Mathematics in Machine Learning

Online Shoppers Purchasing Intention Dataset

Delleani Mattia
s288854

Politecnico di Torino

July 2021

Contents

1 Introduction

Nowadays data driven decision-making processes are widely applied to enhance business by identifying, anticipating and satisfying customers' needs, especially in e-commerce sector. The goal of this project it to analyze the Online Shoppers Purchasing Intention Dataset [5] created by C. Okan Sakar (Bahcesehir University, Istanbul, Turkey) and Yomi Kastro (Inveon Information Technologies Consultancy and Trade, Istanbul, Turkey) and through the implementation of a data science pipeline we make a comparison among different machine learning algorithms that try to solve a binary classification problem that consists in predict if a connected user to the e-commerce website ends with shopping or not.

2 Data overview and Description

The dataset consists of feature vectors belonging to 12,330 sessions and was formed so that each session would belong to a different user in a 1-year period to avoid any tendency to a specific campaign, special day, user profile, or period. There are 18 features: 10 numerical and 8 categorical attributes.

2.1 Categorical features

- **Operating System**: operating system used by the user.
- **Browser**: browser used by the user.
- **Region**: from which geographic region the access is made.
- **Traffic type**: type of data traffic.
- **Visitor type**: if the users is a new visitor or returned.
- **Month**: month of access.
- **Weekend**:(boolean) if the access is made on weekend or not.
- **Revenue**:(boolean) class label, if the session ends with shopping or not.

.

2.2 Numerical features

- **Administrative, Administrative Duration, Informational, Informational Duration, Product Related and Product Related Duration**: represent the number of different types of pages visited by the visitor in that session and total time spent in each of these page categories. The values of these features are derived from the URL information of the pages visited by the user and updated in real time when a user takes an action, e.g. moving from one page to another.

- **Bounce Rate**: refers to the percentage of visitors who enter the site from that page and then leave ("bounce") without triggering any other requests to the analytics server during that session measured by Google Analytics.

- **Exit Rate**: for a specific web page is calculated as for all pageviews to the page, the percentage that were the last in the session, measured by Google Analytics.

- **Page Value**: represents the average value for a web page that a user visited before completing an e-commerce transaction, measured by Google Analytics.

- **Special Day**: indicates the closeness of the site visiting time to a specific special day (e.g. Mother's Day, Valentine's Day) in which the sessions are more likely to be finalized with transaction. The value of this attribute is determined by considering the dynamics of e-commerce such as the duration between the order date and delivery date. For example, for Valentina's day, this value takes a nonzero value between February 2 and February 12, zero before and after this date unless it is close to another special day, and its maximum value of 1 on February 8.

3 Data Exploration

Before proceeding with data preprocessing and classification algorithms, we need to better understand the properties of the dataset in order to conduct a better analysis. So, studying the data and their distributions is the first

step to acquire a thorough understanding of the problem and think about the possible solutions.

3.1 Missing values and NaN/Null

One of the first thing to do is to check the presence of missing values, Nan and Null values because those values can raise errors during the development of the data pipeline and can affect the classifiers. There are different way to handle missing data, such as: replacing them with a custom value (i.e. most present class value, mean, median..) or dropping all the records containing them.
As said before this dataset is made of 12,330 records and simply exploiting the Pandas Library is it possible to check if our dataset contains any of this.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12330 non-null  int64
 1   Administrative_Duration  12330 non-null  float64
 2   Informational            12330 non-null  int64
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  int64
 5   ProductRelated_Duration  12330 non-null  float64
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  category
 11  OperatingSystems         12330 non-null  category
 12  Browser                  12330 non-null  category
 13  Region                   12330 non-null  category
 14  TrafficType              12330 non-null  category
 15  VisitorType              12330 non-null  category
 16  Weekend                  12330 non-null  category
 17  Revenue                  12330 non-null  float64
dtypes: category(7), float64(8), int64(3)
memory usage: 1.1 MB
```

Figure 1: Dataset information

There are no missing, NaN or Null values in the dataset as we can see in Figure 1.

3.2 Data distribution

3.2.1 Class label

Another important thing to check is if the class label distribution is balanced or imbalanced, because in case of imbalance among classes the predictive model developed using conventional machine learning algorithms could be biased and inaccurate. This happens because Machine Learning Algorithms are usually designed to improve accuracy by reducing the error. Thus, they do not take into account the class distribution proportion or balance of classes.
A dataset is "balanced" if it contains equal or almost equal number of samples from each class and "unbalanced" if the samples from one of the classes outnumbers the others.

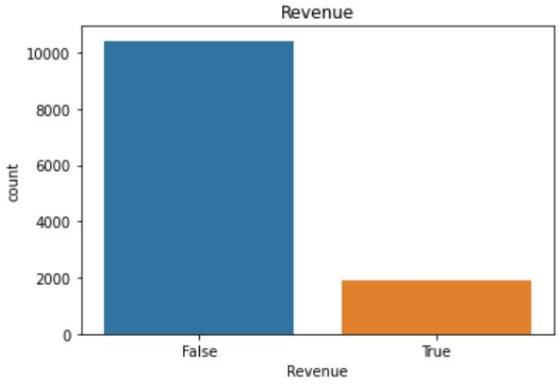


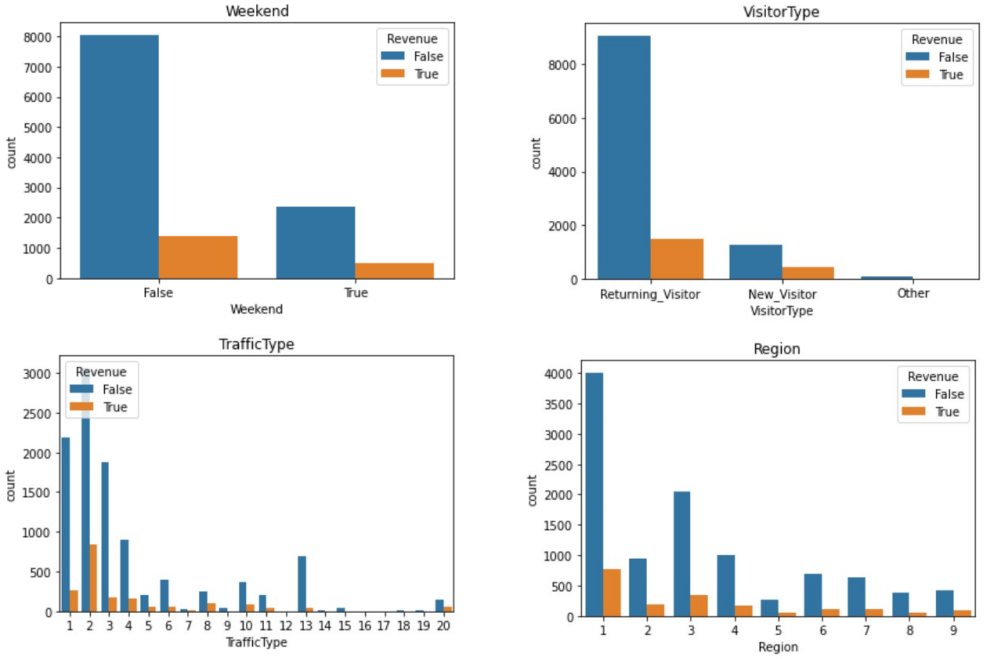
Figure 2: Class label distribution

As Figure 2 shows there is a clear imbalance in the class label data. The data is skewed in favor of the *False* class, which can be translated into visitor that did not shop, and this may cause problems during the training phase, because Machine Learning classifiers, such as Random Forests, in fact, are sensitive to the proportions of classes and with unbalanced training datasets do not work well. As a result, these algorithms tend to be biased towards the class with the largest proportion of observations, called *majority class*, and penalizes the underrepresented class, called *minority class*, that in our case is represented by the *True* class.
In this dataset the proportion are: 84.5% (10,422) are *False* class samples that did not end with shopping, and the rest 15.5% (1908) are *True* class

samples ending with shopping.

3.2.2 Categorical features

In order to analyze the categorical data distribution we can plot the frequencies of each feature with respect to the class label.
Looking at the frequencies plots in Figure 3 allows to easy understand some patterns present in the data. We can see for example that for the *Weekend* attribute most of the data, sold and not sold, has been seen during the working days (week days). Another pattern is that most of the access to the website are done through three Operating System (1, 2, 3). It is also important to notice that there are only 10 months of recording (January and April are missing), and the most popular month in terms of access is May, but the most popular in terms of sold products is November (maybe due to Christmas presents).



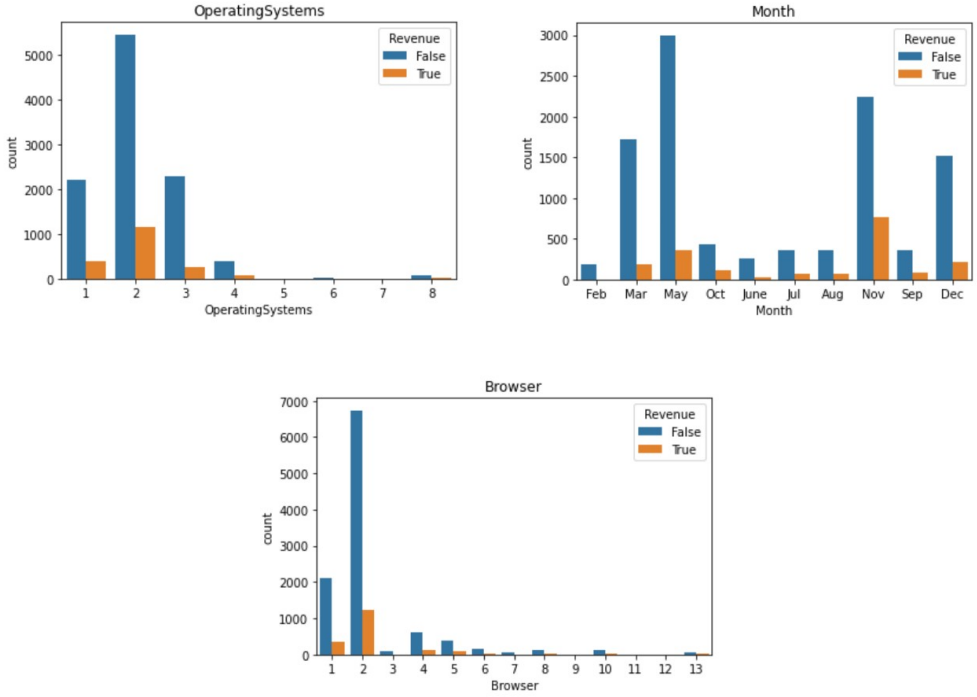


Figure 3: Categorical features distributions

3.2.3 Numerical features

For the numerical attributes the *boxplot* have been used. Boxplots are useful because provide information about data distributions in a clear way and specify five summary number (minimum, maximum, median, first quartile and third quartile), but also can tell you about outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.
As shown in Figure 4, for example, the *ExitRates* distribution for users who buy something has different properties with respect for users that exit the website without shopping.

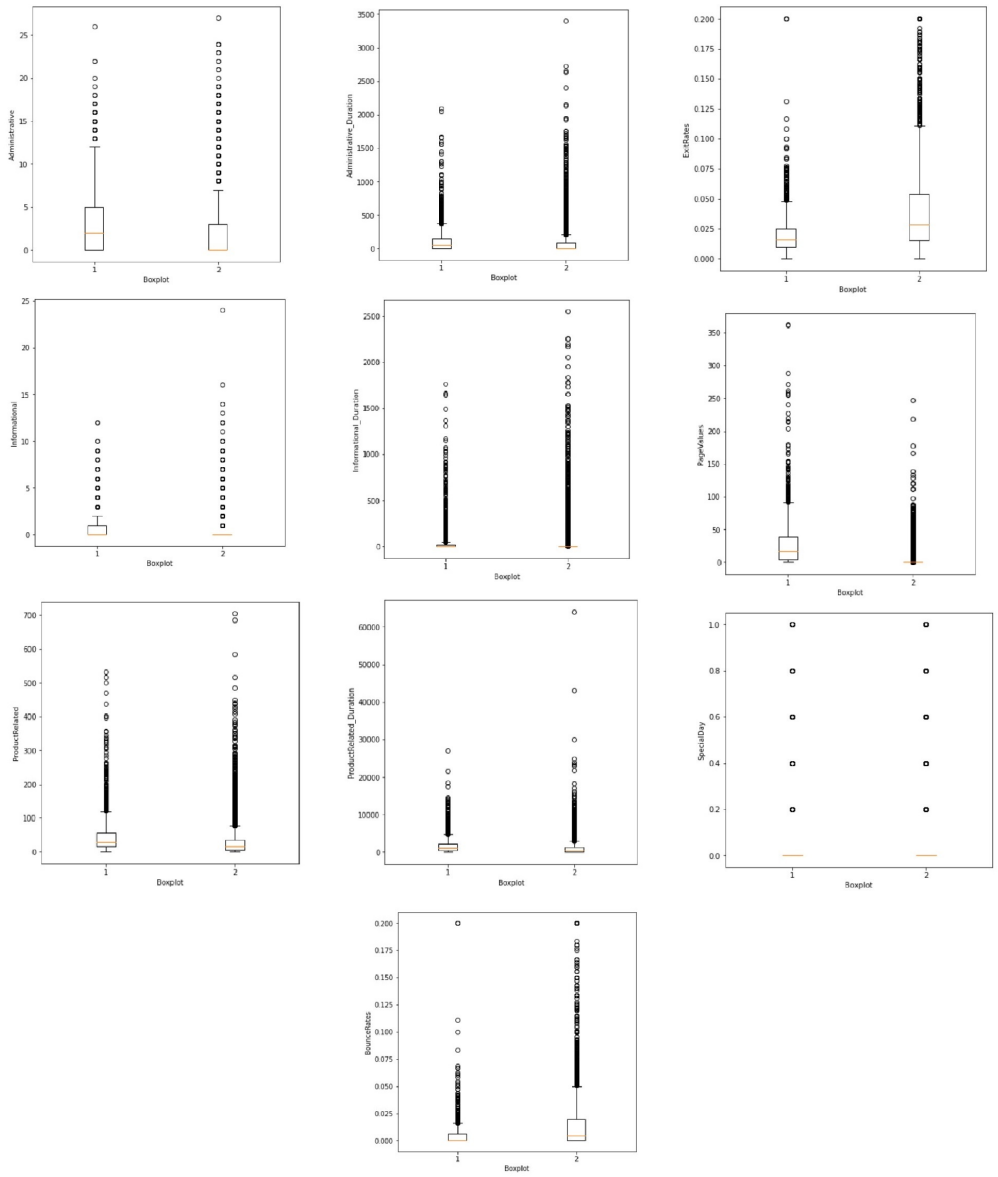


Figure 4: Numerical features boxplots. On x-axis "1" stands for *False* and "2" stands for *True*

3.3 Data Correlation

An important step of the data exploration is the computation of the correlation among the features because the performance of some algorithms can deteriorate if two or more variables are tightly related. An example is linear regression, where one of the offending correlated variables should be removed in order to improve the skill of the model. We may also be interested in the correlation between input variables with the output variable in order provide insight into which variables may or may not be relevant as input for developing a model. Correlation could be positive, meaning both variables move in the same direction, or negative, meaning that when one variable's value increases, the other variables' values decrease. Correlation can also be neutral or zero, meaning that the variables are unrelated. The Pearson correlation coefficient can be used to summarize the strength of the linear relationship between two data samples and is calculated as:

$$Cov(x, y) = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})$$

$$\rho(x, y) = \frac{Cov(x, y)}{S_x S_y}$$

where S_x and S_y are the standard deviation.
Two variables may be related by a nonlinear relationship, such that the relationship is stronger or weaker across the distribution of the variables and in this case, the Spearman's correlation coefficient can be used to summarize the strength between the two data samples.

As we can see in Figure 5 there are some strong correlation between attributes, such as:

- *Administrative* and *Administrative duration* has 0.94 of Spearman correlation and 0.6 of Pearson correlation. So, the correlation has a nonlinear relationship.

- *Informational* and *Informational duration* has 0.95 of Spearman correlation and 0.62 of Pearson correlation. So, the correlation has a non-linear relationship.

- *ProductRelated* and *ProductRelated duration* has 0.88 of Spearman correlation and 0.86 of Pearson correlation. Spearman correlation can also

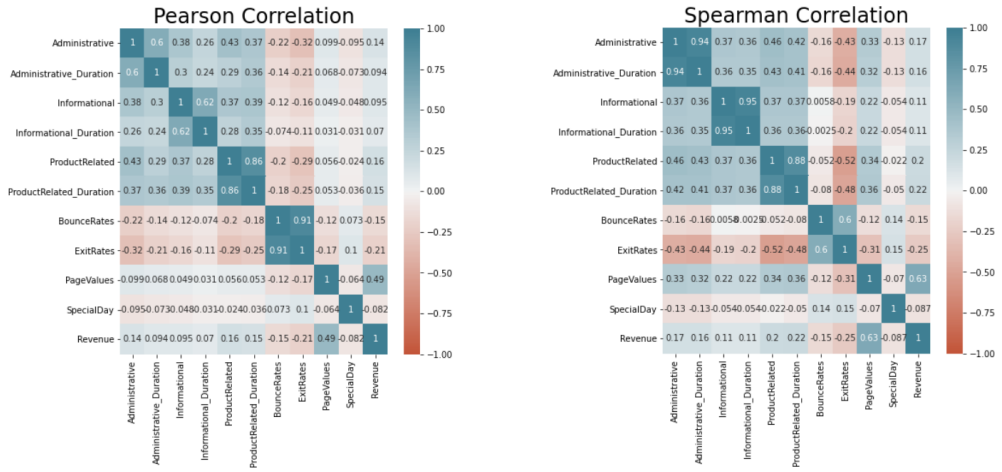


Figure 5: Heatmap correlations among variables

be used if there is a linear relationship between the variables, but will have slightly less power, so in this case the variables are linearly correlated.

- *BounceRates* and *ExitRates* has 0.6 of Spearman correlation and 0.91 of Pearson correlation. In this case the variables are linearly correlated.

4 Data Preprocessing

In this section, we present the data processing pipeline adopted to address the problem. Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format. Data preprocessing can be divided into three main tasks: data cleaning, which mainly consists in handling missing data and outliers; data transformation, which mainly consists in data normalization, feature selection and construction, discretization ecc.; and data reduction, which mainly consists in data aggregation, attribute selection and dimensionality reduction.

4.1 Outliers management & Correlated features

As first, since there are strong correlation in order to address the problem that this phenomenom may cause the attributes *ProductRelated Duration,*

BounceRates, Administrative Duration, Informational Duration have been removed from the dataset.
Then, as said in Section 3.1, there are no missing values so we can proceed to manage the outliers. There are different techniques to remove outliers, in this case Z-score and Interquartile Range (IQR) have been applied. Z-score consists in the z-standardization of the values and then reject values that are above a certain threshold. So, at first, standardization is applied:

$$zscore = \frac{X - \mu}{\sigma}$$

Then we remove values that are above or below a given value of z-score.
IQR method exploit the quartiles for the outliers detection. An observation is defined an outlier if it is 1.5 times the interquartile range greater than the third quartile (Q3) or 1.5 times the interquartile range less than the first quartile (Q1). As seen in Section 3.2.3 in Figure 4, the distributions are

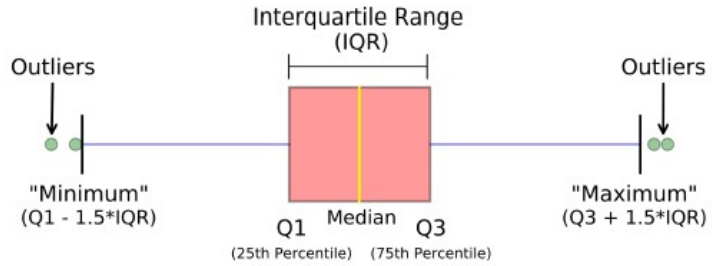


Figure 6: Interquartile Range method

very close to their mean values (seems that most of the values are repeted over the records) and there are a lot of records seen as outliers if we proceed with an IQR method. In fact, if the outliers detection is performed using IQR more than 50% of the dataset is considered as outlier. Proceeding using the Z-score method with threshold set to 3 (the observations that have a $|zscore| >= 3$) the outliers are 16% of the dataset size. Since those two methods have a significant impact in the dataset size and significant affect some features (e.g. *SpecialDay*), none of them have been used and we proceed through an ad hoc outliers detection with the introduction of threshold for some attribute (i.e. for *ProductRelated Duration* records are classified as outliers if their value is $>= 500$).

4.2 One Hot Encoding

Apart from the target variabile, there are 7 categorical features that have to be converted into numbers before proceeding with the Machine Learning algorithms. Binary variables, such as *Weekend*, can be translated into 0,1 econding, whereas the others are nominal ones and require a specific type of encoding. One approach can be to map it into numerical value, in this way each level of a categorical feature corresponds to an integer value. The problem is that numerical values implies an order, but for our data this is not the correct approach. To overcame this problem, the most commonly technique used is *One Hot Encoding*. For each categorical feature, each category value is converted into a new column: we assign 1 to the corresponding value and 0 to everything else. With this type of encoding no order is introduced but as drawback there is the dataset 'explosion', because the number of rows remained the same, but the number of features may increase a lot and affect the performaces of algorithms and may generate the curse of dimensionality problem. In our case, the number of columns of the dataset grew from 14 to 71.

4.3 Standardization

Standardization comes into picture when features of input data set have large differences between their ranges, or simply when they are measured in different measurement units. These differences may causes trouble to many machine learning models, so before proceeding with the data pipeline a z-score standardization has been computed as:

$$X' = \frac{X - \mu}{\sigma}$$

Standardization is required before many procedures and Machine Learning algorithms, such as: Principal Component Analysis (PCA), Clustering, KNN, SVM and many others. There are case in which standardization is not needed, for example with Logistic Regression and Tree based models, because those methods are not sensitive to the magnitude of variables.

4.4 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is is an unsupervised learning algorithm used for dimensionality reduction. Basically, it describes the compo-

sition of variances and covariances through several linear combinations of variables, based on the idea of preserving the variance among the data but reducing the number of features. In another term, it is about obtaining a unique set of orthogonal axes where the data has the largest variance. The reduction of dimensionality should be such that when dropping higher dimensions, the loss of data is minimum. This methods is based on the so called Principal Component (PC) that are a new coordinate system. Given data on $\mathbb{R}^d$, the hope is that the data points can be mapped in a lower dimensional space ($n < d$) and then can be approximately recovered through a second matrix. The PCs decompose the total variance of the data, so the sum of the variance among all the PCs corresponds to the total variance among original data. Each PC is an orthogonal linear transformation of the original data and the first PC is the one which have the higher variance, the last PC is the one having lower variance.

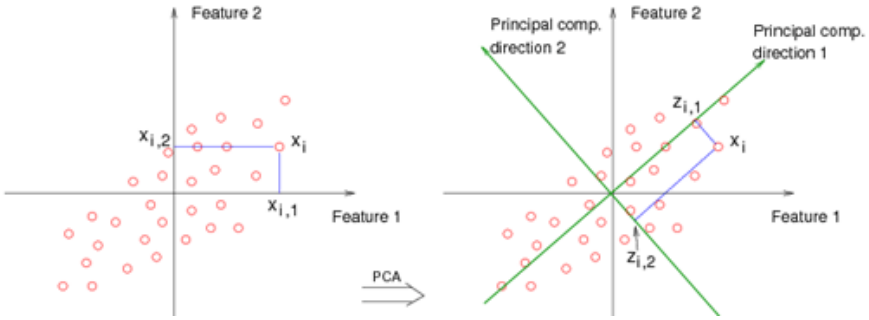


Figure 7: PCA in 2 dimensions

The dimensionality reduction with PCA can be summarized as:
Given data $D = x_1, x_2, \ldots, x_n$. Each x_i is a d-dimensional vector. Follow these steps to use PCA to reduce dimension to k:

1. Find the sample mean $\bar{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$
2. Subtract sample mean from the data $z_i = x_i - \bar{\mu}$
3. Compute the scatter matrix $S = \sum_{i=1}^{n} z_i z_i^t$
4. Compute eigenvectors $e_1, e_2, ..., e_k$ corresponding to the k largest eigenvalues of S

5. Let $e_1, e_2, ..., e_k$ be the columns of matrix $E = [e_1, e_2, ..., e_k]$

6. The desired y which is the closest approximation to x is $y = E^t z$

So after the computation of PCA, is common to plot the cumulative variance expressed by components to see how the variance of our dataset is represented by the components. In our approach, we wanted to proceed with components

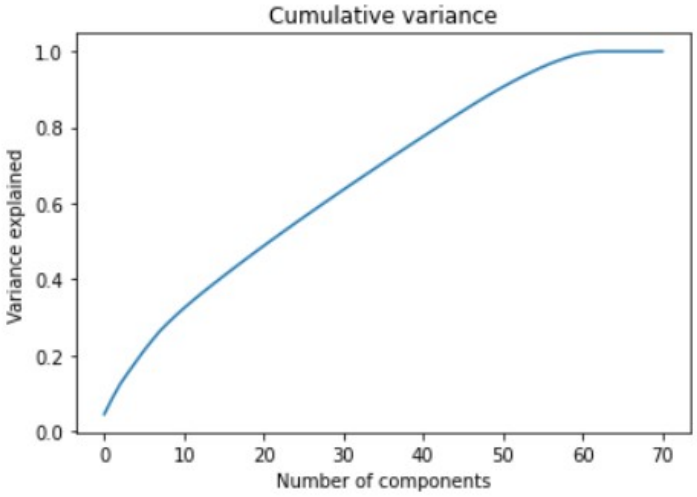


Figure 8: Variance expressed by number of components

that cumulative have 90% of variance and we obtained that we need 50 components over 71.

4.5 Feature selection

Another technique to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model is Feature Selection. Exploiting *SelectKBest* from the *scikit-learn* library of *Python* we are able to extract the best K features of given dataset. Selecting best features is important process when we prepare a large dataset for training. *SelectKBest* perform an Anova F-Test test in order to select the best features, that can be performed on both qualitative and quantitative predictors.

5 Train, validation and test set

5.1 Overview

In order to create a learning algorithm able to classify instances, we need to create a training set S, sampled from the entire dataset D, and to train our model triyng to minimize the training error. This approach is based on the *Empirical Risk Minimization paradigm*, a learning paradigm that starts from the idea that, if the sampled training set S is representative of the entire dataset D, minimizing the error on S we are also minimizing the error on D. Given an hypotesis h, the error on S is defined as:

$$L_s(h) = \frac{\{|i \in [m] : h(x_i) \neq y_i\}}{m}$$

where:

- m is the training size
- $h(x_i)$ is the predicted label
- y_i is the true label

The aim is to find the hypotesis h that minimize the loss $L_s(h)$. While creating the training set S, we want to preserve the proportionality among classes that are present the entire set D. The algorithms are then trained on the training set but they will be evaluated on the other part of the dataset D, the test set. In order to choose the best algorithm (refering to the algorithm itself or the tuned hyperparameters of the algorithm), we need to do a *model selection*: by looking at all the hypothesis h, we want the one that best performs on our data.
Using the test set for the model selection and hyperparameters tuning is not a good idea, because it may cause overfitting. In order to avoid this problem the original dataset is splitted into training, validation and test set, and the main idea behind this is that a very accurate estimation of the true risk can be obtained using both train and validation set: the first for constructing the model and the second to select the best model among all the proposed.

The validation procedure can be summarized as follows:

1. train the algorithm on the training set and construct the hypotesis class $H = h_1, ..., h_d$

2. choose the predictor h_s from H that minimizes the error over the validation set

3. use h_s on the test set

In other words, we are applying a *ERM* approach to the validation set instead of to the training set.

5.2 Cross validation (K-Fold)

Cross-validation is a statistical method used to estimate the performance and perform parameters tuning of machine learning models. It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited. According to this method, the original training + validation set is partitioned into k equal sized subsamples with $\frac{n}{k}$ records each. Then for each fold in the dataset, build the model on k–1 folds of the dataset. Then, test the model to check the effectiveness on k^{th} fold. Repeat this until each of the k-folds has served as the test set and, finally, we compute the average of our k recorded accuracy, called the cross-validation accuracy, and will serve as your performance metric for the model. As said in the previous section, we want training and test to have similar distributions so we stratify to maintain the proportions of the data, considering the binary classification.

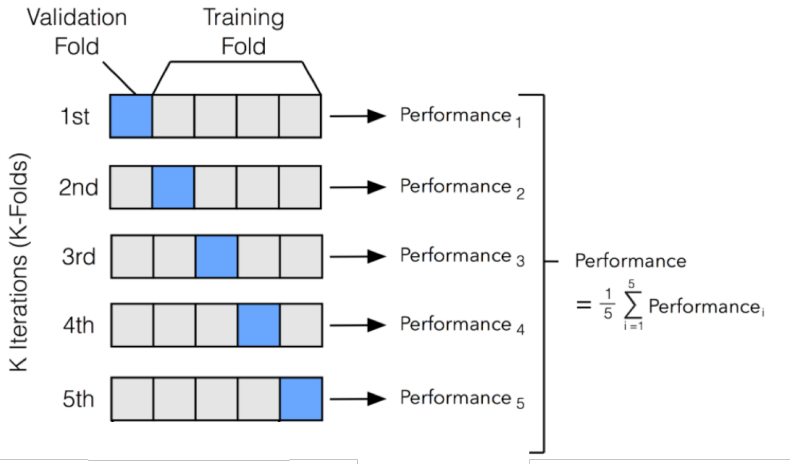


Figure 9: K-fold cross validation with k=5

Before proceeding with K-fold cross validation, a grid with the parameters that we want to try is defined, for example, in features selection (*k-best*) we need to set a priori the hyperparameter K and then we can evaluate how good the model performs only when the classification models gives its results, also the hyperparameters of the classification models need to be tuned.

The k-fold procedure is done many times and, at each iteration, the model uses a different set of parameters from the grid and, finally, the model providing the best score is selected as the best one and the entire dataset is re-trained by using those hyperparameters.

5.3 Over-sampling: SMOTE & ADASYN

As seen in Section 3.2.1, the dataset is unbalance with respect to the class label. In order, to reduce the imbalance there are two main branches: under-sampling and over-sampling. Under sampling consists in downsize the actual data set in such a way that the ratio between minority and majority class decreases. This approach may lead to important information loss. On the other hand, over-sampling method uses synthetic data generation to increase the number of samples in the data set, or better increase the minority class so that the data set becomes balanced by creating synthetic observations based upon the existing minority observations. In this project, SMOTE and ADASYN have been used.

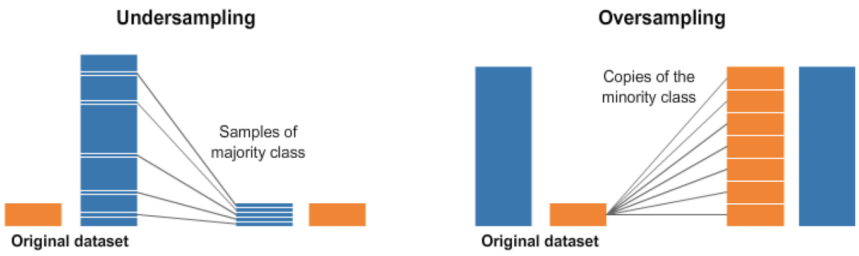


Figure 10: Illustration of under-sampling and over-sampling

An important thing to notice when dealing with over-sampling techniques is that performing those before cross validation can lead to overfitting problems and misleading results because would make the model be trained on instances that are the same of the ones used for validating the model and as

a result we make the validation phase useless. So, it is important to apply these methods only on the training data.

Figure 11: On the right the correct over-sample approach. On the left the wrong one.

SMOTE: stands for Synthetic Minority Oversampling Technique and what it does is to first find the n-nearest neighbors in the minority class for each of the samples in the class . Then it draws a line between the the neighbors an generates random points on the lines as: $x_{new} = x_i + \lambda(x_{z_i} - x_i)$ with $\lambda \in [0, 1]$.

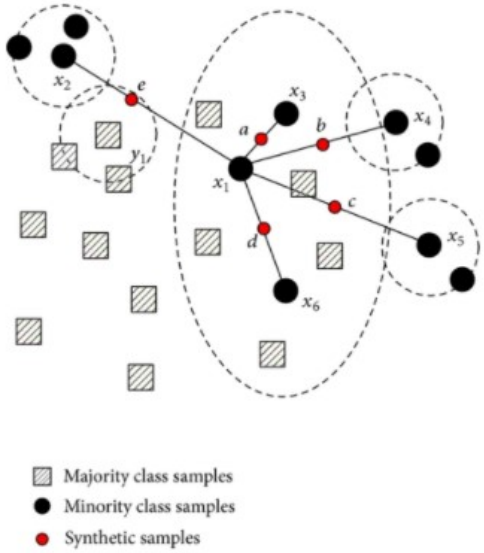


Figure 12: Illustartion of SMOTE with k=5

ADASYN: stands for Adaptive Synthetic and it is the same as SMOTE just with a minor improvement. After creating those sample it adds a random small values to the points thus making it more realistic. In other words instead of all the sample being linearly correlated to the parent they have a little more variance in them i.e they are bit scattered.

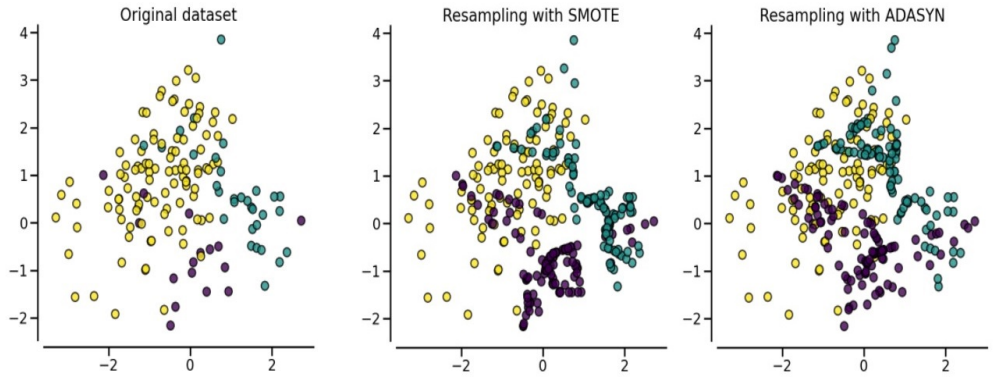


Figure 13: Comparison between SMOTE and ADASYN

6 Evaluation Metrics

In order to compare different models different evaluation measures can be used. However, different metrics measure different performance aspects, so the performance of a model with respect to a given metric does not mean that works well also with other evaluation metrics. For classification problems, metrics involve comparing the expected class label to the predicted class label.
Let: TP = True positives, TN = True negative, FP = False positive, FN = False negative.

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ → Number of correct predictions over the total number of predictions
- $Sensitivity = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$ → The fraction of relevant instances retrieved over the total relevant instances retrieved.
- $Precision = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$ → The number of correct prediction among the predicted to succeed.

- *F1 Score* $= 2 \cdot \frac{\text{Precision} \cdot \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}$ $\rightarrow$ Harmonic mean of sensitivity and precision.

7 Classification Algorithms

7.1 K-Nearest Neighbour (KNN)

K-Nearest-Neighbors is one of the simplest classification algorithm in supervised learning. KNN is an instance-based learning that do not learn weights from training data to predict output (as in model-based algorithms) but use entire training instances to predict output for unseen data. So, features from the same classes need to have similar features so when we project the domain points in the hyper-space of the features, elements with similar features (and so belonging to the same class) are closer one to each other.
KNN algorithm assigns to a point the label of the closest k points, where k is the only hyper-parameter that define the number of neighbors to consider in order to have the classification: $k \in \mathbb{N}$. The closeness is referred to the concept of distance. KNN algorithm usually use an Euclidean Distances defined as:

$$\rho(x, y) = ||x - y|| = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}.$$

k should be choose as an odd number in order to avoid uncertain classification in case of binary classification.

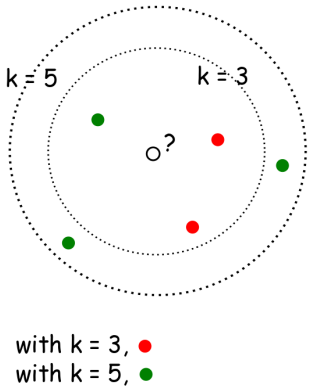


Figure 14: KNN classification

Before applying KNN some data preparation requirement are necessary, see Section 4, such as: data scaling, to locate the data point in multidimensional feature space, it would be helpful if all features are on the same scale; dimensionality reduction, KNN may not work well if there are too many features; and missing value treatment, because these do not allow to calculate the distance.

7.2 Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm widely used in classification problems that want to find an hyperplane that maximize the distance, called margin, between the hyperplane and the closest point (called Support Vector) of classes. So, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate, then the classification is performed by finding the hyper-plane that maximize the margin.
There are some cases that data are not linearly separable solving the hard margin problem so we can introduce slacks variable and allow some mistakes to soften constraints and this is called soft margin problem. The optimization function for the soft margin problem can be written as:

$$\min_{w} \frac{1}{2} w^t w + C \sum_{i} max(0, 1 - y_i w^t x_i)$$

where C is the cost of misclassification and when C is large, the algorithm tries to maximize the number of points correctly classified while a smaller C will encourage a larger margin at the cost of training accuracy.

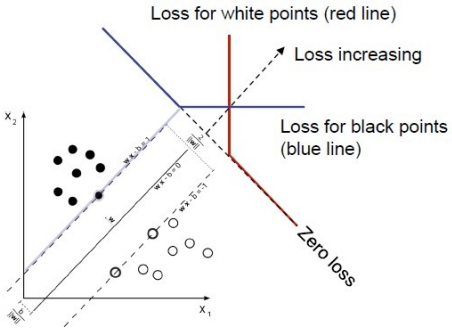


Figure 15: SVM classification

Sometimes happens that data are not linearly separable in their feature space, so we need to map data from the original space into an higher-dimensional space where data are linearly separable. The computation of the mapping phase and then the inner product could be expensive in computational sense, so kernels can be used. Kernels are symmetric functions that correspond to a scalar product in an higher dimensional features space that provide a solution in much more efficient and easier way, defined as:

$$K(x, x') =< \Phi(x), \Phi(x') >$$

Generally, linear support vector machine uses as linear kernel:

$$k(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$$

But there are also non-linear kernels and probably the most popular is the Radial basis kernel:

$$k(\vec{x_i}, \vec{x_j}) = \exp\big(-\gamma \|\vec{x_i} - \vec{x_j}\|^2\big)$$

where $\gamma > 0$ is a parameter to chose.

7.3 Logistic Regression

Logistic regression is the statistical technique used to predict the relationship between the dependent variable (Y) and the independent variable (X), where the dependent variable is binary in nature while the independent can be represented as a features vector $[x_1, x_2, ..., x_n]$ with n number of features.
During the training phase the model wants to learn a vector of weights and bias term. Weights w_i are real number associated with one of the input features x_i and represent the importance of that each input feature has on the classification decision. The bias term b is a real number added to the weighted inputs. Predictions are made summing up the weighted features and the bias term, obtainig z a number that expresses the weighted sum of the evidence for the class, as:

$$z = (\sum_{i=1}^{n} w_i x_i) + b = \vec{w} \cdot \vec{x} + b$$

Since Logistic Regression is based on probability we need $0 \leq z \leq 1$ and in order to respect this constraint the sigmoid function $\sigma(z)$, which is the *logistic function*, is applied as follow:

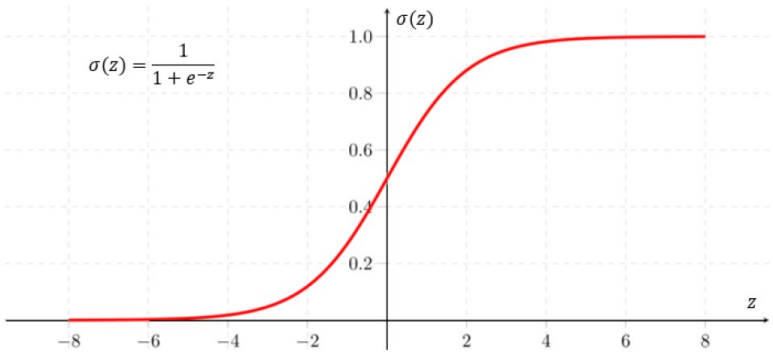


Figure 16: Sigmoid function

and in binary classification the predictions are computed as follow:

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbb{P}(y = 1|x) > 0.5 \\ 0 & \text{if } \mathbb{P}(y = 1|x) \leq 0.5 \end{cases}$$

7.4 Decision Tree

A decision tree is a simple supervised algorithm that predicts the label associated to an instance of x by travelling from the root to the leaf of a tree in which nodes represent features and edges are values. The model starts with a single leaf, the root. The leaf receives as label the one according to the majority vote among all labels over the training set. Then, in a set of iterations, the algorithm examines the possibility of split each leaf and evaluate the gain in splitting it. Among all the possible splits, the one that gave

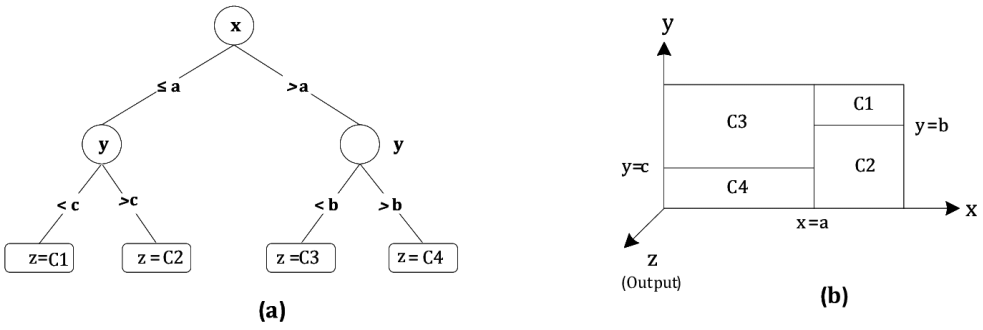


Figure 17: Example of Decision Tree

the best gain is choosen and the gain can be computed, for example, as *Gini index* or *Entropy*. Usually the decision tree learning algorithm are based on a greedy approach: the tree is constructed gradually and at each step the decision is taken by analyze the local optimal decision, so it's not guarantee a global optimal tree.

The process continues until a stopping criterion is reached and since we are training the model on the training set, if the tree grows too much it loses the capability of generalize and this leads overfitting. So, it is recommended to set a maximum depth of the tree that avoid this problem (can be searched through a cross validation with grid search). Tree-based methods are highly interpretable, simple and do not need to create dummy variables for qualitative predictors, however they often have worst prediction accuracy with respect to other supervised learning techniques.

7.5 Random Forest

In order to enforce and overcome the limitation of decision trees Random Forest algorithms have been introduced. Random forest algorithms are based on an ensemble learning technique called *Bagging*, that stands for *Bootstrap Aggregating*, that base models are trained on subsets of the data drawn with replacement (bootstrap replicates). Moreover, for each bootstrap replicate only m random predictors are taken into account (typically $m = \sqrt{p}$ where p is the total number of features), this allows to build uncorrelated trees and as a consequence a more robust model with smaller variance at the expense of less interpretation. Finally, the predictions are made through majority voting.

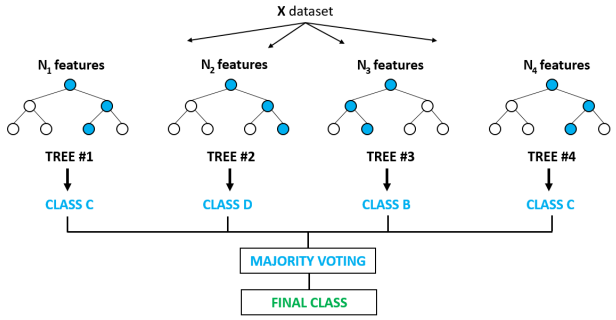


Figure 18: Example of Random Forest

8 Results

In this work, many classification algorithms have been tested under different preprocessing techniques. In this section results are presented comparing algorithms under the same data processing procedure. For further results and hyperparameters selection check the code [6].
The first scenario is the comparison of algorithms without data reduction as follow:

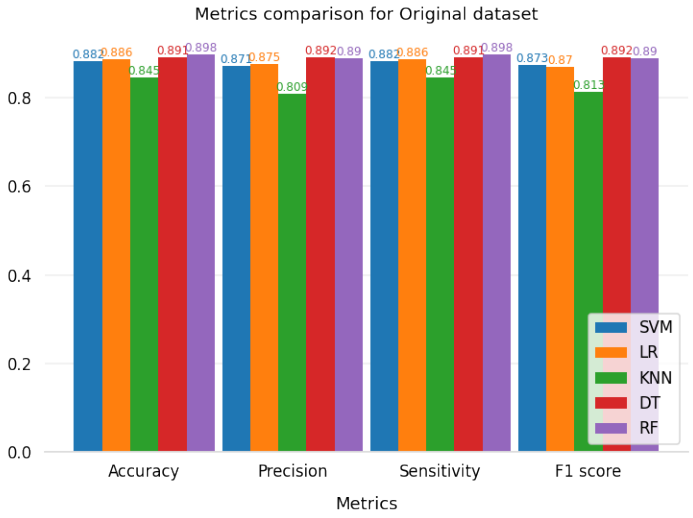


Figure 19: Results without data reduction

As we can see in the plot KNN is the worst model for all the evaluation metrics, while Random Forest and Decision Tree achieve the bests results. Important to emphasize is the fact Decision Tree performs better than Random Forest for F1-Score and Precision and this could be due to the wrong choice of hyperparameters in the tuning phase.
The second scenario is the comparison of algorithms with PCA or Features Selection in the augmentation phase. The first thing to highlights is that apply a dataset reduction is been a good strategy in terms of computational time because the same grid-search combinations of parameters took less time to be discovered and applied. On the other hand seems that the metrics are a bit worst, maybe due to the loss of information. Here, we also compared the effectiveness of the over sampling techniques. Considering F1-score, the Fea-

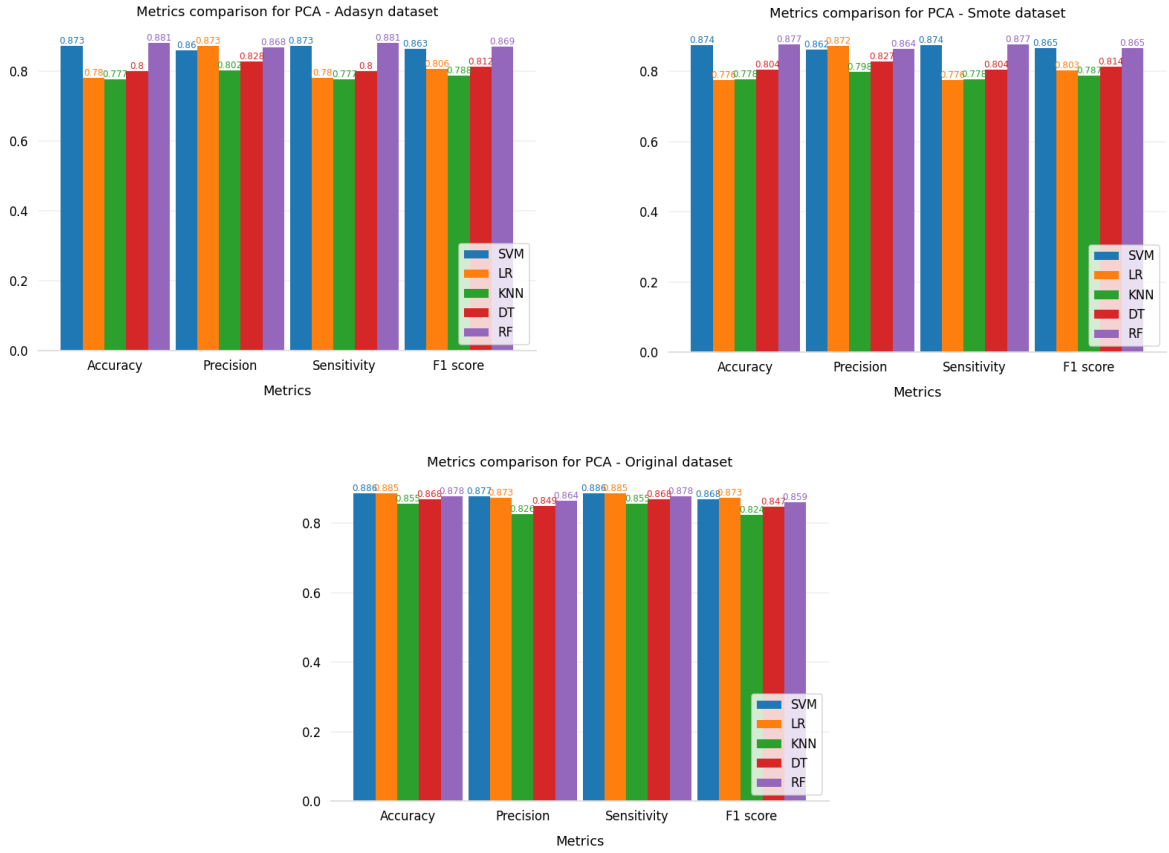


Figure 20: Results with PCA reduction

ture Selection methods achieved better results than PCA and the question that arose during the analysis of the distribution of the class label in terms of unbalancing representation has not a real answer, because it depends on the context. For example if we consider the F1-Score, if PCA is applied the best results is obtained without over-sampling and using Logistic Regression but the second best result is obtained with ADASYN oversampling and Random Forest. Unlike what we saw in the previous scenario, in both PCA and Feature Selection, Random Forest perform better than Decision Tree. Since there is no rule in the choice of augmentation and model choice, in a similar situation, the best solution is to try and test all the possibility and see which is the best in that specific case.

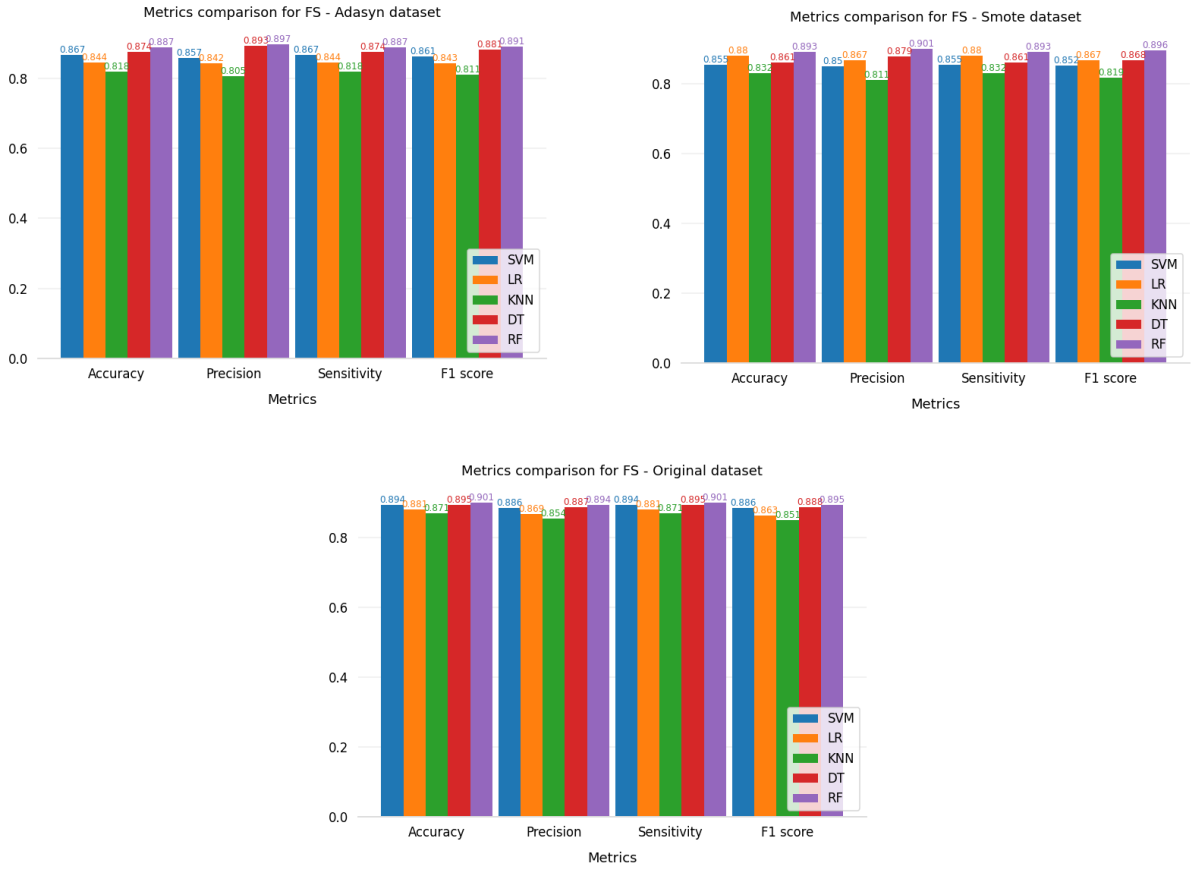


Figure 21: Results with Feature Selection reduction

References

[1] Shai Shalev-Shwartz and Shai Ben-David, *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press, 2014. Available at link

[2] Dirk P. Kroese, Zdravko I. Botev, Thomas Taimre, Radislav Vaisman, *Data Science and Machine Learning*, November 2020.

[3] Jake VanderPlas, *Python Data Science Handbook: Essential Tools for Working with Data*.

[4] Bowyer, Chawla, Hall, Kegelmeyer , *SMOTE: Synthetic Minority Over-sampling Technique*,Journal of Artificial Intelligence Research 16 (2002). https://arxiv.org/pdf/1106.1813.pdf

[5] Online Shoppers Purchasing Intention Dataset, *Dataset*, link

[6] Source code: https://github.com/MattiaDelleani/MML_Thesis