POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Mathematics in Machine Learning

Default of credit card clients

Professors

Prof. F. VACCARINO

Prof. M. GASPARINI

Candidate

Antonio ALBANESE, 282043

A.Y. 2021/2022

Table of Contents

List of Figures

List of Tables

1 Introduction

In the early 2000's, Taiwanese banks over-issued cash and credit cards to unqualified applicants in order to increase market share. At the same time, most cardholders, irrespective of their repayment ability, overused credit cards and accumulated heavy debts.

Credit card *default* occurs when a credit card holder become severely delinquent on his payments. It is a delicate status because it will affect the relationship among the client and the card issuer but more important, it will decrease the probability to get approved for future credit-based services. [1].

In this project, our job is to reliably predict who is likely to default. If so, the bank may be able to prevent the loss by providing the customer with alternative options (such as forbearance or debt consolidation, etc.). Hence, we build an automated model based on the information about the client and historical transactions, capable of identifying key factors and predicting a credit card default.

2 Exploratory Data Analysis

2.1 Dataset description

The study is conducted on the *Default of credit card clients* dataset from the UCI machine learning repository. It gathers $30,000$ observations made by a bank in Taiwan on distinct credit card holders from April 2005 to September 2005. The dataset employs the binary variable `default.payment.next.month` as response variable. It indicates whether or not the credit card holders are defaulters next month (Yes $= 1$, No $= 0$). Then, the following 23 variables are used as explanatory variables. We report the official description from [2] .

- Client personal information

    1. `LIMIT_BAL`: Amount of given credit (in *New Taiwan* dollars): it includes both the individual consumer credit and his/her family (supplementary) credit.
    2. `SEX`: 1 = male; 2 = female.
    3. `EDUCATION`: 1 = graduate school; 2 = university; 3 = high school; 4 = others.
    4. `MARRIAGE`: Marital status, 1 = married; 2 = single; 3 = others.
    5. `AGE`: Age in years.

- History of past payments from April to September 2005, i.e., the delay of the past payment referred to a specific month:

    6. `PAY_0`: Repayment status in September, 2005.
    7. `PAY_2`: Repayment status in August, 2005.
    8. `PAY_3`: Repayment status in July, 2005.

9. PAY_4: Repayment status in June, 2005.
10. PAY_5: Repayment status in May, 2005.
11. PAY_6: Repayment status in April, 2005.

The measurement scale for the repayment status is: -1 = pay duly;
1 = payment delay for one month; 2 = payment delay for two months; ...;
8 = payment delay for eight months; 9 = payment delay for nine months and above.

- Amount of bill statement (in *New Taiwan* dollars), i.e. a monthly report that credit card companies issue to credit card holders in a specific month:

12. BILL_AMT1: Amount of bill statement in September, 2005.
13. BILL_AMT2: Amount of bill statement in August, 2005.
14. BILL_AMT3: Amount of bill statement in July, 2005.
15. BILL_AMT4: Amount of bill statement in June, 2005.
16. BILL_AMT5: Amount of bill statement in May, 2005.
17. BILL_AMT6: Amount of bill statement in April, 2005.

- Amount of previous payment (in *New Taiwan* dollars):

18. PAY_AMT1: Amount of previous payment in September, 2005.
19. PAY_AMT2: Amount of previous payment in August, 2005.
20. PAY_AMT3: Amount of previous payment in July, 2005.
21. PAY_AMT4: Amount of previous payment in June, 2005.
22. PAY_AMT5: Amount of previous payment in May, 2005.
23. PAY_AMT6: Amount of previous payment in April, 2005.

In Figure 1 we can understand what the data looks like. The target default.payment.next.month is renamed DEFAULT to be short, while the PAY_0 column is renamed PAY_1.

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	...	PAY_6	BILL_AMT1	...	BILL_AMT6	PAY_AMT1	...	PAY_AMT6	DEFAULT
1	20000.0	2	2	1	24	2	...	-2	3913.0	...	0.0	0.0	...	0.0	1
2	120000.0	2	2	2	26	-1	...	2	2682.0	...	3261.0	0.0	...	2000.0	1
3	90000.0	2	2	2	34	0	...	0	29239.0	...	15549.0	1518.0	...	5000.0	0
4	50000.0	2	2	1	37	0	...	0	46990.0	...	29547.0	2000.0	...	1000.0	0
5	50000.0	1	2	1	57	-1	...	0	8617.0	...	19131.0	2000.0	...	679.0	0
6	50000.0	1	1	2	37	0	...	0	64400.0	...	20024.0	2500.0	...	800.0	0
7	500000.0	1	1	2	29	0	...	0	367965.0	...	473944.0	55000.0	...	13770.0	0
8	100000.0	2	2	2	23	0	...	-1	11876.0	...	567.0	380.0	...	1542.0	0
9	140000.0	2	3	1	28	0	...	0	11285.0	...	3719.0	3329.0	...	1000.0	0
10	20000.0	1	3	2	35	-2	...	-1	0.0	...	13912.0	0.0	...	0.0	0

Figure 1: Original dataset from UCI machine learning repository, visualized through pandas framework.

Table 1, reported by pandas, shows how there are no missing features for all 30.000 samples in the dataset.

2.2 Data cleaning

2.2.1 Personal Information

With pandas we can also generate descriptive statistics of the dataset, reported in Table 2. We can see how `LIMIT_BAL`, `SEX`, and `AGE` attributes seem to be consistent with the description provided by [2] , while there are some problems with `EDUCATION`, which ranges from 0 to 6 and it should range from 1 to 4, and `MARRIAGE`, which start at 0, while it should range from 1 to 3.

In general, errors in dataset can be handled by correcting them or by deleting the whole associated entry. In this specific case there are 399 entries which present an error for `EDUCATION` or `MARRIAGE`, which is a relatively small number with respect to the dimension of the dataset (ca. 1.33%), for this reason we think that it is reasonable to delete entries with errors. At this step the size of our dataset changes from 30.000 to 29.601 samples.

2.2.2 History of past payments

Attributes relative to the history of payment of the user are columns `PAY_1, ... , PAY_6`. Looking at the relative descriptive report, we found that values range from -2 to 8 instead of -1 to 9. We decide to rescale these attributes' value with the following strategy:

- if value $< 0 \Rightarrow \mathbf{value} = -\mathbf{1}$
- if value $\geq 0 \Rightarrow \mathbf{value} = \mathbf{value} + \mathbf{1}$

#	Column	Non-null Count	Dtype
0	LIMIT_BAL	30000 non-null	float64
1	SEX	30000 non-null	int64
2	EDUCATION	30000 non-null	int64
3	MARRIAGE	30000 non-null	int64
4	AGE	30000 non-null	int64
5	PAY_1	30000 non-null	int64
6	PAY_2	30000 non-null	int64
7	PAY_3	30000 non-null	int64
8	PAY_4	30000 non-null	int64
9	PAY_5	30000 non-null	int64
10	PAY_6	30000 non-null	int64
11	BILL_AMT1	30000 non-null	float64
12	BILL_AMT2	30000 non-null	float64
13	BILL_AMT3	30000 non-null	float64
14	BILL_AMT4	30000 non-null	float64
15	BILL_AMT5	30000 non-null	float64
16	BILL_AMT6	30000 non-null	float64
17	PAY_AMT1	30000 non-null	float64
18	PAY_AMT2	30000 non-null	float64
19	PAY_AMT3	30000 non-null	float64
20	PAY_AMT4	30000 non-null	float64
21	PAY_AMT5	30000 non-null	float64
22	PAY_AMT6	30000 non-null	float64
23	DEFAULT	30000 non-null	int64

Table 1: Info returned by pandas on dataframe cointaining the given dataset

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE
count	30.000,0	30.000,0	30.000,0	30.000,0	30.000,0
mean	16.7484,3	1,6	1,8	1,5	35,4
std	12.9747,6	0,4	0,7	0,5	9,2
min	10.000,0	1,0	0,0	0,0	21,0
25%	50.000,0	1,0	1,0	1,0	28,0
50%	140.000,0	2,0	2,0	2,0	34,0
75%	240.000,0	2,0	2,0	2,0	41,0
max	1.000.000,0	2,0	6,0	3,0	79,0

Table 2: Descriptive report for personal information attributes

2.2.3 Amount of Bill Statement and Amount of Previous Payment

Again it was possible to inspect values of these attributes (`BILL_AMT1, ... , BILL_AMT6` and `PAY_AMT1, ... , PAY_AMT6`) generating the descriptive table using pandas. In this case we do not find any anomaly in the dataset.

2.3 Data Exploration

In this chapter we'll try to better understand the relationship between features in the dataset and the target variable `DEFAULT`.

2.3.1 Target feature - The imbalance problem

At a first look we notice that the dataset we are dealing with is quite **imbalanced**, we have 6605 defaulters out of a total of 29601 entries (22.3%). This imbalance problem will make classification models focusing on the majority class overlooking the minority class if not addressed. [3]

2.3.2 Continuous features

In statistics, the Kernel Density Estimation (KDE) is a well known technique for estimating the probability density function in a non parametric way (i.e. it does not assume any underlying distribution). We will use KDE for inspecting continuous attributes.

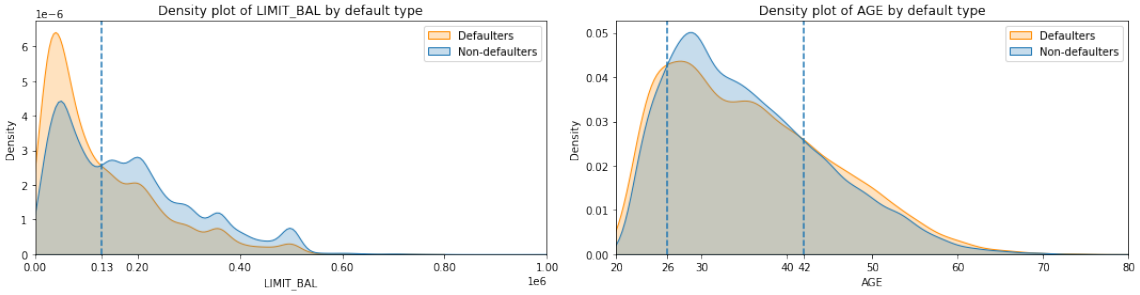

Figure 2: KDE plot of AGE and LIMIT_BAL grouped by class

Exploring `LIMIT_BAL` and `AGE` features, as shown in Figure 2, we can spot some interesting patterns:

- When the given credit amount is below 130.000 the probability of default is higher than non-default.
- Customers with age between 26 and 42 have less probability of default.

We consider these findings to be in line with reality, in fact we have lower probability of default with higher credit limit that usually is given to people with higher creditworthiness and so a lower likelihood to default. In the same way, customers with age between 26 and 42, which tend to have more stable work and family, have a higher capability to repay and so, their probability to default is lower than others. Instead, looking at all other continuous attributes (`BILL_AMTn` and `PAY_AMTn`) we couldn't find any blatant behavior.

2.3.3 Categorical features

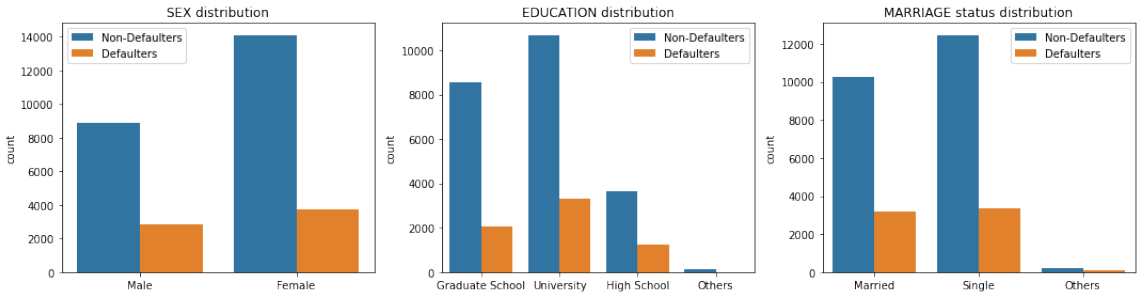

Figure 3: Countplot for SEX, EDUCATION and MARRIAGE features, grouped by DEFAULT class

Regarding features `SEX`, `EDUCATION`, and `MARRIAGE`, looking at distributions in Figure 3 and counts in Table 3 we can see that, even though the values distribution within each attribute is imbalanced (i.e.: we have much more Females than Males and we have that School graduated plus University graduated are almost the totality of the dataset), the proportion of DEFAULTERS and NON-DEFAULTERS per each value reflects more or less the general situation of the entire dataset.

attribute	value	count	defaulters	(%)
SEX	Female	17.855	3.744	$20,96\%$
	Male	11.746	2.861	$24,35\%$
EDUCATION	University	14.024	3.329	$23,73\%$
	Graduate school	10.581	2.036	$19,24\%$
	High school	4.873	1.233	$25,30\%$
	Other	123	7	$5,70\%$
MARRIAGE	Single	15.806	3.329	$21,06\%$
	Married	13.477	3.192	$23,68\%$
	Others	318	84	$26,4\%$

Table 3: Value counts for SEX, EDUCATION and MARRIAGE features

We still have to inspect `PAY_n` features, in this case the box-plot in Figure 4 is very useful. Here we can have a look on the outliers of the features, but also on the impact of each feature on the classification of the customer.

I.e.: the repayment status in September, `PAY_1`, which has well separated boxes of the two classes, holds a greater discriminatory power than the repayment status in the other months.

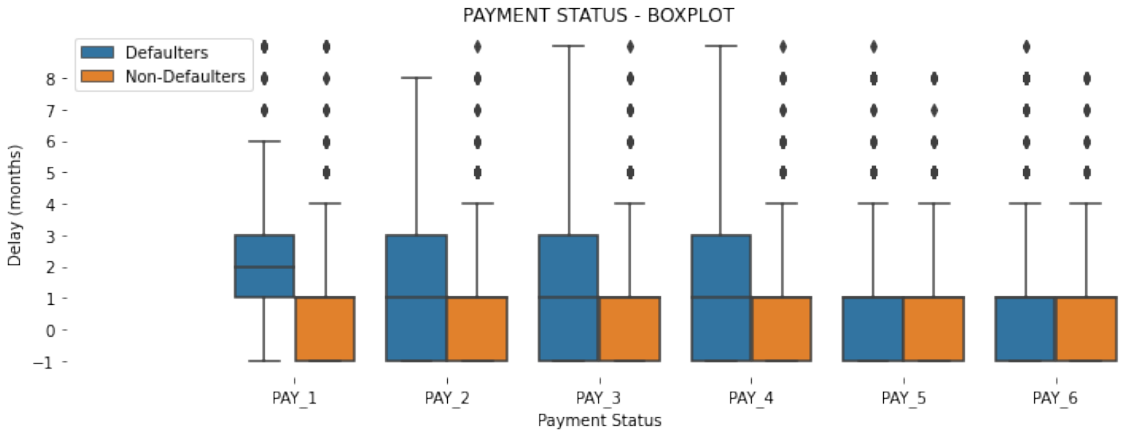

Figure 4: Box-plot for PAYn features, grouped by DEFAULT class

2.4 Checking for normality - QQ plot

Methods we will use later assume that the data have a known and specific distribution, often Gaussian. Applying such methods on data differently distributed my be misleading. To check if we have or not Gaussian distributed data we use the Quantile-Quantile (Q-Q) plot.

Q-Q plot is a graphical method that give us a qualitative evaluation of the data distribution, in this method quantiles of the independent variables are plotted against the expected quantiles of the normal distribution. If the variable is normally distributed the dots in the Q-Q plot will fall along a 45° diagonal.

The Figure 5 shows that numerical features in our data do not follow a Normal distribution.

2.5 Features correlation

The dataset we are dealing with has many attributes. In cases like this it is important to study the correlation among features. In fact, the presence of strongly correlated features may lead to a performances drop. Indeed, there are some methods that assume the independence of predictors and benefit from dimensionality reduction and features selection.

A measure that helps to find out relationship between two variables, measuring the strength of their association, is the Person's Correlation Coefficient (ρ).

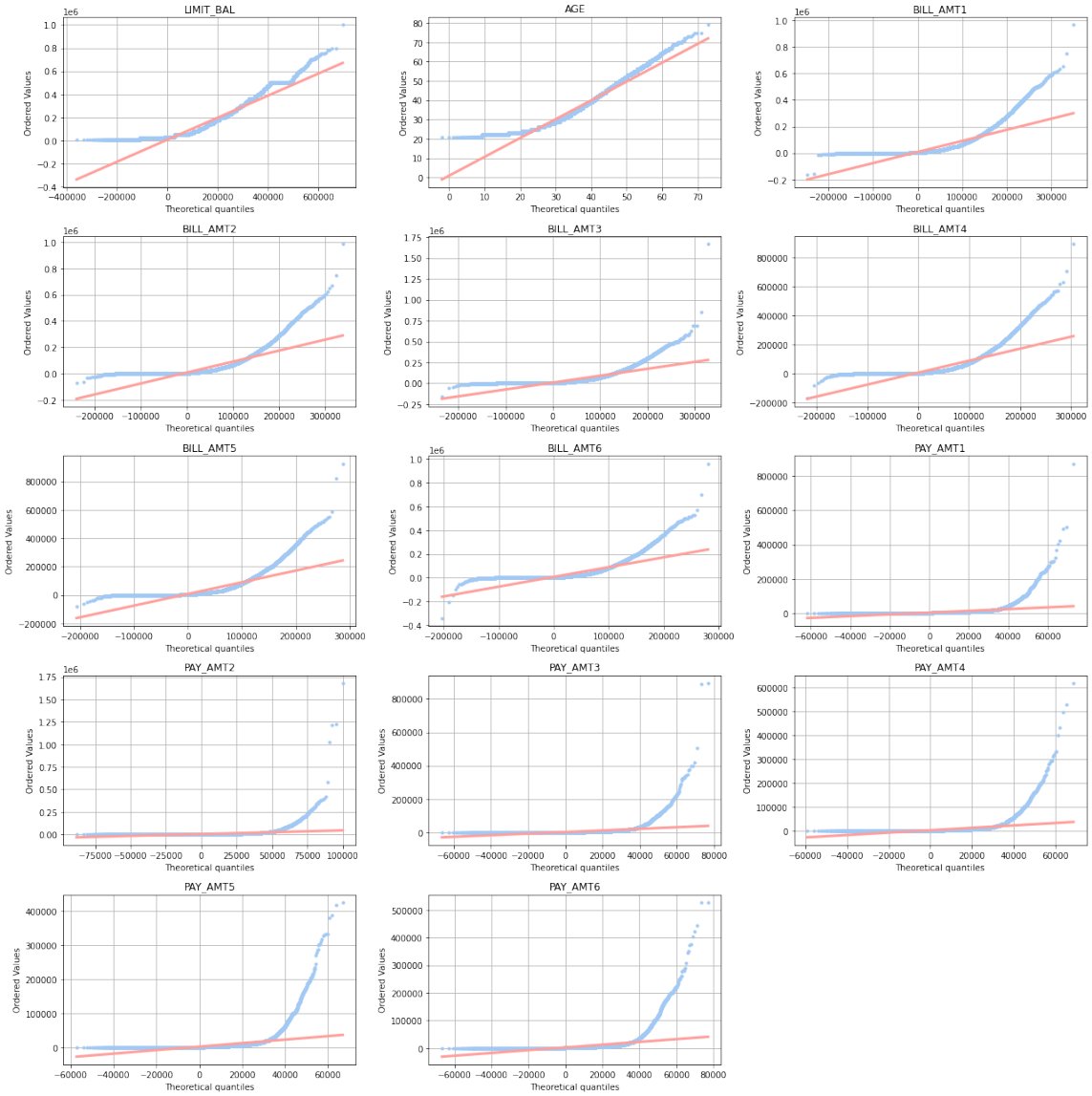

Figure 5: QQ plot for features LIMIT_BAL, AGE, BILL_AMTn, PAY_AMTn

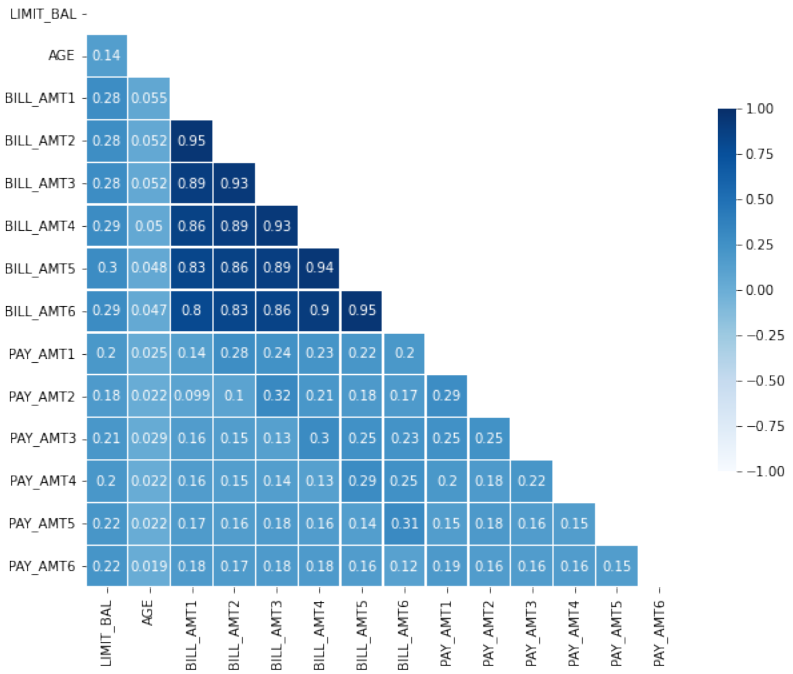

Figure 6: Correlation matrix for all numeric features

Pearson's Correlation Coefficient is defined as the covariance between the two features X and Y (numerator), divided by the product of their deviations (denominator).

$$\rho_{X,Y} = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y} \tag{1}$$

The value of Pearson's Correlation Coefficient can be between -1 and $+1$.

- $+1$ means that they are highly correlated
- 0 means no correlation
- -1 means that there is a negative correlation (inverse proportion).

We also need to remember that Pearson's Correlation Coefficient can only capture linear trends, while it can overlook strongly non-linear correlations even.

A useful tool to inspect correlation in the dataset is the Correlation matrix shown in Figure 6.

We notice that there is a strong positive correlation between the `BILL_AMTn` features. This may indicate a redundancy of information.

3 Data preprocessing

Examine and preprocess our dataset is a critical phase to take before feed it to a learning algorithm as it is know how the quality of the data and the information which they carry are fundamental factors that determines the learning capabilities of an algorithm.

3.1 Categorical features encoding

The categorical features `EDUCATION`, `SEX`, and `MARRIAGE` are already encoded with integer numbers but this may imply an ordering which would be sub-optimal.

One-hot encoding allows us to remove any meaningless ordinal relationship. In practice we create a new dummy binary feature for each unique value in the nominal feature column, following the scheme below:

- `MALE`: {1 = MALE, 0=FEMALE}
- `MARRIED`: {1= MARRIED 0=(OTHERWISE)}
- `SINGLE`: {1= SINGLE 0=(OTHERWISE)}
- `GRAD_SCHOOL`: {1 = SCHOOL GRADUATE, 0=OTHERWISE}
- `HIGH_SCHOOL`: {1 = HIGH SCHOOL GRADUATE, 0=OTHERWISE}
- `UNIVERSITY`: {1 = UNIVERSITY GRADUATE, 0=OTHERWISE}

Using this strategy we do not loose any information.

In Table 4 and Table 5 there is an example that shows variable respectively before and after the application of one-hot encoding.

id	SEX	MARRIAGE	EDUCATION
0	1	1	1
1	2	2	2
2	1	3	3
3	2	1	4

Table 4: SEX, MARRIAGE and EDUCATION features before one-hot encoding

id	MALE	MARRIED	SINGLE	GRAD_SCHOOL	UNIVERSITY	HIGH_SCHOOL
0	True	True	False	True	False	False
1	False	False	True	False	True	False
2	True	False	False	False	False	True
3	False	True	False	False	False	False

Table 5: SEX, MARRIAGE and EDUCATION features after one-hot encoding

3.2 Dataset partitioning

In order to evaluate performances and generalization capability of machine learning algorithm it is needed to separate the dataset into training and test sets.

We split the dataset in two partition, reserving 75% of entries for the training and 25% for the test set, in particular we also apply stratification to ensure that the DEFAULT class proportion is preserved. After applying the partitioning we obtain:

- Training set
  - shape: $(22200, 26)$
  - defaulters: 4.954
  - non-defaulters: 17.246
  - class proportion: $22,30\%$
- Test set
  - shape: $(7401, 26)$
  - defaulters: 1.651
  - non-defaulters: 5.750
  - class proportion: $22,30\%$

3.3 Features scaling

The majority of machine learning and optimization algorithms behave much better if numerical features are on the same scale. Decision trees and Random forest are two of the very few machine learning algorithms where there is no need to worry about feature scaling, as they are scale invariant [4].

There are two common approaches for bringing different features onto the same scale:

- normalization: rescaling of the features to a range of [0,1] , in this case we use min-max scaling:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2}$$

Where X is a particular feature, X_{min} is the smallest value and X_{max} is the largest value.

- standardization: centers the feature columns at mean $\mu = 0$ with standard deviation $\sigma = 1$ so that the feature columns have the same parameters as a standard normal distribution, which makes it easier to learn the weights.

$$Z = X - \mu \tag{3}$$

In this work we decided to use the **normalization** approach.

3.4 Dimensionality reduction

As already mentioned several algorithm would benefit from the removal of highly correlated features, hence in general having lower dimension in the dataset lead to simpler models that are better explainable and which generalize better in the testing environment.

In fact, high dimensional data can make the model overfitting the training set. The problem of overfitting become more serious for example in *Nearest Neighbor classifiers*, where we face the *course of dimensionality* problem. The curse of dimensionality describes the phenomenon where the feature space becomes increasingly sparse for an increasing number of dimensions of a fixed-size training dataset. We can think of even the closest neighbors as being too far away in a high-dimensional space to give a good estimate.

The dimensionality reduction can be carried out by means of:

- **feature selection**: we select a subset of the original features
- **feature extraction**: we construct a new feature subspace deriving information from the original feature set.

We already found that `BILL_AMTn` categories are highly correlated between them so we could manually discard them, instead we decide to keep them to perform in this chapter the *Principal Component Analysis* (PCA).

The dimensionality reduction is closely associated with the concept of *lossy compression*. The idea behind dimensionality reduction is taking data in a high dimensional space and mapping it, applying a linear transformation, into a new space whose dimensionality is much smaller.

That is, if the original data is in $\mathbb{R}^d$, PCA can be seen as embedding it into $\mathbb{R}^n$ ($n < d$). It means to find a matrix $W \in \mathbb{R}^d$ that induces the mapping $x \mapsto Wx$. A natural criterion for choosing W is in a way that will enable a reasonable recovery of the original x. It is not hard to show that in general, exact recovery of x from Wx is impossible.

3.4.1 Principal Component Analysis (PCA)

In PCA, both the compression and the recovery are performed by linear transformations. Let $x_1, .., x_m$ be m vectors in $\mathbb{R}^d$. We would like to reduce the dimensionality of these vectors using a linear transformation.

The aim is to find a matrix $W \in \mathbb{R}^{n,d}$, where $n < d$, that induces a mapping

$$x \mapsto Wx; \qquad x \in \mathbb{R}^d, Wx \in \mathbb{R}^n; \qquad (n < d) \tag{4}$$

and a second matrix $U \in \mathbb{R}^{d,n}$ that can be used to recover each original vector x from its compressed version. That is, for a compressed vector $y = Wx$, where y is in the low dimensional space $\mathbb{R}^n$, we can construct $\tilde{x} = Uy$, so that $\tilde{x}$ is the recovered version of x and resides in the original high dimensional space $\mathbb{R}^d$.

The objective of PCA is to find the compression matrix W and the recovering matrix U so that the total squared distance between original and recovered vector is minimal in the least squared sense:

$$\underset{W \in \mathbb{R}^{n,d},\, U \in \mathbb{R}^{d,n}}{\operatorname{argmin}} \sum_{i=1}^{m} \|\mathbf{x}_i - UW\mathbf{x}_i\|_2^2 \tag{5}$$

The optimal solution (U, W) of this problem takes the form:

- the columns of U are orthonormal: $U^\top U = \mathbb{I}_n \in \mathbb{R}^n$;
- $W = U^\top$.

Hence, we can write:

$$\underset{U \in \mathbb{R}^{d,n} : U^\top U = \mathbb{I}_n}{\operatorname{argmin}} \sum_{i=1}^{m} \|\mathbf{x}_i - UU^\top\mathbf{x}_i\|_2^2 \tag{6}$$

It can be proven that:

$$\|\mathbf{x} - UU^\top\mathbf{x}\|^2 = \|\mathbf{x}\|^2 - \operatorname{tr}\left(U^\top\mathbf{x}\mathbf{x}^\top U\right) \tag{7}$$

where the trace tr of a matrix is the sum of its diagonal entries, i.e., a linear operator. This allows us to rewrite Equation (6) as the following maximization problem:

$$\underset{U \in \mathbb{R}^{d,n} : U^\top U = \mathbb{I}_n}{\operatorname{argmax}} \operatorname{tr}\left(U^\top \sum_{i=1}^{m} \mathbf{x}_i\mathbf{x}_i^\top U\right) \tag{8}$$

Now, we can define the **scatter matrix** $A = \sum_{i=1}^{m} \mathbf{x}_i\mathbf{x}_i^\top$, since this matrix is symmetric, it can be written using its spectral decomposition $A = VDV^\top$, where D is diagonal and $V^\top V = VV^\top = \mathbb{I}_n$. Here, the elements on the diagonal of D are the eigenvalues of A, while

the columns of V are the corresponding eigenvectors. In particular, we can safely assume that the diagonal elements of D are sorted by the largest, and are all positive because A is semidefinite positive:

$$D_{1,1} \geq D_{2,2} \geq ... \geq D_{d,d} \geq 0 \tag{9}$$

From these premises, we can claim that the solution of the optimization problem (6) is the matrix U whose columns $\mathbf{u}_1, ..., \mathbf{u}_n$ are the n eigenvectors of the matrix A corresponding to the largest n eigenvalues, while $W = U^\top$.

In other words, PCA helps us to identify patterns in the data based on the correlation between features, finding the directions of maximum variance in high-dimensional data and projecting it onto a new subspace with equal or fewer dimensions than the original one.

The orthogonal axes (principal components) of the new subspace can be interpreted as the directions of maximum variance given the constraint that the new feature axes are orthogonal to each other as illustrated in the Figure 7. Here, x_1 and x_2 are the original feature axes, and PC_1 and PC_2 are the principal components. [5]

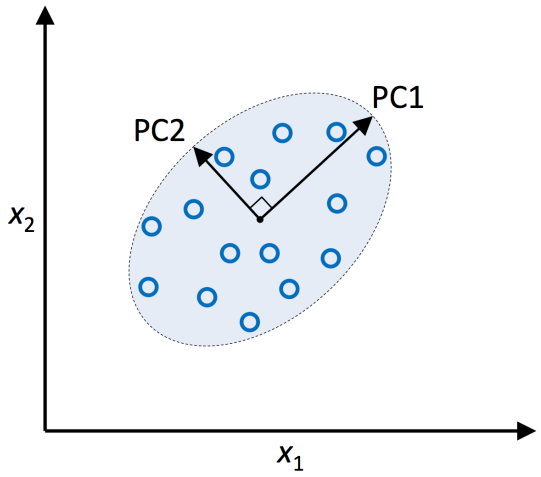

Figure 7: PCA graphical explaination

Since we want to reduce the dimensionality of our dataset by compressing it onto a new feature subspace, we will choose the number of principal components that preserve at the most the information carried by data. We can take such a decision looking at the explained variance ratio, that is how much variance does contain each principal component out of the d components we obtain applying PCA to our training dataset.

Results depicted in Figure 8 show that the first 5 principal components capture more than 90% of the total variance. However, we choose to consider the first 12 principal components that are capable to explain 99% of the variance, even if the number of features is halved.

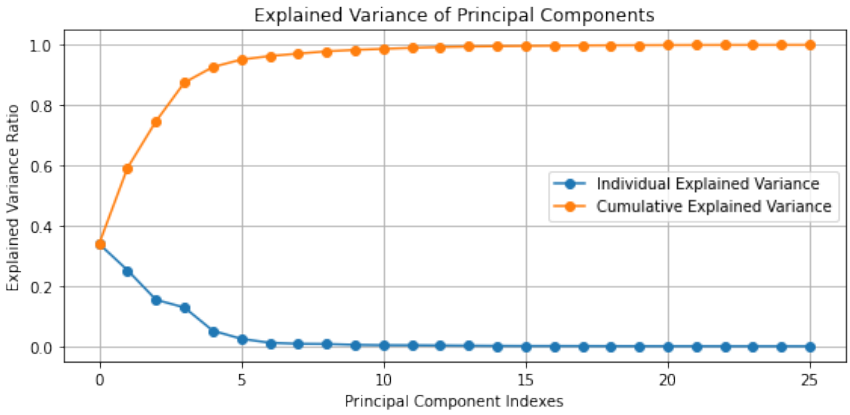

Figure 8: Cumulative and individual explained variance plotted against principal components found

3.5 Class imbalance

Class imbalance is a quite common problem when working with real-world data, it consists of having much more samples that refers to one class than other classes in the dataset.

As already stated in Section 2.3 and reported in Table 6 , the dataset we are dealing with is highly imbalanced, as the non-defaulter samples are over-represented with respect to defaulter samples.

	count	percentage
Non-defaulters	17.246	$77,68\%$
Defaulters	4.954	$22,32\%$

Table 6: Count for the DEFAULT feature, the dataset is highly imbalanced

In our case, a dummy model which predicts only the majority class on a test set that is distributed like the training set could achieve roughly 80% accuracy. Thus class imbalance problem point us also the choice of the right metric to consider when comparing different machine learning models.

Class imbalance can also affect the learning algorithm fitting phase returning a decision rule biased toward the majority class.

There are three options to tackle this problem:

- At training time assign larger penalty to wrong prediction of the minority class
- Oversampling the minority class

- Undersampling the majority class

Unfortunately there is not a generalized best solution, but it is suggested to try out different strategies and chose the technique that seems most appropriate.

Talking about oversampling and undersampling, we exclude the naïve methods, since they randomly duplicate and respectively delete data from the majority/minority class, moreover in naïve undersampling we can not control which information is discarded, while naïve oversampling causes overfitting because the model is trained on many identical samples.

We decide to explore two techniques which mitigate exposed issues, that are Cluster Centroids (undersampling) and SMOTE - Synthetic Minority Oversampling Technique (oversampling).

3.5.1 Cluster centroids

Minority class samples in our dataset are enough to let us perform undersampling. However, one of the major issues of undersampling is that some important information from the majority class may be lost. To overcome this problem the Cluster Centroids method has been proposed by Yen et al. in [6].

The idea in this method is to replace clusters of majority samples with the respective cluster centroids. To this scope a K-means algorithm is fitted to the data with number of clusters set equal to the number of sample of the minority class. Then, the majority of samples from the clusters are entirely substituted by the sets of cluster centroids from K-means [7].
To avoid overfitting issues, the method is fitted only on the training partition of the dataset.

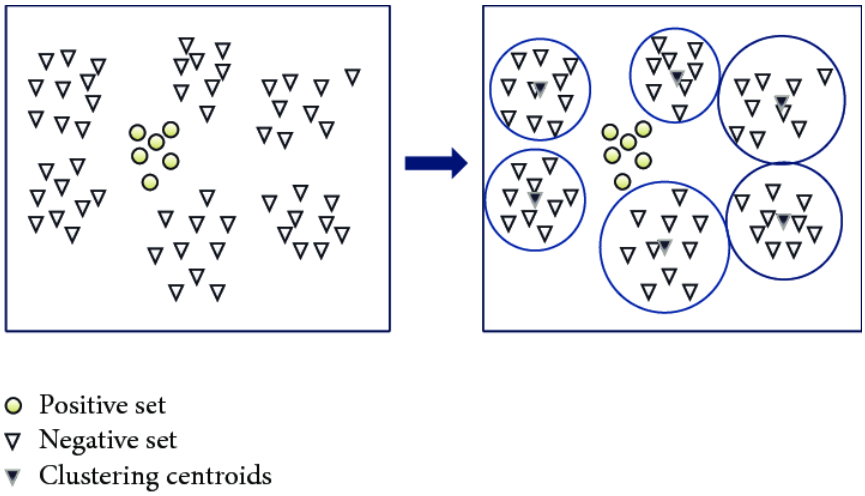

Figure 9: Cluster centroids undersampling [8]

3.5.2 Synthetic Minority Oversample TEchnique (SMOTE)

We already mentioned the overfitting weakness of random oversampling. SMOTE has been proposed in [9] to overcome this issue. Instead of merely replicating existing observations, this technique generates new artificial samples.

SMOTE generate a new sample by linearly interpolating a random selected minority entry and one of its minority neighboring.

Precisely, the algorithm goes through three steps:

1. Choose a random minority observation $\vec{a}$.
2. Among $\vec{a}$'s k nearest minority neighbors select the instance $\vec{b}$.
3. Create a new sample $\vec{x}$ by linearly interpolating $\vec{a}$ and $\vec{b}$ (with w random).

$$\vec{x} = \vec{a} + w \times (\vec{b} - \vec{a}), \qquad w \in [0, 1] \tag{10}$$

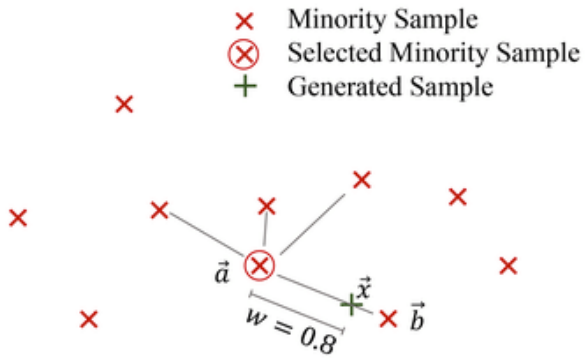

Figure 10: SMOTE linearly interpolates a randomly selected minority sample and one of its k = 4 nearest neighbors [10]

Unfortunately, SMOTE, as other random oversampling techniques suffers from some weakness due to within-class imbalance and noise. In fact, SMOTE choose a random sample from the minority class to start, but if the minority class is not uniformly distributed this can cause densely populated areas to be further inflated with artificial data. Moreover, SMOTE does not recognize noisy minority samples which are located among majority class instances and interpolate them with their minority neighboring, may be creating new noisy entries.

Finally, it has been proven that classification algorithms could benefit from samples that are closer to class boundaries, and SMOTE does not specifically works in this sense, as shown in Figure 11.

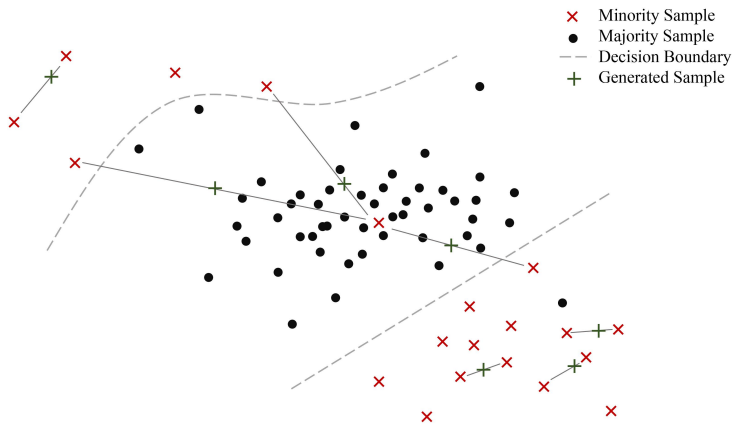

Figure 11: Behavior of SMOTE in presence of noise and with-in class imbalance [10]

3.5.3 K-means SMOTE

The method in [10] employs the simple and popular K-means clustering algorithm in conjunction with SMOTE oversampling in order to rebalance skewed datasets. It manages to avoid the generation of noise by oversampling only in safe areas (i.e., areas made up of at least 50% of minority samples). Moreover, its focus is placed on both between-class imbalance and within-class imbalance, fighting the small disjuncts problem by inflating sparse minority areas.

K-means SMOTE executes three steps: clustering, filtering, and oversampling as show in Figure .

1. Clustering: the input space is clustered into k groups using K-means clustering.
2. Filtering: selects clusters for oversampling, retaining those with a high proportion of minority class samples. It then distributes the number of synthetic samples to generate, assigning more samples to clusters where minority samples are sparsely distributed.
3. Oversampling: SMOTE is applied in each selected cluster to achieve the target ratio of minority and majority instances.

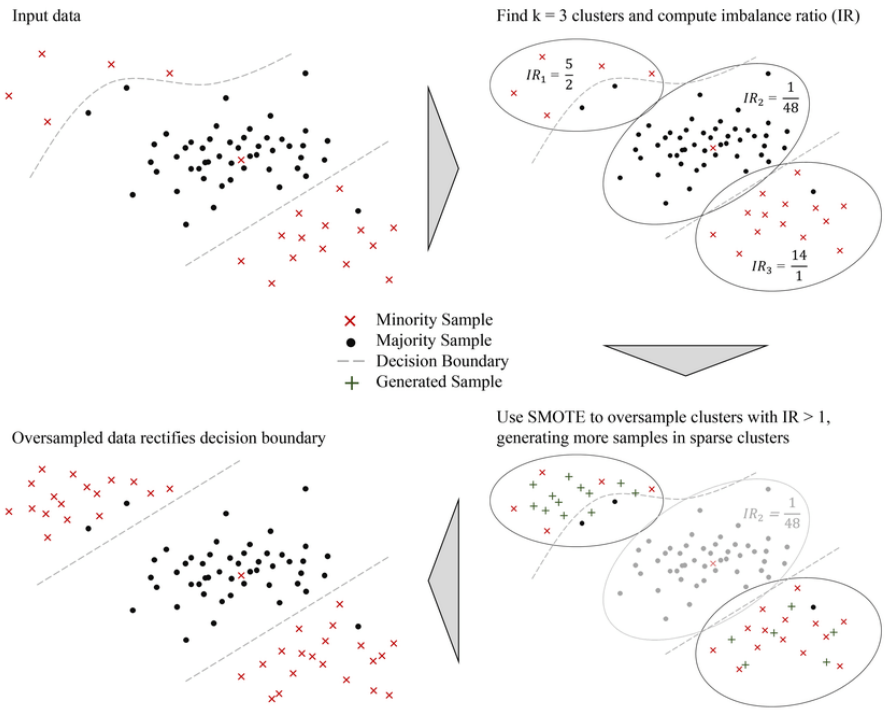

Figure 12: K-means SMOTE oversamples safe areas and combats within-class imbalance [10]

4 Model evaluation strategy

4.1 Validation

While developing a machine learning model, it is extremely important to estimate its performances on data that the model has never seen before. In fact in section 3.2, we split the original dataset into training and test sets. The former will be used during the training phase to make the model learning from it, the latter will be used at the end to test how the obtained model generalizes.

However, typical machine learning models have also some hyperparameters that need to be tuned, to further improve the generalization capability of the model. Tuning these hyperparameters on the test dataset obtained before is not recommended. Reusing over and over the test set in this process would end up of making the test set part of the training set, moreover also partitioning the test set to get a proper validation set (hold-out method) may influence the future performance estimation based on how we decide to partition it.

4.2 K-fold cross validation

A more robust technique than the hold-out is the **k-fold cross-validation** in which we repeat the hold-out method k times on k subsets of the training data.

In practice, we randomly split the training dataset into k folds, for each unique fold we take it as an holdout dataset and use other folds as training set. We then fit the model and evaluate it on the obtained sets. Once obtained the best hyperparameters, we train again the model on the whole training set and finally evaluate it on the test set.

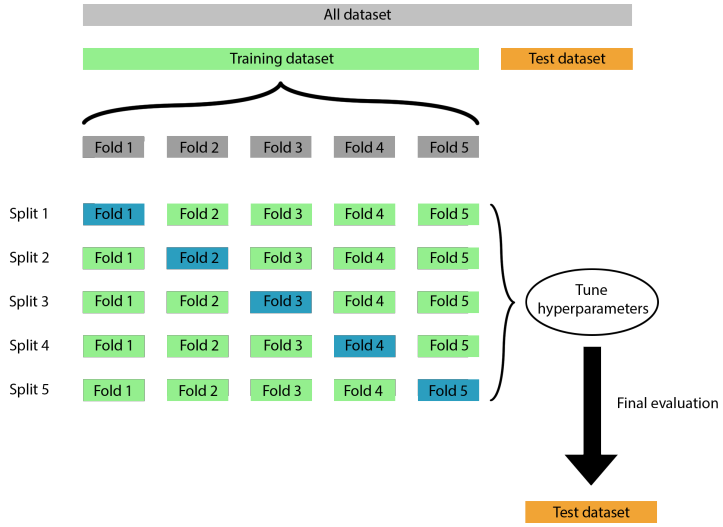

Figure 13: K-fold cross validation (i.e.: $k = 5$)

Since k-fold cross validation is a resampling technique without replacement, its advantage is that each sample of the training set will be used exactly once for validation purpose, and this yields to a lower-variance estimate of the model performances than the holdout method.

A common value for k in k-fold cross validation is 10 [11]. However we are dealing with a large dataset and choosing such a value would be too costly in computational sense, for this reason we decide to use a smaller value for k, i.e. $k = 5$, still taking advantage of the k-fold method.

4.3 Performance evaluation

Classification evaluation metrics compare the predicted class from the classifier with the actual class of the samples. At this scope it is useful to build a **confusion matrix** where the (j, k)-th element counts the number of times that the actual class is j whereas the predicted class is k.

In the binary case (such as our case) the confusion matrix become simpler to build and to interpret. If the two classes to be predicted are *True* and *False* the confusion matrix is the one depicted in Table 7.

		Actual class	
		Positive	**Negative**
Predicted class	**Positive**	TP	FP
	Negative	FN	TN

Table 7: Binary classification confusion matrix

The following terminology is often used when referring to the counts tabulated in a confusion matrix:

- **True positive (TP)**: the number of positive examples correctly predicted by the classification model.
- **False negative (FN)**: the number of positive examples wrongly predicted as negative by the classification model.
- **False positive (FP)**: the number of negative examples wrongly predicted as positive by the classification model.
- **True negative (TN)**: the number of negative examples correctly predicted by the classification model.

4.3.1 Accuracy, Precision and Recall

Starting from the confusion matrix we can define different metrics for the evaluation of our model.

- **Accuracy**: is the performance measure generally associated with machine learning algorithms. It is the ratio of correct predictions over the total number of data points classified.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{11}$$

- **Precision** (also called *positive predictive value*): Indicates how many of a j-object (in binary classification, commonly *True* class is considered) predictions are correct. It is defined as the ratio of correct positive predictions over all the positive predictions.

$$\text{Precision, } p = \frac{TP}{TP + FP} \tag{12}$$

- **Recall** (also called *sensitivity*): Indicates how many of the j-object (in binary classification, commonly *True* class is considered) samples are correctly classified. It is defined as the fraction of j-object predictions over the total number of j-object samples.

$$\text{Recall, } r = \frac{TP}{TP + FN} \tag{13}$$

Accuracy measure treats every class as equally important, for this reason it may be not suitable for imbalanced datasets, where the *rare class* is considered more interesting than the majority class [12]. Hence, *Precision* and *Recall*, that are class-specific metrics, are widely employed in applications in which successful detection of the rare class is more significant than detection of the other.

The challenge is to build a model that is capable of maximize both *Precision* and *Recall*. Hence, the two metrics are usually summarized in a new metric that is **F1-score**. In practice *F1-score* represents the harmonic mean between precision and recall, so, an high value ensures that both are reasonably high.

$$\text{F1-score} = \frac{2}{\frac{1}{r} + \frac{1}{p}} = \frac{2rp}{r + p} \tag{14}$$

5 Classification models

In this section we explore some learning algorithms to build a classification model able to predict credit card default. In particular we will dive into Support Vector Machine, Logistic Regression and some tree based methods, all following the Empirical Risk Minimization paradigm.

5.1 Support Vector Machine

Support Vector Machines (SVMs) are considered among the most effective classification algorithms in modern machine learning [13]. When used for classification tasks, SVMs are supervised learning methods that construct an hyperplane that maximizes the margin between two classes in the feature space.

5.1.1 Hard margin SVM

A hyperplane in a space $\mathcal{H}$ endowed with a dot product $\langle \cdot, \cdot \rangle$ is described by the set

$$\{x \in \mathcal{H} | \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\} \tag{15}$$

where $\mathbf{w} \in \mathcal{H}$ and $b \in \mathbb{R}$

Assuming that $\mathcal{H}$ is linearly separable, such hyperplane naturally divides $\mathcal{H}$ into two half-spaces and hence can be used as the decision boundary of a binary classifier. Given a set of

samples $\mathcal{S} = [x_1, ..., x_m]$, the margin is defined as the distance of the closest point in $\mathcal{X}$ to the hyperplane defined in Equation 15:

$$\min_{i=1,...,m} \frac{|\langle \mathbf{w}, \mathbf{x_i} \rangle + b|}{||\mathbf{w}||} \tag{16}$$

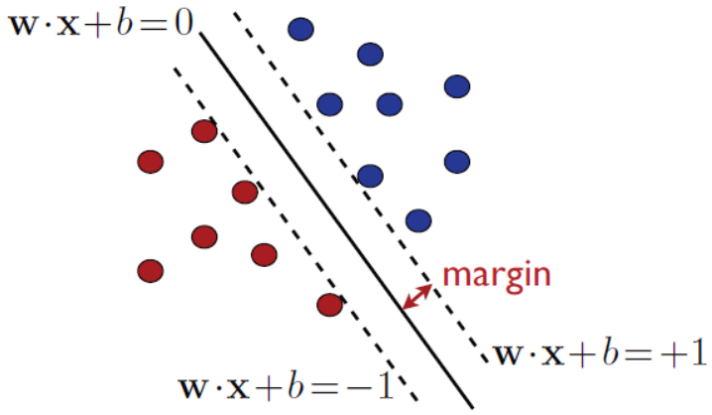

Figure 14: Maximum margin hyperplane [13]

The maximum margin problem the algorithm has to solve can be written as:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{s.t.} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \forall i \end{aligned} \tag{17}$$

5.1.2 From Soft margin SVM to Kernel SVM

In the hard margin formulation, as mentioned, to find a solution to the maximization problem we need the strong assumption that data are linearly separable. In the case in which this assumption does not hold, the Soft SVM formulation is useful. That formulation allows the violation of the constraint for some samples in the training set. Soft margin SVM is formulated introducing non-negative slack variables $\xi_1, ..., \xi_m$ which measure how much the constraint $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ is being violated. Now the problem is to jointly minimize the norm of $\mathbf{w}$ (the margin) and the average of all ξ_i (the average violations to the constraint):

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \qquad \forall i \end{aligned} \tag{18}$$

where C is a penalty parameter, typically determined via k-fold cross validation.

A better looking to the problem in (18) reveals that:

- $\xi_i = 0$ whenever $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$
- $\xi_i = 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ whenever $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq 1$

Let the hinge loss in the context of halfspaces learning be:

$$l^{hinge}(\mathbf{w}, (x, y)) \stackrel{\text{def}}{=} \max\{0, 1 - y\langle \mathbf{w}, x \rangle\}. \tag{19}$$

we have that $\xi_i = l^{hinge}((\mathbf{w}, b), (x_i, y_i))$ and so we can reformulate the Problem in (18) as the unconstrained problem:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}||\mathbf{w}||^2 + L_S^{hinge}(\mathbf{w}, b) \tag{20}$$

Soft SVM is able to handle noise and outliers in almost linearly separable conditions. But most of the time, the data we are dealing with, is not linearly separable, therefore, even softening the margin may not be enough. In these cases it is possible to map the data into an higher dimensional space such that it will be linearly separable.

Let us consider now the dual problem of (18) formulated as:

$$\max_{\alpha \in \mathbb{R}: \alpha \geq 0} (\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle) \tag{21}$$

It is evident that the dual problem only involves inner products between instances, that is nothing but a linear kernel, so there is no restriction of using new kernel functions with the aim of measuring the similarity in higher dimensions.

Given a non-linear mapping $\psi : X \mapsto F$ the kernel function is defined as:

$$K(x, x^{'}) = \langle \psi(x_i), \psi(x_j) \rangle \tag{22}$$

the dual problem in (21) becomes:

$$\max_{\alpha \in \mathbb{R}: \alpha \geq 0} (\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \psi(x_i), \psi(x_j) \rangle) \tag{23}$$

the kernel enables an implicit non-linear mapping of the input points to a high-dimensional space where large-margin separation is sought.

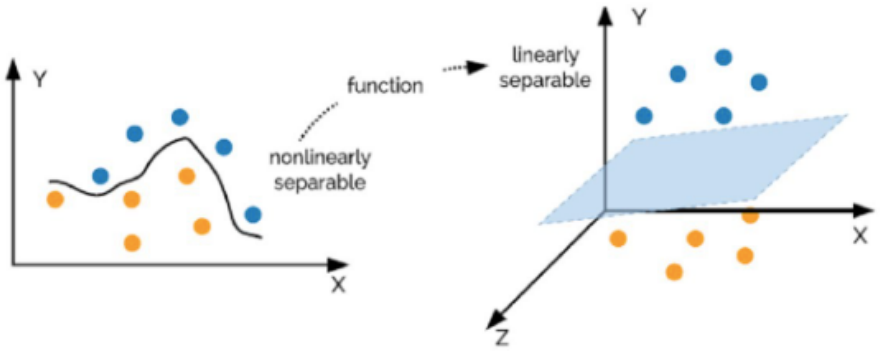

Figure 15: Non linearly separable samples become linearly separable in a higher dimensional space

The complexity now depends on the size of the dataset, because for M data points we need to compute $\binom{M}{2}$ inner products. Kernels explored in our analysis are:

- polynomial kernel: $K(x, x') = (\gamma\langle x, x'\rangle)^d$
- rbf kernel: $K(x, x') = e^{-(\gamma||x-x'||^2)}$

Some combinations of parameters value and kernels have been explored through a Cross-Validated grid search which involved also different choices about the preprocessing phase. Results obtained are reported in Table 8 and Figure 16.

Preprocessing	Parameters	F1-Score
None (Raw data)	kernel: RBF, C : 100, γ : 0.1	$0,4681$
PCA	kernel: RBF, C : 100, γ : 0.1	$0,4665$
PCA + SMOTE	kernel: RBF, C : 100, γ :scale	$0,5247$
PCA + KMeans SMOTE	kernel: RBF, C : 100, γ :scale	$0,4639$
PCA + ClusterCentroids	kernel: RBF, C : 100, γ :scale	$0,4341$

Table 8: Results obtained with SVM model and different preprocessing strategies

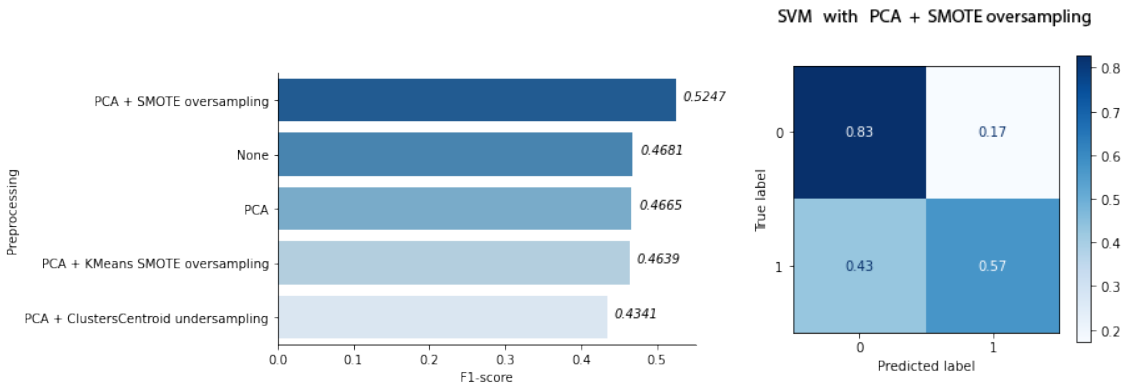

Figure 16: Results obtained with SVM model and different preprocessing strategies on the left, the confusion matrix of the best configuration found on the right

5.2 Logistic Regression

Logistic Regression models are predictors of the family of *Generalized Linear Models* (GLM). GLM are a broad class of models that provide a unifying framework for many commonly used statistical techniques, such as linear regression.

GLM are characterized by three components: the *random component* which identifies the response variable Y and assumes a probability distribution for it, treating the m observations on Y, denoted: $(y_1, ..., y_m)$, as independent; the *linear predictor* that specifies the explanatory variables through a prediction equation that has linear form, such as:

$$\alpha + \beta_1 x_1 + ... + \beta_p x_p = \mathbf{x}^\top \beta; \tag{24}$$

and the *link function* which specifies a function g that relates $\mathbb{E}[Y]$ to the linear predictor such as:

$$g(\mathbb{E}[Y]) = \alpha + \beta_1 x_1 + ... + \beta_p x_p \tag{25}$$

[14].

Hence, in a GLM the expected response for a given feature vector $\mathbf{x} = [x_1, ..., x_n]$ is of the form

$$\mathbb{E}[Y|X = \mathbf{x}] = h(\mathbf{x}^\top \beta) \tag{26}$$

with h, called *activation function*, being the inverse of the link function g. [15]

Rather than directly modeling the distribution of Y, the logistic regression models the probability that Y belongs to a particular class using the logistic function as activation function h:

$$P(Y_i = 1|X = x_i) = h(x_i^\top \beta) = \frac{e^{x_i^\top \beta}}{1 + e^{x_i^\top \beta}} \tag{27}$$

indeed with $x_i^\top \beta$ large we will have an high probability for Y to be 1, and for small $x_i^\top \beta$ we will have an high probability for Y to be 0. The logistic function will always produce an S-shaped curve between 0 and 1,of the form depicted in Figure 17.

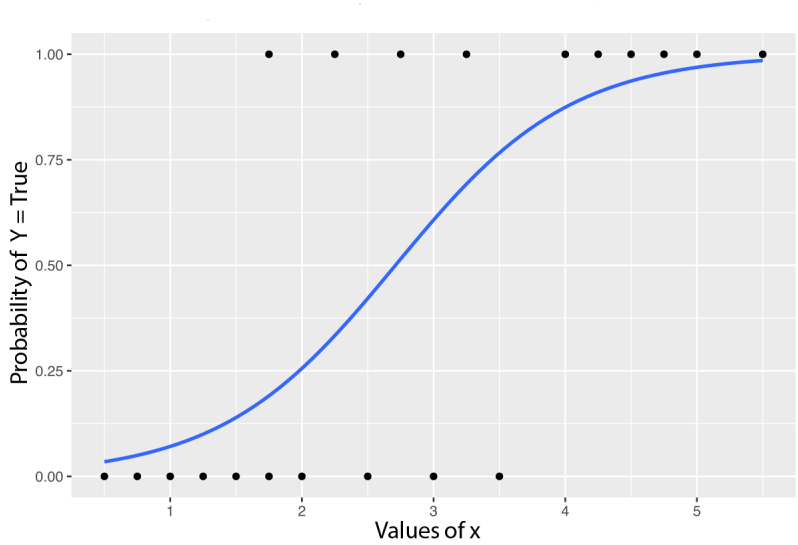

Figure 17: Sigmoid activation function

To estimate the coefficients vector β through the available training data we use the Maximum Likelihood method, which finds $\hat{\beta}$ that is the maximum likelihood estimate of β, this is formalized in such a way:

$$\mathcal{L}(\beta) = \prod_{i=1}^{m} [h(\mathbf{x}_i^\top \beta)]^{y_i} [1 - h(\mathbf{x}_i^\top \beta)]^{1-y_i} \tag{28}$$

where $\mathcal{L}(\beta)$ is the log-likelihood, which maximized with respect to β gives the maximum likelihood estimator of β. In a supervised learning environment that maximization is equivalent to minimizing the function:

$$-\frac{1}{m} \log \mathcal{L}(\beta) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log h(\mathbf{x}_i^\top \beta) + (1 - y_i) \log 1 - h(\mathbf{x}_i^\top \beta)] \tag{29}$$

We can interpret the function in (29) as the *binary cross-entropy training loss* associated with comparing a true conditional probability density function (pdf) with an approximated pdf. In our study we use the Logistic Regression model provided by the `SciKit-learn` python library which by default applies l_2 regularization. Applying a regularization term is useful in reducing the generalization error but not its training error, preventing overfitting on the training set.

The regularization term is added to the objective function, and in practice it penalizes large weights values during the training, in other words regularization term force the weights to go toward 0, and particularly the l_2 regularization does not make them to be 0 (while l_1 does) and it is defined as:

$$l_2(\beta, \lambda) = \frac{\lambda}{2} \sum_{i=1}^{m} \beta_i^2 \tag{30}$$

where λ, called *regularization parameter* help us in tune the regularization strength.

The `Logistic Regression` class in `SciKit-Learn` implements the parameter C as the inverse of the regularization parameter λ.

Combinations of different values of C with different choices about the preprocessing phase have been explored through a Cross-Validated search.

Results obtained are in Table 9 and Figure 18 .

Preprocessing	Parameters	F1-Score
None (Raw data)	C : 50, penalty: l2	$0,3090$
PCA	C : 50, penalty: l2	$0,3028$
PCA + SMOTE	C : 35, penalty: l2	$0,4504$
PCA + KMeans SMOTE	C : 35, penalty: l2	$0,3887$
PCA + ClusterCentroids	C : 20, penalty: l2	$0,4213$

Table 9: Results obtained with Logistic Regression model and different preprocessing strategies

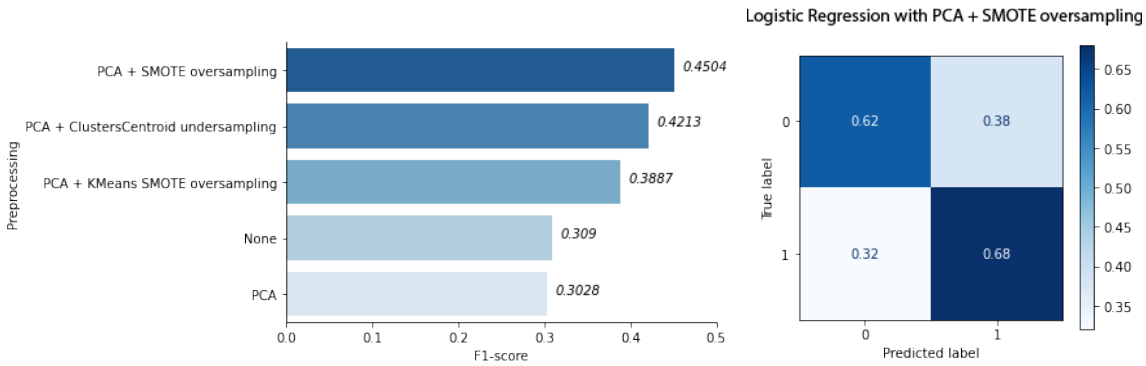

Figure 18: Results obtained with Logistic Regression model and different preprocessing strategies on the left, the confusion matrix of the best configuration found on the right

5.3 Tree based methods

Tree based methods are intuitive and powerful methods which stratify (potentially) complicated feature space into smaller regions and fit a simple prediction function to each region. [15] In order to make a prediction for a given observation we typically use the mean or the mode of the training observations in the region to which it belongs [16].

Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision tree methods [16].

5.3.1 Decision Tree

Decision trees are very interpretable models, they are implemented with binary splits on the predictor variables. Since considering every possible partition is computationally infeasible the tree is constructed with greedy top down approach, known as *recursive binary splitting*, which at each step split the data on the feature that results in the largest *information gain* (IG).

The information gain is simply the difference between the impurity of the parent node and the sum of the child node impurities. In order to split the nodes at the most informative features, we need to maximize the IG at each split.

- if the predictor is continuous, we choose a cut-point that maximizes purity for the two groups created;
- if the predictor variable is categorical, we combine the categories to obtain two groups with maximum purity.

This splitting process is repeated at each child node until we obtain pure leaves. Gini impurity (I_G) and entropy (I_H) are the most commonly used splitting criteria in binary decision trees. Being $p(i|t)$ the proportion of the samples that belong to class i for a particular node t we can define them as:

- Gini Impurity: $I_G(t) = \sum_{i=1}^{c} p(i|t)(1 - p(i|t))$
- Entropy: $-\sum_{i=1}^{c} p(i|t) \log_2(p(i|t))$

However, even though a decision tree is fairly interpretable, it is typically less accurate and robust compared to more sophisticated algorithms.

A more advanced tree based algorithm is *Random Forest*

5.3.2 Random Forest

Random Forest is an ensemble method which reduces the variance of a single model by combining multiple decision trees with the *bagging* technique. The idea behind the bagging

or bootstrap aggregating technique is to generate different *bootstrapped training sets* taking sample with repetition form the dataset.

Random Forest uses the bagging also at each split for the feature selection to *decorrelate* the trees: to generate each splitting rule it is considered a randomly selected subset of features of fixed size.

This is why this algorithm is fairly robust to noise and outliers and will have much less variance rather than a single decision tree. Random forest models are not interpretable as Decision trees, but they allow us to measure the *importance* of each feature. As we will see in the results and in Figure 20 from our analysis on the raw data, it emerges that **repayment status**, **age** and **bill amount** are very important in the decision process.

In the RandomForestClassifier implementation in `SciKit-Learn`, the size of the bootstrap sample is chosen to be equal to the number of training samples in the original training dataset, which usually provides a good bias-variance tradeoff [4]. Therefore we are interested in tuning the number of trees that form the forest (`n_estimators`) and the maximum number of features to consider in each split (`max_features`). Again we perform a Cross-Validated Grid Search to tune these parameters, and results found are reported in Table 10 and Figures 19.

Preprocessing	Parameters	F1-Score
None (Raw data)	`max_features`:None, `n_estimators`: 200	$0,4972$
PCA	`max_features`:None, `n_estimators`: 200	$0,4609$
PCA + SMOTE	`max_features`:sqrt, `n_estimators`: 200	$0,4919$
PCA + KMeans SMOTE	`max_features`:None, `n_estimators`: 50	$0,4607$
PCA + ClusterCentroids	`max_features`:sqrt, `n_estimators`: 100	$0,4153$

Table 10: Results obtained with Random Forest model and different preprocessing strategies

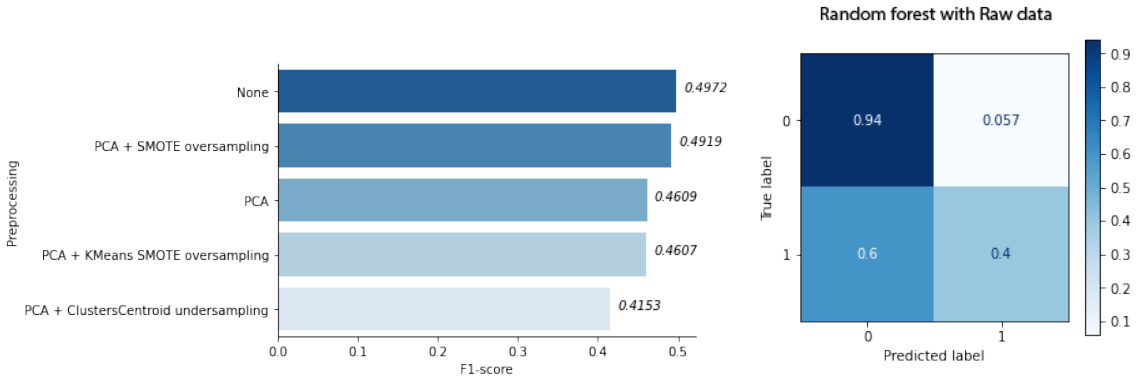

Figure 19: Results obtained with Random Forest model and different preprocessing strategies on the left, the confusion matrix of the best configuration found on the right

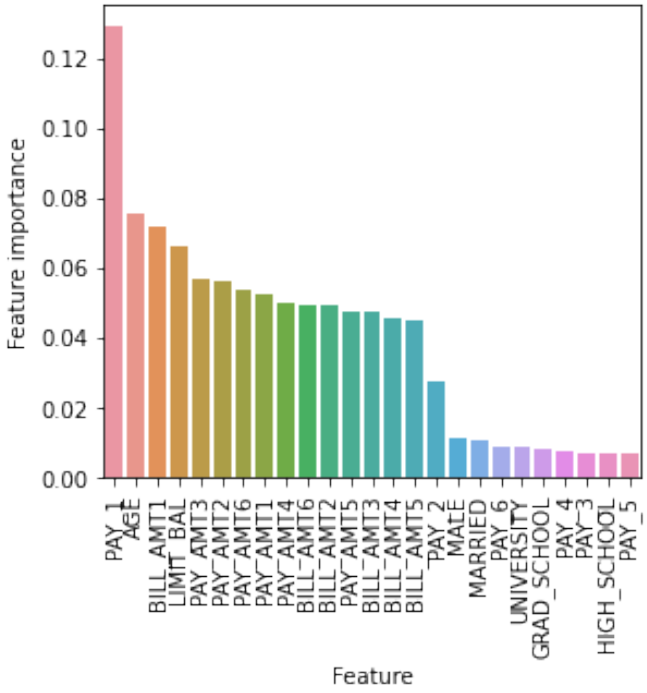

Figure 20: Feature importance obtained training Random Forest on the raw training dataset

6 Conclusion

We tested different machine learning models as well as different preprocessing technique, with the aim of obtaining a good model for predicting the repayment default of credit card users.

We focused on maximizing the F1-score of the models and the best found is SVM with PCA and SMOTE oversampling applied in the preprocessing phase, which gave a F1-score of 0.5247 and recall of 0.57 which is relatively good considering the high imbalance of the dataset.

We also could understand which features in a human analysis of the problem may be investigated first through the feature importance plot of the Random Forest model.

However, our result are quite low and other methods may be explored trying to get better performances. It would be interesting to implement some GradientBoost based models such as Gradient Boosting Classifier or SGD Classifer, and also some outliers management approaches such as Local Outliers Factor or Isolation Forest could help to improve our results.

References

[1] I.-C. Yeh and C. hui Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2473–2480, Mar. 2009. [Online]. Available: https://doi.org/10.1016/j.eswa.2007.12.020

[2] D. Dua and C. Graff, "Uci machine learning repository: default of credit card clients data set," 2019. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

[3] T. M. Alam, K. Shaukat, I. A. Hameed, S. Luo, M. U. Sarwar, S. Shabbir, J. Li, and M. Khushi, "An investigation of credit card default prediction in the imbalanced datasets," *IEEE Access*, vol. 8, pp. 201 173–201 198, 2020. [Online]. Available: https://doi.org/10.1109/access.2020.3033784

[4] S. Raschka, *Python Machine Learning*. Packt Publishing, 2015.

[5] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning*. Cambridge University Press, 2009. [Online]. Available: https://doi.org/10.1017/cbo9781107298019

[6] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, apr 2009. [Online]. Available: https://doi.org/10.1016/j.eswa.2008.06.108

[7] G. Lemaitre, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," Sep 2016. [Online]. Available: https://arxiv.org/abs/1609.06570

[8] Q. Zou, Z. Wang, X. Guan, B. Liu, Y. Wu, and Z. Lin, "An approach for identifying cytokines based on a novel ensemble classifier," *BioMed Research International*, vol. 2013, pp. 1–11, 2013. [Online]. Available: https://doi.org/10.1155/2013/686090

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," 2011. [Online]. Available: https://arxiv.org/abs/1106.1813

[10] F. Last, G. Douzas, and F. Bacao, "Oversampling for imbalanced learning based on k-means and smote," Nov 2017. [Online]. Available: https://arxiv.org/abs/1711.00837

[11] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, p. 1137–1143.

[12] P.-N. Tan, *Introduction to Data Mining*, 2nd ed. Pearson Education Limited, 2019.

[13] e. a. M. Mohri, *Foundations of machine learning*. MIT press, 2018.

[14] A. Agresti, *An Introduction to Categorical Data Analysis*, 3rd ed., ser. Wiley Series in Probability and Statistics. John Wiley and Sons, 2019. [Online]. Available: http://gen.lib.rus.ec/book/index.php?md5=ec387de4af731cc168b9f0506700f5cc

[15] D. P. K. Z. I. B. T. T. R. Vaisman, *Data Science and Machine Learning: Mathematical and Statistical Methods*, ser. Chapman and Hall/CRC Machine Leraning and Pattern Recognition. CRC Press, 2020.

[16] T. H. R. T. Gareth James, Daniela Witten, *An Introduction to Statistical Learning with Applications in R*, ser. Springer Texts in Statistics, Vol. 103. Springer, 2013.