Politecnico di Torino

Master's degree in Data Science and Engineering

Mathematics in Machine Learning

Bank Marketing

Professors
Prof.Francesco Vaccarino
Prof.Mauro Gasparini

Gianluca La Malfa
s290187

September 2022

Contents

Chapter 1

Introduction

A term deposit is a fixed-term investment that includes the deposit of money into an account at a financial institution. Term deposit investments usually carry short-term maturities ranging from one month to a few years and will have varying levels of required minimum deposits. The investor must understand when buying a term deposit that they can withdraw their funds only after the term ends. In some cases, the account holder may allow the investor early termination or withdrawal if they give several days notification. Also, there will be a penalty assessed for early termination.

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit by using the Bank Marketing Data set provided by Moro et al.[1]

Chapter 2

Exploratory Data Analysis

2.1 Dataset

The *Bank Marketing Data Set* is composed by 768587 records. For each record we have information about the personal data about the potential clients. Information about the previous investment can be also found. The target variable indicates whether a term deposit will be subscribed or not. We can distinguish both numerical and categorical features.

The numerical ones are:

1. Age: it indicates the age of the client. Its values are in the interval [18,95]
2. Balance: it indicates the amount of money the client has in his bank account. Its values are in the interval [-8019,102127]
3. Duration: it indicates the last contact duration, in seconds. Its values are in the interval [0,4918]
4. Campaign: It indicates the number of contacts performed during this campaign and for this client. Its values are in the interval [1,63]
5. Pdays: it indicates the number of days that passed by after the client was last contacted from a previous campaign. Its values are in the interval [-1,871]
6. Previous: it indicates the number of contacts performed before this campaign and for this client. Its values are in the interval [0,275]

While the categorical ones are:

1. Job: it indicates the type of job
2. Marital: it indicates the marital status
3. Education: It indicates the level of education
4. Housing: it indicates if the client has a house on loan

5. Loan: it indicates if the client has a personal loan
6. Contact: it indicates the contact on which the communications are received
7. Poutcome: it indicates the result of the previous marketing campaign.

There are no missing values in the dataset. The target value is y, it is also a categorical value which can only assumes "yes" or "no" as values.

2.2 Further studies and hypothesis

As we can see from FIG 2.1 the distribution of the target attribute y is highly unbalanced. We have a percentage of *NO* close to 88% while the percentage of *YES* is approximately 12%

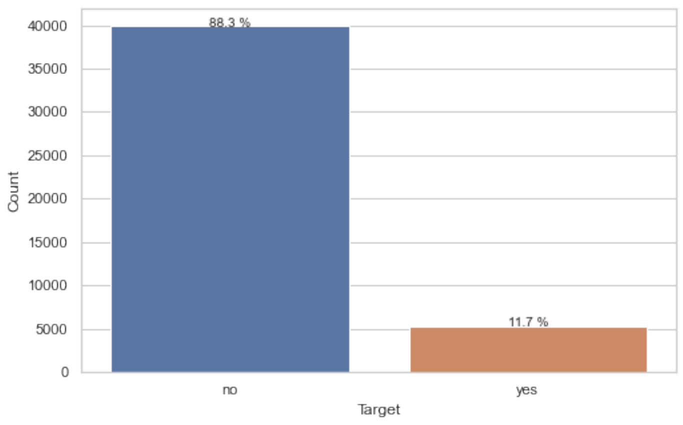


Figure 2.1: Target distribution

We decided to go for further studies in this direction by performing an analysis of the categorical attributes and their relation to the target value. The most used mean of communication is the cellular, while it is possible to see a strictly minor quantity of telephone interviews. Moreover there is an higher percentage of people with a primary school education with respect to secondary and tertiary. The results of the analysis are summoned in FIG 2.2. Generally speaking, people tends to not trust this kind of approach and to be afraid of trying these investments. This can be due a lot of different socio-economical reasons. Only the clients who have already had a successful experience in the past tend to repeat the process, as it can be seen from FIG.2.3

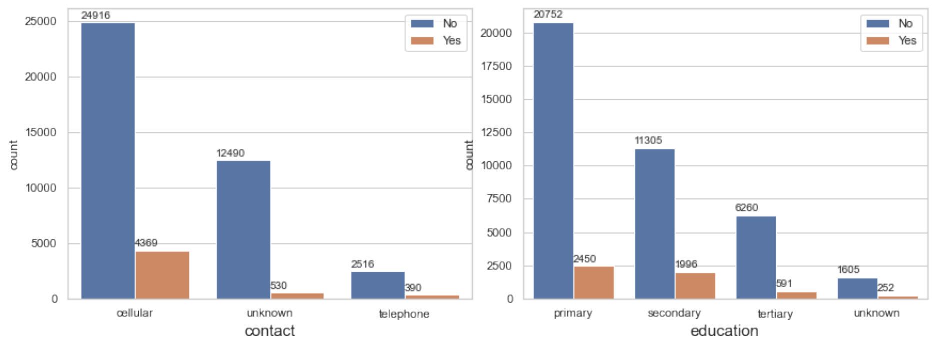


Figure 2.2: Relationship between Contact, Education and the target

Another cause that we supposed had a huge role in the negative trend of the deposit is linked to already present debts. With a deeper analysis we found out that most of the potential clients have already a loan, personal or housing loan. This could for sure lead the potential clients to not risk their savings.

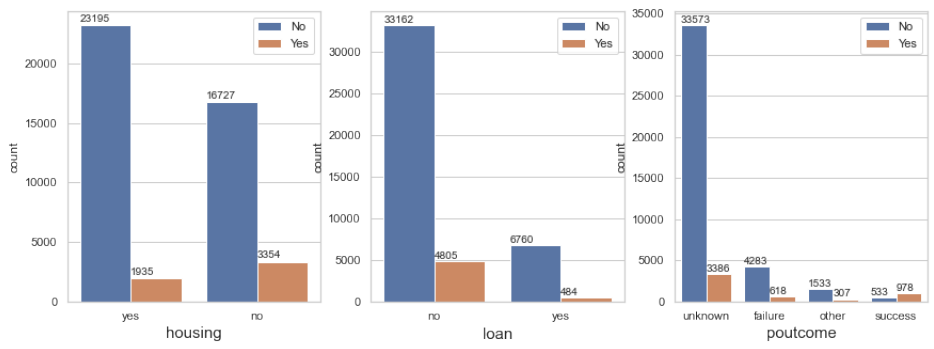


Figure 2.3: Relationship between Housing, Loan, Poutcome and the target

2.3 Correlation

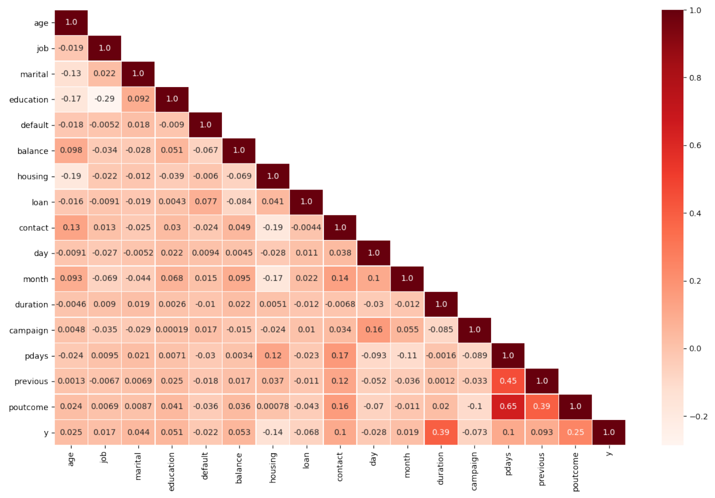
After this analysis we focused on the relationship between all the other attributes of the dataset in order to understand if there was a relationship between the features themselves or with the target. Correlation means association, more precisely it is a measure of the extent to which two variables are related.

- positive correlation: when the correlation coefficient is greater than 0. It signifies that both variables move in the same direction
- negative correlation: when the correlation coefficient is less than 0. This is an indication that both variables move in the opposite direction.

We used the *Pearson correlation*:

$$\rho_{(X,Y)} = \frac{\sigma_{(X,Y)}}{(\sigma_X \sigma_Y)} \tag{2.1}$$

Where $\sigma_{(X,Y)}$ is the covariance between X and Y while σ_X and σ_X are single standard deviations. The values that it can take are in the interval [-1,1]. We computed a Correlation Matrix, as it is possible to see in FIG.2.4, which is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables.



	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
age	1.0																
job	-0.019	1.0															
marital	-0.13	0.022	1.0														
education	-0.17	-0.29	0.092	1.0													
default	-0.018	-0.0052	0.018	-0.009	1.0												
balance	0.098	-0.034	-0.028	0.051	-0.067	1.0											
housing	-0.19	-0.022	-0.012	-0.039	-0.006	-0.069	1.0										
loan	-0.016	-0.0091	-0.019	0.0043	0.077	-0.084	0.041	1.0									
contact	0.13	0.013	-0.025	0.03	-0.024	0.049	-0.19	-0.0044	1.0								
day	-0.0091	-0.027	-0.0052	0.022	0.0094	0.0045	-0.028	0.011	0.038	1.0							
month	0.093	-0.069	-0.044	0.068	0.015	0.095	-0.17	0.022	0.14	0.1	1.0						
duration	-0.0046	0.009	0.019	0.0026	-0.01	0.022	0.0051	-0.012	-0.0068	-0.03	-0.012	1.0					
campaign	0.0048	-0.035	-0.029	0.00019	0.017	-0.015	-0.024	0.01	0.034	0.16	0.055	-0.085	1.0				
pdays	-0.024	0.0095	0.021	0.0071	-0.03	0.0034	0.12	-0.023	0.17	-0.093	-0.11	-0.0016	-0.089	1.0			
previous	0.0013	-0.0067	0.0069	0.025	-0.018	0.017	0.037	-0.011	0.12	-0.052	-0.036	0.0012	-0.033	0.45	1.0		
poutcome	0.024	0.0069	0.0087	0.041	-0.036	0.036	0.00078	-0.043	0.16	-0.07	-0.011	0.02	-0.1	0.65	0.39	1.0	
y	0.025	0.017	0.044	0.051	-0.022	0.053	-0.14	-0.068	0.1	-0.028	0.019	0.39	-0.073	0.1	0.093	0.25	1.0

Figure 2.4: Correlation Matrix

As it is possible to notice, the features *poutcome* and *pdays* are the ones with a higher correlation. Moreover the target variable y does not seem to be strongly correlated to any of the others, having 0.39 as the highest correlation score with *duration*.

Chapter 3

Preprocessing

3.1 Features Selection

Feature selection is the process of reducing the number of input variables. It can lead to both reduction of the computational cost of modeling and, in some cases, to improve the performance of the model. In this first stage we did not remove any feature because of our previous studies made during correlation. We will reserve to select the most important features using PCA in the following steps of our process.

3.2 Features encoding

The first preprocessing operation that was made is the encoding of all the categorical features we had in the dataset. Machine learning models can only work with numerical values. For this reason, it is necessary to transform the categorical values of the relevant features into numerical ones. The encoding techniques that we used are:

- one-hot encoding: Assigns vectors to each category. The vector represent whether the corresponding feature is present (1) or not (0)
- label encoding: it is based on the idea of converting each value in a column to a number

For the categorical values that were simply boolean attributes we used label encoding, while for the more complex ones we used one-hot encoding.

3.3 Data splitting

Data splitting in a very common and fundamental technique in machine learning. It consists of taking the original dataset and divide it in two main splits:

- Training set: The sample of data used to fit the model, that is the actual subset of the dataset that we use to train the model. The model observes and learns from this data and optimize its parameters.

- Test set: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. It is only used once the model is completely trained using the training and validation sets

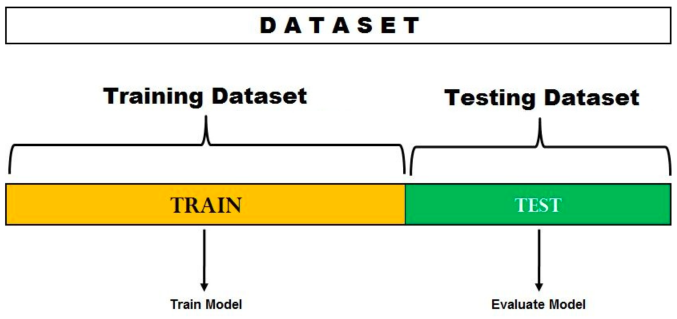


Figure 3.1: Train/Test data splitting

In order to evaluate the performances of the algorithm it is useful to obtain from the training set a new partition, called *Validation set*. To do this we performed a *Stratified K-Fold Cross Validation* with k=5. After shuffling the training data, first it splits the dataset into k groups, it trains on k-1 folds and test on the remaining one.

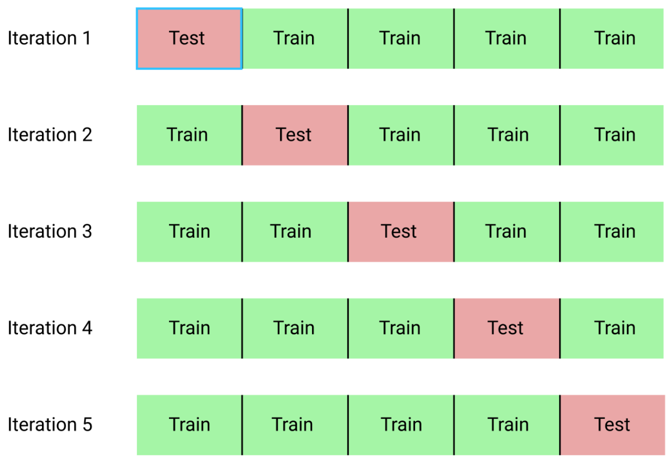


Figure 3.2: K-Fold Cross Validation on training data

3.4 Features Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data. In our case we transformed our distribution in order to have $\mu = 0$ $\sigma = 1$ This step will be fundamental for the following parts of the process, both for PCA and model training. As concerns PCA, scaling is necessary because, as we will see later, it is an algorithm which calculates a new projection of the dataset. The new axis are based on the standard deviation of the variables. So a variable with a high standard deviation will have a higher weight for the calculation of axis than a variable with a low standard deviation. If you normalize your data, all variables have the same standard deviation, thus all variables have the same weight and your PCA calculates relevant axis. As concerns model training, above all for the distance based algorithms, such as SVM or KNN, the presence of different magnitudes of data will affect the computation of distances.[2]

3.5 Dimensionality reduction

With dimensionality reduction we refer to the task of reducing the number of features in a dataset. The higher the number of features, the more difficult it is to model them, this is known as the curse of dimensionality. In order to estimate an arbitrary function with a certain accuracy the number of features or dimensionality required for estimation grows exponentially. When the data is sparse, observations or samples in the training dataset are difficult to cluster as high-dimensional data causes every observation in the dataset to appear equidistant from each other. If data is meaningful and non-redundant, then there will be regions where similar data points come together and cluster, furthermore they must be statistically significant. This can lead to various problems, for example overfitting but above all difficulties in clustering similar features and growing space and computational complexity.

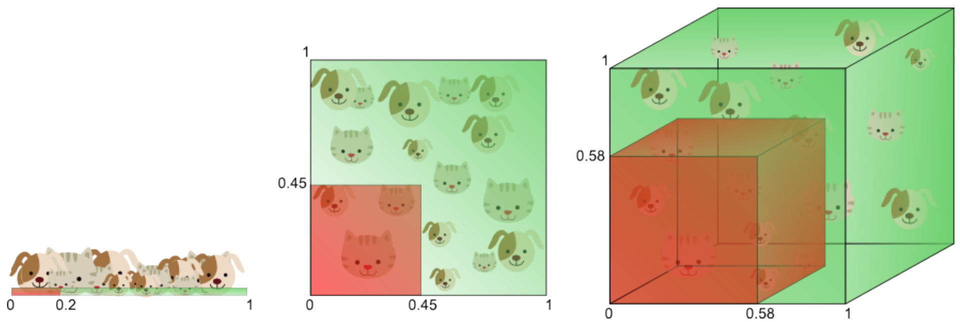


Figure 3.3: Curse of Dimensionality

3.5.1 PCA

Principal Component Analysis, or PCA, is a dimensionality-reduction method to find lower-dimensional space by preserving the variance as measured in the high dimensional input space. It is an unsupervised method for dimensionality reduction. PCA transformations are linear transformations. It involves the process of finding the principal components, which is the decomposition of the feature matrix into eigenvectors. This means that PCA will not be effective when the distribution of the dataset is non-linear.

In PCA we aim to solve the following optimization problem:

$$argmin \sum_{i=1}^{m} ||x_i - UWx_i||_2^2$$

To solve this problem we know from Lemma [7] that the columns of U are orthonormal (namely, $U^T U$ is the identity matrix of R^n) and $W = U^T$.
On the basis of this we can rewrite the optimization problem as:

$$argmin \sum_{i=1}^{m} ||x_i - UU^T x_i||_2^2$$

By exploring some mathematical manipulations and the linear property of the *trace* of matrices, we can express the optimization problem as a maximization one:

$$argmax\, trace(U^T \sum_{i=1}^{m} x_i x_i^T U)$$

Where $A = x_i x_i^T$ is symmetric and therefore it can be written using its spectral decomposition as $A = VDV^T$, where D is diagonal and $V^T V = VV^T = I$. Here, the elements on the diagonal of D are the eigenvalues of A and the columns of V are the corresponding eigenvectors. The solution of the optimization problem is the matrix U whose columns are the n eigenvectors of A corresponding to the largest n eigenvalues.

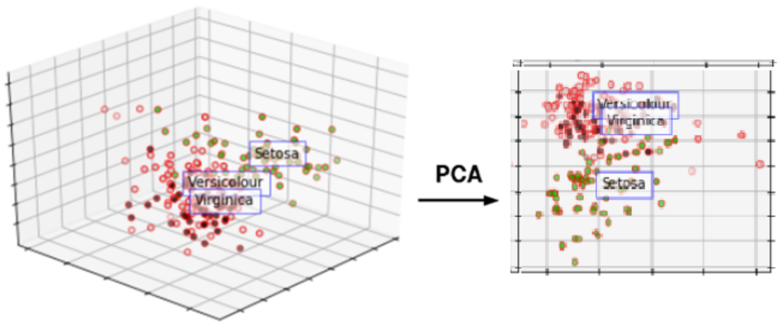


Figure 3.4: PCA

We can sum up the whole process in four fundamental steps:

1. Features scaling: The data has to be transformed to a common scale by taking the difference between the original dataset with the mean of the whole dataset. This will make the distribution 0 centered. See 3.4
2. Compute covariance: Covariance will help us to understand the relationship between the mean and original data. $cov_{x,y} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{N-1}$
3. Determining the principal components: Principal components can be determined by calculating the eigenvectors and eigenvalues. Eigenvectors are a special set of vectors that help us to understand the structure and the property of the data that would be principal components. The eigenvalues on the other hand help us to determine the principal components. The highest eigenvalues and their corresponding eigenvectors make the most important principal components.
4. Final output: It is the dot product of the standardized matrix and the eigenvector. Note that the number of columns or features will be changed.

As previously mentioned, PCA can be also seen as a Variance maximization problem, so the main idea is to find a sort of balance between the number of components selected and the portion of variance explained. Generally, we want to achieve an explained variance at least in the range of [80% - 90%], in fact, as we can see from FIG.3.5, we selected 11 components with 85% of variance explained.

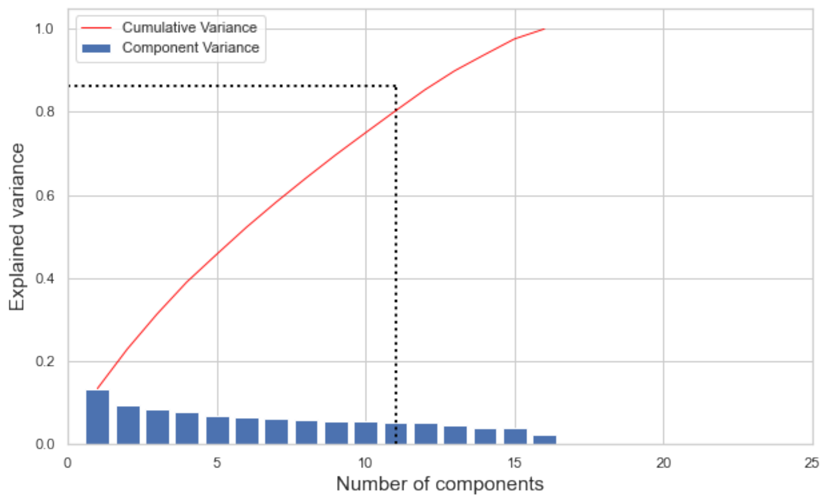


Figure 3.5: Number of selected components by PCA

Of course, with the reduction of the number of features we will have a decrease of the accuracy of the model too, but the benefits will be more influential. We can analyze, visualize and explore data in a simpler way, all of this with a minor computational complexity. The key point of this algorithm is to reduce the number of features while preserving information as much as possible.

3.6 Resampling

Resampling is a very important preprocessing technique which consists of working on the training set in order to reach a more balanced working space than the inital one. Resampling can be done in different ways.

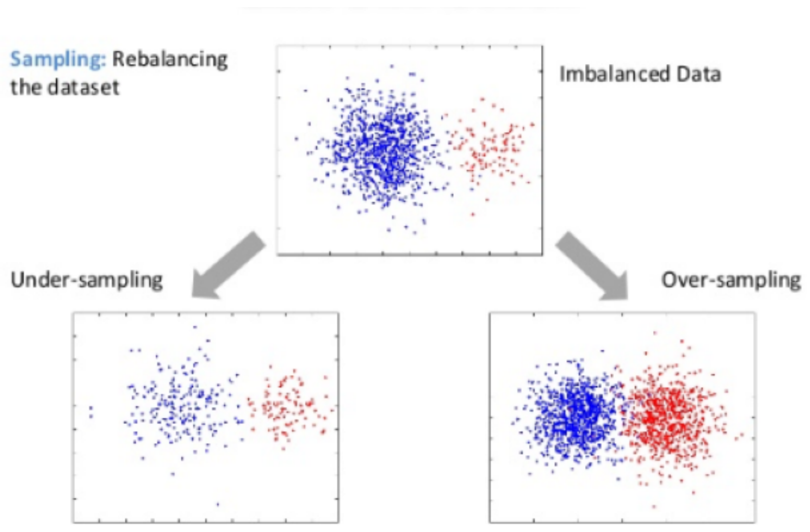


Figure 3.6: Visual Representation of Under and Over Sampling

3.6.1 Under-Sampling

This technique consists of removing records from the majority class in order to balance. The simplest undersampling technique involves randomly selecting examples from the majority class and deleting them from the training dataset. This is referred to as random undersampling. Although simple and effective, a limitation of this technique is that examples are removed without any concern for how useful or important they might be in determining the decision boundary between the classes. This means it is possible, or even likely, that useful information will be deleted. An extension of this approach is to be more discerning regarding the examples from the majority class that are deleted. This typically involves heuristics or learning models that attempt to identify redundant examples for deletion or useful examples for non-deletion. One of the most known technique to understand which records to keep is *Near-miss*. It refers to a collection of undersampling methods that select examples based on the distance of majority class examples to minority class examples. It is possible to distinguish three different approaches:

- *NearMiss-1*: Majority class examples with minimum average distance to three closest minority class examples.

- *NearMiss-2*: Majority class examples with minimum average distance to three furthest minority class examples.

- *NearMiss-3*: Majority class examples with minimum distance to each minority class example.

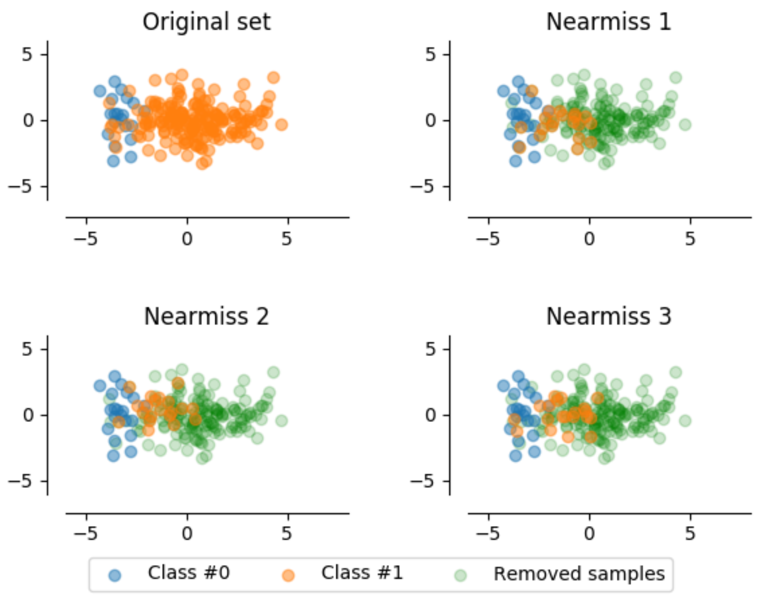


Figure 3.7: Visual Representation of Near Miss

3.6.2 Over-Sampling

It consists of adding synthetic samples of the minority. The goal can be achieved by simply duplicating examples from the minority class in the training dataset prior to fitting a model. This can balance the class distribution but does not provide any additional information to the model.

An improvement on duplicating examples from the minority class is to synthesize new examples from the minority class. This is a type of data augmentation for tabular data and can be very effective. The most effective algorithm in this scenario is *SMOTE* (Synthetic Minority Oversampling Technique) which 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.

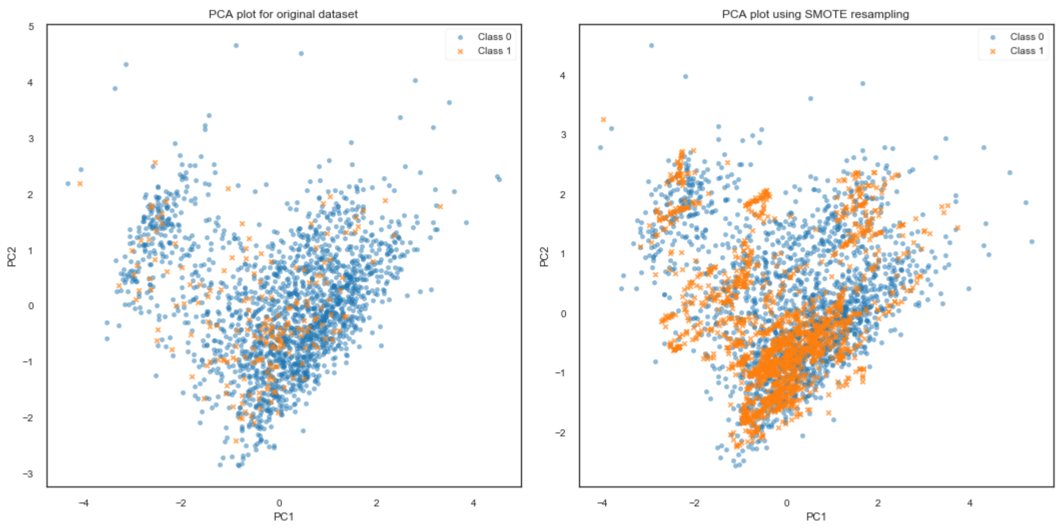


Figure 3.8: Visual Representation of SMOTE

3.6.3 Over and Under Sampling

It is possible to combine both Under and Over sampling, in order to delete the useless records of the majority class while syntethizing new records of the minority class. One of the most common technique is SMOTEEN which is obtained by combining SMOTE an ENN.

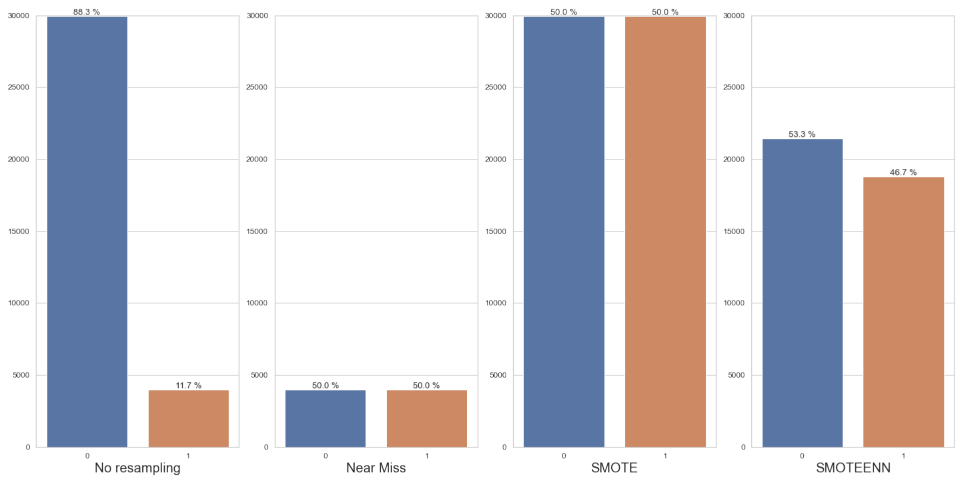


Figure 3.9: Resampling techniques results

In order to find the best performing combination we performed all the test with all the different resampling techniques as we can see in FIG.3.9.

Chapter 4

Model Evalutation

An evaluation metric quantifies the performance of a predictive model. This typically involves training a model on a dataset, using the model to make predictions on a holdout dataset not used during training, then comparing the predictions to the expected values in the holdout dataset. The most used metrics to evaluate a classification model are:

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1 = \frac{2 \times Precision \times Recall}{Precision+Recall} = \frac{2 \times TP}{2 \times TP+FP+FN}$

The challenge of correctly evaluating a model becomes even harder when we deal with an imbalanced dataset like in our case. The reason for this is that many of the standard metrics become unreliable or even misleading. What we are trying to achieve with the F1-score metric is to find an equal balance between precision and recall, which is extremely useful in most scenarios when we are working with imbalanced datasets. The range of F1 is in [0, 1], where 1 is perfect classification and 0 is total failure.

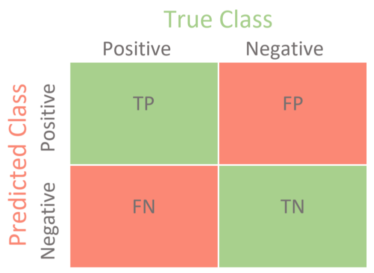


Figure 4.1: Confusion matrix

Chapter 5

Model Selection and Testing

In this section we will explore some of the most important classifiers used to solve the classification problem. In particular, the main algorithms we used are: Decision Trees and Random Forest, Logistic Regression,Support Vector Machine and KNN, following the Empirical Risk Minimization paradigm.

Before going into details we need to specify the environment in which we are working on: We have:

1. A set of observation X
2. A set of labels y
3. A sampling distribution $D\ x\ X$
4. A labelling function $f(.) : X \rightarrow y$

Ideally, given a sample S, our goal is to find the prediction rule $h : X \rightarrow y$ and we could compute the true error defined as:

$$L_{D,f}(h) = P[h(x) \neq f(x) = D(x : h(x) \neq = f(x))$$

However in a real scenario, we do not have the distribution D, neither the labelling function,so this error cannot be computed. In order to estimate the goodness of our algorithm we can compute the *training error*:

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

Even if this could seem the best solution, it fails to generalize. This risk is called *overfitting.* The solution is to apply the ERM rule over a restricted set of hypothesis class, which are chosen exploring prior knowledge. This kind of choice introduces a sort of bias. We select the predictor $h \in H$ that achieves the minimum error:

$$ERM_H(S) \in argminL_s(h)$$

5.1 Decision Trees and Random Forest

A decision tree is a predictor, $h : X \to Y$, that predicts the label associated with an instance x by traveling from a root node of a tree to a leaf. At each node on the root-to-leaf path, the successor child is chosen on the basis of a splitting of the input space. Usually, the splitting is based on one of the features of x or on a predefined set of splitting rules. A leaf contains a specific label.

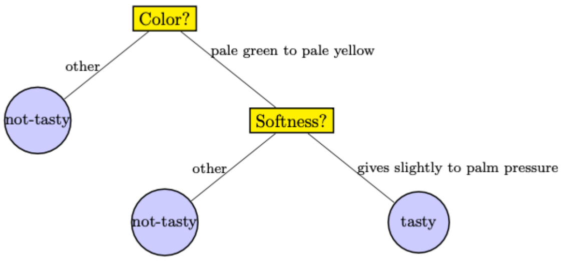


Figure 5.1: Decision Tree classifier

Decision tree is a top-down greedy algorithm.

- It is top-down because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
- It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step

In order to select the best split, measures of purity are minimized:

1. *GINI-index* : $1 - \sum_j p_j^2$
2. *Entropy*: $\sum_j p_j log_2 p_j$

5.1.1 Random Forest

Random Forest belongs to the family of "Decision Trees" algorithms. Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result and gives more robustness with respect to classical tree.

The fundamental concept behind random forest is a simple but powerful one:"the wisdom of crowds", allows it to learn from a diverse set of data, leading to more stable and generalizable results. The reason that the random forest model works so well is that a large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent model. While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction.

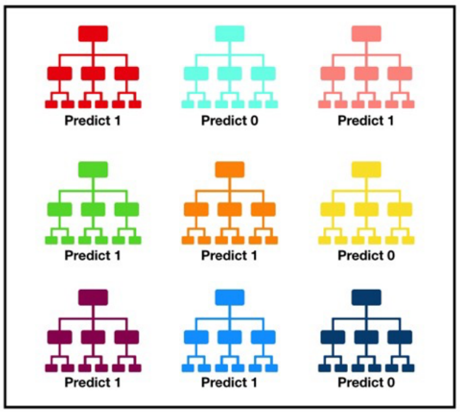


Figure 5.2: Random Forest classifier

Notice that with bagging we are not subsetting the training data into smaller chunks and training each tree on a different chunk. Rather, if we have a sample of size N, we are still feeding each tree a training set of size N (unless specified otherwise). But instead of the original training data, we take a random sample of size N with replacement.

5.1.2 Hyperparameters tuning

As we did in for all classifiers, we tuned all the hyperparameters with all of the possible combinations of PCA (True or False) and all the resampling techniques(Oversampling, Undersampling and both combined). We used the GridSearch approach which makes us able to understand which is the best combination of parameters in order to maximize the metric we chose (F1 score). In particular we tuned:

- "n_estimators":[80,100,120]
- "criterion":["gini", "entropy"]
- "max_features":['sqrt','log2']

The best performing combination was found at:{'criterion': 'gini', 'max_features': 'sqrt', 'n_estimators': 120} combined with PCA and Over-sampling. The F1-score reached was 0,55.

5.2 Logistic Regression

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. As it can be seen in FIG. 5.4, it uses a logistic function called "sigmoid function", to map predictions and their probabilities. The sigmoid function refers to an S-shaped curve that converts any real value to a range between 0 and 1. Moreover, if the output of the sigmoid function (estimated probability) is greater than a predefined threshold on the graph, the model predicts that the instance belongs to that class. If the estimated probability is less than the predefined threshold, the model predicts that the instance does not belong to the class.

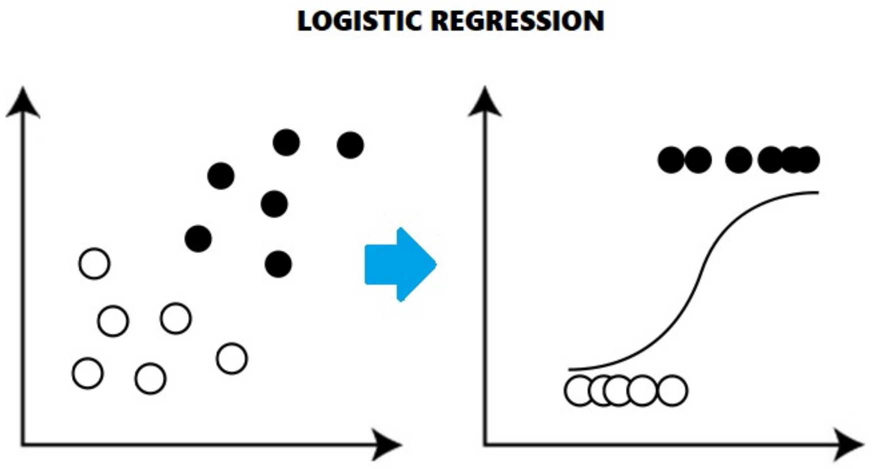


Figure 5.3: Logistic Regression

In particular, the sigmoid function used in logistic regression is the logistic function:

$$\phi_{sig}(z) = \frac{1}{1 + exp(-z)}$$

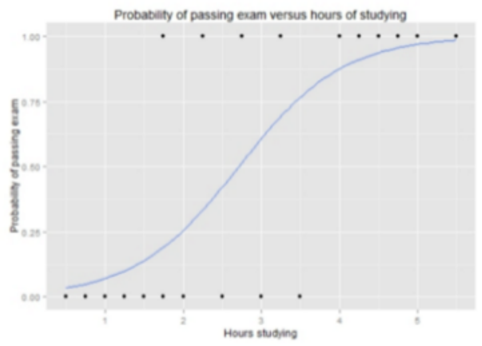


Figure 5.4: Logistic Regression function

Next, we need to specify a loss function. That is, we should define how bad it is to predict some $h_w(x) \in [0; 1]$ given that the true label is $y \in \pm 1$. Clearly,we would like that $h_w(x)$ would be large if $y = 1$ and that $1 - hw(x)$ would be large if $y = -1$
The logistic loss function used is:

$$l(h_w, (x, y)) = log(1 + exp(-y < w, x >))$$

So given a training set S the ERM problem associated with logistic regression is :

$$argmin \frac{1}{m} \sum_{i=1}^{m} log(1 + exp(-y_i < w, x_i >)).$$

The advantage of the logistic loss function is that it is a convex function with respect to w; hence the ERM problem can be solved efficiently using standard methods. It is like finding a Maximum Likelihood Estimator.

5.2.1 Hyperparameters tuning

The parameters that were tuned are:

- "penalty": ['l1','l2']
- "tol": [1e-1, 1e-2, 1e-3, 1e-4]
- "C": [0.6,0.8, 1.0,1.2, 1.4]

The best performing combination was found at:{'C': 1.2, 'max_iter': 500, 'penalty': 'l2', 'solver': 'saga', 'tol': 0.0001} combined with PCA and Oversampling. The F1-score reached was 0,52.

5.3 KNN

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.

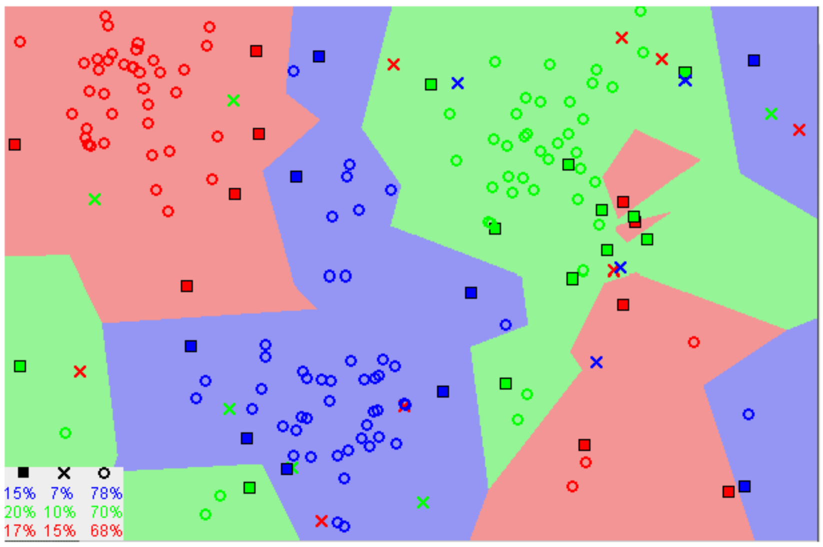


Figure 5.5: KNN

The rationale behind such a method is based on the assumption that the features that are used to describe the domain points are relevant to their labelings in a way that makes close-by points likely to have the same label. The main advantages of using this algorithm are:

- The algorithm is simple and easy to implement.
- There's no need to build a model, tune several parameters, or make additional assumptions.
- The algorithm is versatile. It can be used for classification, regression, and search

While the main disadvantage is related to performances as the dataset with which it is working starts to become bigger.

5.3.1 Hyperparameters tuning

The parameters that were tuned are:

- "n_neighbors": [10, 50, 75]
- "weights": ["uniform", "distance"]
- "algorithm": ["auto", "ball_tree", "kd_tree"]

The best performing combination was found at:{'algorithm': 'auto', 'n_jobs': -1, 'n_neighbors': 10, 'weights': 'distance'} combined with PCA and Oversampling. The F1-score reached was 0,53.

5.4 SVM

5.4.1 Introduction

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.[4] To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes.

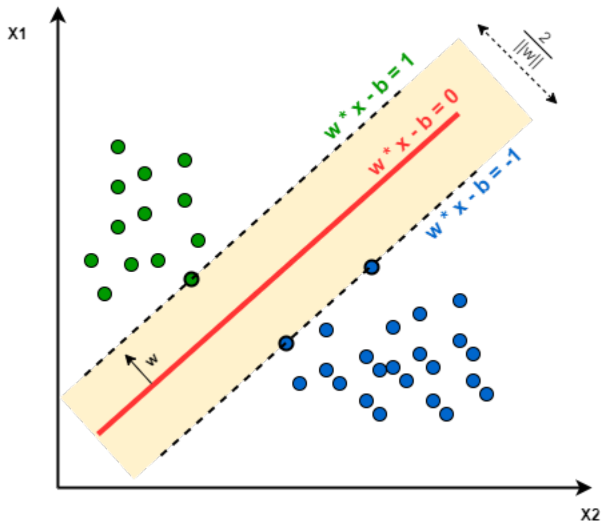


Figure 5.6: SVM

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

5.4.2 Hard-SVM

Hard-SVM is the learning rule in which we return a hyperplane that separates the training set with the largest possible margin. To define Hard-SVM formally, we first express the distance between a point x to a hyperplane using the parameters defining the halfspace. The distance between a point x and the hyperplane defined by (w, b) where $||w|| = 1$ is

$$| < w, x > +b|$$

On the basis of the preceding rule, the closest point in the training set to the separating hyperplane is $min_{i \in m}| < w, x > +b|$.[5] Therefore, the Hard-SVM rule is:

$$argmax \quad min_{i\in[m]}(< w, x > +b) \quad s.t. \quad \forall i, y_i(< w, x > +b) > 0$$

$$argmax \quad min_{i\in[m]} \qquad y_i(< w, x > +b)$$

Which is equivalent to the quadratic optimization problem:

$$argmin_{w,b}||w^2|| \quad s.t. \quad \forall i, y_i(< w, x > +b) >= 1 \quad 5.4.2.1$$

5.4.3 Soft-SVM

The Hard-SVM formulation assumes that the training set is linearly separable, which is a rather strong assumption. Soft-SVM can be viewed as a relaxation of the Hard-SVM rule that can be applied even if the training set is not linearly separable. The optimization problem in 5.4.2.1 enforces the hard constraints $y_i(< w, x > +b) >= 1$ for all i.[6] One plausible way to relax this is to allow some violation for some examples in the training set. We can do this by introducing some *slack variables* $\epsilon_1, \epsilon_2, ..., \epsilon_m$ and replacing each constraint $y_i(< w, x > +b) >= 1$ with $y_i(< w, x > +b) >= 1 - \epsilon_i$ The slack variable measures by how much the constraint is being violated. Soft-SVM jointly minimizes the norm of w (corresponding to the margin) and the average of ϵ_i (corresponding to the violations of the constraints). The tradeoff between the two terms is controlled by a parameter λ. This leads to the Soft-SVM optimization problem:

$$min(\lambda||w||^2 + \frac{1}{m}\sum_{i=1}^{m}\epsilon_i)$$

$$s.t.\ \ \forall i, \ \ y_i(< w, x_i > +b >= 1 - \epsilon_i \ \ and\, \epsilon_i >= 0$$

This equation can be rewritten as a regularized loss minimization problem: We know that the *hinge loss* is:

$$l^{hinge}((w,b),(x,y)) \quad = \quad max[0, 1 - y(< w, x > +b]$$

It becomes:

$$min(\lambda||w||^2 + L_S^{hinge}((w,b)))$$

In 5.7 we can see a summary of the main differences between the two approches.

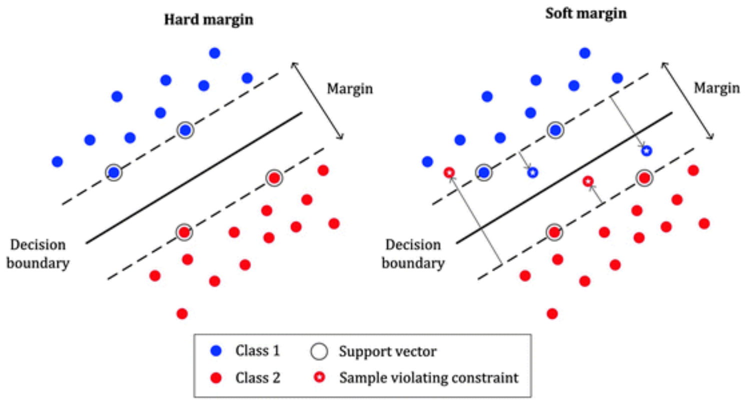


Figure 5.7: Hard SVM vs Soft SVM

5.4.4 Kernel SVM

Support vector machines are more difficult to interpret in higher dimensions. It is much harder to visualize how the data can be linearly separable, and what the decision boundary will look like. Even if with Soft-SVM the process becomes more affordable, in practice data is often very far from being linearly separable, and we need to transform the data into a higher dimensional space in order to fit a support vector classifier anyway. In real applications, there might be many features in the data and applying transformations that involve many polynomial combinations of these features will lead to extremely high and impractical computational costs. The kernel trick provides a solution to this problem. The “trick” is that kernel methods represent the data only through a set of pairwise similarity comparisons between the original data observations x (with the original coordinates in the lower dimensional space), instead of explicitly applying the transformations $\phi(x)$ and representing the data by these transformed coordinates in the higher dimensional feature space.

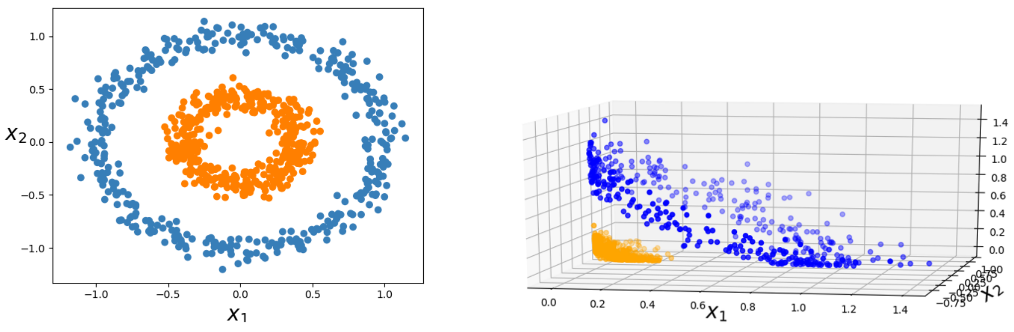


Figure 5.8: Linear separability in one dimensional vs two dimensional space

In kernel methods, the data set X is represented by an n x n kernel matrix of pairwise similarity comparisons where the entries (i, j) are defined by the kernel function: $k(x_i, x_j)$. This kernel function has a special mathematical property. The kernel function acts as a modified dot product. Formally, if we have data $x, z \in X$ and a map $\phi : X -> R^n$ then

$$k(x, z) =< \phi(x), \phi(z) >$$

Our kernel function accepts inputs in the original lower dimensional space and returns the dot product of the transformed vectors in the higher dimensional space.

5.4.5 Hyperparameters tuning

The parameters that were tuned are:

- "kernel": [’linear’, ’rbf’, ’poly’]
- "gamma" : [’scale’, ’auto’]
- ""C": [0.8, 1.0,1.2]

The best performing combination was found at:{’C’: 1.2, ’gamma’: ’auto’, ’kernel’: ’rbf’} combined with PCA and Over-sampling. The achieved F1-score was : 0,55.

5.5 Results summary

Classifier	PCA	Resampling Techniques	Best Combination
SVM	T or F	None,Under,Over,Both	True "Over" {'C': 1.2, 'gamma': 'auto', 'kernel': 'rbf'}
LR	T or F	None,Under,Over,Both	True "Over" {'C': 1.2, 'max_iter': 500, 'penalty': 'l2', 'solver': 'saga', 'tol': 0.0001}
KNN	T or F	None,Under,Over,Both	True "Over" {'algorithm': 'auto', 'n_jobs': -1, 'n_neighbors': 10, 'weights': 'distance'}
RF	T or F	None,Under,Over,Both	True "Over" {'criterion': 'gini', 'max_features': 'sqrt', 'n_estimators': 120, 'n_jobs': -1}

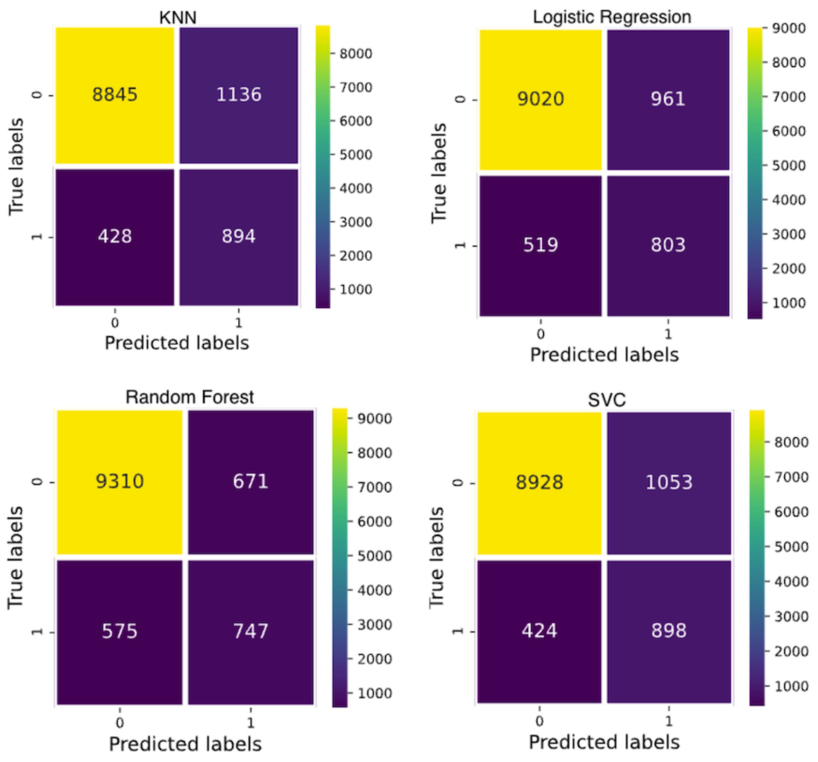


Figure 5.9: Best performing parameters and confusion matrices of the tested classifiers

Chapter 6

Conclusions

After an in depth analysis of the dataset, we proceeded to features encoding and scaling. We applied PCA to select the most important features and then explored some resamping techniques on the training set in order to balance the working space. Different machine learning algorithms have been tested, each of them with different preprocessing steps (PCA and Resampling) and different hyperparameters, in order to estimate the best performing ones. Generally speaking the pipeline with PCA and Oversampling despite being computationally heavier than the other techniques, achieved the best results with all the algorithms. In terms of classifiers the best results we obtained were achieved by SVC and Random Forest while are noticeable the results obtained by Logistic Regression. In the future, various approaches can be explored in order to analyze further points of interest. New classifiers can be tested, for example SGD Classifier approach or further resampling techniques to mitigate class imbalance. In this work, we did not manage outliers, due the complexity of the dataset and the high risk of removing relevant informations, but it for sure will be beneficial to work on them in order to improve the results and the robustness of the model.

Bibliography

[1] [Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

[2] [Introduction to Feature Scaling, www.analyticsvidhya.com]

[3] [Shai Shalev-Shwartz and Shai Ben-David, "Understanding Machine Learning: From Theory to Algorithms, "Logistic Regression"]

[4] [Shai Shalev-Shwartz and Shai Ben-David, "Understanding Machine Learning: From Theory to Algorithms, "SVM"]

[5] [Shai Shalev-Shwartz and Shai Ben-David, "Understanding Machine Learning: From Theory to Algorithms, "Hard SVM"]

[6] [Shai Shalev-Shwartz and Shai Ben-David, "Understanding Machine Learning: From Theory to Algorithms, "Soft SVM"]

[7] [Shai Shalev-Shwartz and Shai Ben-David, "Understanding Machine Learning: From Theory to Algorithms, "Lemma 23.1 from PCA"]