Default of credit card clients

**Contents**

1. [Introduction](#_bookmark0) 2
2. [Exploratory Data Analysis](#_bookmark1) 3
   1. [Dataset Description](#_bookmark2) 3
   2. [Data distribution](#_bookmark3) 4
   3. [Data and cleaning](#_bookmark4) 4
   4. [Categorical features](#_bookmark5) 5
   5. [Continuous features](#_bookmark6) 6
   6. [Check for Normality distribution - QQ-plot](#_bookmark7) 7
   7. [Correlation](#_bookmark8) 7
3. [Data preprocessing](#_bookmark9) 10
   1. [Handling Categorical Features](#_bookmark10) 10
   2. [Outliers and Anomaly detection](#_bookmark12) 11
   3. [Features Scaling](#_bookmark13) 11
   4. [Dimensionality reduction](#_bookmark14) 12
   5. [Class imbalance - Resampling](#_bookmark16) 14
      1. [SMOTE - Synthetic Minority Oversampling Technique](#_bookmark18) 16
4. [Model evaluation](#_bookmark20) 18
   1. [Performance evaluation metrics](#_bookmark22) 18
5. [Classification models](#_bookmark23) 21
   1. [Logistic Regression](#_bookmark24) 21
   2. [Decision Tree and Random Forest](#_bookmark25) 22
      1. [Decision Tree](#_bookmark26) 22
      2. [Random Forest](#_bookmark27) 23
   3. [Support Vector Machine](#_bookmark28) 24
      1. [Hard margin SVM](#_bookmark29) 24
      2. [Soft margin SVM](#_bookmark30) 24
      3. [Kernel SVM - kernal trick](#_bookmark31) 25
6. [Results](#_bookmark32) 27
   1. [Logistic regression](#_bookmark33) 27
   2. [Decision Tree](#_bookmark34) 27
   3. [Random forest](#_bookmark35) 28
   4. [Support vector machine](#_bookmark36) 28
   5. Deep learning
   6. [Overall overview](#_bookmark37) 29
7. [Conclusion](#_bookmark38) 30

**Abstract**

The aim of this study is to exploit some supervised machine learning algorithms to identify the main factors that determine the probability of a credit card default, underlining the mathematical aspects and the methods used. Credit card defaults may occur when you have become severely broke on credit card payments. In order to increase market share, Taiwan’s banks have issued excess cash and credit cards to unskilled applicants. At the same time, most cardholders, regardless of their repayment capability, were using their credit card excessively for consumption and have accumulated huge credits and debts.

The goal is to build an automated model to both identify central drivers and predict credit card default based on customer information and historical transactions. Next, the general concepts of the supervised machine learning paradigm are presented, along with a detailed explanation of all the techniques and algorithms used to build the models. In particular, Logistic Regression, Random Forest and Support Vector Machines algorithms have been applied.

# Introduction

Since 1990, the Taiwanese government has allowed the formation of new banks. In order to increase market share, these banks have issued excess cash and credit cards to unskilled applicants. At the same time, most cardholders, regardless of their repayment ability, have abused their credit card for consumption and piled up heavy credit card debt and cash. Default occurs when a credit card holder is unable to comply with the legal obligation to repay. The crisis has caused a severe blow to confidence in consumer credit and has been a major challenge for both banks and cardholders .

In a well-developed financial system, crisis management is downstream and risk prediction is up- stream. The primary purpose of risk forecasting is to use financial information, such as corporate financial statements, customer transaction and refund records, etc., to predict individual customer business performance or credit risk and reduce damage and uncertainty.

In this project, the aim is to reliably predict who is at risk of defaulting. In this case, the bank may be able to prevent the loss by providing the customer with alternative options (such as forbearance or debt consolidation, etc.). Then, we build an automated model based on customer information and historical transactions that can identify key factors and predict credit card default.

# Exploratory Data Analysis

## Dataset Description

The Default of Credit Card Clients dataset contains 30 000 instances of credit card status collected in Taiwan from April 2005 to September 2005. The dataset employs the binary variable default payment next month as response variable. It indicates if the credit card holders will be defaulters next month (Yes = 1*,* No = 0). In particular, for each record (namely, each client) we have demographic information, credit data, history of payments and bill statements. To be more precise, the following is the complete list of all the 23 predictors.

* 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:

1. PAY 0: Repayment status in September, 2005.
2. PAY 2: Repayment status in August, 2005.
3. PAY 3: Repayment status in July, 2005.
4. PAY 4: Repayment status in June, 2005.
5. PAY 5: Repayment status in May, 2005.
6. 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 com- panies issue to credit card holders in a specific month:

1. BILL AMT1: Amount of bill statement in September, 2005.
2. BILL AMT2: Amount of bill statement in August, 2005.
3. BILL AMT3: Amount of bill statement in July, 2005.
4. BILL AMT4: Amount of bill statement in June, 2005.
5. BILL AMT5: Amount of bill statement in May, 2005.
6. BILL AMT6: Amount of bill statement in April, 2005.

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

1. PAY AMT1: Amount of previous payment in September, 2005.
2. PAY AMT2: Amount of previous payment in August, 2005.
3. PAY AMT3: Amount of previous payment in July, 2005.
4. PAY AMT4: Amount of previous payment in June, 2005.
5. PAY AMT5: Amount of previous payment in May, 2005.
6. 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

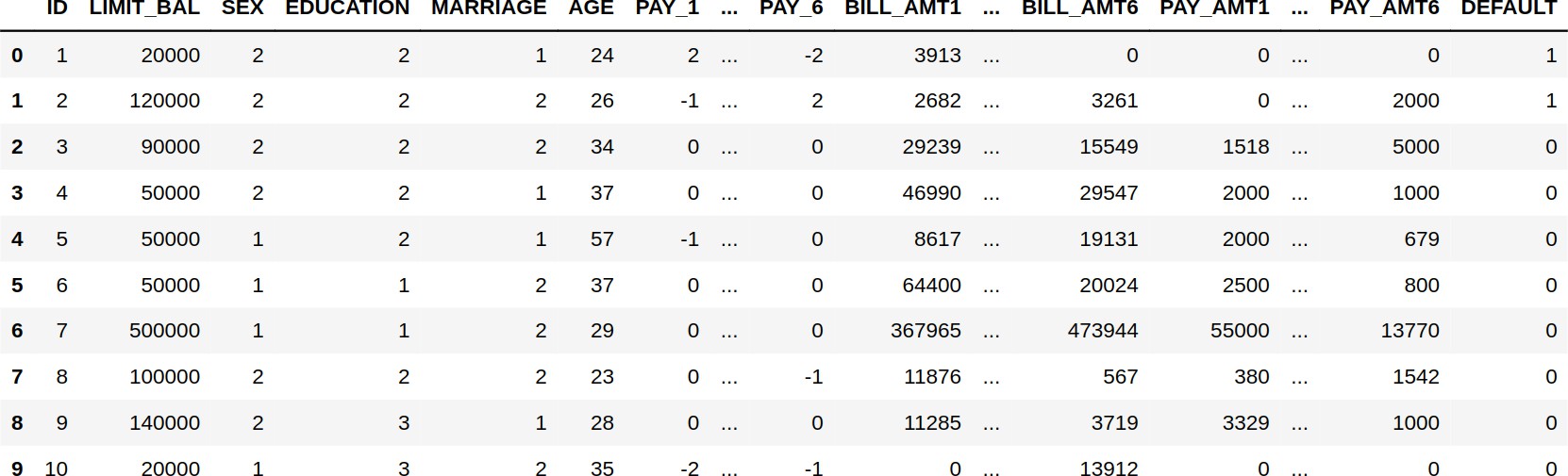
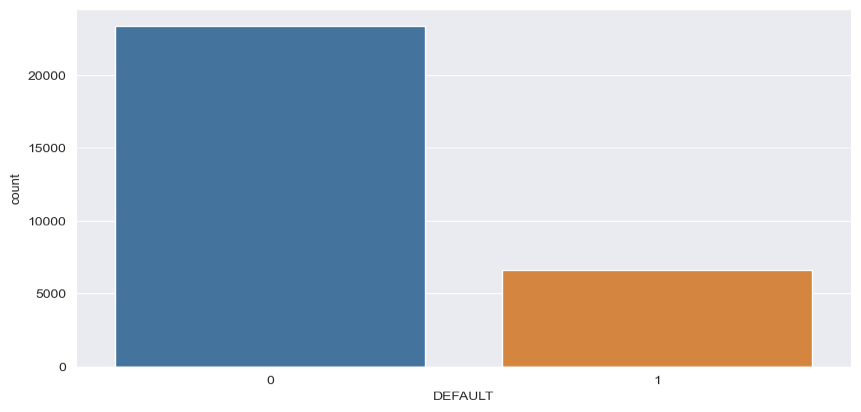


Figure 1: Original dataset from UCI machine learning repository through pandas framework

## Data distribution

From Figure 2 it is possible to see the distribution of the target variable default payment next month. It clearly shows an imbalance towards the 0 class (i.e. no default), with around 78% of the whole dataset. This imbalance problem will make classification models focusing on the majority class over- looking the minority class if not addressed.



## Data Structure and cleaning



So looking at the values present in the attributes some changes have to be done:

* Attribute marriage should present only one of those values: 1,2,3; but in the dataset some records have value 0.
* Attribute education should present only one of those values: 1,2,3,4; but in the dataset some records have values 0,5,6.
* Attributes PAY N should present only one of those values: -1,1,2,3,4,5,6,7,8,9; but in the dataset some records have value -2 and 0.

In first two cases since there is an attribute which represent the Other class (respectively 3 for marriage and 4 for education), all the attributes not-known are mapped in that category.

# Column Non-Null Count Dtype

1. LIMIT BAL 30000 non-null int64
2. SEX 30000 non-null int64
3. EDUCATION 30000 non-null int64
4. MARRIAGE 30000 non-null int64
5. AGE 30000 non-null int64
6. PAY 1 30000 non-null int64
7. PAY 2 30000 non-null int64
8. PAY 3 30000 non-null int64
9. PAY 4 30000 non-null int64
10. PAY 5 30000 non-null int64
11. PAY 6 30000 non-null int64
12. BILL AMT1 30000 non-null int64
13. BILL AMT2 30000 non-null int64
14. BILL AMT3 30000 non-null int64
15. BILL AMT4 30000 non-null int64
16. BILL AMT5 30000 non-null int64
17. BILL AMT6 30000 non-null int64
18. PAY AMT1 30000 non-null int64
19. PAY AMT2 30000 non-null int64
20. PAY AMT3 30000 non-null int64
21. PAY AMT4 30000 non-null int64
22. PAY AMT5 30000 non-null int64
23. PAY AMT6 30000 non-null int64
24. DEFAULT 30000 non-null int64

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

all the values -2 and -1 are mapped in 0. In this way PAY N will indicate for how many months the payment was delayed.

## Categorical features

Regarding the categorical features SEX, EDUCATION and MARRIAGE showed in Figure 3 and counts in Table 2, it is possible to notice that there are much more females than males in the dataset, and, in particular, males have a slightly higher chance to default than females (0.24% vs 0.21%). But in general, whether it is male or female, the proportion of DEFAULTERS and NON-DEFAULTERS per each value, is in line with the categories of the other two features.

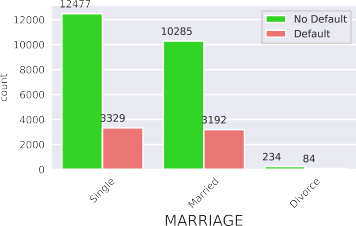
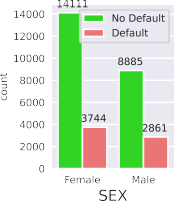


Figure 3: Countplot of SEX, EDUCATION and MARRIAGE grouped by DEFAULT class

We still have to inspect the payment status feature PAY N, boxplots below shown in Figure 4 is very useful. It can be seen that clients who delay payment by one month or less have fewer credit card defaults. I.e. a greater discriminatory power is held the repayment status in September, PAY 1, than the repayment status in the other months.

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

Table 2: Value counts for SEX, EDUCATION and MARRIAGE feature

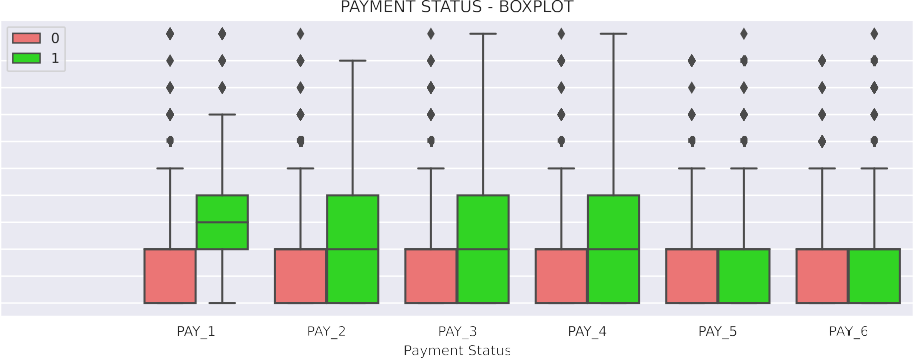


Figure 4: Boxplots of PAYN grouped by DEFAULT class

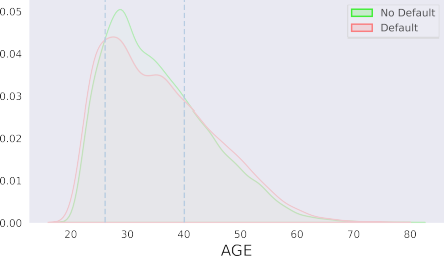
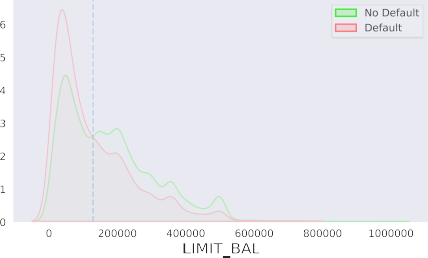
## Continuous features

In statistics, the Kernel Density Estimation (KDE) is a fairly well known technique for estimating the probability density function in a non-parametric way (i.e. it does not assume any underlying distribution). So, for the following continuous feature, we explored their KDE plots. Observing Figure 5 it is possible to notice that most of the default come from credits with a lower LIMITBAL (i.e. credit amount), in particular they are observed in a range among a few thousands Taiwanese dollars to around $140000. The customers above this threshold are more likely to repay their debts.

For the feature AGE, a similar visual analysis is performed. The probability of non-default of age between approximately 25 and 42 is higher, which indicates that consumers are more capable of repaying credit card loans in this age group. An assumption could be that their work and family tend to be stable without too much pressure.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Figure 5: KDE plots of LIMITBAL and AGE grouped by DEFAULT class

## Check for Normality distribution - QQ-plot

Methods we will use later assume that the data should have a known and specific distribution, i.e. Normal distribution. Applying such methods on different data distribution, our final results may be misleading or plain wrong. A way to check whether our data are Normally distributed, we used a graphical method called Quantile-Quantile (QQ) plot that give us a qualitative evaluation. In a QQ-plot, the quantiles of the independent variable are plotted against the expected quantiles of the normal distribution. If the variable is normally distributed, the dots in the QQ-plot should fall along a 45 degree diagonal. The plots show that there is no evidence that numerical features are normally distributed.

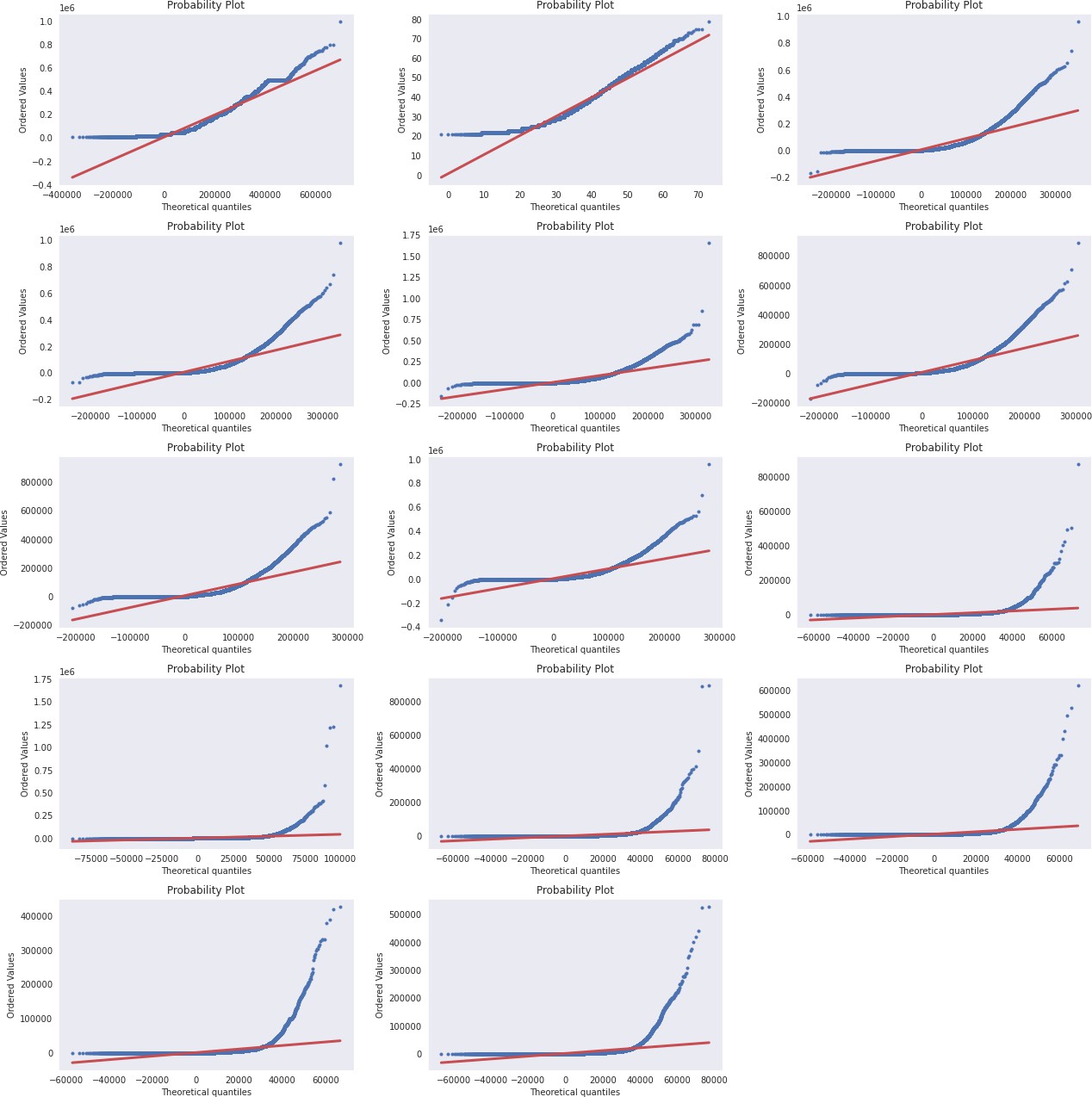


Figure 6: QQ-plot for feature LIMIT BAL, BILL ATM,AGE and PAY ATM

## Correlation

Correlation is a statistical term describing the degree to which two random variables move in coor- dination with one-another. Intuitively, if they are moving in the same direction, then those variables are defined with a positive correlation, or viceversa we define that with a negative correlation. The

Pearson Correlation (*ρ*) is one of the most used linear correlation measures.

*ρ*(*X, Y* ) = *Cov*(*X, Y* ) = *E*[(*X − µX* )(*Y − µY* )]

The value of Pearson’s Correlation Coefficient range is [ -1, 1 ].

• +1 means that they are strongly correlated.

* 0 means no correlation, the two random variables are statistically independent, but it is not true the opposite, because they may have a non-linear relationship.
* -1 means that there is a negative correlation (inverse proportion).

High values of this correlation coefficient with respect to the target is a synonym of data redundancy, so it could be helpful to drop those columns. In Figure 7 is given a graphical representation of the Person Correlation with a Heatmap, where each cell (*i, j*) represents the Person Correlation between the random variables *Xi* and *Xj*

From Figure 7, as we may think, we can observe an ”internal” correlation among the groups of features such as BILL ATM, PAY N. We can also notice that there is no feature with a strong relationship with the target. In fact, there are 15 features with an absolute value of the correlation below than 0.1 and none of the remaining ones have a greater correlation than 0.29.

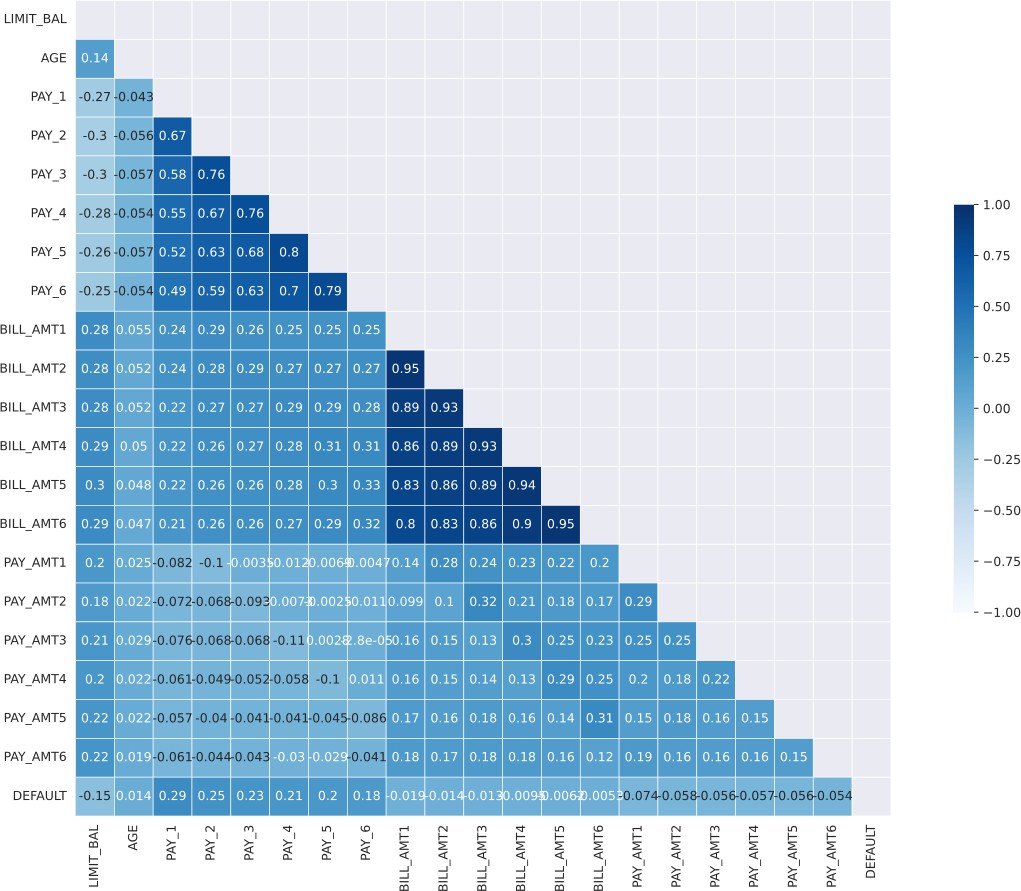


Figure 7: Heatmap correlation

# Data preprocessing

## Handling Categorical Features

The categorical features EDUCATION, SEX, and MARRIAGE are already encoded with integer numbers and could be fed to a machine learning algorithm. However, these are nominal features, for which it would be sub-optimal to assume an ordering. *One-hot encoding* allows us to remove any ordinal relationship, which would be meaningless between these categorical variables. The idea behind this approach is to create a new dummy feature for each unique value in the nominal feature column. Binary values can then be used to indicate the particular class of an example.

Although Scikit-Learn provides methods to perform one-hot encoding automatically, we decide to do the mapping of the features by hand, since there are few. In this way we mitigate the problem of *multicollinearity*, which occurs when there are highly correlated features. Thus, we create the following boolean columns and drop the old ones, EDUCATION, SEX, and MARRIAGE.

* MALE: 1 = male; 0 = female.
* MARRIED: 1 = married marital status; 0 = otherwise.
* GRAD SCHOOL: 1 = graduate school level of education; 0 = otherwise.
* UNIVERSITY: 1 = university level of education; 0 = otherwise.
* HIGH SCHOOL: 1 = high school level of education; 0 = otherwise.

Using this strategy we do not loose any information. In Table 3 and Table 4 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 3: 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 4: SEX, MARRIAGE and EDUCATION features after one-hot encoding

## Dataset Partition

A common practice, in order to evaluate the performances of a classification algorithm, is to divide the dataset into two partitions, called training and test set. The training set is used to fit the machine learning model, whereas the test set is used to evaluate the fit machine learning model. In this case, since there is a adequate number of samples, 75% of the initial dataset is used for the training procedure and the remaining 25% for testing, while preserving the initial data distribution (attribute *stratify* ).

Another data splitting is necessary, in order to tune the hyperparameters. In particular, two new partitions will be considered: train set and validation set. It has been used a Stratified K-Fold Cross Validation (with k = 5). Once obtained the best hyperparameters, train again the model with both train and evaluation set and finally evaluate it on the test set.

* Training set:

**–** shape: (22200,25)

* defaulters: 4955
* non-defaulters: 17245
* class proportion: 22,30%
* Test set:

**–** shape: (7400,25)

* defaulters: 1650
* non-defaulters: 5750
* class proportion: 22,30%

## Outliers and Anomaly detection

An outlier or an anomaly detection is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism. Some of the most common causes of outliers are:

* an entity may seem different because it belongs to another class;
* there is always the probability (even if it is lower) that we record real values far from the regular patterns;
* some technical or human errors occur.

The presence of a significant amount of outliers in some cases could drastically affect the perfor- mances. Therefore, it is a common practice to train the model with and without them in order to catch their contribution. There are several different techniques for removing outliers. It is possible to find them thanks to some graphical representation of the data (e.g. Boxplot).

Figure 8 shows the boxplots for BILL ATM and PAY ATM variables, and are each followed by a de- scription of how outliers were determined for the variables. The BILL ATM variables were depicted the amount of bill statement during the respective months. Since these can actually be considered as re- peated observations of the same variable (per cardholder), then for the purpose of outlier identification, they were analyzed using the same boxplot. In addition, and to minimize the loss of information due to elimination of outliers, a cut-off value was set for these variables based on the general trend observed in the six variables. Consequently, it was assumed that for all BILL ATM variables, any amount exceeding 1,000,000 and any amount going below *min*(BILL ATM4), which was the value 170,000, was considered to be an outlier.

For the PAY ATM variables, the same approach was takes for outlier detection as the one taken for BILL ATM variables. The general trend of the six variables set the upper cut-off point at *max*(PAY ATM4) which was the value 621,000. Any observation exceeding this amount was considered an outlier. On the lower side, no observation had a value lower than 0, hence there were no outliers based on this (taking 0 as the minimum). After the exclusion of outlier values was done, a sample size of 29,993 remained (7 observations were dropped), which was the data used for training and validating the model .

We do not eliminate any sample as an outlier because the literature on the dataset does not provide

information on this , and we lack knowledge about the domain.

## 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 forests are two of the very few machine learning algorithms where there is no need to worry about feature scaling, as they are scale invariant [[6](#_bookmark44)].

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

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

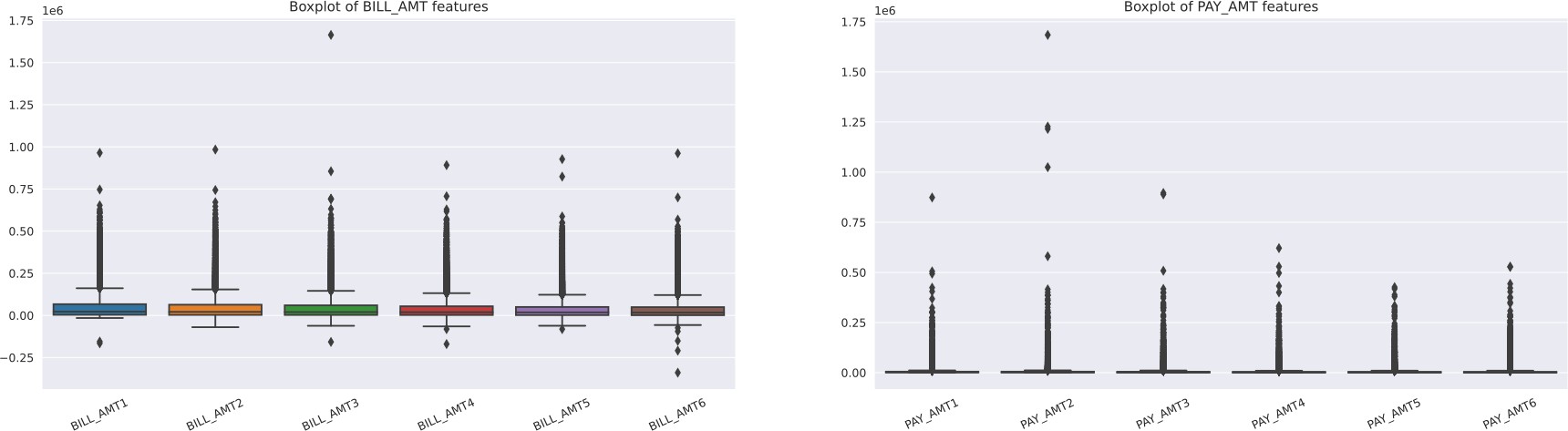


Figure 8: Box-plot for PAY ATM and BILL ATM Variables

where *X* is a particular example, *XMin* is the smallest value in a feature column, and *XMax* is the largest value.

* *Standardization or Z-Score Normalization*: center the feature columns at mean *µ* = 0, with standard deviation *σ* = 1, so that the feature columns have the same parameters as a standard normal distribution, which makes it easier to learn the weights:

*X − µ*

*Z* = (3)

*σ*

While normalization suppress the effect of outliers, standardization maintains information about them. Since the technique we will apply shortly is sensitive to the presence of outliers, we decide to continue our study with the normalized data.

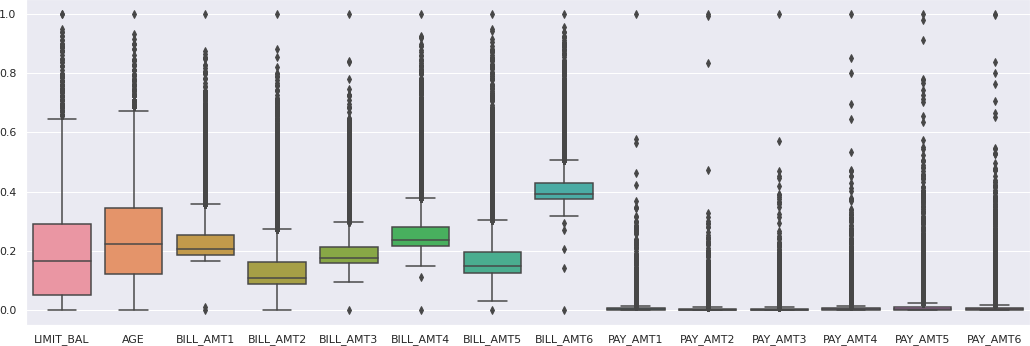


Figure 9: Box-plot scaled features

## Class imbalance - Resampling

Class imbalance is quite a 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. We showed in Section

2.2 and reported in Figure 2 that the dataset we are dealing with is unbalanced, as the non-defaulter examples are over-represented.

In our case, we could achieve almost 80% accuracy by just predicting the majority class (non- defaulters) for all examples, without the help of a supervised machine learning algorithm. Thus, when we will fit classifiers on our datasets, it would make sense to focus on other metrics than accuracy when comparing different models.

Aside from evaluating machine learning models, class imbalance influences a learning algorithm to model fitting itself. Since machine learning algorithms typically optimize cost function that is

computed as a sum over the training examples that it sees during fitting. The decision rule is likely going to be biased toward the majority class.

The option of collecting more data is excluded a priori. There are other option to tackle this problem:

* At training time assign larger penalty to wrong predictions on the minority class.
* Upsampling the minority class
* Downsampling the majority class

Unfortunately, there is not a generalized best solution or technique that works best across different problem domains. Thus, in practice, it is recommended to try out different strategies on a given problem, evaluate the results, and choose the technique that seems most appropriate.

We decide to exclude the na¨ıve methods of oversampling (undersampling), since they randomly duplicate (delete) data from the minority (majority) class, until the desired level is obtained. On the one hand, random undersampling does not allow controlling which information is discarded, on the other hand random oversampling causes overfitting because the model is trained on many identical data. We decide to explore two technique which mitigate exposed issues that are the Cluster Centroid (Undersampling) method and the Synthetic Minority Oversampling Technique (SMOTE).

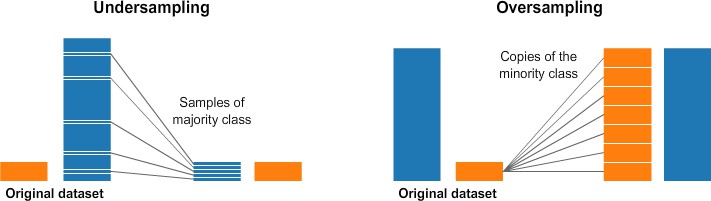


Figure 12: Resampling techniques

### Cluster Centroid Undersampling

Minority class samples in our dataset are enough to allow us to perform undersampling. However, one major problem of using undersampling is that important information may be lost from the majority class, which can cause overly general rules. This cannot be afforded to develop the credit card de- fault prediction model, especially for default samples. Hence, to overcome this problem, the Cluster Centroids method has been introduced in [[8](#_bookmark46)].

The idea of the Cluster Centroids method is to replace clusters of majority samples with the respective cluster centroids. A K-means algorithm is fitted to the data, and the number of clusters is set equal to the number of samples of the minority class. Then, the majority of samples from the clusters are entirely substituted by the sets of cluster centroids from K-means [[9](#_bookmark47)]. Cluster Centroids contain the most representative variations of the majority class in which features values would be visualized at the center.

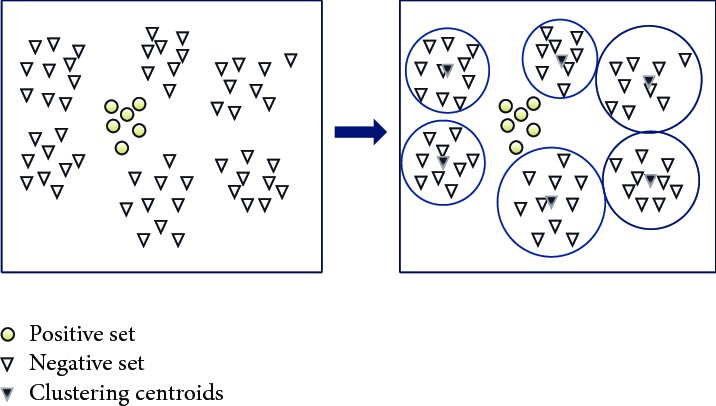


Figure 13: Cluster centroids undersampling [[10](#_bookmark48)]

### SMOTE - Synthetic Minority Oversampling Technique

The Synthetic Minority Oversampling Technique (SMOTE) was proposed in [[11](#_bookmark49)] to avoid the risk of overfitting faced by random oversampling. Instead of simply replicating existing observations, the technique generates artificial samples. As shown in Figure 14, this is achieved by linearly interpolating a randomly selected minority observation and one of its neighboring minority observations. More precisely, SMOTE executes three steps to generate a synthetic sample:

* + - * Firstly, it chooses a random minority observation *⃗a*
      * Among the nearest minority of *⃗a* neighbors select the instance *⃗b*
      * Create a new sample *⃗x* by linearly interpolating *⃗a* and *⃗b* (with *ω* random)

*⃗x* = *⃗a* + *ω ×* (*⃗b − ⃗a*)*, ω ∈* [0*,* 1] (10)

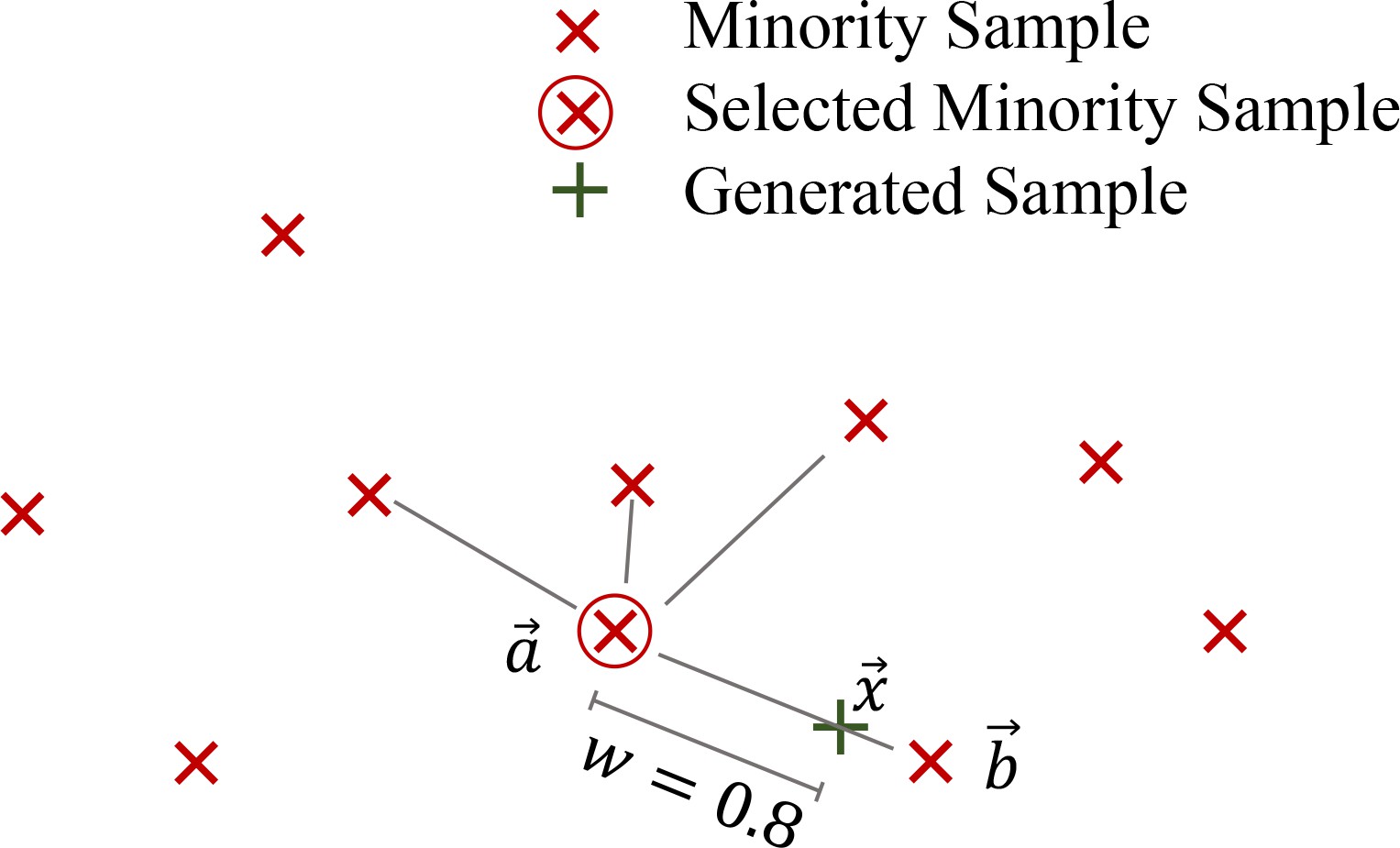


Figure 14: SMOTE linearly interpolates a randomly selected minority sample and one of its k = 4 nearest neighbors [[12](#_bookmark50)]

Unfortunately, SMOTE, as other random oversampling techniques, suffers from weakness due to within-class imbalance and noise. In fact, SMOTE choose a random sample from the minority class to start, but if distribution of the minority class is not uniformed 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 work in this sense, as shown in Figure 15.

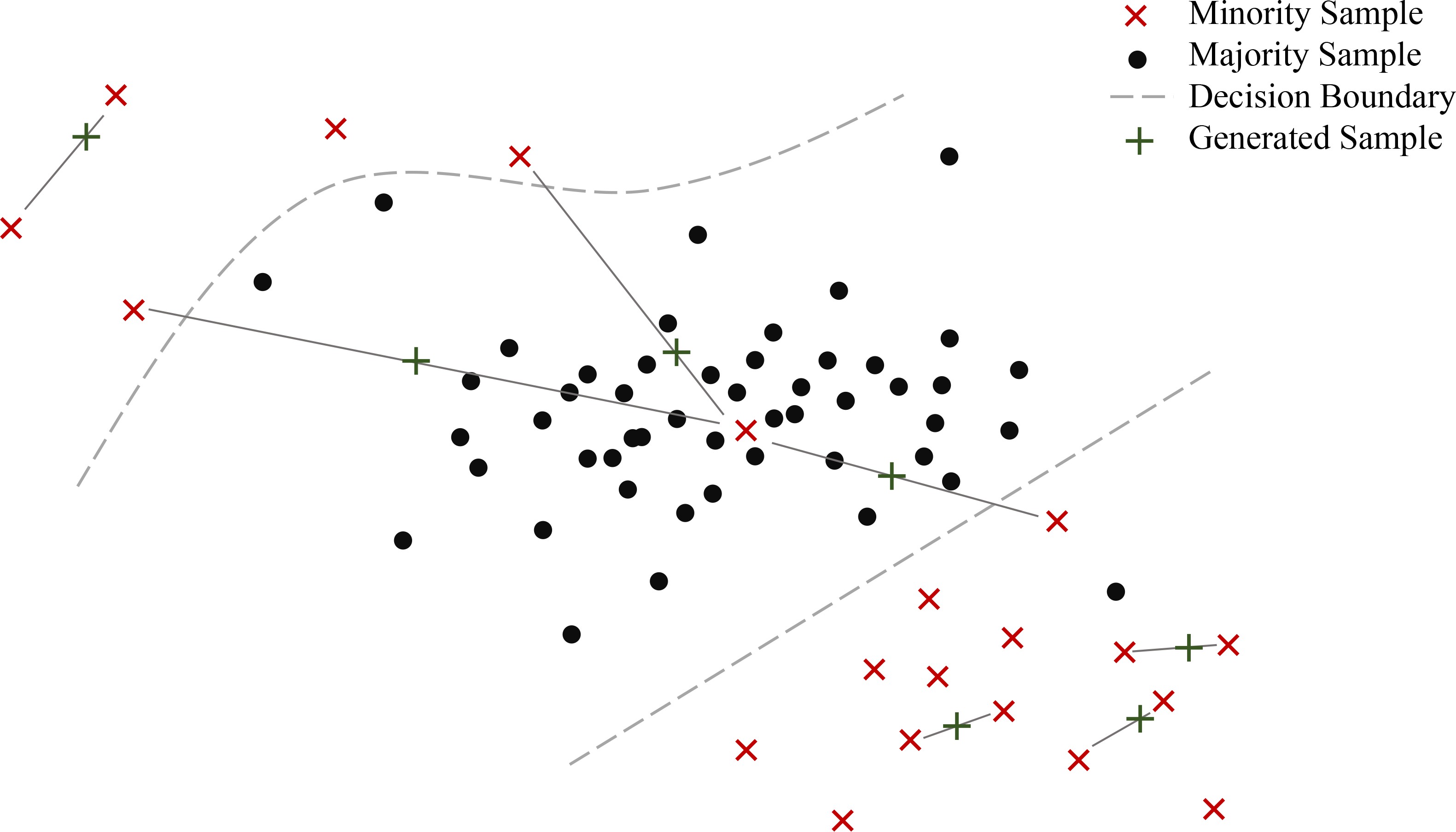
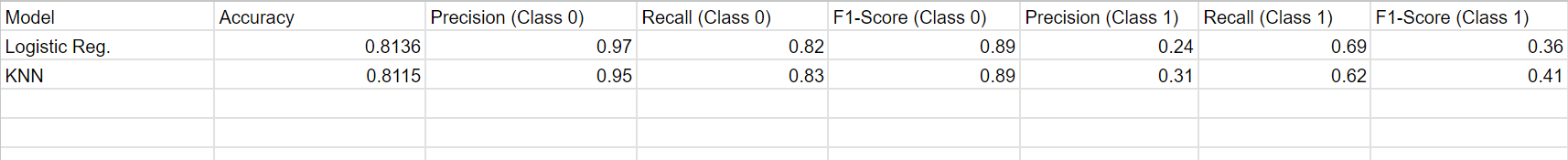


Figure 15: SMOTE behavior in presence of noise and with-in class imbalance [[12](#_bookmark50)]

# Model evaluation

One of the main steps in building a machine learning model is to estimate its performance on data that the model has not seen before. For this reason, in 3.2 initial dataset is splitted into separate training and test datasets. The former will be used for model training, and the latter to estimate its generalization performance. This approach is commonly known as holdout method.

However, in typical machine learning applications, it is interested in tuning and comparing different parameter settings to further improve the performance for making predictions on unseen data. But, if we reuse the same test dataset over and over again during this process, it will become part of our training data and thus the model will be more likely to overfit. A validation set could be held out of the training set, to evaluate on it the performance of the model. However, this is not recommended because performance estimation may be sensitive to how we partition the training set.



**Model Performance Metrics:**

Logistic Regression achieved an accuracy of 81.37%, while K-Nearest Neighbors achieved a slightly lower accuracy of 81.15% on the test dataset.

**Confusion Matrix Analysis:**

**Logistic Regression**:

* TN: 4568, FN: 142
* FP: 976, TP: 314
* **KNN:**
* TN: 4469, FN: 241
* FP: 890, TP: 400
* Precision, Recall, and F1-score:

**Logistic Regression:**

* Precision for class 0: 97%, for class 1: 24%
* Recall for class 0: 82%, for class 1: 69%
* F1-score for class 0: 89%, for class 1: 36%

**KNN:**

* Precision for class 0: 95%, for class 1: 31%
* Recall for class 0: 83%, for class 1: 62%
* F1-score for class 0: 89%, for class 1: 41%

**Model Evaluation Summary:**

* Both models exhibit strengths and weaknesses in predicting defaults.
* Logistic Regression shows higher precision but lower recall for defaults compared to KNN.
* KNN achieves a more balanced performance in terms of precision and recall for defaults.

**Considerations for Model Selection:**

* Depending on the business context, stakeholders may prioritize different metrics such as precision, recall, or overall accuracy.
* Logistic Regression might be preferred if minimizing false positives (precision) is crucial.

KNN could be favored if achieving a balance between precision and recall for default predictions is more important

## Performance evaluation metrics

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 5.

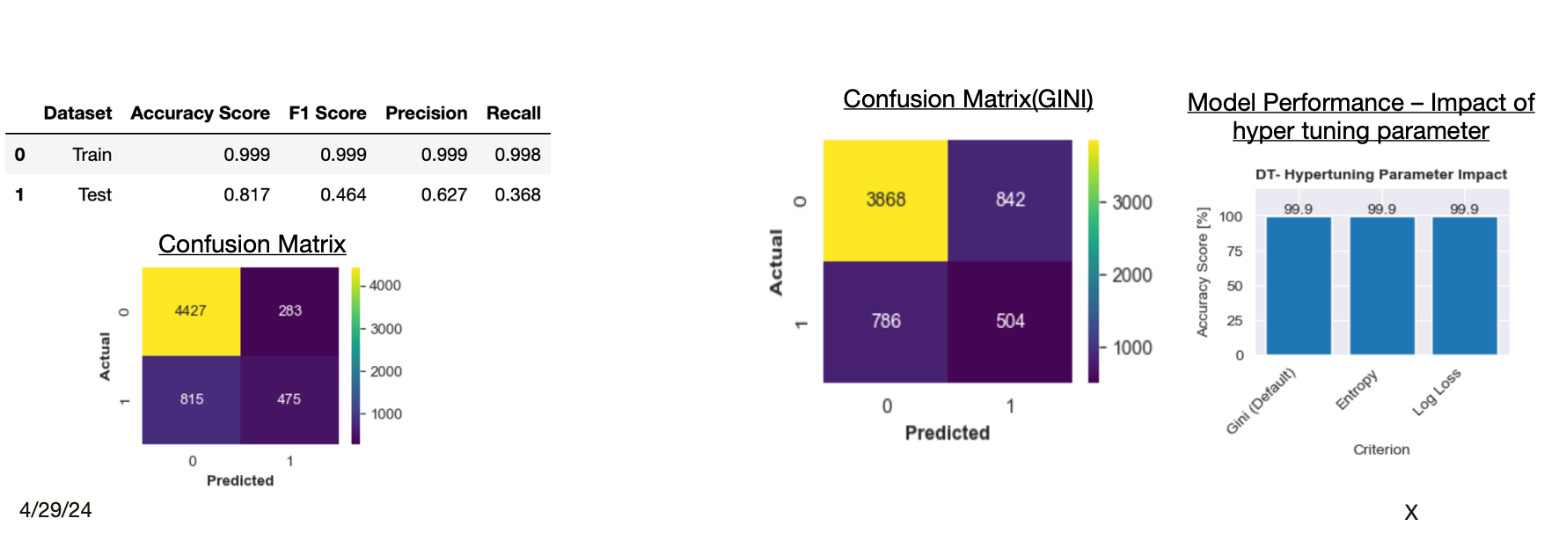


Table 5: Binary classification confusion matrix

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

* *True positive (TP)*, which corresponds to the number of positive examples correctly predicted by the classification model.
* *False negative (FN)*, which corresponds to the number of positive examples wrongly predicted as negative by the classification model.
* *False positive (FP)*, which corresponds to the number of negative examples wrongly predicted as positive by the classification model.
* *True negative (TN)*, which corresponds to the number of negative examples correctly predicted by the classification model.

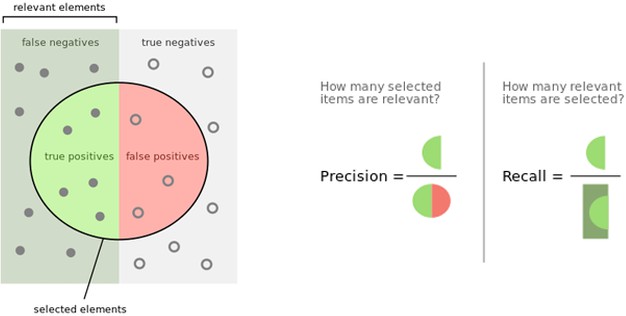


Figure 18: Precision and Recall

Different metrics can be used depending on the task, the data imbalance and other factors. While dealing with classification tasks, these are some of the most used ones:

* *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.
* *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.
* *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.

# Classification models

In this section, we present different supervised learning algorithms with their mathematical details, and we use them on our dataset to build a classification model that is able to predict credit card defaults in the next month. In particular we will dive into Support Vector Machine, Logistic Regression and some tree based methods, all following the Empirical Risk Minimization paradigm.

## 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 dis- tribution for it, treating the *m observations* on *Y* , denoted: (*y*1*, ..., ym*), as independent;
* the linear predictor that specifies the explanatory variables through a prediction equation that has linear form, such as:

*α* + *β*1*x*1 + *...* + *βpxp* = **x***⊤β*; (15)

* the link function which specifies a function *g* that relates *E*[*Y* ] to the linear predictor such as [[15](#_bookmark53)]:

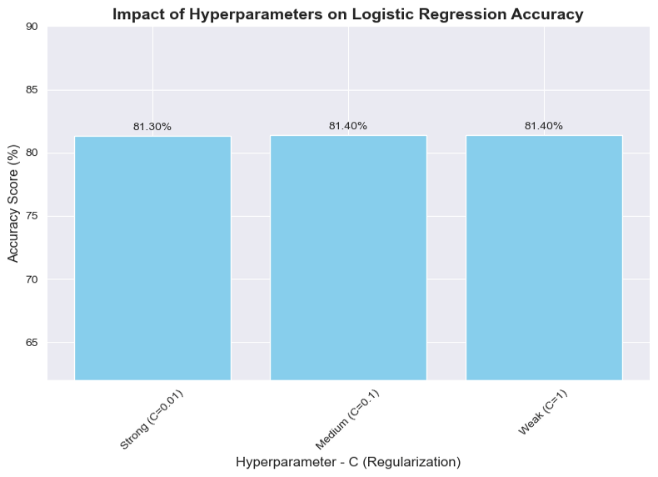
*g*(*E*[*Y* ]) = *α* + *β*1*x*1 + *...* + *βpxp* (16)

Hence, in a GLM the expected response for a given feature vector **x** = [*x*1*, ..., xn*] is of the form:

*E*[*Y |X* = **x**] = *h*(**x***⊤β*) (17)

with *h*, called *activation function*, being the inverse of the link function *g* [[16](#_bookmark54)].

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*:



indeed, with *x⊤β* large we will have a high probability for *Y* to be 1, and for small *x⊤β* we will

*i* *i*

have a high probability for *Y* to be 0. The logistic function will always produce an S-shaped curve

between 0 and 1, as shown in Figure 19.

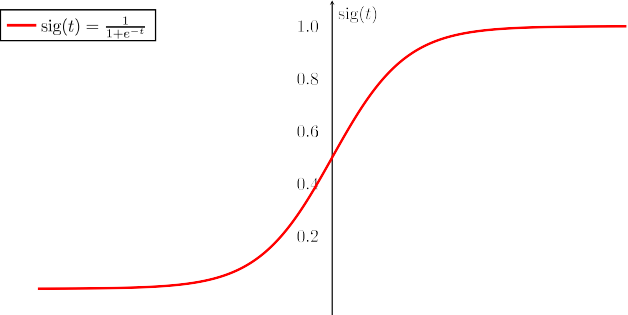


Figure 19: Sigmoid activation function

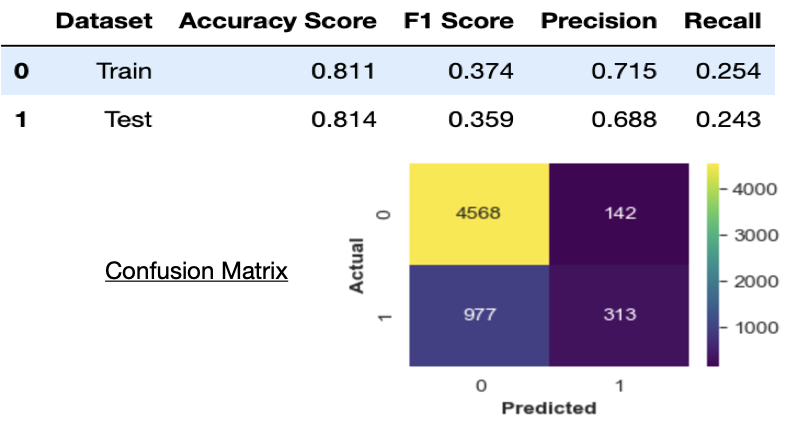
To estimate the coefficients vector *β* through the available training data we use the Maximum Likelihood method, which finds *β*ˆ that is the maximum likelihood estimate of *β*, this is formalized in such a way:

*L*(*β*) = Y[*h*(**x***⊤β*)]*yi* [1 *− h*(**x***⊤β*)]1*−yi* (19)

*m*

*i* *i*

*i*=1



where *L*(*β*) 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:

*—* 1 log *L*(*β*) = *−* 1 Σ[*y* log *h*(**x***⊤β*) + (1 *− y* ) log 1 *− h*(**x***⊤β*)] (20)

*m*

*m m i i i* *i i*=1

We can interpret this function as the *binary cross-entropy* training loss associated with comparing a true conditional probability density function (pdf) with an approximated pdf. In this study, 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* (*β, λ*) = *λ* Σ *β*2

*m*

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

A screenshot of a score

Description automatically generated

A yellow and purple squares with numbers

Description automatically generated

A graph showing a number of blue rectangular objects

Description automatically generated

Varying the hyperparameter C (regularization strength) between 0.01, 0.1, and 1 resulted in a stable accuracy score around 81.4%, indicating robust model performance

## Decision Tree and Random Forest

Tree-based methods provide a simple, intuitive, and powerful mechanism for both regression and classification. The main idea is to stratify a (potentially complicated) feature space into smaller regions and fit a simple prediction function to each region . In order to classify a given observation, we typically use the mode response value for the training observations in the region to which it belongs. 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 .

### Decision Tree

Decision tree classifiers are attractive models if we care about interpretability. They involve creating a set of *binary splits* on the predictor variables in order to create a tree that can be used to classify new observations into one of two groups .

Using the decision algorithm, we start at the tree root and split the data on the feature that results in the largest information gain (IG):

* + - * 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.

In an iterative process, we can then repeat this splitting procedure at each child node until the leaves are pure. This means that the training examples at each node all belong to the same class. Unfortunately, this process tends to produce a tree that is too large and suffers from overfitting. Thus, we typically prune the tree by setting a limit for the maximal depth of the tree.

Gini impurity ( *IG* ) and entropy ( *IH* ) are the most commonly used splitting criteria in binary decision trees. Definying as *p*(*i|t*) the proportion of the examples that belong to class *i* for a particular. 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*.

A screenshot of a graph

Description automatically generated

### 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. An illustration in Figure 20 skim the concept.

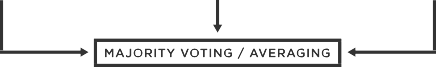
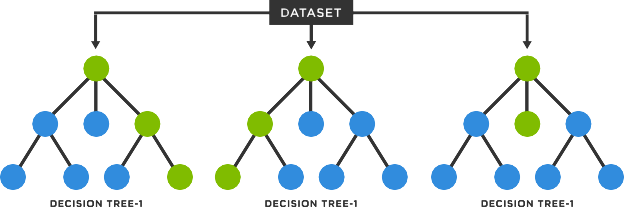


Figure 20: Feature importance obtained training Random Forest on the raw training dataset Random Forest uses the bagging also at each split for the feature selection to *decorrelate* the trees:

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 21 from our analysis on the raw data, it emerges that repayment status, age and bill amount are very important in the decision process.

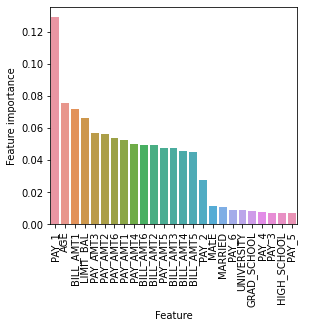


Figure 21: Feature importance obtained training Random Forest on the raw training dataset

## Support Vector Machine

Support Vector Machines (SVMs) are considered among the most effective classification algorithms in modern machine learning [[19](#_bookmark57)]. 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.

### Hard margin SVM

A hyperplane in a space *H* endowed with a dot product *⟨·, ·⟩* is described by the set:

*{***x** *∈ H| ⟨***w,x***⟩* + *b* = 0*}* (22)

where **w** *∈ H* and *b ∈ R*

Such a hyperplane naturally divides *H* into two half-spaces and hence can be used as the decision boundary of a binary classifier: *{***x** *∈ H| ⟨***w,x***⟩* + *b ≥* 0*}* and *{***x** *∈ H| ⟨***w,x***⟩* + *b ≤* 0*}*

Given a set *X* = [**x**1*, ...,* **x***m*], the margin is the distance of the closest point in *X* to the hyperplane:

Since the parametrization of the hyperplane is not unique, we set

*min | ⟨***w***,* **x***i⟩* + *b|* = 1 (24)

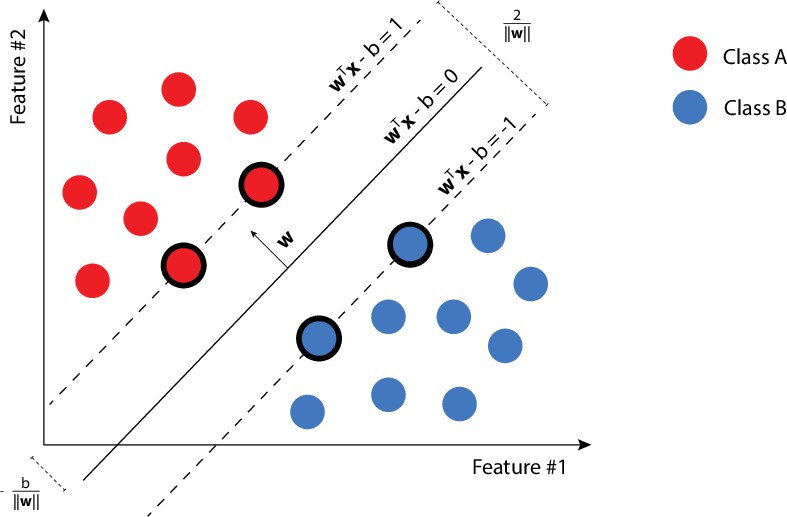


Figure 22: Hard Support Vector Machines

Let *S* = [(**x**1*, y*1)*, ...,* (**x***m, ym*)] be a training set of examples, where each **x***i ∈ H* and *yi ∈ {±*1*}*. Our aim is to find a linear decision boundary parameterized by (**w***, b*) such that *⟨***w***,* **x***i⟩* + *b ≥* 0 whenever *yi* = +1 and *⟨***w***,* **x***i⟩* + *b <* 0 whenever *yi* = *−*1. The SVM solution is the separating hyperplane with the maximum geometric margin, as it is the safest choice. The problem of maximizing the margin can be written as:

s.t. *yi*(*⟨***w***,* **x***i⟩* + *b*) *≥* 1 *∀i*

### Soft margin 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 *ξ*1*, ..., ξm* which measure how much the constraint *yi*(*⟨***w***,* **x***i⟩* + *b*) *≥* 1 is being violated. Now the problem is to jointly minimize the norm of **w** (the margin) and the average of all *ξi* (the average violations to the constraint):

s.t. *yi*(*⟨***w***,* **x***i⟩* + *b*) *≥* 1 *− ξi*

*ξi ≥* 0*, ∀i*

where *C* is a penalty parameter, typically determined via k-fold cross validation. A better looking at the problem reveals that [[20](#_bookmark58)]:

* + *ξi* = 0 whenever *yi*(*⟨***w***,* **x***i⟩* + *b*) *≥* 1
  + *ξi* = 1 *− yi*(*⟨***w***,* **x***i⟩* + *b*) whenever *yi*(*⟨***w***,* **x***i⟩* + *b*) *≤* 1

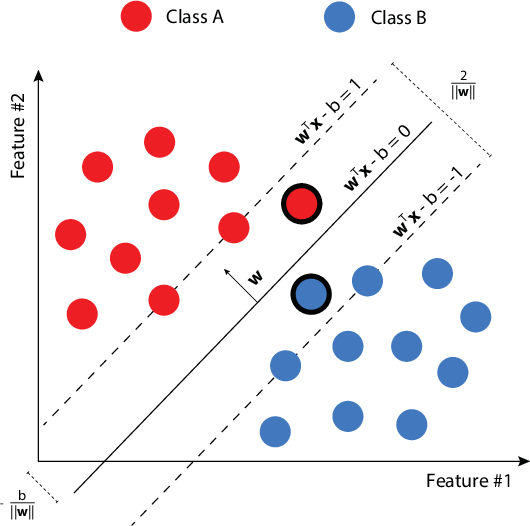


Figure 23: Soft Support Vector Machines Let the hinge loss in the context of half spaces learning be:

*lhinge*(**w***,* (*x, y*) = *max{*0*,* 1 *− y ⟨***w***, x⟩}* (27) we have that *ξi* = *lhinge*((**w***, b*)*,* (*xi, yi*)) and so we can reformulate the Problem in (18) as the

unconstrained problem:

### Kernel SVM - kernal trick

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 a higher dimensional space such that it will be linearly separable.

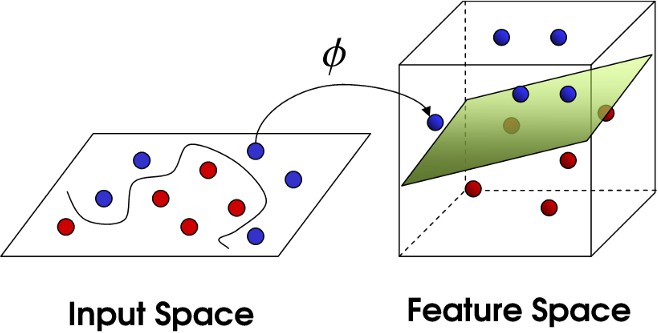


Figure 24: Kernel trick

Let us consider now the dual problem of (26) formulated as:

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 *ψ* : *X '→ F* the kernel function is defined as:

*K*(**x***,* **x***′*) = *⟨ψ*(**x***i*)*, ψ*(**x***j*)*⟩* (30)

the dual problem in (29) becomes:

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

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

2

* polynomial kernel: *K*(**x***,* **x***′*) = (*γ ⟨***x***,* **x***′⟩*)*d*
* RBF kernel: *K*(**x***,* **x***′*) = *e−*(*γ||***x***−***x***′||*2 )

# Results

In the following pages, an overview on the results is given for each classifier. Different preprocessing combinations were tested: applying dimensionality reduction techniques (PCA) or not, using or not different resampling techniques. The metric we choose to adopt is F1-score. Precision-recall curve and Confusion Matrix of the best model have been provided for each model.

## Logistic regression

The *Logistic Regression* class in *Scikit-learn* implements the parameter *C* as the inverse of the regu- larization parameter *λ*. In the Table are shown combinations of different values of C with different choices about the preprocessing phase have been explored through a Cross-Validated search.

A screenshot of a score

Description automatically generated

A yellow and purple squares with numbers

Description automatically generated

## Decision Tree

According to Figure 26, it’s easy to notice how the best performances have been obtained using original data without any manipulation technique. Moreover, the confusion matrix has been provided.

A screenshot of a graph

Description automatically generated

A yellow and purple squares with numbers

Description automatically generated

A graph of blue rectangular bars

Description automatically generated with medium confidence

## Random forest

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 [[6](#_bookmark44)]. 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 7.

According to Figure 27, it’s easy to notice how the best performances have been obtained using PCA and SMOTE oversampling technique. Moreover, the confusion matrix has been provided.

A yellow and purple squares with numbers

Description automatically generated A computer code with text

Description automatically generated

A screenshot of a white background

Description automatically generated

**Variable selection using Bidirectional method**

* Random Forest machine learning model is selected
* 95% significance level is taken into account for backward feature elimination
* 14 out of 24 features are shortlisted from Bidirectional Elimination method

A graph with blue bars

Description automatically generated A graph with blue and white lines

Description automatically generated

* Similar performance is seen for Random Forest with and without bidirectional elimination methods

**Support Vector Machine Classifier**

SVM *– A ML algorithm, separates, by finding hyperplane with maximum margin between different classes in input features*

A yellow and purple squares with numbers

Description automatically generated A yellow and purple squares with numbers

Description automatically generated

A graph of blue squares

Description automatically generated

**Deep learning Neural Network**

Deep Learning Neural Network  *inspired by human brain structure, composed by multiple layers of interactions, with more than 2 hidden layers*

**A screenshot of a graph

Description automatically generated** **A graph of a number of people

Description automatically generated with medium confidence**

**Training Dataset Performance:** The deep learning model achieved an accuracy of 77.8% on the training dataset, with a very low F1 score of 0.049, precision of 0.539, and recall of 0.026. This suggests that the model struggled to capture meaningful patterns in the training data.

**Test Dataset Performance:** On the test dataset, the model performed similarly with an accuracy of 78.3%, but the F1 score dropped further to 0.038, precision to 0.406, and recall to 0.020. These metrics indicate poor performance in correctly identifying positive cases and overall prediction effectiveness.

**Implication:** The low F1 score, precision, and recall on both training and test datasets suggest that the deep learning model is not effectively learning important features or patterns within the data.

## Overall overview

In the following Table the final result obtained with all technique for a confrontation.

A graph of a graph

Description automatically generated with medium confidence

A table with numbers and a number of trees

Description automatically generated with medium confidence

# Conclusion

In this study, different supervised learning algorithms have been inspected and presented with their mathematical details, and finally used on the UCI dataset to build a classification model that is able to predict if a credit card clients will default in the next month. Data preprocessing makes algorithms perform slightly better than when trained with original data: this is likely because the model is trained on a large amount of data. However, all the models implemented achieved comparable results in terms of accuracy.

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

# References

https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients