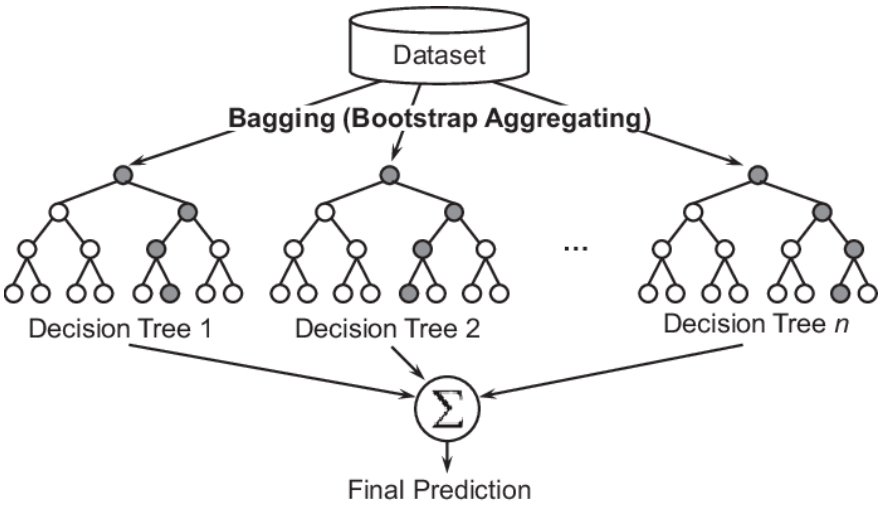


Figure 14: Random Forest with Bagging

4.4.3 K-nearest neighbors

Nearest Neighbor algorithms are among the simplest of all machine learning algorithms. The idea is to memorize the training set and then to predict the label of any new instance on the basis of the labels of its closest neighbors in the training set. 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 labels in a way that makes close-by points likely to have the same label. To compute the distances between points, we use the Minkowski distance in a p-dimensional space, thus:

$$d_p = (\sum_{i=1}^{D} |a_i - b_i|^p)^p. \tag{15}$$

Once the distance has been computed, the k-nearest neighbors are considered and the label is assigned by majority voting. The strong point of K-Nearest Neighbors is that is very easy to implement and to use. On the other hand, it can be very expensive to compute all the distances for large dataset and there may be memory issues. Choosing a small k means that the classification may be more sensitive to outliers, but it's not a good idea making it too big otherwise neighbors that are far apart (maybe irrelevant) will be taken into account. In Figure 15 we show the schema of KNN when a new data point need to be classified in one of the two classes.

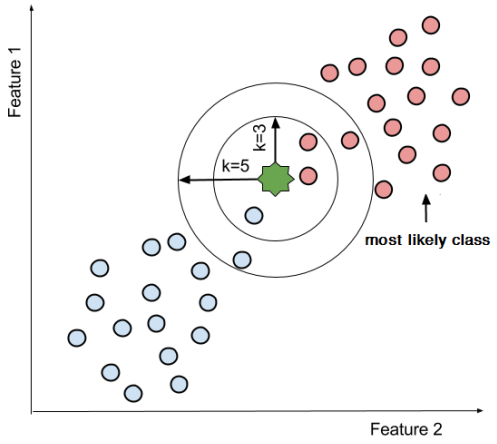


Figure 15: K-Nearest Neighbors scheme

4.4.4 Logistic Regression

Logistic Regression is a General Linear Model that learns a function $h : \mathbf{R} \to [0, 1]$, which is often used for classification task since the output can be interpreted as a probability distribution, where $h(x)$ is the probability that $x = 1$. In order to do this, we define the sigmoid function:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} = \frac{e^{<w,x>}}{1 - e^{<w,x>}} \tag{16}$$

After some computation, the above formula become the log-odds, or logit:

$$\log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 X \tag{17}$$

which is linear in X, thus increasing X the logistic regression function will increase by a quantity β_1. The two coefficients β_0 and β_1 are estimated with the maximum likelihood method (i.e. the two parameters are chosen to maximize the likelihood function).

4.4.5 Support Vector Machine

Support Vector Machine is a supervised learning algorithm, whose main goal is to find an hyperplane that divides the training data belonging from different classes. The classification of new instances is performed considering their position related to the hyperplane.The choice of the hyperplane can be very different and depends on the data that are on the boarder of the distributions named support vectors. In Figure 16, we show a possible representation in a two-dimensional space of a binary classification of SVM.

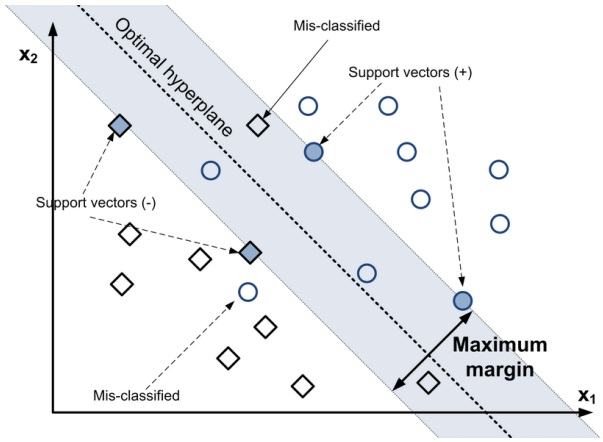


Figure 16: (Soft) Support Vector Machine scheme

A training set S is linearly separable if:

$$\forall i \quad \exists w \in \mathbf{R}^d, b \in \mathbf{R} : y_i(< w, x_i > +b) > 0 \tag{18}$$

For any separable training set there are a lot of possible hyperplanes: the objective is considering the ones that maximize the margin, i.e., the minimal distance between a point of the training set and the hyperplane itself. In order to do that, the hard-SVM problem is defined as follow: beginequation

$$\begin{aligned} &\text{argmax}_{(w,b):||w||=1} \left[\min_{i \in [m]} || < w, x_i > +b|| \right] \\ &\text{s.t.} \quad \forall i, y_i(< w, x_i > +b) > 0 \end{aligned} \tag{19}$$

The solution of the above equation is found with:

$$\hat{w} = \frac{w_0}{||w_0||} \qquad\qquad \hat{b} = \frac{b_0}{||b_0||} \tag{20}$$

This solution is supported by the points that are exactly at distance $1/||w_0||$ from the separating hyperplane. The vectors identified by this solution are called Support Vectors.

The assumption made by the Hard-Margin SVM problem is that data are linearly separable; however, often this is not the case. Soft-Margin SVM is a relaxation of this algorithm that can be applied even when data do not satisfy the strong assumption of linearly separable data. This relaxation consists of allowing the constraint to be not always strictly satisfied, introducing some slack variables $\xi_1, ..., \xi_m$ in the constraint:

$$\begin{aligned} &\text{argmin}_{w,b,\xi} \left(\lambda||w||^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i \right) \\ &\text{s.t.} \quad |< w, x_i > +b| \geq 1 - \xi_i \\ &\sum_{i=1}^{m}\xi_i \leq C \end{aligned} \tag{21}$$

where the parameter ξ_i measures how much the constrains is not being satisfied, C is the non-negative tuning *Regularization parameter*, which determines the number of violations we want to allow. The higher the value of C, the higher the number of errors we are allowing.

If data are non-linearly separable, the **kernel trick** can be used. A non-linear mapping $\psi : X \to F$ is applied to map the instances in a higher dimensional space is a feature space subset (i.e. F). However, computing linear separators in very high dimensional space can be very computational expensive, so it is exploited a kernel that refers to the inner product in the feature space. Formally, given an embedding ψ of some domain X, we define the kernel function as:

$$K(x, x) =< \psi(x), \psi(x) > \tag{22}$$

Using this trick is possible to make training and prediction using K alone without applying ψ. There are different kind of kernel functions that can be used:

- Linear: $K(x, x') = x \cdot x'$
- Polynomial: $K(x, x') = (\gamma \cdot x \cdot x' + r)^d, \gamma > 0$
- Gaussian kernel RBF (Radis Basis Function): $K(x, x) = \exp(\gamma \cdot ||x - x'||^2), \gamma > 0$
- Sigmoid: $K(x, x') = \tanh(\gamma \cdot x \cdot x' + r)$

where γ is an hyperparameter that changes the decision boundaries increasing the number of support vectors considered. Overall small values of γ set decision boundaries similar to the case of linear SVM while high values set more complex boundaries because more influenced by support vectors.

5 Results

In this section we show the results of our experiments. In the first part we deal with the choice of the most suited outlier removal method. Then, we analyze performance of the different classification methods presented in section 4.4: for each one of them we show the ROC curve (explained in section 4.2.4), the confusion matrix (section 4.2.4) and the hyperparameters search space.

5.1 Outliers detection method choice

In order to choose the best suited outliers detection method, we compare the three methods in terms of number of outliers removed, both numerically and graphically. In Figure 17, we show how the original distribution (considering boxplots) are impacted by the deployment of the different outliers removal methods. It is immediate to notice that the major impact occurs with the DBSCAN method, while with Isolation Forest and, mostly, with Local Outlier Factor the effect is much more softened.

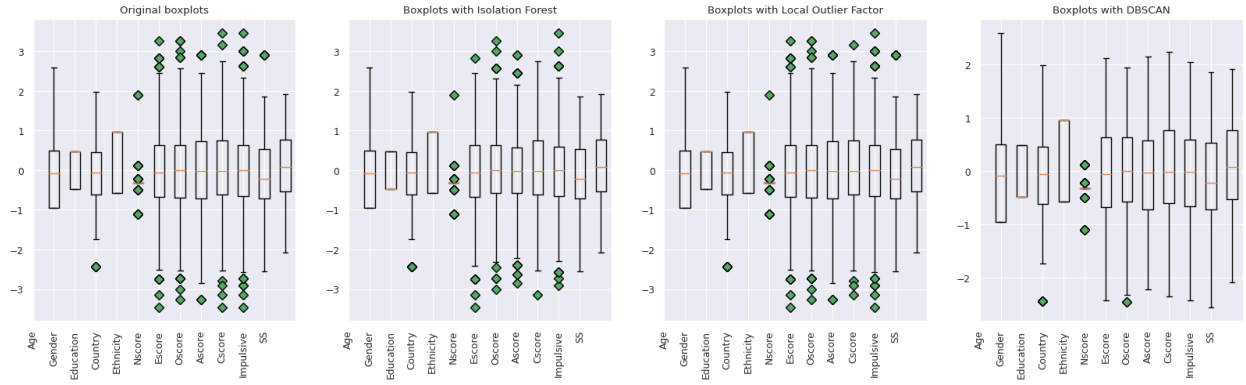


Figure 17: Comparison among the different outliers removal method

Additionally, if we look at number of data points removed from the dataset, it becomes clear how DBSCAN is the method that is able to correctly identify outliers, without compromising the original distribution of data. Table 3 shows some statistics about outliers with the three methods.

	Parameters		# data points	# data points removed	% data points removed
	Parameter	Value			
Starting point			1508	0	0
Isolation Forest	contamination	[0.1]	1357	151	10.01
Local Outlier Factor	[n_neighbors, contamination]	[20, 0.1]	1500	8	0.53
DBSCAN	[ϵ, minPoints]	[0.15, 12]	1351	157	10.41

Table 3: Comparison among the different outliers removal method

5.2 Classification Model choice

In this section we present the results of the different classification model presented in Section 4.4. Before the hyperparameters search for each model, it is important to select the appropriate Scaler and the appropriate sampling method. Two separate grid search are carried out which brings us to use:

- MinMaxScaler as Scaler method (seen in section 3.1);
- ADASYN as Oversampling method (seen in section 3.4.2); during this search we have found that, in case of unbalanced dataset with a relatively small number of samples

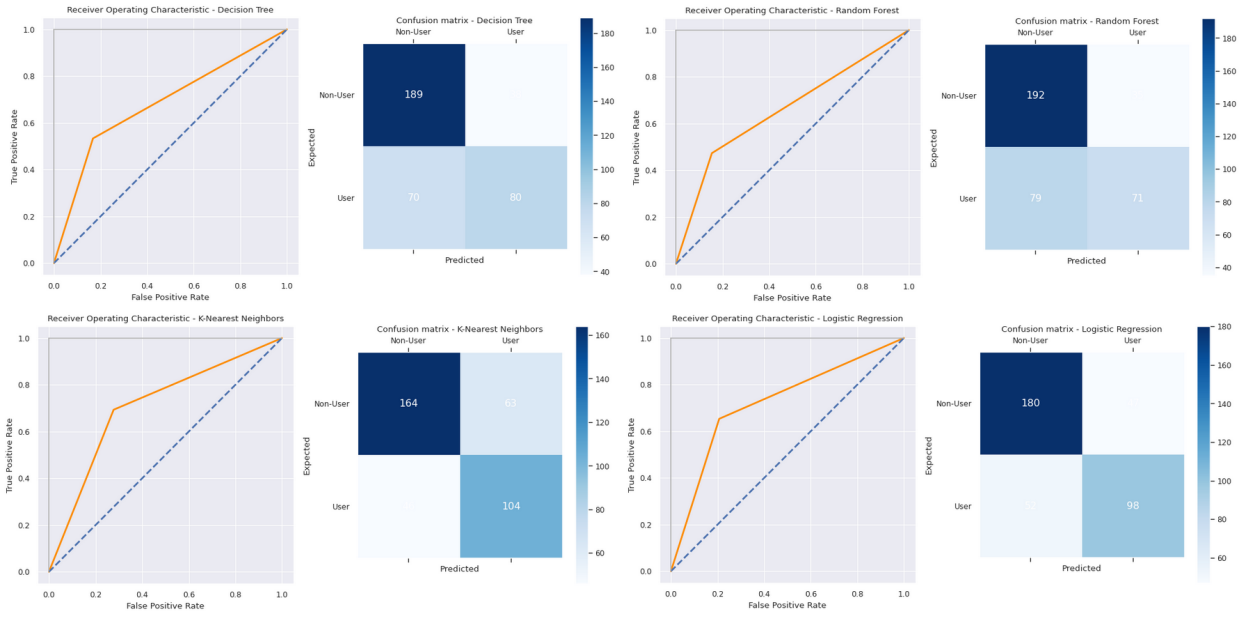
as ours, oversampling methods performs overall better than undersampling ones. A logical explanation might be that the samples not considered from the majority class in the undersampling case contain important and representative information for the algorithm. In our case, ADASYN is run with n_neighbors=5.

Table 4 shows for each model the hyperparameters search space, the best combination of them (highlighted in **bold**) and the best results in terms of accuracy, f1 score and area under curve of the ROC curve.

Model	**Hyperparameter**		**Best Results**		
	Param	Value	Accuracy	F1-score	AUC
Decision Tree	criterion	[gini, entropy, **log_loss**]	0.674	0.667	0.673
	max_depth	[0, 2, **4**, 5, 6, 10, 15]			
Random Forest	n_estimators	[10, 50, 100, 250, 500, **750**, 1000]	0.690	0.664	0.667
	criterion	[gini, **entropy**, log_loss]			
K-nearest Neighbors	n_neighbors	1, 5, 7, 11, 13, 15, **20**]	0.695	0.689	0.695
	algorithm	[auto, ball_tree, kd_tree, **brute**]			
Logistic Regression	penalty	[l2, **None**]	0.753	0.734	0.734
SVM	C	[0.001, 0.1, 1, 10, **100**, 1000, 10'000]	0.759	0.755	0.766
	γ	[0.0001, **0.001**, 0.01, 0.1, 1]			
	kernel	[linear, poly, **rbf**, sigmoid]			

Table 4: Performance and Parameters of Classification models

For completeness' sake, we report here below the Confusion Matrix and the ROC curve for each classification model tested. These two graphics are found considering the best parameters for each model, which have been found with the 5-Cross Validation technique already explained. Logistic Regression and SVM are the models that perform better with respect to others.



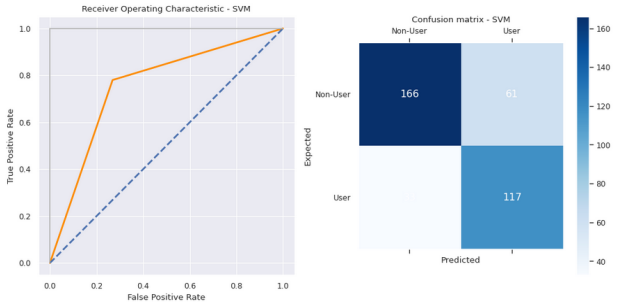


Figure 18: ROC Curve and Confusion Matrix for each model

5.3 Dimensionality Reduction Impact

In this last section, the objective is to obtain a model which is less complex and lighter, without compromising too much its performance. In order to do that, we consider the model with the best performance in the previous section and we see how it is impacted, in terms of performance, by dimensionality reduction.

In Figure 19 we can see how accuracy, f1 score and area under curve change with respect to the number of components selected for PCA. We can clearly see that performance of the model worsen with the reduction of the number of components, but just slightly manner. For example, take a look at the case with 4 components, which explain 78% of the total variance: in this case, all the three performance metrics considered do not fall under 70%. Considering the the initial maximum performance is 75%, a 5% drop in performance may be justified by the huge simplification of the model.

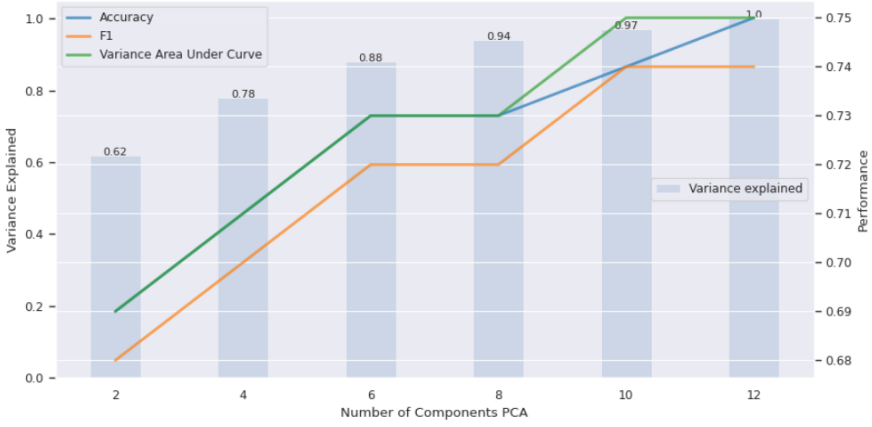


Figure 19: PCA impact on model performance

In Figure 20, we show the ROC Curve and the Confusion Matrix of the SVM model already considered in section 5.2, but deployed together with PCA with n_components=4, for dimensionality reduction. With respect to the previous case, without the PCA (Figure 18, we can immediately see that performance worsen, especially in terms of False Positive, but they are still satisfactory results with a much more simpler model. It is worth noticing also that, PCA works really well if paired up with Outlier Removal Method, as in our case.

Removing outliers before the dimensionality reduction step allows us to decrease the variance that need to be explained by the components of the PCA.

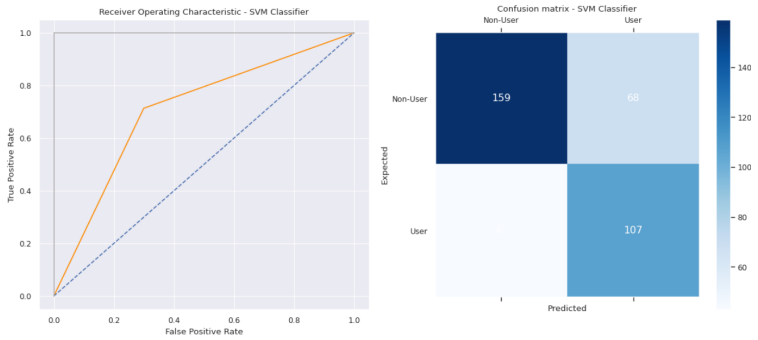


Figure 20: ROC Curve and Confusion Matrix of SVM with PCA (n_components = 4)

6 Conclusions

Figure 21 summarizes the performance of the different methods deployed, through a grouped barplot with accuracy, f1 score and auc for each model, together with the plot of the ROC curve. Logisitic Regression and Support Vector Machine Classifier provide the better results: they are comparable in terms of accuracy, but SVM is slightly better in terms of f1 score. This is due to the fact that, even if True Positive predicted are lower, also number of False positive and False Negative are lower with respect to the Logistic Regression Case. This means that, with SVM, we have a model that is less accurate when predicting the User case, but suffers less in the Non-User case: if we consider the working environment of a model like this one, where we deal with central nervous system psychoactive drugs, which have an high impact on the User, the fact that we can count on a model that is less susceptible to False Negative and Positive is more important than the higher number of True Positive.

In this analysis, we explored several methods at different levels of a classification pipelines: starting from the data exploration and selection of features, through the pre-processing step considering scalers, outliers, dimensionality reduction, and sampling techniques to deal with unbalanced data, up to the choice of the best suited classification for the task, considering multiple alternatives. The pipeline chosen is able to reach satisfactory results, with an accuracy of over 75% for the Ecstasy drug case. The best results were obtained considering the SVM Classifier Model, combined with MinMaxScaler as scaler, DBSCAN Clustering as outlier removal method and ADASYN as sampling method. PCA was then explored to reduce the complexity of our model: performance worsened, but they were still acceptable, considering the reduced dimensionality of our space.

Possible future works may take into consideration the drugs that have been excluded, for brevity's sake, by this analysis: for example, a model for each one of the drug could be tuned, trained and tested, one at a time. Another possible implementation may be considering the 7-class classification problem already explained in the Problem definition section (2.1).

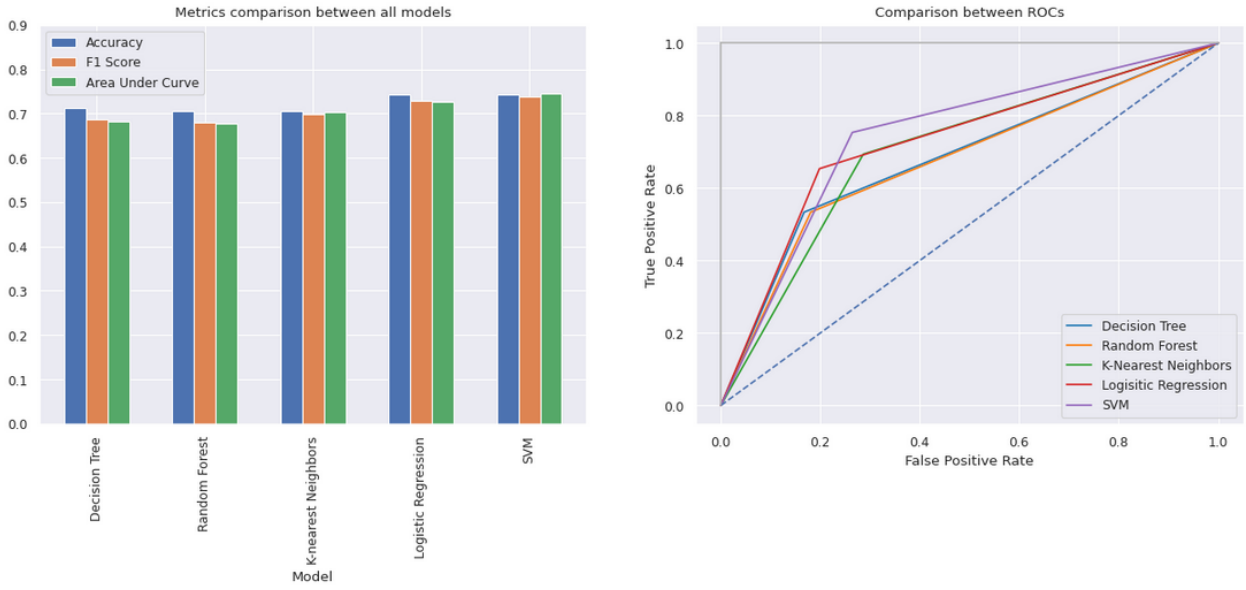


Figure 21: Comparison between models performance

References

[1] Dataset Link http://archive.ics.uci.edu/ml/datasets/Drug+consumption+(quantified)

[2] Code Link: https://github.com/lucabnf/drug-consumption

[3] Mitteroecker P, Bookstein F. The conceptual and statistical relationship between modularity and morphological integration. Systematic biology. 2007; 56(5):818-836.

[4] Data Science and Machine Learning, Mathematical and Statistical Methods, Dirk P. Kroese, Zdravko I. Botev, Thomas Taimre, Radislav Vaisman, 25th November 2020

[5] Understanding Machine Learning: From Theory to Algorithms, Shai Shalev-Shwartz, Shai Ben-David, 2014, Cambridge University Press, http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning