TESI DI DOTTORATO

Dipartimento di Economia e Metodi Quantitativi

Statistical models for the corporate financial distress prediction

Modelli statistici per la previsione dell'insolvenza aziendale

Carmen CUTUGNO

Tutor: Prof. Salvatore Ingrassia

Coordinatore Dottorato: Prof. Benedetto Matarazzo

Dottorato di Ricerca in "Matematica per le decisioni economiche e finanziarie", XXIII Ciclo - 2010

Settore Scientifico Disciplinare: SECS-S/01

Università degli Studi di Catania

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Corporate Distress Analysis

Corporate distress analysis is a matter of interest for many parties: lending institutions, regulatory authorities and investors in general. Research on financial distress prediction models has relevance to lending institutions both in deciding whether to grant or not a loan (and its related conditions) and in devising policies to monitor exiting loans.
Furthermore, distress prediction models may provide earlier warnings of financial problems and assess the likelihood of a company experiencing problems in principal repayments. Corporate distress could be consider in the more general framework of credit scoring, referring to every kind of evaluation on credit recovery at a fixed period of time (consumer loans, mortgages, and so on) where a financial institution needs to distinguish between good risk and bad risk applicants for credit. In reality there are not well-defined classes of good and bad risks. Accurate models assign each applicants (consumer or company) a probability of defaulting on repayment. Furthermore, there are complex issues of what is meant by ”default”. If we focus on the case of corporate loans, there is a big issue if considering the bankruptcy criterion an appropriate proxy for considering a company reliable (in terms of solvency) or not, or if the strictly legal limitation of a filing for bankrupt could be replaced by other proxy criteria.
The problem stands largely on the data available to implement a model, because if we

consider a bank data-warehouse it could provide us all sort of information connected both to the financial statement data and to the loan-repayments referred to the applicant. But, if we refer to a new applicant without a credit ”history” on debt-recovery or if we simply have no such information, we could only refer to quantitative data (financial or economic ratios) comparable to the industry sector and to previous financial default data used as learning sample to implement forecasting models.

In the past decades, classification into these two classes (good and bad risks) was made by human judgement on the basis of past experience. The scale of the current credit industry, the advance in computer technology along with the development of classification methods and then the legislation that aimed at preventing subjective prejudice from influencing decision have modified the decision process towards more objective analysis models.

Financial institutions such as banks, insurance and loan companies are subject to overview by regulatory bodies for solvency and stability, in particular in Europe they are under Basel II regulation framework.[1]

Basel II requires banks to assess appropriate internal credit risk measurement and management systems, compelling banks to improve their risk rating systems. The estimation of risk parameters, namely probability of default, loss given default and exposure at default, is the basis for the regulatory capital calculation, because these risk parameters determine the minimum capital requirements for a bank. Credit model scores are key inputs for pooling retail portfolios and estimating the risk components. According to the first pillar of Basel II accord: minimum capital requirements consider that ”for corporate and bank exposures, the PD is the greater of the one-year PD associated with the internal borrower grade to which that exposure is assigned, or 0.03%”, [130].

Consequently, monitoring the performance of the underlying rating systems is a key com-

[1]The Basel Committee on Banking Supervision (BCBS), that maintains its secretariat at the Bank of International Settlements (BIS) in Basel, is implementing a new revision of the Basel Accords, *Basel III*, a comprehensive set of reform measures to strengthen the regulation, supervision and risk management of the banking sector. The update aims at 1) improving the banking sector’s ability to absorb shocks arising from financial and economic stress, whatever the source, 2) improving risk management and governance, 3) strengthening banks’ transparency and disclosures. The reforms target are: bank-level, or micro-prudential, regulation, which will help raise the resilience of individual banking institutions to periods of stress; macro-prudential, system wide risks that can build up across the banking sector as well as the pro-cyclical amplification of these risks over time.

ponent of the supervisory review process.

Regulators expect that a Basel II implementation leads to better risk management and, ultimately, produces tangible benefits for the business.

Some authors, [127], created the binomial dependent variable (default/no default) by observing the situation of each firm at the end of the next financial year and (following Basel II definition, only if the company is ninety or more days past it is considered as a default). In order to apply the Basel II formulas for the A-IRB approach, we had to provide four inputs: probability of default (PD), loss given default (LGD), exposure at default (EAD) and maturity (M). The default (distress) prediction models are aimed at estimating PD, in particular the one year PD required under Basel II. In their sample,they observed an high variability in the distributions of the financial ratios for the two dependent variable groups. They interpreted it as due to the different sectors in which the companies operate (for example real estate firms have financial data completely different from agricultural companies). After processing the default prediction models, they create seven rating classes and the probability of default (PD) for each rating class is calculated by dividing the number of default by the total number of enterprises in each class. Rating classes have been created in order to obtain the value of PD closest to the one showed by bond equivalent PD distribution.

The response in a prediction problem could be quantitative or qualitative; in a quantitative problem the key standard variable is numerical, whereas in a qualitative problem the key standard is categorical. Typically, in credit applications a qualitative response has two categories (default or not, for example) though there are exceptions (8-class market segmentation system in [132]). So as above-mentioned, because of its ubiquity and importance, most of the works refer to the two-class case. In the two-class case, predictive models can be though of as yielding estimated probabilities (or monotonic transformations of these) that the applicant will belong to each of the classes- and as a classification (prediction to the qualitative category) is made by comparing the prediction with a threshold, [128]. If the predicted probability of belonging to class 0 is greater than the threshold, the point is classified into class 0, and otherwise into class 1. The threshold thus defines a (decision) surface, with new points which fall on one side being classified into class 0 and

those which fall on the other side being classified into class 1. (the relationship between the underlying continuum and the probabilities of predicting the memberships of each of the two classes has been explored in [131]). A subtlety, arising in both quantitative and qualitative predictive situations, is that the response variable is often a proxy for something else of real interest. For example, default might be an easily measured alternative standing in for creditworthiness or profitability, both of which are difficult to define, let alone measure (and this is were the measurement approach can be valued, yielding more appropriate response variables based on and derived from those that can be easily measured). The generic two-class prediction problem in credit scoring begins a set of data describing previous customers, among with a class variable indicating the outcome (good or bad). The aim is to use this information to construct a model relating the descriptive data to the outcome indicator so that new customers, for whom one has the only descriptive data, can be allocated to a likely class. In the retail banking context such a model is called a *scorecard*, after the early pre-computer system, and this is the generic term usually adopted in literature and industry.[2]

Neural networks are complex models and they can be highly effective, at least in those problems where the decision surface is complex and also well separates the classes. On one hand, the complexity of such models means that they defy simple interpretation. Moreover, if the classes are not well separated, if there is class overlapping in the predictors space, neural networks could better perform in the decision surface being non linear. This is often the case in application scoring situations where the predictor variables simply do not permit highly accurate separation of the classes.

Improving the accuracy of a credit risk model is likely to have beneficial effects on the Basel II capital requirement when the Advanced Internal Rating (A-IRB) approach is used, [126]. Indeed, applying a model with higher accuracy will result in lower capital requirement regardless the companies are classified as retail customers or corporates. The new Basel Capital permits banks the possibility to choose whether to classify firms (with

[2]The earliest form of predictive model assigned a numerical weight to each category of each variable and calculated a score for a new applicant by summing its weights over the variables. The result was displayed in the form of a table showing the weights for each category of each variable, hence the term scorecard, essentially assigning larger weights to less risky applicants.

sales less than €. 50 million and exposure less than that €. 1 million) as corporate or as retail. In Pillar 1 of Basel II Accord, the rules to calculate bank capital requirements for each of the different segments are clearly explained. All formulas (both for Non-SME's and SME's)[3] follow the same calculation steps involving inputs for correlation (R), capital requirement (K) and risk-weighted assets (RWA). The most important input variables to be provided by the banks are three: PDs, LGDs and exposure at default EAD; while the asset correlation (R) is implicitly given by the Basel formulas.[4].

Banks, in particular, and most financial institutions worldwide, have either recently developed or modified existing internal credit risk systems or are currently developing methods to conform with best practice systems and processes for assessing the probability of default (PD), and, possibly, loss-given-default (LGD) on credit assets of all types.

As well defined by Altman, [124], the assignment of appropriate default probabilities on corporate credit assets is a three-step process involving from the development of:

1. credit scoring models;
2. capital market risk equivalents- usually bond ratings;
3. assignment of PD and possibly LGDs on the credit portfolio.

It is important to notice that a statical methodology (such as logit-regression or neural networks) can combine steps 1) and 2) where the output from 1) automatically provides estimates of PD. This is one of the reasons why logit-regression approaches has been preferred by some consultancy modellers developers rather than the discriminant model.

Financial distress is used to mean severe liquidity problems that cannot be resolved without a sizeable rescaling of the entity's operations or structure.

Filing for bankruptcy is the criterion used in most studies, even if this event is a legal one that could be heavily influenced by the actions of bankers or other creditors. Ambiguity caused by the presence of a non financially distressed company but bankrupt and

[3] SME stands for Small Medium Enterprise.

[4] Asset correlation (R) represents the correlation between the assets in the specific portfolio (retail, corporate, equity, etc.) and in the Accord different formulas are given to calculate this value based on the nature of the assets, [130]

vice versa is an inherent limitation connected with bankruptcy research studies. Even so, we may observe that prior to bankruptcy companies strongly tend to pass common and preferred dividends, go into technical defaults and engage in forced sales of assets, all to the detriment of securities values.

The evaluation of a firm financial status depends on many factors, financial and economic, that could be predictive of a weak level of solvency or a pre-condition for a liquidation proceeding, stating the inability of the firm to pay its financial obligations.

There are several indicators of, or information sources about, the likelihood of financial distress: *cash flow* analysis, *corporate strategy* of the firm in relation to the referring industry, *financial statements* and *external variables* (such as security returns and bond rating potentially encoding information about cash flows and financial statements items). The estimate of cash flow analysis is critically dependent on the assumptions underlying the preparation of the budget. The financial statements analysis of the firm and those of a comparison set of firms could focus on a single financial variable (univariate analysis) or on a combination of financial variables (multivariate analysis). Moreover, the information about the financial status of a firm and its positioning in the belonging industry-sector is crucial for investors, stockholders, loan-analysts or creditors, taking into concern the description and the representative characteristics of the analysed sector.

In the industry, objective statistical methods of allocating applicants to risk classes are known as credit scoring methods. This term derives from summated rating scale with a threshold methods, very widespread in the industry, a sort of application score.

In the literature, several estimation methods have been suggested to predict financial distress, from the simple univariate analysis [1], to multiple discriminant analysis (MDA) [2], [3], logit [4] and probit models [5], artificial neural network models (ANN) [6], [7], rough set theory [8], Bayesian network (BN) models [9], and genetic programming [10]. The above-mentioned methodologies are considered as supervised classification methods, where a collection of labelled patterns are provided and the problem is to label a new unlabelled item. The historical training patterns are used to learn and to derive rules of classes [11]. Some authors, [125], compared the performance of different methodologies applied on corporate distress diagnosis. Even if their work could not be said to disclose that a

clear dominance[5] of neural network (NN) compared to traditional statistical techniques (in their work, linear discriminant analysis, LDA), they concluded in a more balanced way considering both advantages and shortcomings of the black-box NN technique.[6] The results of their work are near or superior to the ones obtained by LDA, so finally, taking into consideration the not transparent economic interpretation of NN coefficient, they suggest to use the two techniques in tandem.

Supervised learning can, usually, achieve high prediction accuracy if the training data and the analysed data have similar characteristics. The disadvantage emerges if there are no data records with known class labels.

Otherwise, in bankruptcy studies, samples tend to be small or the information about the failure status may not be readily available for training a supervised methodology.

Differently, data mining techniques and clustering methods belong to the unsupervised classification methods dealing with the problem of predicting unobserved features (cluster labels) from observed ones, so category labels are data driven.

Recent researches have proposed the application of clustering analysis and data mining in the field of the performance evaluation of industrial organization [12], [13] and financial distress prediction, [14], [15].

In the literature, the variables generally considered in the analysis concerning distress prediction models consist of financial ratios or of a set of financial and economic ratios, or sometimes of different variable classes composed of both ratios and balance-sheet items.

Rules for classifying applicants into good or bad classes would develop from a design set; the speed of developing and implementing these rules is important and they probably need to be changed quite frequently (depending on the precise application area).

There are important issues to be considered when we refer to credit scoring or corporate distress analysis, that are: the quality of the data, regional and macroeconomic effects, individual factors, default probability dynamic, default events correlation, monitoring process and scoring overrides.

The data quality refers to many aspects. Firstly, the correct formulation of the object

[5] On the contrary, several works in the same period, concluded considering NN approach in financial distress classification superior to the other methodologies.

[6] It should be underlined that their work was conducted on a induced 50% balanced sample.

population in order to set the learning and the control sample. Very often, the scarcity of data induces to collect information belonging different institutions in a single data-base, so risking to have bias factors that could affect the results of the model.[7] Secondly, the quality of the data is related to the variable taken into consideration in the model. In corporate credit scoring, financial and economic ratios are generally considered and the issue refers to the correct indicators to be selected, and their capability to provide information in relation to the problem and the industrial belonging sector.

In the model implementation, the effect of macroeconomic scenario is generally ignored. Several works, [123], confirmed that the omission of these factors tend to create biased estimations on the individual characteristics. The more is the time considered and the data geographically dispersed, the more the bias effects increase.

Generally, models do not include complete information referred to an applicant because of not observed or the observation refers to a previous period.

One of the principal criticism on the scoring model is that they do not consider the time dynamic of the default probability of an applicant.

In the corporate credit, it is possible that the default probability of a company is correlated to other companies default in the economic system they operate. So we are in presence of correlation between units to be considered in the model.

The issue of monitoring the scoring model is current and related to the capability to remain accurate and updated.

Scoring overrides issue relates to the operating problem to change or modify a pre-existing risk-analysis system in a bank or financial institution.

In **part I** of the work, we present an overview of the unsupervised statistical methodologies that could be applied for the industrial sector analysis, in the segmentation perspective and potential distress level classification. In particular, we delve into the model-based clustering methodology and its application as an unsupervised classification methodology for sector segmentation. We present a case-study A concerning an industry sector analysis focused on clustering two industrial segments into sub-groups according to financial and operating characteristics by using the above-mentioned classification procedure, the

[7] For further details, see [103]

model-based clustering, operating in a multivariate.

The **part II** of the work, is focused on the supervised statistical methodologies applied on credit scoring or corporate distress analysis, latu sensu, by drawing the outlines of two methodologies: *Logistic regression* and *Neural Networks.*

Moreover, an overview is presented of the *Evaluating rules* considered for selecting and evaluating the performance of a model.

We report a case-study B focused on a comparison of the two above-mentioned supervised methodologies for predicting financial distress: *Logit* and *Neural Networks*, and to the problem of *class imbalance.* Furthermore, the *class imbalance* problem, generally faced in credit scoring model, is analysed and the related literature presented.

A case-study C is reported, analysing the effect of different unbalance levels in a neural network distress prediction model.

Part I

Unsupervised statistical methodologies

Chapter 2

Unsupervised methods

Unsupervised methods are statical models used when no classes are defined or assigned *a priori* and aiming at finding structure or identifying specific patterns in the data by the use of data mining techniques. They are generally implemented in biological and diagnosis studies, marketing and data warehouse analysis.

The main characteristic of unsupervised methods is to detect a common structure and differentiation elements in the data, and are generally used to organize, to classify and to visualize data. The exploratory data analysis methods could be divided into two main typologies: *clustering* methods, consisting in grouping elements into clusters according to specific distance measures; *projection* methods, focused on reducing data dimensionality in lower dimensional space maintaining some initial features as to preserve the core of the information.

If we consider a set D = ((x_1),(x_2), ..., (x_n)) of independent variables (x_i) $\in \mathbb{R}^m$, we denote an *unsupervised estimation problem* the one concerning the subdivision of the set D into disjoint subsets as that the **x** vectors, belonging to the same subset, are similar according to a similarity measure.

The importance of implementing *data-driven* methodologies of analysis rely both due to the problem of having *a priori* information about a phenomenon or because it could be a latent one and to the presence of a wide set of informations stored in data warehouse in a variety of sectors: finance, banking, retail sales, manufacturing, marketing and medical diagnosis, to be exploited to derive association rules. Data may be investigated to find

functional dependencies and to search for patterns considered as representations of a process, trying to extract information from the data as intended in the general definition of *knowledge discovery analysis.*

In the following chapter, we analyse in the section 2.1 a projection method, the **principal component analysis** and in section 2.2 a clustering methodology, the **model based clustering**. Section 2.3 presents a case study on **industry sector analysis** processed by the use of a model based clustering.

2.1 Principal component analysis

In exploring high-dimensional data sets for group structure, it is typical to rely on "second-order" multivariate techniques, in particular, *principal component analysis* (PCA) in order to reduce dimensionality.[1] As a first objective principal component analysis seeks the standardized linear combination of the original variable which has maximal variance.[2]
Let $\mathbf{x}$ be a random vector with mean μ and covariance matrix Σ, than the principal component transformation is the transformation

$$\mathbf{x} \rightarrow \mathbf{y} = \Gamma'(\mathbf{x} - \mu) \tag{2.1}$$

where Γ is orthogonal, $\Gamma'\Sigma\Gamma = \Lambda$ [3] is diagonal and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$. The strict positivity of the eigenvalues λ_i is guaranteed if Σ is positive definite. The *ith principal component* of $\mathbf{x}$ may be defined as the ith element of the vector $\mathbf{y}$:

$$y_i = \gamma'_{(i)}(\mathbf{x} - \mu) \tag{2.2}$$

Here $\gamma_{(i)}$ is the ith column of Γ, and may be called the ith vector of *principal component loadings*. If $\mathbf{x} \sim (\mu, \Sigma)$ and $\mathbf{y}$ is defined in (2.1), then some properties of principal components:

$$\begin{aligned} E(y_i) &= 0; & Var(y_i) &= \lambda_i; \\ Cov(y_i, y_j) &= 0 \quad for\ i \neq j; & Var(y_1) &\geq Var(y_2) \geq \cdots \geq Var(y_p) \geq 0; \\ \sum_{i=1}^{p} Var(y_i) &= tr\Sigma; & \prod_{i=1}^{p} Var(y_i) &= |\Sigma| . \end{aligned}$$

The sum of the first k eigenvalues divided by the sum of all the eigenvalues represents the *proportion of total variation explained by the first k components*,

$$(\lambda_1 + \cdots + \lambda_k)/(\lambda_1 + \cdots + \lambda_p) \tag{2.3}$$

[1] Principal component analysis was developed by Hotelling (1993) after the former work by Karl Pearson (1901).
[2] This attempt to reduce dimensionality can be described as "parsimonious summarization" of the data.
[3] The representation of Σ follows from the spectral decomposition.

One disadvantage of principal component analysis is that the principal components are not scale-invariant. The correlation between the ith variable x_i and the jth principal component y_j,

$$\rho_{ij} = \gamma_{ij}(\lambda_j/\sigma_{ii})^{1/2} \tag{2.4}$$

When Σ is a correlation matrix, $\sigma_{ii} = 1$ so $\rho_{ij} = \gamma_{ij}(\lambda_j)^{1/2}$.
If we consider the p standardized variables Z, the score of the jth principal component on the hth individual unit is expressed by

$$y_{hj} = \gamma_{j1}z_{h1} + \gamma_{j2}z_{h2} + \cdots + \gamma_{jp}z_{hp}; for h = 1, ..., n \tag{2.5}$$

where γ_{ji} is the coefficient of the jth principal component and ith variable.

2.2 Model based clustering

Cluster analysis consists in the identification of groups of observations that are cohesive and separated from other groups. Most clustering methodologies are based on heuristic procedures, as hierarchical agglomerative clustering or iterative partitioning methods such as k-means method. The statistical properties of these methods are generally unknown, precluding the possibility of formal inference. Recently in literature clustering procedure has been related to probability models, [117], and it has also been shown that most popular heuristic methods are approximate estimation for certain probability methods. For example, standard k-means clustering and Ward's method have been considered equivalent to known procedures for approximately maximizing the multivariate normal classification likelihood when the covariance matrix is the same for each component and proportional to the identity matrix.

In literature, finite mixture models have been proposed in the context of clustering by several authors ([118], [119], [120], [121], [122]) but only recently these models have been recognized to provide a principled statistical approach to the problems arising in applying clustering methods ([17], [108]).

In finite mixture models, each component probability distribution corresponds to a cluster. The problem of determining the number of clusters and of choosing an appropriate clustering method can be recast as statistical model choice problems and models that differ in numbers of components and/or in component distributions can be compared.

Fraley and Raftery (2002), [19], presented a clustering strategy combining model-based hierarchical agglomeration and the EM algorithm for maximum likelihood estimation of multivariate mixture models. So they exploit the capability of the first one, hierarchical agglomeration, of producing reasonably good partitions even when started without any information about groupings, whereas in EM initialization represents a critical point for the optimization process in presence of local minima.

Then, by initializing EM with reasonable starting partitions from hierarchical agglomeration, they obtain improved estimated partitions.

The Bayesian Information Criterion (BIC) approximation was proposed to determine the

number of group in the data, firstly in a work by [115], and then extended by [108] to select simultaneously the parametrization of the model and the number of clusters.
Given data $\mathbf{y}$ with independent multivariate observations $\mathbf{y}_1, \cdots, \mathbf{y}_n$, the likelihood for a mixture model with g components is

$$\mathcal{L}_{MIX}(\theta_1, \cdots, \theta_g|\mathbf{y}) = \prod_{j=1}^{n} \sum_{i=1}^{g} \pi_i f_i(\mathbf{y}_j, \theta_i) \tag{2.6}$$

where f_i and θ_i are the density and parameters of the ith component in the mixture and π_i is the probability that an observation belongs to the ith component ($\pi_i \geq 0; \sum_{i=1}^{g} \pi i = 1$).
In *model based clustering*, each cluster is, generally, represented by a Gaussian model. Let f_i be a multivariate normal density ϕ_i, parametrized by its mean μ_i and covariance matrix Σ_i, the model is

$$\phi_i(\mathbf{y}|\mu_i, \Sigma_i) = (2\pi)^{-\frac{p}{2}} |\Sigma_i|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}(y_j - \mu_i)^T \Sigma_i^{-1} (y_j - \mu_i) \right\} , \tag{2.7}$$

where $\mathbf{y}$ represents the data, and *i* is an integer subscript specifying a particular cluster. Formerly, the mixture models for clustering analysis considered only equal covariance matrix Σ. Model-based clustering offers different modelization of the covariance matrix Σ, that could be parametrized by spectral decomposition, in the form:

$$\Sigma_i = \lambda_i D_i A_i D_i' , \tag{2.8}$$

where $\lambda_i = |\Sigma_i|^{\frac{1}{d}}$ is a scalar, D_i is the orthogonal matrix of eigenvectors of Σ_i and A_i is a diagonal matrix whose elements are proportional to the eigenvalues of Σ_i, [17]. The orientation of the principal components of Σ_i is determined by D_i, while A_i determines the shape of the density contours; λ_i specifies the volume of the corresponding ellipsoid, which is proportional to $\lambda_i^d |A_i|$, where *d* is the data dimension.
Characteristics (orientation, volume and shape) can vary between clusters, or be constrained to be the same across clusters.
Data generated by mixture of multivariate Gaussian density are characterized by components or clusters that are ellipsoidal and centred at the means μ_i, whereas the covariances Σ_i determine their other geometric features.

Model-based clustering procedure was introduced by Banfield and Raftery (1993), [17], that proposed a general framework for geometric cross-cluster constraints in multivariate normal mixtures by parametrizing covariance matrices through eigenvalue decomposition. By allowing some but not all of the three features λ_i, D_i and A_i to vary between clusters, parsimonious and easily interpreted models could be obtained which are appropriate to describe various clustering situations, [112].

Their idea was to treat λ_i, D_i and A_i as independent sets of parameters and either constrain them to be the same for each cluster or allow them to vary among clusters. When parameters are fixed, clusters will share certain geometric properties. This approach generalize the work of Murtagh and Raftery (1984), [116], who used the equal shape/equal volume model for clustering in character recognition. It also subsumes the three most common models: λI, equal variance and unconstrained variance; $\Sigma_i = \lambda_i I$, where the clusters are spherical but have different volumes; $\Sigma_i = \lambda_i A_i$, where all covariances are diagonal but otherwise their shapes, sizes and orientation are allowed to vary.

This parametrization includes well-known models such as uniform spherical variance ($\Sigma_i = \lambda I$) which gives the sum of squares criterion, constant variance and unconstrained variance. In one dimension, there are just two models: E for equal variance and V for varying variance. For more than one dimension, in MCLUST, the model identifiers code geometric characteristics of the model (volume, shape, orientation) with an (E) if equal, (I) identity and (V) variable, as shown in Table 2.1. In the first two columns, there are specified the identifier and the name of the model. Then, the third column reports the correspondent distribution and the other columns indicate volume, shape and orientation. The last column indicates the cost of the model, in terms of number of parameters to estimate. Parameters associated with characteristics designated by E or V are determined from the data. The first family of models is referred to spherical shapes, as $A_i = I$ where I is the identity matrix. This parametrization leads to two models: $\lambda I; \lambda_i I$. The second family of modelizations consists in assuming that the covariance matrices Σ_i are diagonal, it means that in the parametrization the orientation matrices D_I are permutation matrices. In such a case variations on the shape matrices seem to be of any particular interest, and the models arising are four: $\lambda A; \lambda_i A; \lambda A_i$ and $\lambda_i A_i$. In the third case, the

Id	Model	Distribution	Volume	Shape	Orientation	Number of parameters
E		(univariate)	equal			1
V		(univariate)	variable			g
EII	λI	Spherical	equal	equal	NA	$\alpha + 1$
VII	$\lambda_i I$	Spherical	variable	equal	NA	$\alpha + d$
EEI	λA	Diagonal	equal	equal	coordinates axes	$\alpha + d$
VEI	$\lambda_i A$	Diagonal	variable	equal	coordinates axes	$\alpha + d + g - 1$
EVI	λA_i	Diagonal	equal	variable	coordinates axes	$\alpha + dg - g + 1$
VVI	$\lambda_i A_i$	Diagonal	variable	variable	coordinates axes	$\alpha + dg$
EEE	λDAD^T	Ellipsoidal	equal	equal	equal	$\alpha + \beta$
EEV	$\lambda D_i AD_i^T$	Ellipsoidal	equal	equal	variable	$\alpha + g\beta - (g-1)d$
VEV	$\lambda_i D_i AD_i^T$	Ellipsoidal	variable	equal	variable	$\alpha + g\beta - (g-1)(d-1)$
VVV	$\lambda_i D_i A_i D_i^T$	Ellipsoidal	variable	variable	variable	$\alpha + g\beta$

Table 2.1: Parametrizations of the covariance matrix Σ_i, available in MCLUST. In the analysed unrestricted case (mixture) $\alpha = gd + g - 1$, whereas $\beta = (d(d+1)/2)$.

ellipsoidal family (the more expensive in terms of number of parameters), it is possible to assume variable volume, equal shape and variable orientation (VEV model, $\lambda_i D_i AD_i^T$), or the other three ellipsoidal models by varying the assumptions on the geometric features ($\lambda DAD^T; \lambda D_i AD_i^T; \lambda_i D_i A_i D_i^T$).

In model-based clustering, we assume a mixture model with incomplete data and we use the EM algorithm, Dempster et al. (1977). EM algorithm maximizes the likelihood function $\mathcal{L}(\psi|y_1, ..., y_n)$ indirectly by proceeding iteratively in two steps, E-step and M-step, applied on the complete data log likelihood function, $\log L_c(\psi)$. EM is strongly sensitive to initialization, being a local maximizer seeker, and also, because of the unboundiness of the likelihood function, the optimization could fail, converging to some singularities, [18] for constrained ML formulations. A procedure is to initialize EM with the model based hierarchical results and to use approximate Bayes factors with the BIC (Bayes Information Criterion) to determine the number of clusters, see [19].

The EM algorithm procedure for mixture model is considered in details in **Subsection 2.2.1.**. For further details on MCLUST programming procedure, see **Appendix A**.

Model selection There is a trade-off between the choice of the number of clusters and that of the clustering model. If a simpler model is chosen, then more clusters may be needed to provide a good representation of the data. If a more complex model is used,

then fewer clusters may be considered. As regards model based clustering, we notice that mixture-model approach to clustering allows the use of approximate Bayes factors to compare models so we select both the parametrization of the model and the number of clusters. The Bayes factor is the posterior odds for one model against another assuming neither is favoured a priori. In MCLUST, EM algorithm is used to find the maximum mixture likelihood, and the criterion used for selection corresponds to the BIC[4] equals to twice the log Bayes factor:

$$2\log(p(x|\mathcal{M})) + constant \approx 2l_{\mathcal{M}}(x,\widehat{\theta}) - m\mathcal{M}\log(n) \equiv BIC \tag{2.9}$$

where $p(x|\mathcal{M})$ is the likelihood of the data for the model $\mathcal{M}$, $l_{\mathcal{M}}(x,\widehat{\theta})$ is the maximized mixture log-likelihood for the model and $m\mathcal{M}$ is the number of independent parameters to be estimated in each model. The number of clusters is not considered an independent parameter in computing BIC. If each model is equally likely a priori, the $p(x|\mathcal{M})$ is proportional to the posterior probability that the data conform to the model $\mathcal{M}$. The larger the value of the BIC, the stronger the evidence for the model. Kass and Raftery, see [106], define the BIC to have opposite sign to that given here, in which the smaller (more negative) the BIC, the stronger the evidence for the model.

Fraley and Raftery, see [108], chose to reverse this convention considering easier the interpretation of the BIC values plots.[5]

Likelihood cannot be used directly in the evaluation of models for cluster analysis, because the fit of a mixture model can only improve as more terms are added to the model. This explains the reason why in BIC, it is added a term to the likelihood to penalize the complexity of the model, in order to maximize for more parsimonious parametrizations and smaller numbers of groups. The BIC can be used to compare models with different parametrizations, different number of components or both. The R statistical package MCLUST provides a function to compute the Bayesian Information Criterion (BIC) given the maximized log-likelihood for model, the data dimensions, and the number of components in the model, and allows comparison of models with differing parametrizations and/or differing numbers of clusters. In general, the larger the value of the BIC the

[4] Schwarz 1978, see [107].

[5] Fraley and Raftery use that convention in MCLUST R Statistical Package, see [109].

stronger the evidence for the model and number of clusters, see [109].

2.2.1 EM algorithm

The Expectation-Maximization (EM) algorithm[6] is a broadly applicable iterative procedure for computing MLEs in the context of incomplete-data problems. On each iteration of the EM algorithm, there are two steps, called the *expectation step* or E-step and the *maximization step* or M-step.

Let $\mathbf{Y}$ denote the random vector corresponding to the observed data $\mathbf{y}$, having p.d.f. postulated as $f(\mathbf{y};\Psi)$, where $\Psi = (\Psi_1, ..., \Psi_d)^T$ is a vector of unknown parameters with parameter space Ω.

The observed data vector $\mathbf{y}$ is viewed as being incomplete and is regarded as an observable function of of the so-called complete function. The notion of incomplete data includes the conventional sense of missing data, but it also applies to situations where the complete data represent what would be available from some hypothetical experiment. In the latter case, the complete data may contain some variables that are never observable in a data sense. Within this framework, let *x* denote the vector containing the augmented or so-called complete data, and let *z* denote the vector containing the additional data, referred to as the unobservable or missing data.

Let $f_c(\mathbf{x};\Psi)$ denote the p.d.f. of the random vector $\mathbf{X}$ corresponding to the complete-data vector $\mathbf{x}$, $\mathbf{X} = (\mathbf{Y}, \mathbf{Z})$. Then the complete-data log likelihood function that could be formed for Ψ if $\mathbf{x}$ were fully observable is given by:

$$\log \mathcal{L}_c(\Psi) = \log f_c(\mathbf{x};\Psi) \tag{2.10}$$

Formally, we have two samples spaces χ and $\mathcal{Y}$ and a many-to-one mapping from χ to $\mathcal{Y}$. Instead of observing the complete-data vector $\mathbf{x}$ in χ, we observe the incomplete-data

[6]The name EM was given by Dempster, Laird and Rubin (1977), for a complete overview on the topic see [99].

vector $\mathbf{y} = \mathbf{y}(\mathbf{x})$ in $\mathcal{Y}$. It follows that:

$$f(\mathbf{y}; \Psi) = \int_{\chi(y)} f_c(\mathbf{x}; \Psi) d\mathbf{x} \tag{2.11}$$

where $\chi(y)$ is the subset of χ determined by the equation $\mathbf{y} = \mathbf{y}(\mathbf{x})$.
The EM algorithm approaches the problem of solving the incomplete-data likelihood equation, $\partial \log \mathcal{L}(\Psi)/\partial\Psi = 0$, indirectly by proceeding iteratively in terms of the complete-data log likelihood function, $\log \mathcal{L}_c(\Psi)$. As it is unobservable, it is replaced by its conditional expectation given $\mathbf{y}$, using the current fit for Ψ. More specifically, let $\Psi^{(0)}$ be some initial value for Ψ. Then on the first iteration, the E-step requires the calculation of

$$\mathcal{Q}(\Psi; \Psi^{(0)}) = \mathcal{E}_{\Psi^{(0)}} \{\log \mathcal{L}_c(\Psi) | \mathbf{y}\} \tag{2.12}$$

The M-step requires the maximization of $\mathcal{Q}(\Psi; \Psi^{(0)})$ with respect to Ψ over the parameter space Ω. That is, we choose $\Psi^{(1)}$ such that

$$\mathcal{Q}(\Psi^{(1)}, \Psi^{(0)}) \geq \mathcal{Q}(\Psi; \Psi^{(0)}) \tag{2.13}$$

for all $\Psi \in \Omega$. The E- and M-step are then carried out again, but this time with $\Psi^{(0)}$ replaced by the current fit $\Psi^{(1)}$. On the (k+1)th iteration, the E- and M-steps are defined as follows:

E-Step Calculate $\mathcal{Q}(\Psi; \Psi^{(k)})$, where

$$\mathcal{Q}(\Psi; \Psi^{(k)}) = \mathbf{E}_{\Psi^{(k)}} \{\log \mathcal{L}_c(\Psi) | \mathbf{y}\} \tag{2.14}$$

M-Step Choose $\Psi^{(k+1)}$ to be any value of $\Psi \in \Omega$ that maximizes $\mathcal{Q}(\Psi; \Psi^{(k)})$; that is

$$\mathcal{Q}(\Psi^{(k+1)}, \Psi^{(k)}) \geq \mathcal{Q}(\Psi; \Psi^{(k)}) \tag{2.15}$$

for all $\Psi \in \Omega$.

The E- and M-steps are alternated repeatedly until the difference

$$\mathcal{L}(\Psi^{(k+1)}) - \mathcal{L}(\Psi^{(k)}) \tag{2.16}$$

changes by an arbitrarily small amount in the case of convergence of the sequence of likelihood values $\{\mathcal{L}(\Psi^{(k)})\}$. It is demonstrated the monotony of EM algorithm, that is the (incomplete-data) likelihood function $\mathcal{L}(\Psi)$ is not decreased after an EM iteration, that is

$$\mathcal{L}(\Psi^{(k+1)}) \geq \mathcal{L}(\Psi^{(k)}) \tag{2.17}$$

for k=0,1,2,....Hence, convergence must be obtained with a sequence of likelihood values that are bounded above.

Another way of expressing (2.8) is to say that $\Psi^{(k+1)}$ belongs to

$$\mathcal{M}(\Psi^{(k)}) = \arg\max_{\Psi} \mathcal{Q}(\Psi; \Psi^{(k)}) \tag{2.18}$$

which is the set of points that maximizes $\mathcal{Q}(\Psi; \Psi^{(k)})$.

EM algorithm for estimation of mixture model parameters We now consider the application of the EM algorithm for the ML fitting of the parametric mixture model,

$$f(\mathbf{y}_j; \Psi) = \sum_{i=1}^{g} \pi_i f_i(\mathbf{y}_j; \theta_i) \tag{2.19}$$

to an observed random sample, $\mathbf{y} = (\mathbf{y}_1, \cdots, \mathbf{y}_n)$, where

$$\Psi = (\pi_1, \cdots, \pi_{g-1}, \xi^T)^T \tag{2.20}$$

is the vector containing all the unknown parameters in the mixture model and ξ is the vector containing all the parameters in $\theta_1, \cdots, \theta_g$ known a priori to be distinct. In detail, $\xi = (\mu_i, \Sigma_i)$ consists of the elements of the mixture components means, $\mu_1, \cdots, \mu_g$, and the distinct elements of the component covariance matrices, $\Sigma_1, \cdots, \Sigma_g$. The log likelihood for Ψ that can be formed from the observed data is given by,

$$logL(\Psi) \quad = \quad \sum_{j=1}^{n} logf(y_i; \Psi) = \sum_{j=1}^{n} log \sum_{i=1}^{g} \pi_i f_i(\mathbf{y}_j; \theta_i) \tag{2.21}$$

Computation of MLE of Ψ requires solving the above-mentioned likelihood equation, $\partial \log \mathcal{L}(\Psi)/\partial\Psi = 0$, so that the MLE of Ψ, $\hat{\Psi}$, satisfies,

$$\hat{\pi_i} = \sum_{j=1}^{n} \tau_i(\mathbf{y}_j; \hat{\Psi})/n; (i = 1, \cdots, g) \tag{2.22}$$

and

$$\sum_{g=1}^{g}\sum_{j=1}^{n}\tau_i(\mathbf{y}_j;\hat{\Psi})\partial log f_i;\hat{\theta}_i/\partial\xi = 0 \tag{2.23}$$

where

$$\tau_i(\mathbf{y}_j;\Psi) = \pi_i f_i(\mathbf{y}_j;\theta_i)/\sum_{h=1}^{g}\pi_h f_h(\mathbf{y}_j;\theta_h) \tag{2.24}$$

is the posterior probability that $\mathbf{y}_j$ belongs to the ith component of the mixture.

If we consider the mixture problem in the EM framework, the observed-data vector $\mathbf{y}$ is viewed as being incomplete, as the associated component-label g-dimensional vectors $\mathbf{z}_j$, are not available. Each y_j is conceptualized as having arisen from one of the components of the mixture model being fitted, where $\mathbf{z}_{ij} = (\mathbf{z}_j)_i = 1 or 0$, according to whether $\mathbf{y}_j$ arises or not from the ith component of the mixture $(i = 1,\cdots,g; j = 1,\cdots,n)$. The complete-data vector is expressed as

$$\mathbf{x} = (\mathbf{y},\mathbf{z}) \tag{2.25}$$

where

$$\mathbf{z} = (\mathbf{z}_1,\cdots,\mathbf{z}_n) \tag{2.26}$$

The component-label vectors $\mathbf{z}_j$ are taken to be realized values of the random vectors $\mathbf{Z}_j$ that are assumed to be distributed unconditionally according to the multinomial distribution. The complete-data log likelihood for Ψ is given by

$$\log L_c(\Psi) = \sum_{i=1}^{g}\sum_{j=1}^{n} z_{ij}\left\{\log\pi_i + \log f_i(\mathbf{y}_j;\theta_i)\right\} \tag{2.27}$$

The EM algorithm is applied to the mixture problem by treating the z_{ij} as missing data. The addition of the unobservable data to the problem, the $\mathbf{z}_j$, is handled by the E-step, which takes the conditional expectation of the complete-data log likelihood, $\log L_c(\Psi)$, given the observed data $\mathbf{y}$, using the current fit for Ψ. On the (k+1)th iteration of the EM algorithm, the E-step requires the computation of $\mathcal{Q}(\Psi;\Psi^{(k)})$, where $\Psi^{(k)}$ is the value of Ψ after the kth EM iteration. The E-step, on the (k+1)th iteration, requires the calculation of the current conditional expectation of $\mathbf{Z}_{ig}$ given the the observed data $\mathbf{y}$,

$$\mathbf{E}_{\Psi^{(k)}}(\mathbf{Z}_{ij}|\mathbf{y}) = pr_{\Psi_k}\left\{\mathbf{Z}_{ij} = 1|\mathbf{y}\right\} = \tau_i(\mathbf{y}_j;\Psi^{(k)}) \tag{2.28}$$

where

$$\tau_i(\mathbf{y}_j;\Psi^{(k)}) = \pi_i^k f_i(\mathbf{y}_j;\theta_i^k)/f(\mathbf{y}_j;\Psi^k) = \pi_i^k f_i(\mathbf{y}_j;\theta_i^k)/\sum_{h=1}^{g}\pi_h^k f_h(\mathbf{y}_j;\theta_h^k) \tag{2.29}$$

corresponds to the posterior probability that the jth observation belongs to the ith component of the mixture, given observed value $\mathbf{y}_j$ and the parameter estimate Ψ^k at the kth iteration, for $i = 1,\cdots,g$ and $j = 1,\cdots,n$.
After these premises, the E-step could be described as

$$\mathcal{Q}(\Psi;\Psi^{(k)}) = \mathbf{E}_{\Psi^{(k)}}\left\{\log\mathcal{L}_c(\Psi)|\mathbf{y}\right\} = \sum_{i=1}^{g}\sum_{j=1}^{n}\tau_i(\mathbf{y}_j;\Psi^{(k)})\left\{\log\pi_i + \log f_i(\mathbf{y}_j;\theta_i)\right\} \tag{2.30}$$

If the z_{ij} were observable, then the complete-data MLE of π_i would be given by

$$\hat{\pi}_i = \sum_{j=1}^{n} z_{ij}/n \qquad for\ \ (i = 1,\cdots,g) \tag{2.31}$$

As the E-step simply involves replacing each z_{ij} with its current conditional expectation $\tau_i(\mathbf{y}_j;\Psi^{(k)})$ in the complete-data likelihood, the update estimate of the mixing proportions π_i is given by replacing each z_{ij} by $\tau_i(\mathbf{y}_j;\Psi^{(k)})$ to give

$$\pi_i^{(k+1)} = \sum_{j=1}^{n}\tau_i(\mathbf{y}_j;\Psi^{(k)})/n \qquad for\ \ (i = 1,\cdots,g) \tag{2.32}$$

As concerns the updating of the parameters ξ estimate on the M-step at the (k+1) iteration, ξ^{k+1} is obtained as an appropriate root of

$$\sum_{i=1}^{g}\sum_{j=1}^{n}\tau_i(\mathbf{y}_j;\Psi^{(k)})\partial\log f_i(\mathbf{y}_j;\theta_i)/\partial\xi = 0 \tag{2.33}$$

whose solution often exists in closed form, obtaining the estimates of the component means μ_i and component covariance matrices Σ_i at the (k+1)th iteration. Considering that,

$$\mu_i^{(k+1)} = \sum_{j=1}^{n} z_{ij}^{(k)}\mathbf{y}_j/z_{ij}^{(k)} \tag{2.34}$$

and

$$\Sigma_i^{(k+1)} = \sum_{j=1}^{n} z_{ij}^{(k)} (\mathbf{y}_j - \mu_i^{(k+1)})(\mathbf{y}_j - \mu_i^{(k+1)})^T / z_{ij}^{(k)} \tag{2.35}$$

are the MLE's estimates of μ_i and Σ_i, if the z_{ij} were observable. As $\log L_c(\Psi)$ is linear in the z_{ij}, it follows that the z_{ij} are replaced by their current conditional expectations estimates $\tau_{ij}^{(k)} = \tau_i(\mathbf{y}_j; \Psi^{(k)})$, so that

$$\mu_i^{(k+1)} = \sum_{j=1}^{n} \tau_{ij}^{(k)} \mathbf{y}_j / \tau_{ij}^{(k)} \tag{2.36}$$

and

$$\Sigma_i^{(k+1)} = \sum_{j=1}^{n} \tau_{ij}^{(k)} (\mathbf{y}_j - \mu_i^{(k+1)})(\mathbf{y}_j - \mu_i^{(k+1)})^T / \tau_{ij}^{(k)} \tag{2.37}$$

for $i = 1, \cdots, g$ and $j = 1, \cdots, n$.

In *model-based clustering*, for the M-step, estimates of the means and probabilities have closed form, as above-exposed, involving the data from the E-step, see [108], [19].

Whereas, it is important to underline that, in the model-based framework, the computation of the covariance estimate $\hat{\Sigma}_i^{(k)}$ depends on its specific parametrization, according to the model considered. Details of the M-step for $\Sigma^{(k)}$ parametrized by the eigenvalue decomposition have been implemented in Celeux and Govaert (1995), see [112].

2.3 Case study A: Sector analysis via model based clustering

2.3.1 Outlines

The work presents an unsupervised procedure for the evaluation of the firm financial status, aiming at identifying a potentially weak level of solvency of a company through its positioning in a segmented sector. Model Based Clustering is, here, used to segment real datasets concerning sectoral samples of industrial companies listed in five European stock exchange markets. The analysis does not forecast failure or non-failure, rather it compares company's operating and financial characteristics mean and median of the different groups identified by the clustering process.

The underlying idea in this work is that the financial and economic features, defining the financial structure level, are strictly connected with the industry sector the firm belongs to [16], and that seeking for industry-sector key indicators levels may lead to a more appropriate evaluation of the firm financial profile.

In the case-study, an unsupervised procedure of classification analysis is presented, aiming at identifying the position of a company in a segmented sector. The choice of a data-driven methodology of analysis is due to the consideration that the information about a liquidation proceeding does not identify potential failure pre-condition because it refers to a stated insolvency status.

The time period considered in the evaluation process is from 2005 to 2007, in order to verify the classification dynamic of a firm in the sectoral segmented framework.

The procedure presented starts from the assessment of the financial and economic indicators that influence the specific industry sector, by a principal component analysis (PCA), and then it proceeds with operating the segmentation, in a financial and economic perspective, of the sector by clustering methodology.

We propose the use of the model based clustering because it allows the modelization of the covariance matrix and because of its capability to assign every unit to a n-group with a probability of belonging. The model based methodology is compared to another clus-

tering method, the K-means clustering. In model based clustering, it is assumed that the data are generated by a mixture of underlying distributions in which each component represents a different group or cluster and the problem of determining the number of clusters and of choosing an appropriate clustering method can be recast as statistical model choice problems, [17],[19].
The rest of the paper is organized as follows, a brief review about the model based clustering methodology, then the presentation of two industrial sector analysis, as Constructions and Technology Hardware sector, and the reported results of the proposed procedure.

2.3.2 Two sector analysis

We consider two datasets of yearly financial statement data of companies, selected according to the Industry Classification Benchmark (ICB), listed on Frankfurt, Madrid, Paris, London and Milan stock exchange markets, for the period 2005-2007. The first sample refers to the Technology Hardware sector and consists of 92 units for the period 2005-2007. The second sample refers to the Constructions sector and consists of 105 units for 2005, 107 units for 2006 and 113 units for 2007. We calculate a set of 13 financial and economic ratios: Quick ratio, Current ratio, Leverage, Debt Coverage, Cost of Debt, Equity to liabilities, Asset turnover, Expected time of liabilities refunding indicator, Ebitda to Sales ratio, Return on Asset (Roa), Return on Investment (Roi), Return on Sales (Ros) and Return on Equity (Roe).

The Statistical Analysis and Results

Firstly, we process a principal component analysis (PCA), as a pre-step examination on the variables and their influence on the data variability, then we run a model based clustering on the scores obtained by the PCA, in order to classify the companies into groups related to different financial structure levels. The variables (financial and economic ratios) are calculated by operating transformations of accounting data measured in the same unit. We do not standardize the variables to compute the principal components because measurements are on comparable scale, see [20], [21]. The clustering analysis has been

Table 2.2: Model and number of clusters selected

Year	Constructions Sector	Technology Hardware Sector
2005	VVI,3	VEV,3
2006	VEI,4	VEV,4
2007	VEV,3	VEV,4

processed by using the package MCLUST vers.3.3.1 (Fraley and Raftery, 2009) of the statistical software R. We select the best model according to the BIC corresponding to the different parametrisation of the covariance matrix Σ_j, and indicating the number of the component of the mixture.

Each unit is assigned to the component to which it has the highest estimated posterior probability of belonging and each distribution component of the mixture may correspond to a cluster and thus, in our analysis, to a group of companies, [19].

By examining the cluster centroids, mean and median, of the ratios, we may define different financial and economic structure levels in each component of the mixture.

For the first dataset, Technology Hardware sector, we select, by the PCA, three components explaining about the 72% of variance in 2005, 66% in 2006 and 68% in 2007. In all the three years, 2005-2007, the components extracted are strongly influenced by the evaluation of working capital, as expressed by the operating return and the relation between short-term debts and current assets, and also by the evaluation of the weight of net equity. In 2005 and 2007, we found a strong influence on the evaluation of the economic return for investors (or the ownership), expressed by the Roe, less strong in 2006. The evaluation of the firm's ability to refund financial debts and on the expense on debts, expressed by Debt Coverage and Cost of Debt, is, strongly, captured by the components extracted in the whole period. The asset turnover is relevant in 2005 and 2006, less in 2007. The operating return evaluation, is strongly captured in the whole period. In 2005, by the application of the model based clustering on the scores, as shown in Fig.1a and Fig.1b, the data have been fitted by a three-components mixture of Gaussian distributions, connected with a VEV,3 model, thus a ellipsoidal, equal shape model, presenting cluster 1 with about 53% of the observations, cluster 2 with 36% and cluster 3 with 11%, also see Table 1. Cluster 1

presents high levels of turnover, economic returns and indebtedness, see Table 2. Cluster 2 shows a medium level of indebtedness, low operating returns and asset turnover. Cluster 3, a marginal group, presents high level of indebtedness and low operating returns. In 2006, the data have been fitted by a four-components mixture of Gaussian distributions, connected with a VEV,4 model, thus an ellipsoidal, equal shape model. Cluster 1 with about 37% of the observations, cluster 2 with 23%, cluster 3 with 28%, and cluster 4 with 11%. Cluster 1 presents high economic return levels and quite high level of indebtedness. Cluster 2 presents a lower level of indebtedness, compared to group 1, but a very low level of operating economic return. Cluster 3 is highly indebted, not very high economic return levels, even if with a current financial management better than cluster 2. Cluster 4, a residual group. In 2007, the data have been fitted by a four-components mixture of Gaussian distributions, connected with a VEV,4 model, thus an ellipsoidal, equal shape model. Cluster 1 with about 48% of the observations, cluster 2 with 27%, cluster 3 with 17 %, and cluster 4 with 7%. Cluster 1 presents an average level of indebtedness and asset turnover, with economic return levels not very high, connected with a quite high level of cost of debt. Cluster 2 shows low economic return levels, low indebtedness, medium levels of asset turnover, and a quite good current financial situation. Cluster 3 presents good levels of asset turnover, but quite high levels of indebtedness and low economic return levels. Cluster 4 is a marginal group, presenting group centroids not very coherent from the economic point of view.

For the second dataset, Construction sector, we select, by the PCA, three components explaining about the 69% of variance in 2005, 76% in 2006 and 79% in 2007. In all the three years, 2005-2007, the components extracted are, strongly, influenced by the evaluation of working capital, as expressed by the operating return and the relation between short-term debts and current assets, and also by the evaluation of the weight of net equity. In 2006 and 2007, we found a strong influence on the evaluation of the economic return for investors, expressed by the Roe, less strong in 2005.

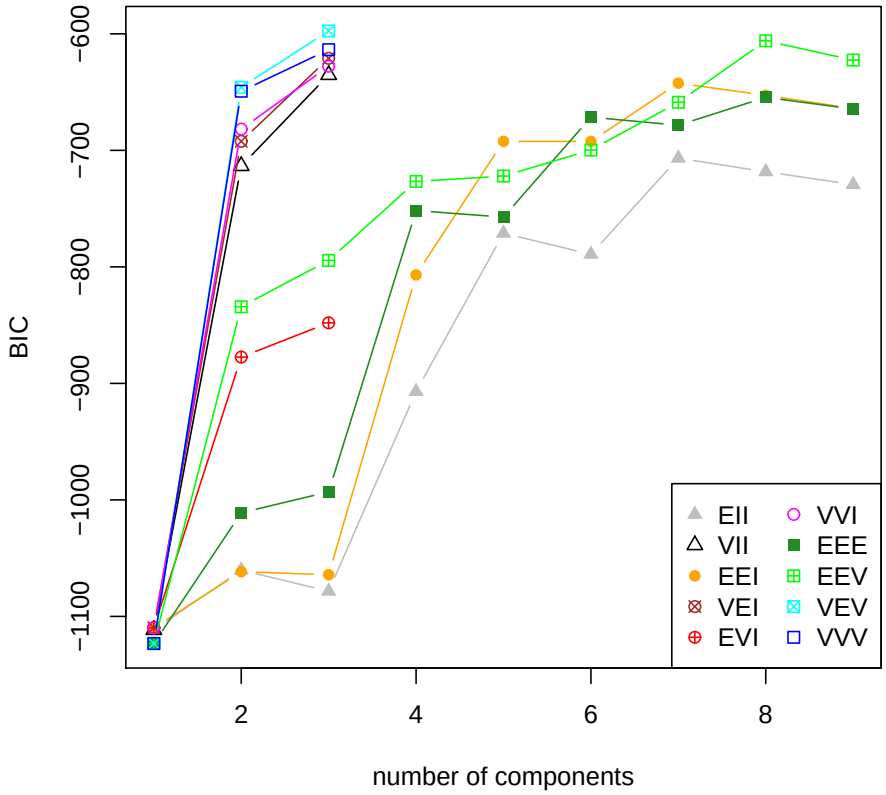

Figure 2.1: Model based clustering for Technology Hardware Sector, year 2005: BIC values and model selection.

The evaluation of the firm's ability to refund financial debts, expressed by Debt Coverage and Cost of Debt, is captured by the components extracted in the whole period. The asset turnover is relevant in 2005, less in 2006 and 2007. In 2005, the data have been fitted by a three-components mixture of Gaussian distributions, connected with a VVI,3 model, thus a diagonal, varying volume and shape model. Cluster 1 with about 56% of the observations, cluster 2 with 13 % and cluster 3 with 31%. Cluster 1 presents low Asset Turnover and an high indebtedness, even if better levels of economic return. Cluster 2 shows an high level of indebtedness with low level of turnover and economic return. Cluster 3 presents average levels of economic return and a good level of turnover,

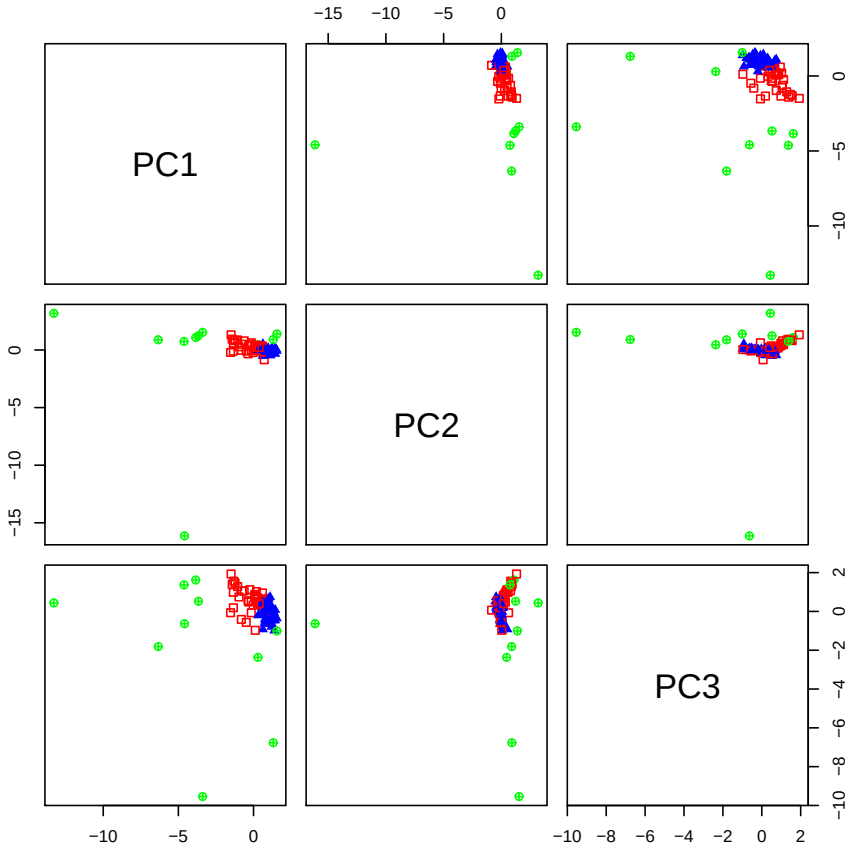

Figure 2.2: Model based clustering for Technology Hardware Sector, year 2005: Pairs plot of model based classification of the data.

even if characterized by high indebtedness level. In 2006, the data have been fitted by a four-components mixture of Gaussian distributions, connected with a VEI,4 model, thus a diagonal, equal shape model. Cluster 1 with about 23% of the observations, cluster 2 with 63 %, cluster 3 with 7%, and cluster 4 with 7%. Cluster 1 presents good levels of economic return and an average level of indebtedness. Cluster 2 shows economic returns and indebtedness levels on average. Both cluster 3 and cluster 4 represent marginal groups with few elements. In 2007, the data have been fitted by a three-components mixture of Gaussian distributions, connected with a VEV,3 model, thus an ellipsoidal, equal shape model. Cluster 1 with about 5% of the observations, cluster 2 with 41 % and cluster 3

Table 2.3: Industry sector analysis

Technology Hardware	Cluster 1	Cluster 2	Cluster 3	Cluster 4
2005	High econ. returns	Low econ. returns-	Low econ. returns- -	
	High indebtedness	Medium indebt.	High indebtedness+	
	High turnover	Low turnover	Low turnover	
2006	High econ. returns	Low econ.returns-	Low econ. returns	Low econ. returns-
	High indebtedness	Medium indebt.	High indebtedness	High indebtedness
	High turnover	Medium turnover	Medium turnover	Low turnover
2007	Medium econ. ret.	Medium econ. ret.	Low econ. returns-	Low econ. returns- -
	Medium indebt.	Low indebtedness	High indebtedness	Not coherent
	Medium turnover	Medium turnover	High turnover	
Constructions	Cluster 1	Cluster 2	Cluster 3	Cluster 4
2005	High econ. returns	Low econ. returns	Medium econ. ret.	
	High indebtedness--	High indebt.++	High indebtedness	
2006	High econ. returns	Medium econ. ret.	Low econ. returns	Low econ. returns--
	Medium indebt.	Medium indebt.	High indebtedness	High indebt.++
2007	Low econ.returns	High econ. returns	Medium econ. ret.	
	High indebt.++	Medium indebt.	Medium indebt.	
	Low refund. capab.-	Medium ref. capab.	Low refund. capab.	

with 54 %. Cluster 1, a marginal group, with high leverage. Cluster 2 shows medium level of indebtedness and high economic returns. Cluster 3 presents an average economic and financial situation. The identification, in both three years, of marginal groups with dispersed elements, could be interpreted as a signal of the presence of potential outliers. These findings have been detected both in the first and the second dataset. In order to compare the methodology presented in the paper, we have processed on the two datasets for the three years, 2005-2007, a K-means clustering, and we found, for all the runs, classifications dissimilar to the ones obtained in the model based clustering. We observed that the model based methodology detects more clusters and one or two residual clusters compared to the K-means procedure that often tends to identify fewer larger cluster and some singletons.

2.3.3 Results and comments

In this part of a more extended company analysis framework, we have considered a two-ways data, and we have followed a sequential procedure by applying firstly the PCA and then the clustering to the scores, a procedure usually used in literature.
The results provided by our analysis show that a clustering procedure applied on a specific industrial sector could report a segmentation according to the financial and economic level of the companies providing a "'scenario"' analysis.
By applying this procedure, it would be possible to have "'financial level"' information on the analysed sector and a segmentation of it with the correspondent average key-indicators values. Next analysis could be processed according to the reported classification of the companies belonging to a specific level-class. For example, a supervised analysis could be applied taking into consideration the specific segmentation of the industrial sector to be analysed, by considering the presence of a specified number of classes to be predicted according to the financial-level classes.
Our intent is both to extend the analysis to other industrial sectors and to consider a different procedure, consisting in the simultaneous combination of dimensionality reduction and clustering operation, see [22].

The results may suggest that model based clustering is a more flexible methodology, compared to the other clustering methods, because every unit is assigned to every of the n-group with a probability of belonging (the posterior probability), giving the possibility to better identify borderline unit. In our application, on average, we found that model based clustering tends to detect more clusters than K-means. Similar findings have been, also, found in previous papers, see [23], where MCLUST is compared to another robust clustering method. The identification of the number of the Gaussian components of the mixture with the number of clusters may need further analysis in order to verify possible misleading association, signalled by the presence of components with few elements, or units with not very high posterior probability of belonging and not very well separated groups, that could be connected with the merging problem of normal distributions or not Gaussian distributions. Moreover, both dispersed few elements group or very low probability of belonging of an element to a cohesive group could indicate the presence of potential outliers. We intend to proceed with further research in order to provide a more robust model based approach for clustering, by considering mixture of *t* distributions instead of Gaussian mixture, see [24] or other robust clustering methodology.

Part II

Supervised statical methodologies

Chapter 3

Supervised methods

3.1 Introduction

Supervised methodologies belong to the confirmatory data analysis procedures, whereas unsupervised methods belong to exploratory procedures.
In supervised estimation we are provided a collection of outputs with given class (classification) or given values (continuous outputs, regression) patterns, and the problem consist in assigning a predicted label or value to a new pattern. The given labelled or valued patterns are used to learn the underlying data structure and the functional dependencies between the output variable *t* and the inputs/predictors set **x**. So, each object is described in terms of a feature vector **X**, belonging to a suitable space and has a true unknown class or value *t* apart from a *learning* or *training set* D = ((x_1, t_1),(x_2, t_2), ..., (x_n,t_n)) where these outputs are known and from whom we derive association rules.
If we consider the feature vectors **X** of i.i.d. random vectors according to a probability law f(x), the learning process could be summarized as the estimation of an output *t* according to a conditional probability $f(t|x)$, fixed but unknown, where x belongs to the class *t*.
Supervised methods are applied to banking application for credit scoring, fraud detection, medical diagnosis and speech recognition.
The following chapter presents two supervised methodologies, in Section 3.2 the **Logistic regression** model and in Section 3.3 the **Neural networks**.

Section 3.4 concerns the **evaluating rules** regarding both a presentation of the model selection criteria and of the accuracy measures used for evaluating classifier performance. In Section 3.5, it is presented a case study on a comparative analysis between **Prediction models: Logit versus Neural Networks**. Section 3.6 refers to the **Class imbalance** problem connected with bankruptcy prediction analysis, and in Section 3.7 it is presented a case study on **Neural Networks distress prediction model on unbalanced dataset**.

3.2 Logistic regression

Over the last decades, in many fields and in particular in credit scoring procedures, the logistic regression model has been considered a reliable method of analysis when the outcome variable is discrete (binary or dichotomous).
The specific form of the logistic regression model is:

$$\pi(\mathbf{x}) = \frac{\exp(\mathbf{x}'\beta)}{1 + \exp(\mathbf{x}'\beta)} \tag{3.1}$$

where $\pi(\mathbf{x}) = \mathbb{E}[Y|\mathbf{x}]$ represents the conditional mean of output response vector Y given the predictor variables vector $\mathbf{x}$. We express the value of the outcome variable, coded as 0 or 1, given $\mathbf{x}$, as $Y = \pi(\mathbf{x}) + \varepsilon$. Here, the quantity ε may assume one of two possible values. If y=1 then $\varepsilon = 1 - \pi(x)$ with probability $\pi(x)$, and if $\mathbf{y} = 0$ then $\varepsilon = -\pi(x)$ with probability 1 - $\pi(x)$. Thus, ε has a distribution with mean zero and variance equal to $\pi(x)(1 - \pi(x))$. That is, the conditional distribution of the outcome variable follows a binomial distribution with probability given by the conditional mean, $\pi(\mathbf{x})$ and maximum-likelihood methods are required for the parameters estimation. Furthermore, assume that the outcome variable has been coded as 0 or 1, representing the absence or the presence of the characteristic, respectively. Consider a collection of p independent variables denoted by the vector $\mathbf{x}' = (x_1, x_2, \cdots, x_p)$. Let the conditional probability that the outcome is present be denoted by $P(Y = 1|\mathbf{x}) = \pi(\mathbf{x})$.
A transformation of $\pi(x)$ is the logit transformation[1] defined as:

$$g(\mathbf{x}) = \ln \frac{\pi(\mathbf{x})}{1 - \pi(\mathbf{x})} = \mathbf{x}'\beta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \tag{3.2}$$

The importance of this transformation is that $g(\mathbf{x})$ has many of the desirable properties of a linear regression model.[2]
Assume that we have a sample of n independent observations $(\mathbf{x}_i, y_i)$, $i = 1, 2, \cdots, n$. Fitting the model requires that we obtain estimates of the vector $\beta' = (\beta_0, \beta_1, \cdots, \beta p)$. The method of estimation used is the maximum likelihood.

[1] In the logistic regression model the link function is the logit transformation.
[2] The logit $g(\mathbf{x})$ is linear in its parameter, may be continuous and may range from $-\infty$ to $+\infty$ depending on the range of $\mathbf{x}$.

The likelihood function expresses the probability of the observed data as a function of the unknown parameters,

$$l(\beta) = \prod_{i=1}^{n} \pi(\mathbf{x}_i)^{y_i}(1 - \pi(\mathbf{x}_i))^{1-y_i} \tag{3.3}$$

Or expressed as log likelihood, it is defined as

$$\mathcal{L}(\beta) = ln(l(\beta)) = \sum_{i=1}^{n} \{y_i ln\left[\pi(\mathbf{x}_i)\right] + (1 - y_i) ln\left[1 - \pi(\mathbf{x}_i)\right]\} \tag{3.4}$$

To find the value of β that maximizes $\mathcal{L}(\beta)$, we differentiate $\mathcal{L}(\beta)$ with respect to β and set the resulting expressions equal to zero. There will be p+1 likelihood equations that are obtained by differentiating the log likelihood function with respect to the p+1 coefficients. The likelihood equations that result may be expressed as follows:

$$\sum_{i=1}^{n} \left[y_i - \pi(\mathbf{x}_i)\right] = 0 \tag{3.5}$$

and

$$\sum_{i=1}^{n} x_{ij} \left[y_i - \pi(\mathbf{x}_i)\right] = 0 \tag{3.6}$$

for $j = 1, 2, \cdots, p$.

Let $\hat{\beta}$ denotes the solution to these equations, thus the fitted values for the multiple logistic regression model are $\hat{\pi}(\mathbf{x}_i)$, computed using $\hat{\beta}$ and $\mathbf{x}_i$.

3.3 Neural networks

An artificial neural network (ANN) is a statistical model for estimation of non-linear dependencies between input variables and output response vector.[3]

The original research in neural networks was inspired and motivated by biological studies focused on understanding and modelling the structure of the brain functions. The first structure of learning machine was the *perceptron* proposed by Rosenblatt (1962) for pattern recognition, [133].[4]

The first result that links the generalization properties of the perceptron and the error minimization on the training sample was provided by Novikoff (1962), [135].

Afterwards, Vapnik (1982), [136], formulated an inferential paradigm based on the inductive principle of the *empirical risk minimization*.[5]

Neural network architecture is based on the linear combination of activation functions, and could be represented by a graph where patterns are m-dimensional vectors assigned to graphs nodes (input and output) and the transformations between patterns are operated by means of optimization algorithms. The choice of the activation function $\tau(.)$ depends on the choice of the output distribution (the $\tau(.)$ function codomain).

For example, the *identity activation function*:

$$\tau(z) = z \tag{3.7}$$

does not operate any transformation to the input values.

It is appropriate for output variable not constrained.

When referring to sigmoid activation functions[6], we generally considered the *logistic ac-*

[3]For further details see [30], [31].

[4]The Rosenblatt perceptron consists in an architecture, according to McCulloch-Pitts model (1943), [134], with a single neuron which takes m inputs, $\mathbf{x} = (x_1, \cdots, x_m)$ and delivers one output y $\in \{-1, 1\}$. The input-output relation is given by $g(\mathbf{x}) = sign(\mathbf{w}'\mathbf{x} - w_0)$ where $\mathbf{w} \in \mathcal{R}^m$ and $w_0 \in \mathcal{R}$ are the neuron coefficients named weights and threshold, respectively. The sign(.) is a function as sign(u)=1 if $u>0$ and $sign(u) = -1$ if $u<0$.

[5]Subsequently, Vapnik & Chervonenkis (1991) demonstrated the necessary and sufficient conditions for the consistency of the principle of empirical risk minimization, [137].

[6]Another sigmoid function is $\tau(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, assuming values in (-1, 1).

tivation function:

$$\tau(z) = \frac{1}{1+e^{-z}} \tag{3.8}$$

assuming values in (0,1). It is appropriate for dichotomous (0/1) output variables where the expected value is a probability.

Neural networks are families of models with large but not unlimited flexibility given by a large number of parameters.[7] The two most widely used neural networks architectures, belonging to this family, are *multi-layer perceptrons* (MLP) and *radial basis functions* (RBF).[8] Multi-layer perceptron is used in supervised learning for forecasting and classification. A feed-forward network is a network in which vertices can be numbered so that all connections go from a vertex to one with a higher number. The vertices are arranged in layers, with connections only to higher layers. In feed-forward networks the information moves towards only one direction, from a level to the following one, without feedback loops. If the connections are bi-directional, we refer to feedback networks.

Let $(\mathbf{X}, Y)$ be a pair of random vector $\mathbf{X}$ and a random variable Y with joint probability distribution $p(\mathbf{x}, y)$, where $\mathbf{X}$ is the m-dimensional input vector (predictor variables) assuming values in some space $\mathcal{X} \subseteq \mathbb{R}^m$ and Y is a response variable with values in $\mathcal{Y} \subseteq \mathbb{R}$. We assume that the input-output relation can be written as $Y = \phi(\mathrm{x}) + \varepsilon$, where ε is a random variable with zero mean and finite variance. We then assume that the unknown functional dependency $\phi(\mathrm{x}) = \mathbb{E}[Y|\mathrm{x}]$ is estimated by means of the function $f_p(\mathbf{x})$ realized by an MLP with m inputs, p neurons in the hidden layer and one neuron in the output,

$$f_p(\mathbf{x}) \;=\; \sum_{k=1}^{p} c_k \tau(\mathbf{a}_k'\mathbf{x} + b_k) + c_0 \tag{3.9}$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_p \in \mathbb{R}^m$, $b_1, \ldots, b_p, c_{p+1}, c_1, \ldots, c_p \in \mathbb{R}$ and $\tau(.)$ is a sigmoidal function.[9] We denote by $\mathbf{A}$ the p x m matrix with rows $\mathbf{a}_1', \ldots, \mathbf{a}_p'$ and set $\mathbf{b} = (b_1, \ldots, b_p)$ and

[7] The traditional methods of statistics and pattern recognition are either *parametric* based on a family of models with a small number of parameters, or *non parametric* in which the models used are totally flexible.

[8] The main difference between RBF and MLP relates to the activation function in the hidden nodes. In MLP, the activation function is a linear combination of inputs and node-weights. In RBF, it is a function of the distance between the input vector and the reference vector of the jth node.

[9] Logistic activation function or hyperbolic tangent.

$\mathbf{c} = (c_1, \ldots, c_p)$,. Such quantities are called *weights* and we denoted them by $\mathbf{w}$, so that $\mathbf{w} \in \mathbb{R}^{p(m+2)+1}$.

Let $\mathcal{F}_p$ be the set of all functions of type (3.9) for a fixed p, for $1 \leq p \leq N$, referring to $\mathcal{F}$ for simplicity of notation. The problem is to find the function $f^{(0)} = f(\mathbf{w}^{(0)})$ in the set $\mathcal{F}$ such that the *generalization error* (or *expected risk*),

$$\widehat{\mathcal{E}}(f) = \int [y - f(\mathbf{x})]^2 \, p(\mathbf{x}, y) d\mathbf{x} dy \tag{3.10}$$

where the integral is over $\mathcal{X}$ x $\mathcal{Y}$, attains its minimum,

$$f^0 = \arg\min_{f \in \mathcal{F}} \widehat{\mathcal{E}}(f) \tag{3.11}$$

and $\mathbf{w}_0$ denotes the weights of f^0. The distribution $p(\mathbf{x}, y)$ is unknown, so we compute, from the sample $\mathcal{L}$=$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ (called the learning set: of N i.i.d. realizations of $(\mathbf{X}, Y)$), the *empirical error*:

$$\widehat{\mathcal{E}}(f, \mathcal{L}) \;\; = \;\; \sum_{(\mathbf{x}_n, y_n) \in \mathcal{L}} (y_n - f(\mathbf{x}_n))^2 \tag{3.12}$$

and estimate the least squares parameters by minimizing (3.12).[10] If a sum-of-squares error is used, however, the quantities which can be determined are the x-dependent mean of the distribution (given by the outputs of the trained network) and a global average variance (given by the residual value of the error function at its minimum).[11] The sum-of-squares error does not represent the only one error function. There exist other error functions (for example, entropy for classification problem, [30]).

If we consider a sample $\mathcal{D}$, generally the procedure consists in partitioning it in three independent sub-samples: a *learning set* (or training set) $\mathcal{L}$, a *validation set* $\mathcal{V}$ and a *test set* $\mathcal{T}$. The *learning set* $\mathcal{L}$ is a set of examples used to fit the parameters of the model. To estimate the parameters of the model, we refer to the *learning error*. The *validation*

[10] We refer to the principle of *empirical risk minimization*. The sum-of-squares error function is derived from the principle of maximum likelihood on the assumption of Gaussian distributed target data. Otherwise, the use of a sum-of-squares error does not require the target data to have a Gaussian distribution, [30].

[11] If we consider the conditional distribution $p(y|\mathbf{x})$, and we specify the hypothesis of normality, homoskedasticity and not correlation, then the minimization of the sum-of-squares error conducts to the maximum likelihood estimation for the network weights.

set $\mathcal{V}$ is a set of examples used to tune the parameters of the model (e.g., to choose the number of hidden nodes of the neural network). To select the model with the best generalization properties, the approach would consist in evaluating the error function on the validation set (*validation error*). The *test set* $\mathcal{T}$ is a set of examples used only to assess the performance of a fully specified model, by referring to the *test error.* However, very often the sample $\mathcal{D}$ is split into only two sub-groups, the learning set $\mathcal{L}$ and the test set $\mathcal{T}$.[12] Both the learning and test set are used to estimate parameters $\hat{\mathbf{w}}^{(0)}$, because $\widehat{\mathcal{E}}^*(f;\mathcal{L})$ is the function to be minimized and $\widehat{\mathcal{E}}^*(f;\mathcal{T})$ is the function used to control overfitting. As the learning process proceeds, at the beginning both the learning and the test errors generally decrease, but at a certain point the test error begins to increase even thought the learning error is still decreasing. It is judged the point in which the overfitting occurs. The learning process is then stopped, and the current estimate of the weights is chosen to be $\hat{\mathbf{w}}^{(0)}$. This technique is named *early stopping.* If the test error never decreases during the training process then the network is considered under-parametrized. Otherwise, a local minimum is achieved by the optimization algorithm and initialization parameters should be changed. The *early stopping* technique represents a regularization method, because if starting values are small in magnitude and the weight increase as the learning process proceeds, then stopping the training before convergence force the weights to remain small. When the data are not split into three different sets and validation error cannot be computed for estimating the generalization error, model selection criteria are used to compare different networks.

Neural network models can suffer from either underfitting or overfitting. A network that is not sufficiently complex can fail to detect fully the signal in a complicated data set, leading to underfitting. A network that is too complex may fit the noise, leading to overfitting. The complexity of a network is related to both the number of weights and the size of the weights. The generalization performance depends more on the size of the weights than on the size of the networks (number of parameters). Large weights cause the sigmoids to saturate, and this leads to quite irregular surfaces. A regularisation method to smooth the size of the weights is the *weight decay*, where a penalty term is added to the error function

[12] For further details about the procedure implemented in NNET R statistical package, see Appendix B.

to be minimized during the learning process over the sum of all weights of the network:

$$\widehat{\mathcal{E}}^*(f;\mathcal{L}) = \widehat{\mathcal{E}}(f;\mathcal{L}) + \lambda \sum w_i^2$$

where λ is the *decay* or *smoothing* parameter. An approach consists in processing networks with different λ values and estimating the correspondent generalization error, so to select the λ parameter related to the minimum error value.

Optimization algorithm From the numeric point of view, the problem of learning in neural networks consists in finding a weight vector $\mathbf{w}_{(opt)}$ that minimize the error function $\widehat{\mathcal{E}}(\mathbf{w})$. For neural networks in general form, in particular those with more than one layer of adaptive weights, the error function will typically be a highly non-linear function of the weights, and there may exist many local minima. As a condition of the non-linearity of the error function, it is not in general possible to find a closed-form solutions for the minima, and iterative optimization algorithms are required. These algorithms involve a search through weight space consisting of a succession of steps of the form

$$\mathbf{w}^{(s+1)} = \mathbf{w}^{(s)} + \delta\mathbf{w}^{(s)} \tag{3.13}$$

where s labels the iteration step. Different algorithms involve different choices for the weight vector increment $\delta\mathbf{w}^s$. For some algorithms, such as conjugate gradients and the quasi-Newton algorithms, the error function is guaranteed not to increase as a result of a change to the weights. One potential disadvantage of such algorithms is that if they reach a local minimum they will remain there. The choice of initial weights for the algorithm then determines which minimum the algorithm will converge to. The majority of initialization procedures involve setting the weights to randomly chosen small values.[13]
The simplest network training algorithm is the *gradient descent* (also known as *steepest descent*). This method and the *back-propagation* algorithm are currently considered

[13]The initial weights values are chosen to be small so that sigmoid activation functions are not driven into the saturation regions but not too small in order not to lead to slow training. In literature, it has been suggested that the summed inputs to the sigmoid functions should be of order unity. The weights in the hidden layer are generated by a normal distribution having zero mean and and σ^2. The choice of variance σ^2 is important. It is generally suggested that the standard deviation of the distribution used to generate the initial weights should scale like $\sigma \propto m^{1/2}$, [30].

inefficient at some extent and, at the moment current, moment analysed also from the theoretical perspective. There are other currently processed optimization methods. If we consider a small number of parameters, we refer to *Newton* and *Gauss-Newton*, included *Levenberg-Marquandt.* If we consider a large number of parameters, we refer to the *conjugate gradient* algorithm. In the intermediate case, we refer to the Quasi-Newton methods. Then, we provide an overview referring only to the above-mentioned currently used algorithms.

In the ***Conjugate gradient*** algorithm, the search directions are orthogonal to each other with respect to the scalar product $(Hd^{(s)}, d^{(s+1)})$, or H-conjugate, that is

$$d^{(s+1)} H d^{(s)} = 0 \tag{3.14}$$

where H is the Hessian matrix evaluated at the point $\mathbf{w}^{(s+1)}$.[14] Search directions which satisfy (3.14) are said to be conjugate. It is possible to construct a sequence of successive search directions $d^{(s)}$ such that each direction is conjugate to all previous directions, up to dimensionality W of the search space.

In the ***Newton's method***, if we refer to local quadratic approximation, we consider a search direction based on the inverse of the Hessian of the error function,

$$d^{(s)} = -H^{-1} \nabla \widehat{\mathcal{E}}_{(s)} \tag{3.15}$$

where the vector $H^{-1}\nabla\widehat{\mathcal{E}}_{(s)}$ is known as *Newton direction* or *Newton step.*[15]

The weight vector $\mathbf{w}^*$ corresponding to the minimum of the error function satisfies,

$$\mathbf{w}^* = \mathbf{w} - H^{-1} \nabla \widehat{\mathcal{E}} \tag{3.16}$$

The exact evaluation of the Hessian for non-linear networks is computationally demanding, since it requires $O(NW^2)$ steps and $O(W^3)$ steps for the computation of its inverse.

The ***Quasi-Newton's methods*** are based on the Newton direction and involve generating a sequence of matrices $G^{(s)}$ which represent increasingly accurate approximations to the inverse Hessian H^{-1}, using information on the first derivatives of the error function. The

[14] The successive search direction $d^{(s)}$ are chosen such that, at each step of the algorithm, the component of the gradient parallel to the previous search direction is unaltered.

[15] The *Newton direction* forms the basis of a variety of optimization strategies.

problems arising from Hessian matrices which are not positive definite are solved by staring from a positive-definite matrix and ensuring that the update procedure is such that the approximation to the inverse Hessian is guaranteed to remain positive definite.
If we express,

$$g = \nabla\widehat{\mathcal{E}} = H(\mathbf{w} - \mathbf{w}^*) \tag{3.17}$$

with w^* corresponding to the minimum of the error function.
From the Newton direction formula (3.16), we consider that the weight vectors at step s and s+1 are related to the corresponding gradients by

$$\mathbf{w}^{(s+1)} - \mathbf{w}^{(s)} = -H^{-1}(g^{(s+1)} - g^{(s)}) \tag{3.18}$$

which is known as the Quasi-Newton condition. The approximation G of the inverse Hessian is constructed so as to satisfy this condition also.
The two most commonly used update formulae are the *Davidson-Fletcher-Powell* (DFP) and the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) procedures. Here, we give only the BFGS expression,[16]

$$G^{(s+1)} = G^{(s)} + \frac{pp^T}{p^T v} - \frac{(G^{(s)}v)v^T G^{(s)}}{v^T G^{(s)} v} + (v^T G^{(s)} v)uu^T \tag{3.19}$$

where we define the following vectors:

$$p = \mathbf{w}^{(s+1)} - \mathbf{w}^{(s)} \tag{3.20}$$

$$v = g^{(s+1)} - g(s) \tag{3.21}$$

$$u = \frac{p}{p^T v} - \frac{G^{(s)}v}{v^T G^{(s)} v} \tag{3.22}$$

At each step of the algorithm, the direction -Gg is guaranteed to be a descent direction, since the matrix G is positive definite. However, the weight vector is updated using,

$$\mathbf{w}^{(s+1)} = \mathbf{w}(s) + \alpha^{(s)} G^{(s)} g^{(s)} \tag{3.23}$$

where $\alpha^{(s)}$ is found by line minimization.
An advantage of the Quasi-Newton method approach over the Conjugate gradient is that

[16] BFGS method is generally regarded as superior. In NNET R Statistical package, BFGS is the method implemented for optimization.

the line search does not need to be performed with great accuracy since it does not form a critical factor in the algorithm. The disadvantage stands on the computational storage requirement.

The ***Levenberg-Marquardt*** algorithm is specifically designed for minimizing a sum-of-squares error.

Consider the sum-of-squares error function,

$$\mathcal{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\epsilon_n^2 = \frac{1}{2}\|\epsilon\|^2 \tag{3.24}$$

where ϵ_n is the error for the nth pattern, and e is a vector with elements ϵ_n. The elements of the Hessian matrix take the form,

$$(\mathbf{H})_{ik} = \frac{\partial^2\mathcal{E}}{\partial\mathbf{w}_i\partial\mathbf{w}_k} = \sum_{n=1}^{N}\left\{\frac{\partial\epsilon_n\partial\epsilon_n}{\partial\mathbf{w}_i\partial\mathbf{w}_k} + \epsilon_n\frac{\partial^2\epsilon_n}{\partial\mathbf{w}_i\partial\mathbf{w}_k}\right\} \tag{3.25}$$

If we neglect the second term, the Hessian can be written in the form

$$\mathbf{H} = \mathbf{Z}^T\mathbf{Z} \tag{3.26}$$

where we have defined the matrix $\mathbf{Z}$ with elements,

$$(\mathbf{Z})_{ni} = \frac{\partial\epsilon_n}{\partial w_i} \tag{3.27}$$

In the Levemberg-Marquand algorithm, the variation $\Delta\mathbf{w}$, at each step, is expressed by

$$\mathbf{w}^{(s+1)} = \mathbf{w}^{(s+1)} - (\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^T\epsilon^{(s)} \tag{3.28}$$

where $\mathbf{Z}$ is computed on the basis of the errors $\epsilon^{(s)}$ at s-th step, the parameter λ governs the step size, and $\mathbf{I}$ is the unit matrix.

Interpretive methods Artificial neural networks (ANNs) are generally referred as ”black box” procedures, not disclosing the relation between the explicative variables and the dependent variable and the interpretation of the weights of the network or the activation values in the hidden layers with respect to the set of data analysed. In ANNs field, a

particular research direction is focused to the implementation of procedures dedicated to understand the nature of the internal mechanism to transpose information in the activation process and to interpret the effect of the inputs variables on the output.

These methods can be principally grouped into two main groups: *magnitude of weights* ans *sensitivity analysis.*[17]

Analysis based on the *magnitude of the weights* refers to those procedures based on the values stored in the static matrix of weights to determine the relative influence of each input variable on each one of the network outputs.

The equations proposed for weights magnitude analysis are characterized by the calculation of the product of the weights w_{ij} (connection weight between input neuron i and hidden neuron j) and v_{jk} (connection weight between hidden neuron j and output neuron k) for each hidden neurons, obtaining the sum of the calculated product. For example, Garson (1991), [138], proposed

$$Q_{ik} = \frac{\sum_{j=1}^{L} \left(\frac{w_{ij}}{\sum_{r=1}^{N} w_{rj}} v_{jk} \right)}{\sum_{i=1}^{N} \left(\sum_{j=1}^{L} \left(\frac{w_{ij}}{\sum_{r=1}^{N} w_{rj}} v_{jk} \right) \right)} \tag{3.29}$$

where $\sum_{r=1}^{N} w_{rj}$ is the sum of the connection weights between the N input neurons and the hidden neuron j, and Q_{ik} represents the percentage of influence of the input variable x_i on the output y_k, in relation to the rest of the input variables, so that the sum of this index returns the 100% for all the input variables.[18]

Tchaban et al. (1998), [141], proposed another sensitivity measure,

$$S_{ik}^{T} = \frac{x_i}{y_k} \sum_{j=1}^{L} w_{ij} v_{jk} \tag{3.30}$$

[17] Montano et al. (2003) defined a different procedure NSA(Numeric sensitivity analysis) to analyse the effect of input variable on output, based on the calculation of the slopes that are formed between the inputs and the outputs, without assumptions about the nature of the variables included (quantitative or qualitative), [32].

[18] But Garson's formula does not take into account the signs of the weights, so weights with opposite signs can cancel each other out.

representing a variant to the Garson's equation.

However, in literature the analysis based on the magnitude of weight is not considered effective whereas *sensitivity analysis* procedures have been more implemented.

Sensitivity analysis could be split into two directions: *analysis based on the network output* and *analysis based on the error function.*

Sensitivity analysis is based on the measurement of the effect that is observed in the output y_k due to the change produced in the input x_i. The analytical version of sensitivity analysis starts from the *Jacobian matrix*, whose elements are given by the derivatives of the network outputs with respect to the inputs, $J_{ki} \equiv \frac{\partial y_k}{\partial x_i}$, where each such derivatives is evaluated with all other inputs held fixed.[19] The Jacobian matrix could provide a measure of the local *sensitivity* of the outputs to changes in each of the inputs variables, [30].

Zurada et al. (1997), [139], proposed a measure for sensitivity S_{ik} of the output y_k due to changes in the input variable y_k, based on Jacobian matrix, expressed as,

$$S_{ik} = \frac{\partial y_k}{\partial x_i} = f'(net_k) \sum_{j=1}^{L} v_{ik} f'(net_j) w_{ij} \tag{3.31}$$

where $f'(net_k)$ and $f'(net_j)$ are the derivative of the activation function of the hidden neuron j and the output neuron k, respectively.[20]

As above-mentioned, sensitivity analysis could be applied to the effect observed in the error function, provoking a perturbation in the input. Wang et al. (2000), [140], consisting of comparing the error made by the network from the original patterns with the error made when restricting the input to a fixed value (in general the average value) for all patterns. Thus, the greater the increase in the error function upon restricting the input the greater the importance of the input on the output.

[19] The term Jacobian matrix is also used to refer to the derivatives of the error function with respect to the network weights, as calculated using back-propagation.

[20] This method is limited to networks presenting quantitative variables.

3.4 Evaluating rules

Classifier performance may be assessed for two basic reasons: to *compare* classifiers, in order to select the best one, or to determine an *absolute measure of quality* of performance, in order to verify the adequacy of the classifier to the problem.
In this chapter, in **Sub-section 1** the commonly criteria used for selecting a model over another are depicted and in **Sub-section 2** an overview of main evaluating metrics for goodness of classifier is presented, in particular considering the performance evaluation in the case of unbalanced datasets.

3.4.1 Model selection criteria

It is presented an overview of the selection criteria used in literature to compare statistical models and select the optimal one. It is important to notice that when using these model selection criteria in the case of a MLP (Multilayer perceptron), the number of K degree of freedom is set equal to the number W of weights, that means W=p(m+2)+1, where *p* indicated the hidden nodes and *m* the input vector dimensionality.[21]
Recalling the functional dependency $\phi(\mathbf{x}) = \mathbb{E}[Y|\mathbf{x}]$, for a fixed p and $1 \leq p \leq N$, let $\mathcal{F}$ be the set of all functions of kind:

$$f_p(\mathbf{x}_n) \;=\; \sum_{i=1}^{p} c_i \tau(\mathbf{a}_i' \mathbf{x}_n + b_i) + c_{p+1} \tag{3.32}$$

where $i = 1, ..., N$, $\mathbf{a}_1, \ldots, \mathbf{a}_p \in \mathbb{R}^m$, $b_1, \ldots, b_p, c_{p+1}, c_1, \ldots, c_p \in \mathbb{R}$ and τ is a sigmoidal function. The problem is to find the function $f^{(0)} = f(\mathbf{w}^{(0)})$ in the set $\mathcal{F}$ such that the *generalization error*:

$$\mathcal{E}(f) \;=\; \int [y - f(\mathbf{x})]^2 \, p(\mathbf{x}, y) \, d\mathbf{x} \, dy \;, \tag{3.33}$$

where the integral is over $\mathcal{X} \times \mathcal{Y}$, attains its minimum, that is $f^{(0)} = \arg\min_{f \in \mathcal{F}} \mathcal{E}(f)$ and $\mathbf{w}^{(0)}$ denotes the weights of $f^{(0)}$. In practice, the distribution $p(\mathbf{x}, y)$ is unknown, but we

[21] S. Ingrassia et I. Morlini investigated the case of a neural network for small dataset, in which W > N. They referred to the *equivalent number of degree of freedom* by setting K=p+1, see [98].

have a sample $\mathcal{L} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, called learning set, of N i.i.d. realizations of $(\mathbf{X}, Y)$ so that we can compute the *empirical error*:

$$\widehat{\mathcal{E}}(f, \mathcal{L}) \quad = \quad \sum_{(\mathbf{x}_n, y_n) \in \mathcal{L}} (y_n - f(\mathbf{x}_n))^2 \tag{3.34}$$

In the general framework of model selection, we suppose there are $f_{p_1}, \ldots, f_{p_K}$ models of the form (3.6). As the estimation in statistical models may be thought of as the choice of a single value of the parameter chosen (according to some criterion) to represent the distribution, model selection may be thought of in this framework as the estimation applied to the model f_{p_h}, with $h = 1, \ldots, K$. The only special issue is that the set of models is discrete and has finite range. There may be occasions when one model clearly dominates the others and the choice is unobjectionable, and other occasions when there are several competing models that are supported in some sense by the data. Due to the *unidentifiability* of the parameters, there may be no particular reasons for choosing a single best model over the others according to some criterion. On the contrary, it make more sense to "deselect" models that are obviously poor, maintaining a subset for further considerations regarding, for example, the computational costs.

Let f_k be a statistical model based on K degrees of freedom, N denoting the size of the learning (training) set. Some of the model selection criteria derive from the maximum likelihood and could be referred, generally, to the form:

$$\Pi \quad = \quad \widehat{\mathcal{E}}(f_k) + C_k \tag{3.35}$$

where the term $\widehat{\mathcal{E}}(f_k) = \widehat{\mathcal{E}}(f_k, \mathcal{L})$ is the empirical error of the model f_k based on the learning set and C_k represents a complexity term expressing the penalty connected to the number of the degree of freedom K of the model. The complexity term increases as the number K of degree of freedom grows, so compensating the effect on the selection criteria given by a decrease in the empirical error term.

As follows, an overview of indexes generally used for model selection is presented. They are expressed in a general form so to handle with different kind of problems and be interpreted in several application cases. One of them is the AIC (Akaike 1974):

$$\mathrm{AIC} = \log(\widehat{\mathcal{E}}(f_k)) + \frac{2K}{N}$$

where N is the size of the learning set. Another criterion is the BIC (Bayesian information criterion), according to the minimum value of:

$$\mathrm{BIC} = \log(\widehat{\mathcal{E}}(f_k)) + \frac{K\log(N)}{N}$$

Then the GCV error,

$$\mathrm{GCV} = \widehat{\mathcal{E}}(f_k)\left(1 - \frac{K}{N}\right)^{-2}$$

the UEV criterion

$$\mathrm{UEV} = \frac{\widehat{\mathcal{E}}(f_K)}{N - K}$$

and finally the FPE (Akaike, 1970),

$$\mathrm{FPE} = 6\widehat{\mathcal{E}}(f_k)\left(\frac{1 + K/N}{1 - K/N}\right).$$

where K denotes the number of degrees of freedom of the model f_k. For AIC and BIC there are different forms in literature; here we follow Raftery, see [110]. Some of these criteria obey the likelihood principle, that is they have some frequentist asymptotic justification; some others correspond to a Bayesian decision problem.
In the case-studies presented, we used both AIC, BIC and GCV in order to select the optimal neural network structure.

3.4.2 Evaluation metrics

Classifier performance evaluation is a crucial stage in assessing and developing learning techniques. To evaluate the performance of a classifier different metrics can be applied, each one referring to a part of information in respect to another.

Furthermore, evaluation procedure should be applied to several domains, presenting different characteristics such as dimensionality, type of features, data complexity, to understand the classifier's general behaviour.

Clearly, the main objective of constructing a classification model is to correctly classify as many future instances as possible. The most popular measure of performance a classifier is of *error rate* or *misclassification rate*, and it represents the proportion of objects misclassified by the rule. It will refer to future objects to be classified. *Error rate* is the commonest assessment criterion, probably because it is a default criterion, but it is not necessarily the most appropriate one. In particular, the main weakness of error rate is that it implicitly assumes that the cost of different types of misclassification are equal, and this is unlikely to be in most real applications. Furthermore, error rate does not inform about the misclassification level because it does not distinguish between different instances misclassified in relation to their distance from the threshold chosen. Moreover, error rate gives no information about accuracy of the probability estimate in the classifier. If the threshold is not determined a priori then it would be important to have accurate estimation over a range of potential threshold, by considering the fact of setting different thresholds connected to implicitly varying misclassification costs. In classifier performance evaluation, the most common case is the two-class situation, where π_0 represents the prior probability of class 0, $\pi_1 = 1 - \pi_0$ is the prior probability of class 1, p_0 is the proportion predicted to have come from class 0, $p_1 = 1 - p_0$ the predicted proportion from class 1, and n=a+b+c+d is the overall sample size, as shown in table 3.1.

When the performance criterion is error rate, a *confusion matrix* is the cross-classification

			True	class	
			Positive class	Negative class	
			0	1	
Predicted Class	Positive prediction	0	True positive (a)	False positive (b)	p_0
	Negative prediction	1	False negative (c)	True negative (d)	p_1
			π_0	π_1	n

Table 3.1: Confusion matrix for a two-class problem. Different types of errors and hits.

of the predicted class by the true class. The off-diagonal elements show where the main

misclassification occur. The confusion matrix, based on error rate, is asymmetric because error rate is an asymmetric measure, as the number of elements from class *i* misclassified into class *j* could not equal the number from class *j* misclassified into class *i.* Confusion matrix generally refer to error-rate, even if any measure of the distance between two classes can be used as the basis of confusion matrix.

The *error rate* and the *accuracy* are widely used metrics for measuring the performance of learning systems. However, when the prior probabilities of the classes are very different, such metrics might be misleading.

$$\textbf{Error rate} = E = \frac{c+b}{a+b+c+d} \tag{3.36}$$

$$\textbf{Accuracy} = Acc = \frac{a+d}{a+b+c+d} = 1 - E \tag{3.37}$$

Maybe, *accuracy* is the most common evaluation metric but it is not suitable to evaluate imbalanced datasets since the minority class has much lower precision and recall than the majority class. For instance, it is straightforward to create a classifier having 99% accuracy (or 1% error rate(if the data set has a majority class with 99% of the total number of cases, by simply labelling every new case as belonging to the majority class. Furthermore, these metrics consider different classification errors as equally important. Additional metrics have been proposed from other domains. They are ROC and AUC, F-value, maximum geometric mean (MGM) of the accuracy on the majority class and the minority class, maximum sum (MS) of the accuracy. All the metrics can be divided in two categories: metrics based on confusion matrix directly and that based on accuracy of binary classes or precision and recall directly. Accuracy, precision and recall, FP rate, TP rate, ROC and AUC fall into the first, while F-value and other more complex metrics, such as MGM of the accuracy on the majority class and the minority class, MS, fall into the other.

False negative rate:

$$FN = \frac{c}{a+c} \tag{3.38}$$

is the percentage of positive cases misclassified as belonging to the negative class;

False positive rate:

$$FP = \frac{b}{b+d} \tag{3.39}$$

is the percentage of negative cases misclassified as belonging to the positive class;

True negative rate:

$$TN = \frac{d}{b+d} = 1 - FP \tag{3.40}$$

is the percentage of negative cases correctly classified as belonging to the negative class, also reported as *specificity*;

True positive rate:

$$TP = \frac{a}{a+c} = 1 - FN \tag{3.41}$$

is the percentage of positive cases correctly classified as belonging to the positive class, also reported as *sensitivity* or *recall.*

These four class performance measures have the advantage of being independent of class costs and prior probabilities. It is obvious that the main objective of a classifier is to minimize the false positive and negative rates or, similarly to maximize the true negative and positive rates. In particular, if class 0 represents *cases* and class 1 represents *non cases*, *sensitivity*(Se) and *specificity*(Sp) define performance in terms of predicted classifications within each true classes.

Then, let us underline the two above mentioned measures *recall* and *precision*:

$$\textbf{Precision} = \frac{a}{a+b} \tag{3.42}$$

$$\textbf{Recall} = \frac{a}{a+c} \tag{3.43}$$

Precision of a classification rule is the percentage of times the predictions associated with the rule are correct. As above-mentioned, *recall* is the percentage of all examples belong to class X that are covered by a rule.

For the purpose of comparison, it is convenient to combine *precision* and *recall* into a single measure of performance, the *F-measure*

$$\textbf{F-Measure} = \frac{(1+\beta^2) * recall * precision}{\beta^2 * recall + precision} \tag{3.44}$$

The value of β, a non-negative real, is adjusted according to the importance between precision and recall.

F-value (or F-measure) is high when both recall and precision are high and can be adjusted through changing the value of β, where, in details, β corresponds to the relative importance of precision versus recall.

The more common F-measure (F_1) fixes the β value to 1:

F_1= (2*recall*precision)/(recall+precision).

The F_1 measure lies between zero and one, with values close to one indicating better performance. It is a useful performance metric because it gives low scores to methods that obtain high precision by sacrificing recall or vice versa.

Furthermore, on domains where misclassification cost is relevant, a cost matrix could be used. A cost metrics defines the misclassification cost, and in this case the objective of the classifier is to minimize classification cost instead of error rate.

Perhaps, the most common metric to assess overall classification performance is ROC analysis and the associated use of the area under the ROC curve (AUC). ROC curve is a two-dimensional graph in which TP rate(benefits) is plotted on the y-axis and FP rate(costs) is plotted on the x-axis. To produce a 2x2 confusion matrix as in Table ? it is necessary to settle on a specific threshold, whereas with a receiver operating characteristic (ROC) curve the performance is shown at each threshold. In particular, ROC curve presents simultaneously, for a range of possible classification thresholds for the classifier, the true positive rate(sensitivity) on the vertical axis against false positive rate(1-specificity) on the horizontal axis. Different points in the curve correspond to different thresholds used in the classifier. If we consider the 45° line as a benchmark, the closer the ROC curve is to that line the worse the performance is, because it would mean that it classify the same proportion of the cases and the non-case into the case class at each value of the threshold, that is it would not separate the class at all. On the contrary, the best classifier performance is associated to a ROC curve following the two axis, because it would classify 100% of the cases into the case class and 0% of the non-case into the case class for some threshold points. For most real world applications, there is a trade-off between FN and FP and similarly between TN and TP. Some classifiers have parameters for which

different settings produce different ROC points. A classifier that produces probabilities of an instance to be in each class (for example, neural networks produce continuous outputs that can be mapped to probability estimate) can have a threshold parameter biasing the final class selection. The ROC methodology allows for ranking of examples based on their class memberships, whether a randomly chosen majority class example has a higher majority class membership than a randomly chosen minority class, see [28]. Plotting all the ROC points that can be produced by varying these parameters produces a ROC curve for the specific classifier. Generally, this is a discrete set of points, including (0,0) and (1,1), which are connected by line segments. The lower left point (0,0) represents a strategy that classifies every example as belonging to the negative class and the upper right point represents a strategy that classifies every example as belonging to the positive class. As said before, the point (0,1) represents the perfect classification, and the line x=y represents the strategy of random guessing the class. We would study classifiers dominance relationship by comparing the associated ROC curves. So, if a curve, for a given specificity, has a greater sensitivity the associated classifier provide a superior performance, and vice versa for a given sensitivity. ROC curves obtained by data are not smooth but step functions. There is a strong connection between the prior probability of a class and its error cost.
If the costs of misclassifying class 0 (c_0) and class 1 (c_1) are known, it is possible to compute the **Total cost**,

$$\pi_0 c_0(1 - Sensitivity) + \pi_1 c_1(1 - Specificity) \tag{3.45}$$

It could be found the threshold on the ROC curve that minimizes this cost, where the curve slope equals $s = (\pi_1 c_1)/(\pi_0 c_0)$.
Alternatively, if error costs are unequal and known, then we can adjust the decision threshold to minimize the overall cost of errors. Two curves could have the same performance if they intersect in a point, corresponding to a specific threshold. Generally, one curve dominates in some intervals of thresholds and another dominates in other intervals. Just in few situations, one classifier curve results superior to another at all thresholds values.
The curve is plotted by connecting points at intervals on the threshold scale so that several instances change classes between each threshold level. An alternative is to smooth

the curve by fitting some parametric form. There is a series of methods for estimating the ROC curve for a continuous test. A fully parametric approach that results in a smooth curve models the constituent distribution function parametrically in order to arrive at the induced estimator of the ROC curve. A non-parametric method that results in a step function is to use the empirical estimate whose properties have been derived, see [38]. An intermediate strategy between these two is a semi-parametric approach. A more commonly taken strategy to semi-parametric estimation of the ROC curve is to model the ROC curve parametrically, but avoid making additional assumptions about the distribution of the tests results. They produce smooth estimated curves while requiring less stringent assumptions than a fully parametric approach. The bi-normal ROC curve is perhaps the most popular of these intermediate strategies. Parametric ROC analysis is based on a bi-normal assumption, meaning that the actually positive cases are normally distributed and the actually negative cases are normally distributed. It is the overlap between these two distribution that results in the Bayes error rate. Once we have characterized in some way the training examples drawn from these two distributions, then we are free to set a decision threshold that minimizes the Bayes error rate. Other approaches have used logistic and negative exponential distributions. Sampling variability of ROC plots can be analysed by plotting confidence bands instead of single curves. When comparing two ROC curves it could be applied a statical test. To conduct a statistical analysis of ROC curves and their area, one can use traditional tests, such as t-test or analysis of variance (Anova), [39], but these procedures do not take into account the case-sample variance. To produce an average ROC-curve, some authors fit the curve of maximum likelihood to case ratings under a bi-normal assumption, averaged the ROC-curve parameters a and b (or a and Δ m) in the ROC estimation function, and produced an ROC curve using these averaged parameters, [40].

A major disadvantage of ROC analysis is that it does not deliver a single performance measure. If we would reduce the information about performance on a single criterion, we could choose a particular threshold or using another single measure as the area under the ROC curve (AUC). AUC, calculated from a ROC graph, is an overall measure of accuracy that considers the curve in its entirety. AUC does not place more emphasis on one class

over the other, so it is not biased against the minority class. In contrast to error rate, AUC is invariant to the prior probabilities. Area under the ROC curve is most appropriate when each curve dominates another. The limit of AUC occurs when there is no one curve dominating another overall (if one curve is superior in some regions and the other elsewhere), resulting difficult to identify which curve is superior to the other. If multiple curves dominate in different parts of the ROC space, then it could be used the ROC Convex Hull method to select the optimal classifier, see [33]. There are also analysis for when only a portion of the ROC curve is of interest. Cost curves are equivalent to ROC curves, but plot expected cost explicitly, [41].

There is a three-way equivalence between AUC, the Wilcoxon-Mann-Whitney statistic and the probability of a correct ranking of randomly chosen (negative and positive) pair. The AUC is equivalent to the two independent sample Wilcoxon-Mann-Whitney non parametric test statistic, expressing the probability that a randomly selected class 0 example will be consider to belong to class 0 than a random class 1 example. Furthermore, the *Gini coefficient* is defined as twice the area between the ROC curve and the 45% line. Other measures reporting the information on performance to an unique criterion are the *sensitivity/specificity ratio*, expressed by:

$$\frac{a(b+d)}{b/(a+c)} = (a/(a+c)) \div (b/(b+d)) \tag{3.46}$$

Another single degree of freedom criterion is the *odds ratio* or cross-product, expressed by

$$\frac{a}{c} \div \frac{b}{d} = \frac{ad}{cb} = \frac{Se * Sp}{(1-Se)(1-Sp)} \tag{3.47}$$

defining the ratio of the odds of being classified into class 0 given that the example actually belongs to class 0 and class 1 respectively.

The performance evaluation criteria of learning procedures from imbalanced datasets is an outstanding problem. Metrics are used to evaluate the data learning results, so if they do not adequately value rarity or minority class then the learning process, in general, is not likely to handle rare classes and rare cases very well. So particular mention should be made on research focused on *rare classes* [22] emphasizing that they have less impact on

[22] Rare cases correspond to a meaningful but relatively small subset of the data, or equivalently, define a small region of the instance space. Much of the research on rarity relates to rare classes, or, more generally,

accuracy than common classes (classification accuracy computes the fraction of examples that are correctly classified), that is, the minority class has much lower precision and recall than majority class. In literature, different approaches and measures have been proposed and evaluated in relation to the learning process of unbalanced data. Accuracy places more weight on the common classes than on rare classes, which makes it difficult for a classifier to perform well on the rare classes. Additional metrics that could result more appropriate are: the above mentioned ROC analysis and the associated use of the area under the ROC curve (AUC)to assess overall classification performance. In fact, AUC does not place more emphasis on one class over the other, so it is not biased against the minority class. ROC curves, like precision-recall curves, can also be used to assess different trade-off.

With rare cases/small disjuncts different metrics have been managed in literature. One of these measures is the *Laplace estimator*,

$$\textbf{La} = \frac{N - n + K - 1}{N + k} \tag{3.48}$$

k: is the number of classes

N: is the number of instances

n: number of N example belonging to the majority class

in the case of two classes the formula becomes, La_2= (N-n+1)/(N+2).

A more sophisticated error-estimation metric for handling rare cases and small disjuncts was proposed by Quinlan, [37]. This method improves the accuracy estimates of the small disjuncts by taking the class distribution (class priors) into account. Rather than using the entire training set to estimate the class priors, a more representative (local) set of examples is used, relating to training examples close to the small disjunct.

Quinlan modifies the laplace formula this way,

$$\textbf{QLa} = \frac{N - n + I}{N + I} \tag{3.49}$$

where I=1/(1-C) and C represents the disjunct context error C=*e'*/*n'*, with *e'* representing the examples that do not belong to the class associated with the disjunct and *n'* the number of examples in the context of a disjunct.

class imbalance. [36]

Quinlan's experimental results report that, in presence of highly skewed class distributions, applying this modified metric to the learning process, improves classification performance. To summarize, RMSE could be considered as reflecting the classifier's ability to estimate posterior probability, AUC with information about its ranking capabilities and Error Rate metric as a threshold metric.

Recent literature is moving towards classifiers spatial comparison. Some authors studied the issue of selecting appropriate metrics through a visualization method or focused on aggregating the results obtained by different classifiers on different domains by visualization, [42] [43]. In the extreme case, all the performance data are expressed in a single number (projection to one dimension) and the classifiers are compared on the basis of a single quantity, i.e. a scalar metric. However, this involves the maximum amount of information loss and single value indicators of classifier performance are most likely to be unsatisfactory in conveying information about classifier performance.

Finally, to complete the overview we consider classifiers comparison on an exploratory basis rather than through standard evaluation. Different tools may be useful, according to the data available, that could vary from simple approaches to plotting the results in a convenient way (such as histograms, scatter graphics) to dimensionality reduction techniques such as multidimensional scaling or self organizing maps.

3.5 Case study B: Prediction models: Logit Vs. Neural Networks

3.5.1 Outlines

Corporate distress prediction models have been introduced, in literature, to classify companies according to the failure forecast. Here, we propose a comparative analysis of two classifiers: logit and neural networks, on aggregate datasets of Italian companies, referred to the period 2004-2008. We compare the accuracy of these two methodologies and verify the capability of the chosen distress prediction model. Results are expected to provide information on the degree of forecast accuracy of the different methodologies and on the predictive power of proxy variables. The present work attempts to provide evidence of neural networks models outperforming accuracy over logit for predicting the potential financial distress of a company. Corporate distress consists in the inability of a company to refund its financial obligations. An insolvency or a liquidation proceeding implies costs connected with the credit recovery. Thus, increasing the accuracy of a company distress prediction is crucial for banks or investors in relation to the decision process to grant or not a bank-loan or a credit line. In particular, according to the provisions of the First Pillar of the Basel II framework, the increasing quality of the credit risk assessment will result in a reduction of capital allocation. The methodologies considered for corporate distress prediction are supervised classification methods, where a collection of labelled patterns are provided and the problem is to label a new unlabelled item by learning and deriving rules of classification from historical training patterns. In the following work, we compare the predictive accuracy of two different classifiers: a logit model and a neural network, for a corporate distress model [25].

3.5.2 Analysis on balanced dataset

The analysis has been carried out on a sample of 570 large industrial Italian companies, where the related financial data referred to the period 2004-2008 (Source Amadeus). We started from an unbalanced sample of 774 failed companies and 38,480 healthy ones. We

randomly downsized the larger class [27] of healthy units to obtain a balanced sample of 50% failed and 50% healthy companies. Then, we reduced the sample by selecting only firms having complete financial data for three consecutive years, in order to calculate all the six indicators. We considered two datasets in order to analyse the prediction at two different time lags: T1, referred to one-year prior to failure financial statement data and T2, referred to two-years prior to failure ones. We proceeded with the analysis by splitting the data into two sub-samples: a training dataset (402 units, about 70%), to estimate the model parameters of the classifier, and a control dataset (168 units, about 30%), to evaluate the ability of the estimated model in predicting different cases not in the training sample. Further, the related ROC curve area is considered as an accuracy evaluation criterion of the obtained prediction [28].

We considered six input variables, consisting of financial and economic ratios to capture both the financial and the economic perspective, see Table 3.2.

These variables have been chosen because commonly regarded both by banks and scholars [29] as the key indicators to set up their failure prediction models. They are related to two principal company investigation areas: *financial status*, concerning the relationship between positive cash-flows and liabilities, and *performance*, relating to the company efficiency expressed by profitability indicators.

In details, the *Refunding capability*, expressing the potentiality of the company to generate positive cash flows to cover financial obligations, is investigated by the Financial debt coverage ratio. The *Growth*, indicating an increasing economic dynamic, is investigated by the Sales variation ratio. The *Debt cost*, expressing the degree of the economic incidence of the debt exposure, is investigated by the Interest paid on sales ratio. The *Indebtedness*, indicating the debt exposure level, is investigated by the Leverage. The *Efficiency*, expressing the capability to generate operating returns, is investigated by the Ebit on sales ratio. The logarithm of Sales represents a size control variable.

PROXY	VARIABLE	COMMENT	FORMULA
Refunding Capability	$FDebtCov_t$	Financial Debt Coverage	$cashflow_t/financialDebt_t$
Growth	$SalesVar_t$	Sales Variation	$\Delta(sales)/sales_{t-1}$
Debt cost	$IPSales_t$	Interest paid on Sales	$interestpaid_t/sales_t$
Indebtedness	$Leverage_t$	Leverage	$shareholdersfunds_t/TotAsset_t$
Efficiency	$EbitSales_t$	Ebit on Sales	$Ebit_t/Sales_t$
Size	$SIZE_t$	Size variable	$Log(sales_t)$

Table 3.2: Description of the input variables in the distress prediction model

3.5.3 Numerical results for balanced data

The output variable value 1 corresponds to a not-failed company whereas the value 0 is related to a failed one. Thus, positive coefficients are associated with decrease in the probability of failure while negative ones correspond to increase in the probability of bankruptcy. As concerns the logistic regression processed, both in T1 and T2 lag periods, the model coefficients estimated that result significant at 5% level are *leverage* and *ebitsale*, in T2 also *ipsales* is significant. The other estimated coefficients do not result significant in both two lag periods, even if these ratios are commonly considered by banks and loan analysts in credit scoring evaluations.

A plausible economic explanation could be connected with the disclosure accounting rules for the Financial statement items in Italy and their weak capability to render the financial dimension. In particular, we refer to information regarding the cash-flows dynamics and the correct distinction between long and short-term debt exposure. As regards the growth dimension, the generally used item: sales, could not result always appropriate to describe different sector characteristics that instead could be better investigated by other economic revenue items. In T1, the logistic regression results, see Table 3.3, show that the log odds of healthy (versus failed) increases by 7.41 for a one unit increase in *leverage* and it is enhanced by 5.02 for a one unit increase in *ebitsales*. In T2, the logistic regression results, see Table 3.4, indicate that the log odds of healthy (versus failed) increases by 6.56 for a one unit increase in *leverage* and it is enhanced by 6.73 for a one unit

					No. of obs	= 402
					LR chi2(6)	= 178.00
					Prob > chi2	= 0.0000
					PseudoR2	= 0.3194
Log	likelihood =	-189.64759				
y	Coef.	Std. Err.	z	$P > \|z\|$	[95%	Conf. Interval]
fdebtcovt1	.3143562	.2541972	1.24	0.216	-.1838611	.8125735
salesvart1	-.0086909	.2370037	-0.04	0.971	-.4732095	.4558278
ipsalest1	-1.246083	1.378483	-0.90	0.366	-3.94786	1.455695
leveraget1	7.410037	1.127743	6.57	0.000	5.199701	9.620373
ebitsalest1	5.019519	1.433156	3.50	0.000	2.210584	7.828454
sizet1	.1516105	.1323923	1.15	0.252	-.1078737	.4110947
cons	-2.406548	1.320317	-1.82	0.068	-4.994321	.1812255

Table 3.3: Logistic regression results: one-year prior to failure (T1).

increase in *ebitsales*. For a one unit increase in *ipsales*, the log odds of being not-failed decreases by 7.28.

The coefficients of *leverage* and *ebitsales* are always positive and significant at the 5 % level in both two periods, T1 and T2, indicating that the indebtedness and efficiency dimensions are strongly related to failure events. The coefficient of *ipsales* results negative and significant only in T2.

As regards the neural network, we iterated the estimation process on the training dataset for combinations of hidden nodes from 2 to 10 and values of the decay parameter: 0.1, 0.01, 0.05, 0.005. The optimization process is done via a quasi-Newton method. The initial parameter vector has been chosen at random but setting the same random seed for every combination. We obtained 36 models and we computed the *empirical error* on the training set for each model estimated. We calculated the related goodness-of-fit criteria AIC, BIC and GCV, by considering the number of degree of freedom equals to the number of weights of the model. Then, we selected three best models according to each selection criterion for both the two lag periods. Thus the training process is completed, the best NN models selected were used for prediction applied to the control dataset. From the

					No. of obs	= 402
					LR chi2(6)	= 108.34
					Prob > chi2	= 0.0000
					PseudoR2	= 0.1944
Log	likelihood =	-224.47624				
y	Coef.	Std. Err.	z	$P > \|z\|$	[95%	Conf. Interval]
fdebtcovt2	-.2836873	.339502	-0.84	0.403	-.949099	.3817244
salesvart2	.099439	.1683394	0.59	0.555	-.2305002	.4293783
ipsalest2	-7.2858	2.914459	-2.50	0.012	-12.99803	-1.573565
leveraget2	6.556056	1.039662	6.31	0.000	4.518357	8.593755
ebitsalest2	6.730001	1.890963	3.56	0.000	3.023782	10.43622
sizet2	.0354015	.1175283	0.30	0.763	-.1949497	.2657527
cons	-1.377883	1.17262	-1.18	0.240	-3.676175	.920409

Table 3.4: Logistic regression results: two years prior to failure (T2).

	bestvalue	nodes	decay	Area	p-value	binorm.area
LOGIT				**0.8605**	3.5e-16	0.8608
AIC	4.086	3	0.005	0.8968	3.3e-19	0.8927
BIC	106.048	2	0.005	0.8634	2.1e-16	0.8755
GCV	59.756	3	0.005	0.8968	3.3e-19	0.8927

Table 3.5: T1 (one-year prior to failure). Area under the ROC curve for Logit and best selected Neural Networks. The first column contains the selection criteria best values. The p-value addresses to the null hypothesis H_0: ROC Area=0.5. The sixth column contains the binormal curve area.

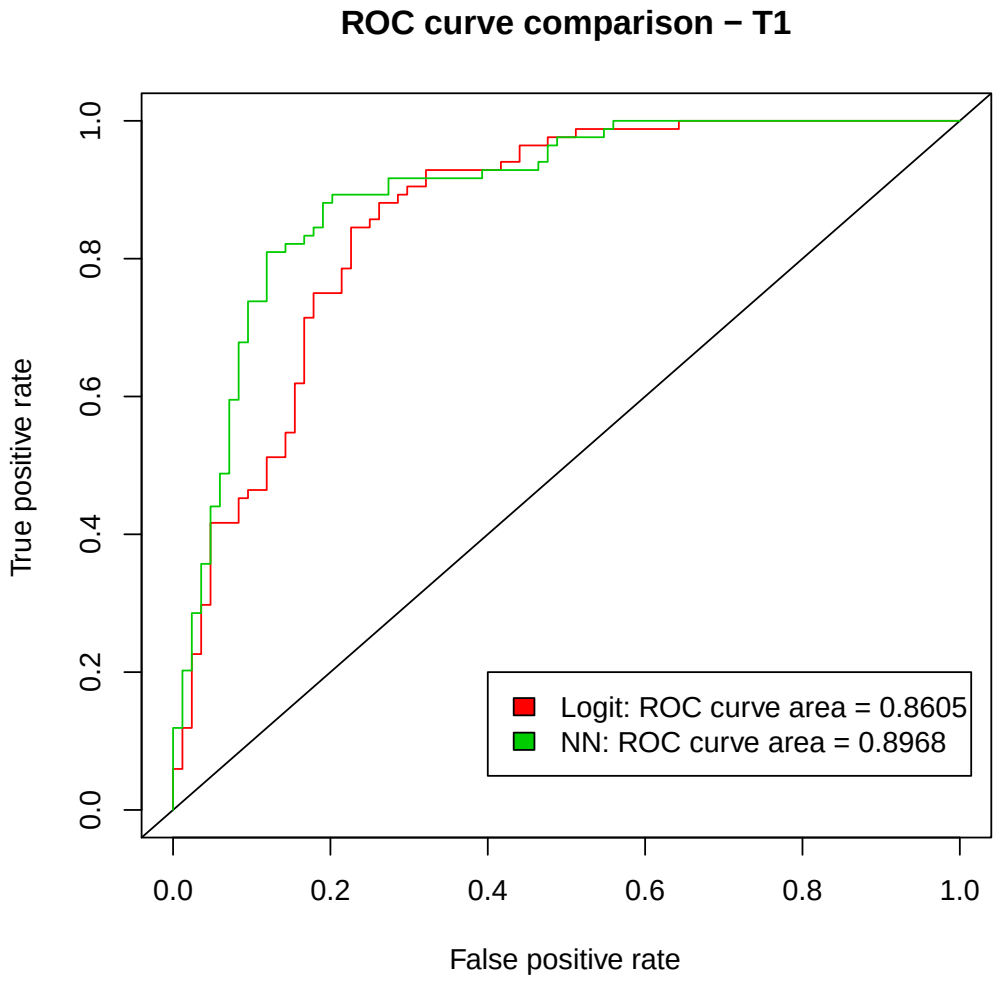

Figure 3.1: T1- ROC curve comparison between Logit and selected NN (3 hidden nodes, decay=0.005).

	bestvalue	nodes	decay	Area	p-value	binorm.area
LOGIT				**0.7728**	5.1e-10	0.7325
AIC	4.313	4	0.005	0.8202	3.9e-13	0.8151
BIC	106.2	2	0.010	0.7684	9.5e-10	0.7446
GCV	75.219	4	0.005	0.8202	3.9e-13	0.8151

Table 3.6: T2 (two-years prior to failure). Area under the ROC curve for Logit and best selected Neural Networks. The first column contains the selection criteria best values. The p-value addresses to the null hypothesis H_0: ROC Area=0.5. The sixth column contains the binormal curve area.

predictions of both logit and NN models, we traced the ROC curve in order to compare the forecasting performance of the two models, as shown in Fig. 3.1 and Fig. 3.2, and the dominance of a curve on the other. Furthermore, we calculated the area under the ROC curve, as an alternative forecast accuracy criterion considering the curve in its entirety, to objectively compare the two classifiers.

For T1, the AIC and GCV criteria selected the same best model with three hidden nodes and decay equals to 0.005, presenting an higher value of the area under the ROC curve than the model selected by the BIC criterion, see Table 3.5.

In the same way for T2, the AIC and GCV criteria selected the same best model, in this case with four hidden nodes and decay factor equals to 0.005, presenting an higher value of the area under the ROC curve than the model selected by the BIC criterion, see Table 3.6. For both two lag periods, neural network models, selected with the criteria AIC and GCV, presented dominant ROC curves and higher ROC area values over the logistic model, indicating a better forecast accuracy, see Figures 3.1 and 3.2.

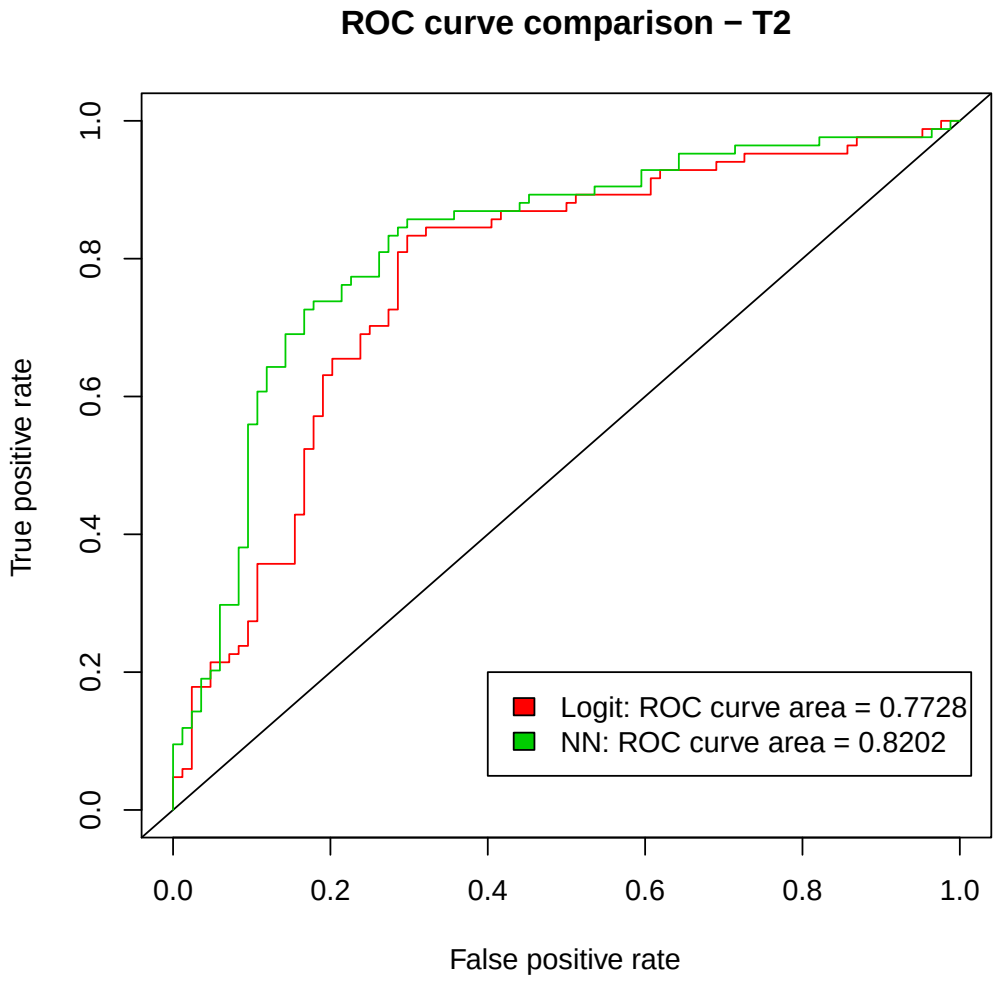

Figure 3.2: T2- ROC curve comparison between Logit and selected NN (4 hidden nodes, decay=0.005).

3.5.4 Comments

The comparative analysis underlines a superior prediction accuracy of the neural networks models over the logistic model for both two failure lag periods.
The neural networks more accurate fitting is important for the costs of wrong prediction in credit scoring. Otherwise, the logistic regression offers a more readable economic interpretation of the predictor variables influence on the output response vector.
In the case study, leverage and ebitsales in T1 (also ipsales in T2) result strongly significant whereas the other variables not, maybe due to the Italian disclosure rules for the Financial statement items.
From the economic point of view, further research may be carried on aiming at setting a more significance stable predictors frame, also for other European countries.
Further investigations may focus on input variables sensitivity analysis to interpret the impact of predictors on the output vector in neural networks [32].
As concerns the methodological procedure, further developments may be conducted aiming at improving the tuning in the neural network optimization process and in managing with unbalanced data.

3.6 Class Imbalance

3.6.1 The problem and the literature

Learning with skewed[23] class distributions is an important issue in supervised[24] learning because for a number of application domains, a huge disproportion in the number of cases belonging to each class is common.

Working with unbalanced dataset is still a major obstacle in classifier learning and many traditional learning systems are not prepared to induce a classifier that accurately classifies the minority class under such situation. Frequently, the classifier has good classification accuracy for the majority class but its accuracy for the minority class is unacceptable. The problem arises when the misclassification cost for the minority class is much higher than the misclassification cost for the majority one.

The class imbalance problem is encountered by inductive learning system on skewed datasets where one class is represented by a large number of instances while the other is heavily under-represented, presenting only few instances, and that class unbalance could affect the performance compared to the one attainable by standard learning methods which assume a balanced class distribution.

Several domains present the class imbalance problem, such as text categorization, information retrieval, fraud detection, rare medical diagnosis, financial risk models and image recognition.[25]

[23] The class imbalance problem occurs when there is a large discrepancy between the prior probabilities of the individual classes, that is one class is represented by a greater number of training examples than the other.

[24] Supervised learning is the process of creating a classification model from a set of examples, called the training set, which belong to a set of classes. Once a model is created, it can be used to automatically predict the class of other unclassified examples. In supervised learning, a set of n training examples is given to an inducer. Each example X is an element of the set F1*F2*..Fj where Fj is the domain of the *jth* feature. Training examples are tuples (X,Y) where Y is the label, output or class. The Y values are typically drawn from a discrete set of classes 1,...,K in the case of classification. Given a set of training examples, the learning algorithm (inducer) outputs a classifier such that, given a new example, it accurately predicts the label Y, [88].

[25] In many applications, such as medical diagnosis, fraud detection, intrusion prevention and risk management, the primary interest is in fact the small classes. In these applications, it is only the data distributions that are skewed, but so are the misclassification costs. Most classical learning algorithms assume that all misclassification errors cost equally, and ignore the difference between types of misclassification errors.

In corporate distress prediction analysis, we would refer to a two-classes problem, indicating the failure event and the non-failure one. Even if we will consider the case of two-classes problems, the discussion could be extended also to multi-class problems.

In literature, there have been various attempts at dealing with the class imbalance problem, [26], [44], [27], [45]), [46], [47], at the beginning mostly sparse, and recently several studies and research are conducted aiming at connecting specific types of imbalances to the degree of inadequacy of standard classifiers and to test and compare the various methods proposed to remedy the problem and the response on accuracy of different classifiers. Some early studies attempted to systematize research on the class imbalance problem, in one of them [48] different degrees of imbalances are linked to the performance of a decision tree learning system on a large number of real-world data sets, in another [49] a number of specific approaches are proposed to deal with class imbalances in the context of neural networks and on a few real-world datasets.

Several methods have previously been proposed to deal with this problem including prior scaling, probabilistic sampling, post-scaling and equalizing class membership, [52].

Class imbalance impacts on classifiers like Decision tree and Multi-layer perceptrons designed to optimize overall accuracy without taking into account the relative distribution of each class, [84]. These classifiers tend to ignore small classes while concentrating on classifying the large one accurately, [83].

The imbalance in the data can be more characteristics of ”sparseness” in feature space than the class imbalance, [61]. In addition to the problem of inter-class distribution, another problem arising due to the ”sparsity” in the data is the distribution of data within each class. This problem was also linked to the issue of small disjuncts.

A large number of approaches have been proposed to deal with the class imbalance problem: internal approach, consisting in creating algorithm ad hoc for the problem, [85] [26], or external approach consisting in re-sampling the data so to diminish the effect, [86], independently of the classifier used.

Recent research has shown that using an uneven distribution of class examples in the learning (training) process can leave the learning algorithm with a performance bias: poor accuracy on the minority class but high accuracy on the majority class; even if some au-

thors confirm only partly this assumption and connected it to certain conditions, [48] [34]. In literature, major research directions on class imbalance have, principally, focused on:

- Re-sampling methods for balancing the dataset;
- Modification of existing learning algorithms;
- Measuring the classifier performance in imbalance domains;
- Relationships between class imbalance and other data complexity features.

An important area of research, related to the learning bias deriving from unbalance, concerns finding classifiers that are accurate on both classes and adopting metrics that are insensitive to the learning bias such as the area under the ROC curve (AUC) or the average accuracy of each class. As mentioned in **Subsection 3.4.2**, the AUC (area under the ROC curve) is generally chosen as the primary evaluation measure because it is known to be a good estimator of classification ability in class imbalance learning. The AUC is insensitive to the class imbalance learning bias and considers the classification performance across varying classification thresholds. It is important to notice that particular attention must be kept on the trade-off between different class accuracy, while the techniques applied to deal with the unbalanced problem to improve classifier performance are often focused on increasing minority class accuracy without controlling the effects of the overall classification ability of the classifier. As said before, increasing the performance of one class might result in a trade-off in performance for the other. Analysis of learning bias is usually conducted by computing individual class performances, that is majority class accuracy and minority class accuracy, in order to analyse if as the level of class imbalance increases more instances are classified as belonging to the majority class. The main aim would be to adapt the learning function managing to maintain good overall classification

ability across both the minority and the majority classes using the AUC, as well as increasing individual class accuracy.
The unbalanced class problem could be framed according to different directions:

- What is the nature of the class imbalance problem?
- How do different approaches proposed for dealing with the class imbalance problem compare?
- Does the class imbalance problem hinder the accuracy performance of classifier?

As concerns the nature of class imbalance problem, some authors [27] provided evidence that the imbalance problem is a relative problem depending on both the complexity of the concept (corresponding to the number of sub-clusters into which the classes are subdivided) represented by the data in which the imbalance occurs and the overall size of the training, in addition to the degree of class imbalance present in the simulated data. In particular, [27] concentrated on explaining both the relationship between concept complexity, size of the training set and the class imbalance level, to identify the class imbalance situations that are most damaging for a standard classifier that expect balanced class distributions and how to deal with the class imbalance problem.
For imbalanced datasets, the decision boundary established by standard machine learning algorithm tends to be biased towards the majority class; therefore, the minority class instances are more likely to be misclassified. There are many problems that arise from learning with imbalanced datasets. The first problem concerns with measures of performance. If the evaluation metrics does not take the minority class into consideration, the learning algorithm will not be able to cope with class imbalance very well. With standard evaluation metrics, such as the overall classification accuracy, the minority class has less impact compared to the majority class. The second problem is related to the lack of data. For a class consisting of multiple clusters, some clusters may contain a small number of

samples compared to other clusters; therefore the lack of data can occur within the class itself. The third problem is noise. Noisy data have a serious impact on minority classes than on majority classes. Furthermore, standard machine learning tend to treat samples from a minority class as noise.

In learning estimation, imbalances in the distribution of the data can occur either between the classes (inter-class) or within a single class (intra-class), that is the case where a single class is composed of various sub-clusters of different sizes, some of them being tiny. A *between-class* imbalance corresponds to the case where the number of examples representing the positive class differs from the number of examples representing the negative class; whereas a *within-class* imbalance corresponds to the case where a class is composed of a number of different sub-clusters and these sub-clusters do not contain the same number of examples.[26] In the inter-class imbalance, the degree of imbalance can be represented by the ratio of sample size of the minority class to that of the majority class. Most classification techniques such as decision tree, discriminant analysis and neural networks assume that the training samples are evenly distributed amongst different classes. In real-world applications, the ratio of minority to majority samples can be as low as 1 to 100, 1 to 1000 or 1 to 10,000. Hence, the standard classifiers are effected by the prevalent class and tend to ignore or treat the small classes as noise. When the performance is measured using classification accuracy, the best ratio is near to the natural ratio; on the other hand, when the AUC measure is used, the best ratio is near the balanced ratio [34]. Although both types of imbalances are known to affect negatively the performance of classifiers, in general only methods for dealing with between class imbalance have been implemented. Deepening the analysis, the imbalance ratio between classes should not be considered as the only factor causing reduction in classifier performance and that other factors such as training size and concept complexity also affect performance. In literature, some authors delved the within-class unbalance by referring to the problem of small disjuncts[27]

[26]The within-class imbalance problem occurs when a class consists of several sub-clusters or sub-concepts and these sub-clusters do not have the same number of samples, [51].

[27]The within-class along with the between-class imbalance problem are expressions of the general problem known as the problem of small disjuncts, in which classifiers are biased towards recognizing large disjuncts correctly, but over-fitting and misclassifying samples represented by small disjuncts, [50]

and concept complexity in learning estimation, [50] [51] [52] [53]. Systems that learn from examples do not usually succeed in creating a purely conjunctive definition for each concept. Instead, they create a definition that consists of several disjuncts, where each disjunct is a conjunctive definition of a sub-concept of the original concept. The coverage of a disjunct is defined as the number of training examples it correctly classifies. A disjunct is called small if its coverage is low. In literature, rare cases are considered to cause small disjuncts to occur, which are known to be more error prone than large disjuncts. Learning system usually create concept definitions that consist of several disjuncts. The learned decision boundary much approximates more closely the true decision boundaries when more data is available. One study, [36], which employed synthetically generated datasets, showed that rare cases have a much higher misclassification rate than common cases, because the associated lack of data where the number of samples is small affects the estimation of the true decision boundary. We refer to this as the problem with rare cases. Rare cases cause small disjuncts in the learned classifier. The problem with small disjuncts, observed in many empirical studies, is that small disjuncts generally have a much higher error than large disjuncts. One explanation is that some small disjuncts may not represent rare, or exceptional, cases, but rather something else-such as noisy data. Each disjunct, in turn, is a conjunctive definition of a sub-concept of the original concept. The coverage of a disjunct corresponds to the number of training examples it correctly classifies, and a disjunct is considered a small disjunct if the coverage is low. What makes small disjuncts more error prone are the bias of the classifiers as well as the effect of attribute noise, missing attribute, class noise and training set size on the rare cases which cause them. Within a class, the data could be distributed according to a mixture density whose components have relative densities that may vary greatly. Some authors, [52], underlined that techniques usually used while decreasing the difference between the prior probabilities of the classes (between-class imbalance), they would probably increase the difference between the relative densities of the sub-components within each class (within-class imbalance). In most classification tasks, the presence of within-class imbalance is implicit. It is known to have negative effects on the performance of standard classifiers and increases the complexity of concept learning. Although both types of imbalances are known to affect negatively

the performance of classifiers, most existing methods for class imbalance focused mainly on dealing with between class imbalance have been implemented consisting in rectifying the between-class imbalance and ignoring the case where imbalance occurs within each class. As concerns the concept complexity in data, it corresponds to the level of separability of classes within the data. Some authors, [27], reported that for simple data sets that are linearly separable, classifier performances are not susceptible to any amount of imbalance. Indeed, as the degree of data complexity increases, the class imbalance increases, the class imbalances factor starts impacting the classifier generalization ability. High complexity refers to inseparable datasets with highly overlapped classes, complex boundaries and high noise level. When samples of different classes overlap in the feature space, finding the optimum class boundary becomes hard. In fact, most accuracy-driven algorithms bias toward the prevalent class.
That is, they improve the overall accuracy by assigning the overlapped area to the majority class, and ignore or treat the small class as noise.
We will examine in more detail the small-disjuncts problem and data complexity after a general literature overview coping with the class imbalance problem.
In literature, different solutions to the class imbalance problem have been proposed both at the data and the algorithmic level.[28]
Different methods have been proposed to deal with class imbalance

1. Random re-sampling methods;

2. Focused re-sampling methods;

3. Cost-learning methods;

[28]For a comprehensive quite updated review of the ”state of the art” for the class imbalance research area and for the applied techniques, see [54]

In details, at the *data level*, different forms of re-sampling have been proposed such as random oversampling with replacement, random under-sampling, directed oversampling (in which no new examples are created, but the choice of samples to replace is informed rather than random), directed under-sampling(where again the choice of examples to eliminate is informed), oversampling with informed generation of new samples, and combination of the above techniques. At the *algorithmic level*, applied procedures include adjusting the costs of the various classes so as to counter the class imbalance (by altering the relative costs of misclassifying the small and the large classes), adjusting the probabilistic estimate at the tree leaf (when working with decision tree), adjusting the decision threshold, and recognition-based (learning from one class) rather than discrimination based (two class) learning. The random over-sampling method consists of oversampling the small class at random until it contains as many examples as the other class. The random under-sampling method consists of eliminating, at random, elements of the over-sized class until it matches the size of the other class. The cost-modifying methods consist of modifying the relative cost associated to misclassifying the positive and the negative class so that it compensates for the imbalance ratio of the two classes.
External approaches could be divided into two groups: one focused on the best data for inclusion in a training set, [86], and the other focused on studying the best proportion of positive and negative examples to include in a training set:

1. Oversampling or under-sampling?

2. At what rate oversampling or under-sampling?

3. Can combination of re-sampling improve classification accuracy?

The simplest way to balance a dataset is by under-sampling (randomly or selectively) the majority class while keeping the original population of the minority class.
Random under-sampling aims at balancing the data through the random removal of negative examples. The major problem of this technique is that it can discard data potentially

important for the classification process and could result in information loss for the majority class, [27]. Unlike the random method, it has been proposed to remove only those negative instances that are redundant or that border minority class examples, [59].
Experimental results show that under-sampling produces better results than over-sampling in many cases. Instead, other authors, [58] [65], considered oversampling a correct method in datasets with a a very high majority/minority ratio.
In another work, [89], the authors tested a C4.5 decision tree classifier and they found under-sampling more effective for sensitivity than oversampling (duplication method) because of the reduction in pruning n case of oversampling. Under-sampling often renders pruning unnecessary. Oversampling tends to reduce the amount of pruning that occurs. In the best classifier seeking they define the curve of the expected cost of a classifier across all possible choices of misclassification costs and class distributions by considering the normalized error rate. They include the misclassification cost in the probability distribution (the x in the cost function) by multiplying the original value of probability of positive events by the cost of misclassifying a positive instance as negative and then normalizing so that x ranges from 0 to 1. The process tend to exactly balancing the misclassification costs to the class distribution. They generally found that using under-sampling established a reasonable baseline for algorithm comparison. However, one problem with under-sampling is that introduces non-determinism into what is otherwise a deterministic learning process. The values obtained from cross-validation estimate the mean performance of a classifier based on a random sub-sample of the dataset. With a deterministic learning process any variance in the expected performance is largely due to testing on a limited sample. But for under-sampling, there is also variance due to the non-determinism of the under-sampling process. If our measure of success is purely the difference between the means then this is not important. But the choice between two classifiers might also depend on the variance and then using under-sampling might be less desirable.
Above all, the belief is that although over-sampling does not lose any information about the majority class, it introduces an unnatural bias in favour of the minority class.
The simplest oversampling method is the random over-sampling that consists in increasing the size of the minority class by randomly replicating positive examples. Some

authors,[56], proposed to oversample the minority class by generating new instances by interpolating between several positive examples that lie close together, (method called SMOTE) that allows the classifier to build larger decision regions that contain nearby instances from the minority class.
Using synthetic examples to augment the minority class is believed better than oversampling with replacement, even if it creates noise which could result in a loss of performance. SMOTE stands for Synthetic Minority over-sampling Technique.
In additional works, [45], they continue not using oversampling with replication, by considering this procedure not always results to improve minority class prediction and could lead to over-fitting. They interpret the underlying effect in terms of decision regions in feature space.[29] So, they generate synthetic examples by operating in the ”feature space” rather than the data space, causing the classifier to create larger and less specific decision regions, rather than smaller and more specific regions. Synthetic samples are generated by taking the difference between the feature vector (sample) under consideration and its nearest neighbour. Then this difference is multiplied by a random number between 0 and 1, and added to feature vector under consideration. In other words, neighbours from the k nearest neighbours are randomly chosen and added.
Their results show that, on average, under-sampling is usually better than oversampling with replication; SMOTE in better than under-sampling; pruning is detrimental to learning from imbalanced datasets. A modification of this method is Borderline SMOTE [57], that consists in using only positive examples close to the decision boundary, since they are more likely to be misclassified.
In other works,[30] external and internal cross-validation are used to guide sampling, [60]. In details, it is used a wrapper-based algorithm to select under-sampling percentages, a down-step procedure by controlling a performance threshold to be not violated and determining the sampling levels which maximize the classifier’s (a decision tree) f-measure (calculated from precision and recall to summarize the effects of the two types of errors)

[29]The imbalance in the data can be more characteristic of ”sparseness” in feature space than the class imbalance, [45].

[30]For a complete overview of the re-sampling strategy: random oversampling with replacement, random under-sampling, focused oversampling, focused under-sampling, oversampling with smote, see [61].

or the AUC (area under the ROC curve).

Other authors, [52], proposed a ”guided re-sampling”, by firstly processing an unsupervised clustering algorithm on each class of the training data in an attempt to find any within-class imbalances and then the clusters are found they use them to guide the re-sampling. The elements in each sub-components within each class can then be re-sampled until each subcomponent has the same number of examples as the largest sub-component. Then the between-class imbalance can be eliminated by randomly selecting and duplicating members of the under-represented class (equalizing class membership). Their results show improvements by using guided re-sampling on ”very imbalanced” (between and within) datasets when methods of blind re-sampling fail in allowing a classifier to be trained to recognize members of the under-represented class.[31]

To summarize, in literature, part of research has focused in overcoming the drawbacks of both random under-sampling and over-sampling. If we could know beforehand the conditional probabilities which make the construction of a true bayes classifier possible, class distribution should not be a problem. Conversely, classifier it is likely to suffer from poor estimates due to few data available for the minority class. Nevertheless, due to low overlapping between the classes, the effect of class imbalance in this case is lower than when there is a high overlapping. In some cases, the over-fitting problem is avoided in over-sampling forming new minority classes by interpolating between several minority class examples that lie together, in order to cause the decision boundaries for the minority class to spread further into the majority class space. Other methods for reducing the training set size are based on k-nearest neighbour. To cite one work, [58], in the procedure two controlled parameters are considered: the first one is the clusters centroids distance and the second one the imbalance degree (so the overlapping degree), in order to verify if the degree of imbalance is an element by itself for degrading performance or if it also due to the degree of the overlapping between the two class distributions. They found that the more the overlapping degree (the smaller centroids distance) is high the more the decrease in prediction accuracy (AUC). Their results show that oversampling methods in general

[31] These results are subject to the assumption to know the correct number of sub-components per class as well as their nature, [52].

and SMOTE-based methods in particular, are very effective even with highly imbalanced and overlapped datasets. Furthermore, when they apply SMOTE + ENN methods in high degree of overlapping they reach the better results because of the cleaning effect of the sampling method.[32]

Other re-sampling procedures have been proposed taking into account the small disjuncts problem, see [62]. In the work, it is suggested the following re-sampling strategy:

a) using a clustering algorithm on each class to identify the sub-clusters that constitute it;

b) each sub-cluster of the large class is re-sampled until it reaches the size of the biggest sub-cluster in that class. At this point, the overall size of the large will be max-classized and there will be no within-class imbalance in the large class.

In order to prevent a between-class imbalance as well as within class imbalances in the smaller class, each sub-cluster of the small class is re-sampled until it reaches size max-classize/N_smallclasses, where N_smallclasses represents the number of sub-clusters in the small class. In many real-world domains, the class distribution where the data is skewed the cost of misclassifying the minority class could be substantially greater than the cost of misclassifying the majority class. Typical classifiers such as decision tree induction system or multilayer perceptions are designed to optimize overall accuracy without taking the relative distribution of each class. These classifiers tend to ignore small classes while concentrating on classifying the large ones accurately, [53].

The reason that altering, by sampling, the class distribution of the training sets aids learning with highly-skewed data sets is that it effectively imposes non-uniform misclassification costs. This equivalence between altering the class distribution of the training data and altering the misclassification cost ratio was formally established by [46].

There are three regular approaches for feature selection: *filter-based, wrapper-based* and *embedded-based* ones. The filter-based selects relevant features before the classification algorithm is applied. The wrapper approach assumes to perform many times the learners on candidate feature subsets to choose relevant features. In the embedded approach, the feature selection is occurred as a part of the learners.

[32] ENN stands for Wilson's Edited Nearest Neighbour Rule. ENN removes any example whose class label differs from the class of at least two of its three nearest neighbours. ENN is applied to the oversampled training set, as a data cleaning method, [58].

Sampling methods modify the prior distributions of the majority and minority class in the training set to obtain a more balanced number of instances in each class, and, finally, we could classify them in three categories:

Basic sample methods The two basic methods of reducing class imbalance in training data are under-sampling and oversampling. Cost-sensitivity is obtained by altering the ratio of positive to negative examples in the training data, or, equivalently, by adjusting the probability threshold using to assign class labels. Both of the sampling techniques decrease the overall level of class imbalance, thereby by making the rare class less rare. These sampling methods have several drawbacks. Under-sampling discards potentially useful majority-class examples, and thus can degrade classifier performance. Because oversampling introduces additional training cases often by making exact copies of examples, it may lead to over-fitting. More importantly, oversampling introduces no new data, so it does not address the fundamental ”lack of data”. This explains why some studies have shown simple over-sampling to be ineffective at improving recognition of the minority class and why under-sampling may be a better choice.

Advanced sampling methods They may combine under-sampling and over-sampling techniques. One of the popular oversampling approaches is SMOTE [56], which attempts to add information to the training set by introducing new, non replicated minority class examples. A probability distribution is selected to model the available minority class examples. In an under-sampling scheme, instead of eliminating instances randomly ([90]) proposed to use vector quantization, which is a loss compression method on the majority class to build a set of representative local models and use them for training SVM. An alternative method would be to use clustering to identify possible rare cases and then sample to equalize these cluster sizes. The informative re-sampling is a cluster-based under-sampling, where clustering is employed for selecting the representative samples to improve the predictive accuracy for the minority class. Some authors,[91], also proposed to use clustering to reduce the imbalance ratio, called CLass purity maximization (CPM), by partitioning the

data space into clusters and filtering out regions of high majority class, in order to use only regions containing minority samples to build a predictive model. In a work, [92], proposed an active learning query technique that creates query instances near the classification boundary rather than selecting randomly.

Ensamble-learning methods In which multiple classifiers are trained from the original data and their predictions are combined to classify new instances.

BOOSTING ([72]) and BAGGING ([93]) are the two widely known ensemble-based approaches. Boosting is an iterative algorithm that places different weights on the training distributions each iteration. After each iteration boosting increases the weights associated with the incorrectly classified examples and decreases the weights associated with the correctly classified examples separately. This forces the learner to focus more on the incorrectly classified examples in the next iteration. Note that boosting effectively alters the distributions of the training data. At each boosting iteration, the distribution of training data is altered by updating the weight associated with each sample. When the datasets are severely skewed, under-sampling and oversampling methods are often combined to improve generalization of the learner, [71].

Boosting has been analysed from a theoretical perspective to determine whether it is guaranteed to improve the classification performance of any base learner for the rare class([95]). This analysis shows that no such guarantee exists. Rather, the performance improvement from boosting is shown to be strongly tied to the choice of the base learning algorithm.

Examples of algorithms that use boosting approach are SMOteBoost ([56]), AdaBoost ([72]), DataBoost-IM that balances not only the class distribution but also the total weight within the class ([94]), and cost-sensitive boosting. Most bagging methods use a similar learning procedure that consists in re-sampling subsets from a given training set, building multiple base classifiers on those subsets and combining their predictions to make final prediction ([93]). In Under-bagging, each subset from the training set is created by under-sampling the majority class randomly.

Re-sampling methods have mainly been criticized because of altering the original class distribution. Sampling is a ”wrapper-based method” that can make any learning algorithm cost-sensitive, whereas the ”‘cost-sensitive learning algorithm”’ is not a wrapper-based method since the cost-sensitivity is embedded in the algorithm.
In a cost-sensitive learning, no cost is assigned to correct classifications. Since the positive (minority) class is often more interesting than the negative (majority) class, typically (cost of false negative) **Cfn** $>$ **Cfp** (cost of false positive), (Note that a false negative means that a positive example has been misclassified).
As concern the solutions at the algorithm level, we can define three main directions:

- Cost-sensitive learning
- One-class classifiers
- Classifier ensembles

As regards *cost-sensitive learning*, traditional learning models implicitly assume the same misclassification costs for all classes. In some domains the cost of a particular kind of error can be different from others. Some works assign distinct costs to the classification errors for positive and negative examples. In a work, [27], it has been proposed the use of non-uniform error costs defined by means of the class imbalance ratio.
As concerns *one-class classifiers*, the minority class can be viewed as the target class, whereas the majority class will be the outlier class. Some authors, [63], show that one-class learning is particularly useful on extremely unbalanced data sets with high dimensional noisy feature space.
As regards *classifier ensembles*, several studies, [64], [65], have been proposed consisting in generating multiple training samples in order to cover all the elements and then combining the results of different classifiers. Other data complexity characteristics have

been investigated in many studies focused on the effect of other complexity features in imbalanced domains causing loss of performance as distribution of the data within each class, small disjunct [53], density and overlap complexity [66].

A cost-sensitivity learning technique takes costs, such as misclassification cost, into consideration during model construction and produces a classifier that has the lowest cost. In a two class problem, C(+,-) signifies the cost of misclassifying a positive sample as the negative sample, and C(-,+) denotes the cost of the contrary case. Cost-sensitive learning methods take advantage of the fact that it is more expensive to misclassify a true positive instance than a true negative instance, that is C(+,-) $>$ C(-,+). For a two-class problem, a cost-sensitive learning method assigns a greater cost to false negative than to false positives, hence resulting in a performance improvement with respect to the positive class.

Existing cost-sensitive learning for dealing with imbalances datasets can be divided into two different categories. The first category consists of learning algorithms that are designed to optimize a cost-sensitive function directly (cost-sensitive decision tree, [67] , that directly takes costs into model building). The second category is a collection of existing cost-insensitive learning algorithms that are converted into cost-sensitive ones. This category, also known as cost-sensitive meta-learning, can be further divided into sampling, weighting, thresholding and ensemble learning. Methods in the weighting group [68], convert sample-dependent costs into sample weights by assigning heavier weights to the minority training instances. Different weighting strategies have been proposed, [70], to weight samples of the minority class based on the local data distributions, and others suggested to weigh training samples based on posterior probability [69].

Cost-sensitive learning approach assumes the misclassification costs are known. In practice, specific cost information is often unavailable because costs often depend on a number of factors that are not easily compared. Moreover, [36] found that cost-sensitive classifiers may lead to over-fitting during training.

Learners can implement cost-sensitive learning in a variety of ways. One common method is to alter the class probability thresholds used to assign the classification value. In this case, no data is discarded or replicated.

Cost sensitive learning, besides changing the class distributions they incorporate costs in

decision making is another way to improve classifier's performance when learning from imbalance datasets.

Cost models takes the form of a cost matrix where the cost of classifying an example from true class *i* to class *j* corresponds to the matrix entry λ_{ij} and vice versa, see table 3.7. This matrix is usually expressed in terms of average misclassification costs for the

		True	class
		Positive (class *i*)	Negative (class *j*)
Predicted Class	Positive prediction (class *i*)	0	λ_{ij}
	Negative prediction (class *j*)	λ_{ji}	0

Table 3.7: Cost matrix for a two-class problem.

problem. The diagonal elements are usually set to zero, meaning correct classification has no cost. The aim in cost-sensitive classification is to minimize the cost of misclassification, which can be realized by choosing the class with the minimum conditional risk, [71] *Metacost* [47] begins to learn an internal cost-sensitive model then estimates class probabilities using bagging and then re-labels the training examples with their minimum expected cost classes, and finally relearns a model using the modified training set.

In *Adaboost*, [72], an ensemble learning method, initially all weights are set equally, but on each round the weights of incorrectly classified examples are increased so that the weak learner is forced to focus on the hard examples in the training set.

Some authors, [41], proposed *Cost-curves*, where the x-axis represents the fraction of the positive class in the training set, and the y-axis represents the expected error rate grown on each of the training sets. The training sets generated by over(under)sampling. The error rate not represented are constructed by interpolation. They define two cost-sensitive components for a machine learning algorithm: firstly by producing different classifiers for different distributions and secondly by choosing the most appropriate classifier for right distribution. however, when the misclassification costs are known, the x-axis can represent the "probability cost function", which is the normalized product of $C(-|+) * P(+)$; the y-axis represents the expected cost. In Boosting, the classifiers in the ensemble are trained serially with the weights on the training instances adjusted adaptively according

to the performance of the previous classifiers. As above-mentioned, AdaBoost updates the weights of examples according to the misclassification costs. SMOTEboost embeds SMOTE procedure during boosting iterations.

To summarize, when misclassification cost are known, the best metric for evaluating classifier performance is total cost., Total cost=(fn*C_{fn}) + (fp*C_{fp}), [74].

Oversampling and under-sampling can be used to alter the class distribution of the training data. The disadvantage with under-sampling is that it discards potentially useful data. The main disadvantage with oversampling is that by making exact copies of existing examples, it makes over-fitting likely.

A reason that may have contributed to the use of sampling rather than a cost-sensitive algorithm is that misclassification costs are often unknown.

Some authors, [74], cannot conclude that the degree of class imbalance favours one method over another(comparison between cost-sensitive, oversampling and under-sampling). Oversampling appears to be the best for small datasets, cost-sensitive for datasets more than 10,000 examples. They found that which sampling method performs best is highly dependent on the dataset, with neither method a clear winner over the other.

Other researchers, [53], stated that while cost-based methods are, in some cases, reported to perform better than random re-sampling approaches, they do not have the flexibility offered by the sampling methods.

In a work, [76], internal approaches are considered to have the disadvantage of being algorithm specific and it might be quite difficult to transport the modification proposed for the class imbalance problem from one classifier to the other. On the contrary, external approaches are independent of the classifier used and are, thus, more versatile, . Similar conclusions in [75], where they found that there is no general support for considering cost-sensitive learners to outperform sampling for obtaining the best classifier performance (calculated by the lower total cost), but that it is verified in presence of larger datasets where it is possible to generate accurate probability estimates.

Another school of thought is a *recognition based approach* in the form of a *one-class learner.* In a Recognition-based or one-class learning approach, the classifier is modelled on the examples of the target class (the small class) in the absence of examples of the non-

target class. One of the early systems that utilizes this recognition-based approach was proposed in [26]. It uses neural networks and attempts to learn only from the target class examples and thus recognizing the target concept, rather than differentiating between majority and minority instances of a concept. The *one-class* learners provide an interesting alternative to the traditional discriminative approach, where in the classifier is learned on the target class alone ([77], [63], [78]). In particular, [63], focused on extreme imbalance where the minority class consists of around 1-3% of the data.

They considered the possibility of single class learning with support vector machine and they found that positive one-class learners (SVM, linear kernel) perform significantly better than two-class learners.

One-class learning approach has been also applied to auto-encoder-based classifiers ([87]) and ensemble one-class classifiers. Similar patterns from positive instances of a concept are learnt, classifiers are then presented with unseen samples and classification is accomplished by imposing a threshold on the similarity value.

Since threshold draws the boundaries that separate the two classes, choosing an effective threshold is crucial in one-class learners.

Coming back to a general overview on results provided by research works for assessing the class imbalance problems, in their experimental analysis, [27], concluded with the result that it is the class imbalance and not the decrease in overall training set size that caused a decrease in classification accuracy. Furthermore, from other experiments they sorted out that rather than being a problem because of the relative size of the large and the small class, the class imbalance becomes a problem only when the size of its small class is very small with respect to the concept complexity, that is when it contains very small sub-clusters. They concluded that in very large domains in which there is a good chance that the sub-clusters of each class are represented by a reasonable number of examples, the class imbalance will be of no consequence and contrarily if considering smaller domains in which small sub-clusters will be present. In their work they found, on simulated datasets, that under-sampling is by far the least effective method, but in their work the role of the positive and the negative class is symmetrical and no example are irrelevant. They found oversampling appears to be quite effective ways of dealing with the problem,

because the false positive rate decreases (the oversampled one) and the false negative does not significantly increase indicating that the oversampling does not shift the error distribution and preserve a low false negative error rate while learning the false positive error rate. They found that modifying the relative cost of misclassifying each class allows to achieve the same results of oversampling without increasing the training set size. They also found that MLPs (multilayer perceptron) do not seem to suffer from the class imbalance problem in the same way classification trees do, and that are affected both by oversampling and under-sampling, and under-sampling results less effectiveness.
Other authors [47] stated the contrary, that means under-sampling results a more effective strategy than over-sampling. In those works, the authors considered the minority class as the class of interest(as in distress corporate model) and the under-sampling was applied on the majority class, that included a lot of data irrelevant to the classification task that are worth eliminating by under-sampling techniques.
As concerns the effect of class distribution on classifier learning, there are different works focused on the correct class distribution to adopt in learning process and on verifying the effectiveness of using natural data distribution.[33] Some authors [48] stated that the naturally occurring class distribution often is not the best for learning and often substantially better performance can be achieved by using a different class distribution.
According to them, minority-labelled classification rules perform worse than their majority labelled counterparts in part because the test set contains more majority-class examples than minority-class ones. A second reason that classifiers perform worse on the minority-class test examples is that, all else being equal, a classifier is less likely to fully flesh out the boundaries of the minority concept in the concept space because there are fewer examples of that class to learn from. When learning from the balanced versions of the unbalanced datasets, the induction algorithm generally produces fewer but more accurate classification rules for the minority class than for the majority class. They evaluated 12 minority class distributions at different percentage: 2%, 5%, 10% 20%...80% 90% 95%, evaluating the performance additionally compared to the naturally occurring class distri-

[33]"It has been a tacit assumption in much machine learning research that the naturally occurring class distribution is best for learning. However, this assumption has been coming under increased scrutiny, partly because many of the data sets now being learned from have high degree of class imbalance." [48]

bution, in terms of error rate and AUC. Rather than trying to determine the optimal class distribution, they try to identify an optimal range of class distributions. They expected classifier performance to be unimodal with respect to changing class distribution and that an optimal distribution exists and that as we move further away from this optimal distribution, in either direction, classifier performance will degrade progressively. Their results confirmed that hypothesis generally for AUC. They performed t-tests to compare, for each dataset, the performance of the classifiers. If a t-test yields a probability of $\leq .10$ then they concluded that the best distribution is statistically different from the other distribution, otherwise they cannot conclude that and therefore group the distributions together to form an optimum range, within which they were confident that the true optimum class distribution fell. They were interested in whether the optimum range included the natural distribution. In general they found that, for AUC, the optimum ranges appear to be centred to the right of the 50:50 class distribution, so between 50% and 90%. In this situation, the strategy of always allocating half of the training examples to the minority class, while it will not always yield optimal results, will generally lead to results which are not worse than, and often superior to, those which use the natural class distribution. Thus, if one does not know the true misclassification costs and is unwilling to determine experimentally the optimal training distribution, they suggest that for maximizing AUC the training set be formed from equal numbers of examples of each class.

However, their work does not examine the effect of concept complexity nor training set size in the context of their relationship with class imbalances, nor does it look at ways to remedy the class imbalance problem or the effect of class imbalances on classifier other than the decision tree classifier.

From further analysis, [34], it results that, in same cases the best distribution does not differ from the natural one in any consistent manner and that, in some case, a balanced class distribution also is not the best class distribution for training, to minimize undifferentiated error rate. They also found that, in some case, the error rate values curve (according to different percentage of unbalancing) usually form a unimodal, or nearly unimodal, distribution and that the best distribution that minimizes errors is not balanced, since the classifiers induced from class distributions deviating from the naturally occurring distri-

bution would be improperly biased. Then they consider AUC, instead of accuracy, and they found that a balanced class distribution generally performs well although it does not always performs optimally. AUC, unlike error rate, is unaffected by the class distribution of the test set. So, the curve generated using the balanced class distribution almost always outperforms the curve associated with the natural distribution.

Moreover, they found that the performance curves tend to flatten out as the size of the dataset grows, indicating that the choice of class distribution may become less important as the training-set size increases.

Other authors, [34], proposed a method for adjusting the posterior probabilities to account for the difference between r_train (the fraction of positive examples) and p (the true prior probability). Larger AUC values indicate generally better classifier performance and, in particular, a better ability to rank cases by likelihood of class membership. More generally, induction algorithms that maximize accuracy should be biased to perform better at classifying majority-class examples than minority-class examples, since the former component is weighted more heavily when calculating accuracy. Therefore, the training data are less likely to include enough instances of all of the minority-class sub-concepts in the concept space, and the learner may not have the opportunity to represent all truly positive regions. Because of this, some minority-class test examples will be mistakenly classified as belonging to the majority class.

As concerns the above-mentioned data concept complexity[34] and small disjuncts, there are several works related and focused on it. Small disjuncts are those disjuncts in the learned classifier that cover few training examples.

Some authors [79]investigated the problem of over-fitting in case of sparse data, whereas others [44] linked the relationship between the problem of over-fitting the data and dealing with class imbalances. In a work, [62], the author found that no matter what the size of the training set is, linearly separable domains (concept complexity level c=1) do not appear sensitive to any amount of imbalance and as the degree of concept complexity increases so does the system's sensitivity to imbalances. Furthermore, as the size of the

[34]The concept complexity corresponds to the number of sub-clusters into which the classes are subdivided.

training set increases, the degree of imbalance yielding a large error rate decreases. This suggests that in very large domains the class imbalance problem may not be a hindrance to a classification system. Her study suggests that the imbalance problem is a relative problem depending on both the concept complexity represented by the data in which the imbalance occurs and the overall size of the training set, in addition to the degree of class imbalance present in the data. Briefly, a huge class imbalance will not hinder classification of a domain whose concept is very easy to learn nor will we see a problem if the training set is very large. On the contrary, a small class imbalance could greatly harm a very small dataset or one representing a very complex concept. According to her results, rather than being a problem because of the relative size of the large and the small class, the class imbalance problem is only a problem when the size of its small class is very small with respect to the concept complexity, when it contains very small sub-clusters-that means that in very large domains in which the sub-clusters of each class are represented by a reasonable number of examples, the class imbalance will be of no consequence.

In other works, [83], the authors found that class imbalance causes a sharp decrease in accuracy given a single target concept complexity. The more complex the target the more negative the effect class imbalance is.

When we cope with the topic of concept learning and small disjuncts, we have to consider that, firstly, many concepts include rare or exceptional cases and it is desirable for induced definitions to cover these cases, even if they can only be covered by augmenting the definitions with small disjuncts. Secondly, small disjuncts constitute a significant portion of an induced definition.

As depicted by some authors, [50], there exist three kinds of approach to the problem of small disjunct:[35]

Approach 1: The most direct means of eliminating error-prone small disjunct is to eliminate all small disjuncts by explicitly refusing to create disjuncts whose coverage is below a certain threshold. The first objection to this method is that it has the undesirable effect of creating definitions that do not include the unusual cases of a concept (represented by

[35]Classification methods have a tendency to over-fit and misclassify the examples represented by small disjuncts, [50].

small but significant disjuncts). Secondly, eliminating small disjuncts from a definition may significantly increase the definition's error rate.

Approach 2: Significance testing and error rate estimation to determine whether or not including a disjunct in a definition. Disjuncts whose coverage is too low do not pass significance tests. Error rate can only be estimated and suffers from the same problem. But mixed procedure is desirable, if we consider that for small disjuncts error rate is not related to significance in any simple way.

Approach 3: To select among disjuncts that are indistinguishable on the basis of the training set, it is possible to use different bias for large (maximum generality-the opposite of specificity) and small (a selective specificity bias) disjuncts.

Rare cases, like rare classes, can be considered the result of a form of data imbalance and have in fact been referred to as within-class imbalances.

Much of the research on rarity relates to rare classes, or, more generally, class imbalance. This type of rarity requires labelled examples and is associated with classification problems. A second type of rarity concerns rare cases. Rare cases correspond to a meaningful but relatively small subset of the data, or equivalently, define a small region of the instance space. Rare cases depend only on the distribution of the data and therefore are defined for both labelled and unlabelled data, and for supervised and unsupervised data mining tasks. In the case of labelled data, a rare case corresponds to a sub-concept, or sub-class, that occurs infrequently. Unfortunately, except for artificially generated domains, rare cases are not easily identified. However, unsupervised learning techniques such as clustering may help to identify, and they also manifest themselves, as small disjuncts in classifiers induced from the data.

There have been several attempts, [36], to improve the performance of data-mining systems with respect to rarity by choosing a more appropriate bias. The simplest approach involves modifying existing systems to eliminate some small disjuncts based on tests of statistical significance or using error estimation techniques. The hope is that these will remove only improperly learned disjuncts. Unfortunately, this approach was shown not only to degrade performance with respect to rarity, but also to degrade overall classification performance. More sophisticated approaches have been developed and to judge their

efficacy on rare cases it is considered whether they improve the performance of the small disjuncts-based on the assumption that rare cases manifest themselves as small disjuncts. Segmenting the data is one way to deal with rarity, by separating the problem into separate sub-problems and analysing them separately. In many data mining tasks, it is the rare classes/cases that are of primary interest. One solution is to use cost-sensitive learning methods. These methods can exploit the fact that the value of correctly identifying the positive (rare) class outweighs the value of correctly identifying the common class. For two-class problems this is done by associating a greater cost with false negative than with false positives. Assigning a greater cost to false negatives than to false positives will improve performance with respect to the positive (rare) class. One problem with this approach is that specific cost information is rarely available. Thus, without specific cost information, it may be more practical to only predict the rare class and generate an ordered list of the best positive-predicting rules. Then one can decide where to place the threshold after data-mining is complete.

Some authors, [53], proposed to approximate the rare cases, using an unsupervised method (k-means clustering) and assuming that ,firstly, the small disjuncts constructed by their unsupervised method do correspond fully to the unknown rare cases of the domain and, secondly, there is a correspondence between the small disjuncts learned by the unsupervised method and those subsequently learned by the supervised method.

In their work, [53], the re-sampling strategy they propose consists of clustering the training data of each class separately and performing random oversampling cluster by cluster. They found that it is the small disjunct problem more than the class imbalance problem responsible for the decrease in accuracy, and the cluster-based oversampling is shown to outperform the other methods.

Induction techniques that deal with rare classes must try to maximize precision and recall. Most induction techniques try to optimize these two competing measures, PNrule, [95], uses two-phase rule induction to focus on each measure separately. In the first phase, if high precision rules cannot be found then lower precision rules are accepted, as long as they have relatively high recall. So, the first phase focuses on recall. In the second phase precision is optimized. This is accomplished by learning to identify false positives within

the rules from phase one. The presence of the second phase permits the first phase to be sensitive to the problem of small disjuncts, while the second phase allows the false positives to be grouped together addressing the problem of data fragmentation. Different rare cases may have little in common between them, making it difficult for one learner to assign the same class value to all of them.

One possible solution is to reformulate the original problem so that rare cases are viewed as separate classes. In their work, [53], the re-sampling strategy they propose consists of clustering the training data of each class separately and performing random oversampling cluster by cluster. Their implemented approach consists in: 1) separating each class into subclasses using clustering, 2) relabelling the training examples based on the subclasses (clusters) and then 3) re-learning on the revised training set. They found that it is the small disjunct problem more than the class imbalance problem responsible for the decrease in accuracy, and the cluster-based oversampling is shown to outperform the other methods.

The class imbalance problem is more significant when the data sets have a high level of noise. Noise in datasets can emerge from various sources, such as data samples are poorly acquired or incorrectly labelled, or extracted features are not sufficient for classification. In particular, one of the sources could be a set of features used for classification not sufficient to draw class boundaries. Data noise in classification problems can be generally described as data examples on different classes inseparable in the feature space. If a dataset is considered noisy, the class boundary to separate different class examples in the feature space is almost impossible to draw.

It is known that noisy data affect many machine learning algorithms; however, it has been showed that noise has even more serious impact when learning with imbalanced data [36]. The problem occurs when samples from the small class are mistakenly included in the training data for the majority class and vice versa. For the prevalent class, noise samples have less impact on the learning process. In contrast, for the small class it takes only a few noise samples to influence the learned sub-concept.

In literature, different approaches have been used for measuring noise levels based, generally, on inter and intra class distances. In a work, [55], the authors tested three different neural networks BP, RBF and Fuzzy ARtmap, and they found that classifiers performance

on unbalanced data very much depends on how well the two classes are separated, and that when data is noisy, none of the three networks performs well without any additional processes over unbalanced data, by generating artificial noisy data examples along the classification boundaries.

To cope with the within-class imbalance, other procedure, including clustering have been proposed. In a work, [97], it is presented a procedure that considers no re-sampling by focusing on sub-clustering analysis. The procedure consists in:

1) separating each class into a number of sub-classes, using an unsupervised learning technique (clustering) and re-labelling each training example as a function of these new subclasses;

2) applying supervised learning to various versions of these new problems;

3) the results obtained on each version are combined in a decisive vote.

In another work, [51], the author attempted to set procedures for dealing simultaneously with both types of unbalances with stratification methods (re-sampling or downsizing approaches) and the misclassification rate of multi-layer perceptrons classifier in case of symmetrical and asymmetrical sub-clusters composition. The optimal procedure consisted in creating both between-class than within-class balanced data.

Other authors, [82], proposed a learning approach for imbalanced dataset consisting in a under-sampling method based on clustering. They applied a clustering technique to partition the training instances of each class independently into a smaller set of training prototype patterns. Then a weight is assigned to each prototype to address the class imbalance problem. The weighting strategy is introduced in the cost function such that the class distributions become even. After clustering, the dataset is reduced to K exemplars; each represented by a cluster centroid and same cluster size for all the sub-clusters. In other words, they chose to select the cluster centroids as representative samples. Then they apply a feed-forward neural networks for the supervised classification.

In another work, [96], the authors proposed a selection procedure MFMP involving selective sampling and random sampling of the majority class according to the minority clusters and test four learners. They found that for moderately unbalanced datasets where minority class exhibits cluster nature their approach outperforms random under-sampling.

Recently, connected to the topic of within-class unbalance Elkan pointed out that ROC curves could result unable to deal with within-class imbalances and different within-class misclassification, suggesting for a revision.[36]

It was argued that the reason why class imbalances and overlapping classes are related is that misclassification often occurs near class boundaries where overlap usually occurs as well. It is important to select features that can capture the high skew in the class distribution, [81], [80].

There are not comprehensive empirical studies that evaluate all of the existing methods. Sampling techniques have generated the most research in this area and there are a few studies that compare sampling methods. Unfortunately, even in this case the conclusions are not consistent. One study, ([48]) on class distribution shows that by altering the class distribution of the training data so that it deviates from the natural, underlying distribution, improved classifier performance is possible. However, classifier performance was improved more when the bias just described was removed by adjusting the decision thresholds within the classifier.

[36] For a detailed overview of the literature debate on class unbalance problem, see [54].

3.7 Case study C: Neural networks distress model on unbalanced data

3.7.1 Outlines

Distress prediction analysis is characterized by the presence of an outstanding unbalance between a minority class of failed firms and a larger class of not-failed ones.
In the previous literature on class imbalance problem, applied to several real-world domains, the majority of research has been focused on testing and analysing effects of imbalance on Decision tree classifier and only few works referred specifically to neural networks, see [26] [77] [27] [55][82].
The following work attempts to verify the effect of the class imbalance problem on the accuracy of a neural network classifier. In specific, the predictive accuracy is tested on a neural network model for a distress analysis prediction, at different imbalance levels.
Some previous works concern on evaluating classifiers' performance at different imbalance degrees compared to the natural class distribution, [48] [34].
The present work aims at verifying the effect of imbalance on neural network model's accuracy, in term of AUC, as we move from a resized balanced distribution towards the naturally occurring distribution.

3.7.2 Analysis on unbalanced dataset

The analysis starts from the original unbalanced sample of of 774 failed companies and 38,480 healthy ones large industrial Italian companies, period 2004-2008 (Source Amadeus), used in the previous case study B. The aim of the analysis is to test the effects of different imbalance levels on overall classifier performance.
The imbalance levels compared to the balanced benchmark one will be:

- level A: 40% minor class (failed) and 60% larger class (not-failed)

- level B: 30% minor class (failed) and 70% larger class (not-failed)

We would analyse the presence of an increase/decrease in accuracy in the case of augmented imbalance levels. In the previous case-study B, we have processed a neural network model on a balanced dataset, composed of 570 companies. In the following work, we will resize the original sample at different imbalance degrees by increasing the size of the balanced sample by maintaining the minor class size invariant and applying a random under-sampling on the larger natural class (the not-failed one) to obtain the two chosen imbalance levels A and B. As in the balanced case analysis, for both imbalance levels we considered two datasets in order to analyse the prediction at two different time lags: T1, referred to one-year prior to failure financial statement data and T2, referred to two-years prior to failure ones. We proceeded with the analysis by splitting the data into two sub-samples: a training dataset, to estimate the model parameters of the classifier, and a control dataset, to evaluate the ability of the estimated model in predicting different cases not in the training sample. The training dataset is made up of the 70% of the entire sample, whereas the control dataset covers the 30%.

Further, the related ROC curve area is considered as an accuracy evaluation criterion of the obtained prediction [28] and to evaluate the influence of imbalance on classifier performance compared to the benchmark balanced dataset. The dataset referred to the unbalance level A is composed of 713 units, corresponding to an unbalance sample of 40% of failed companies and 60% of healthy ones. Whereas, the dataset referred to the unbalance level B is composed of 950 units, corresponding to an unbalance sample of 30% of failed companies and of 70% of healthy ones.

As in the case study B, we considered six input variables, consisting of financial and economic ratios, commonly regarded both by banks and scholars as distress key-indicators, to capture both the financial and the economic perspective, see Table 3.2.

To recall the economic interpretation of the selected variables, the *Refunding capability* expresses the potentiality of the company to generate positive cash flows to cover financial obligations, (Financial debt coverage ratio). The *Growth* indicates an increasing economic dynamic, (Sales variation ratio). The *Debt cost* expresses the degree of the

economic incidence of the debt exposure, (Interest paid on sales ratio). The *Indebtedness* indicates the debt exposure level, (Leverage). The *Efficiency* expresses the capability to generate operating returns, (Ebit on sales ratio). The logarithm of Sales represents a size control variable. The output variable value 1 corresponds to an healthy company whereas the value 0 is related to a failed one.

3.7.3 Numerical results for unbalanced data

We iterated the estimation process on the training dataset for combinations of hidden nodes from 2 to 10 and values of the decay parameter: 0.1, 0.01, 0.05, 0.005. The optimization process is done via a quasi-Newton method. The initial parameter vector has been chosen at random but setting the same random seed for every combination. We obtained 36 models and we computed the *empirical error* on the training set for each model estimated, for both unbalance level A and B datasets in period T1 and T2. We calculated the related goodness-of-fit criteria AIC, BIC and GCV, by considering the number of degree of freedom equals to the number of weights of the model.
We selected three best models according to each above-mentioned selection criterion. Once the training process is run, the best NN models selected were used for prediction applied to the control dataset. By considering the NN models prediction vectors and the output variable, we traced the ROC curve in order to compare the forecasting performance of the three models at the different unbalance levels (50%, 40% and 30% of minor class elements) and the dominance of a curve on the others in T1 lag period, as shown in Fig. 3.3. The same procedure has been applied to the T2 lag period data, see Fig. 3.4.
Furthermore, we calculated the area under the ROC curve, to objectively compare the models by considering the curve in its entirety.
As regards the balanced model in T1, the AIC and GCV criteria selected the same best model with three hidden nodes and decay equals to 0.005, presenting an higher value of the area under the ROC curve than the model selected by the BIC criterion, see Table 3.8.
In the same way, as concerns the unbalance level A model (40% minor class) in T1, the AIC and GCV criteria selected the same best model with three hidden nodes and decay

	bestvalue	nodes	decay	Area	p-value	binorm.area
AIC_{50}	4.086	3	0.005	**0.8968**	3.3e-19	0.8927
BIC_{50}	106.048	2	0.005	0.8634	2.1e-16	0.8755
GCV_{50}	59.756	3	0.005	**0.8968**	3.3e-19	0.8927
AIC_{40}	4.18	3	0.005	**0.8903**	4.4e-22	0.8917
BIC_{40}	109.975	2	0.005	0.854	1.7e-18	0.8688
GVC_{40}	65.559	3	0.005	**0.8903**	4.4e-22	0.8917
AIC_{30}	4.346	4	0.005	**0.887**	4.2e-25	0.8888
BIC_{30}	114.944	2	0.005	0.8593	6.6e-22	0.874
GVC_{30}	77.395	4	0.005	**0.887**	4.2e-25	0.8888

Table 3.8: T1 (one-year prior to failure). Area under the ROC curve for best selected Neural Networks in 50, 40 and 30 unbalance levels. The first column contains the selection criteria best values. The p-value addresses to the null hypothesis H_0: ROC Area=0.5. The sixth column contains the binormal curve area.

equals to 0.005, with higher AUC value than the one selected by the BIC criterion.

As concerns the unbalance level B model (30% minor class) in T1, the AIC and GCV criteria selected the same best model with four hidden nodes and decay equals to 0.005.

As regards the balanced model in T2, the AIC and GCV criteria selected the same best model with four hidden nodes and decay equals to 0.005, presenting an higher value of the area under the ROC curve than the model selected by the BIC criterion, see Table 3.9. Whereas as concerns the unbalance level A model (40% minor class) in T2, the AIC, BIC and GCV criteria both selected a best model with two hidden nodes and decay equals to 0.005. As concerns the unbalance level B model (30% minor class) in T2, the AIC and GCV criteria selected the same best model with six hidden nodes and decay equals to 0.005, with higher AUC value than the one selected by the BIC criterion.

For both two lag periods, in all the considered models, the criteria AIC and GCV tend on average to select neural networks presenting dominant ROC curves and higher AUC.

A classifier A performs better than a classifier B if it is located to the north-west area of B in ROC space. The overall quality of classifier is measured by the AUC.

Based on the AUC values, we notice that, in period T1, the balanced class distribution generates a slightly superior overall classifier compared to both the other two ones. The superiority is more evident if we consider the curves in the ROC space, where NN_{50}

	bestvalue	nodes	decay	Area	p-value	binorm.area
AIC_{50}	4.313	4	0.005	**0.8202**	3.9e-13	0.8151
BIC_{50}	106.2	2	0.010	0.7684	9.5e-10	0.7446
GCV_{50}	75.219	4	0.005	**0.8202**	3.9e-13	0.8151
AIC_{40}	4.459	2	0.005	**0.7909**	4.4e-13	0.7794
BIC_{40}	110.108	2	0.005	**0.7909**	4.4e-13	0.7794
GVC_{40}	86.535	2	0.005	**0.7909**	4.4e-13	0.7794
AIC_{30}	4.635	6	0.005	**0.8047**	2.8e-16	0.8052
BIC_{30}	115.147	2	0.005	0.8003	7.4e-16	0.7975
GVC_{30}	103.576	6	0.005	**0.8047**	2.8e-16	0.8052

Table 3.9: T2 (two-years prior to failure). Area under the ROC curve for best selected Neural Networks in 50, 40 and 30 unbalance levels. The first column contains the selection criteria best values. The p-value addresses to the null hypothesis H_0: ROC Area=0.5. The sixth column contains the binormal curve area.

curve outperforms both the NN_{40} and NN_{30} curves in the north-west area associated to cut-off (thresholds) values up to 0.5, see Fig. 3.3.[37] The hierarchy between NN_{40} and NN_{30} curves, as regards the AUC values, results towards the superiority of the NN_{40} (less unbalanced), even if in a part of the north-west ROC space NN_{30} is outperforming, probably due to the trade-off between unbalance level and increasing number of elements in the training process. It is important to underline that the natural distribution of the minor class, in the original real-world dataset, is strongly more unbalanced (about 2%), so our analysis simply is aimed at detecting the tendential effect on classifier performance of an increase in the unbalance level.

The results in T1 (one-year prior to failure) are likely to confirm a tendency of the classifier accuracy to decrease when we move towards the natural distribution even if this effect seems to be smoothed over by the counter-effect of the benefit of an augmented number of units in the training process. Based on the AUC values, we notice that, also in period T2, the balanced class distribution generates a quite slightly superior overall classifier compared to both the other two ones. As a premise, it is not secondary to un-

[37] NN_{50}, NN_{40} and NN_{30} correspond, respectively, to the neural network model processed on the balanced dataset, the dataset with an unbalance level of 40% in the minor class and the one with an unbalance level of 30% in the minor class.

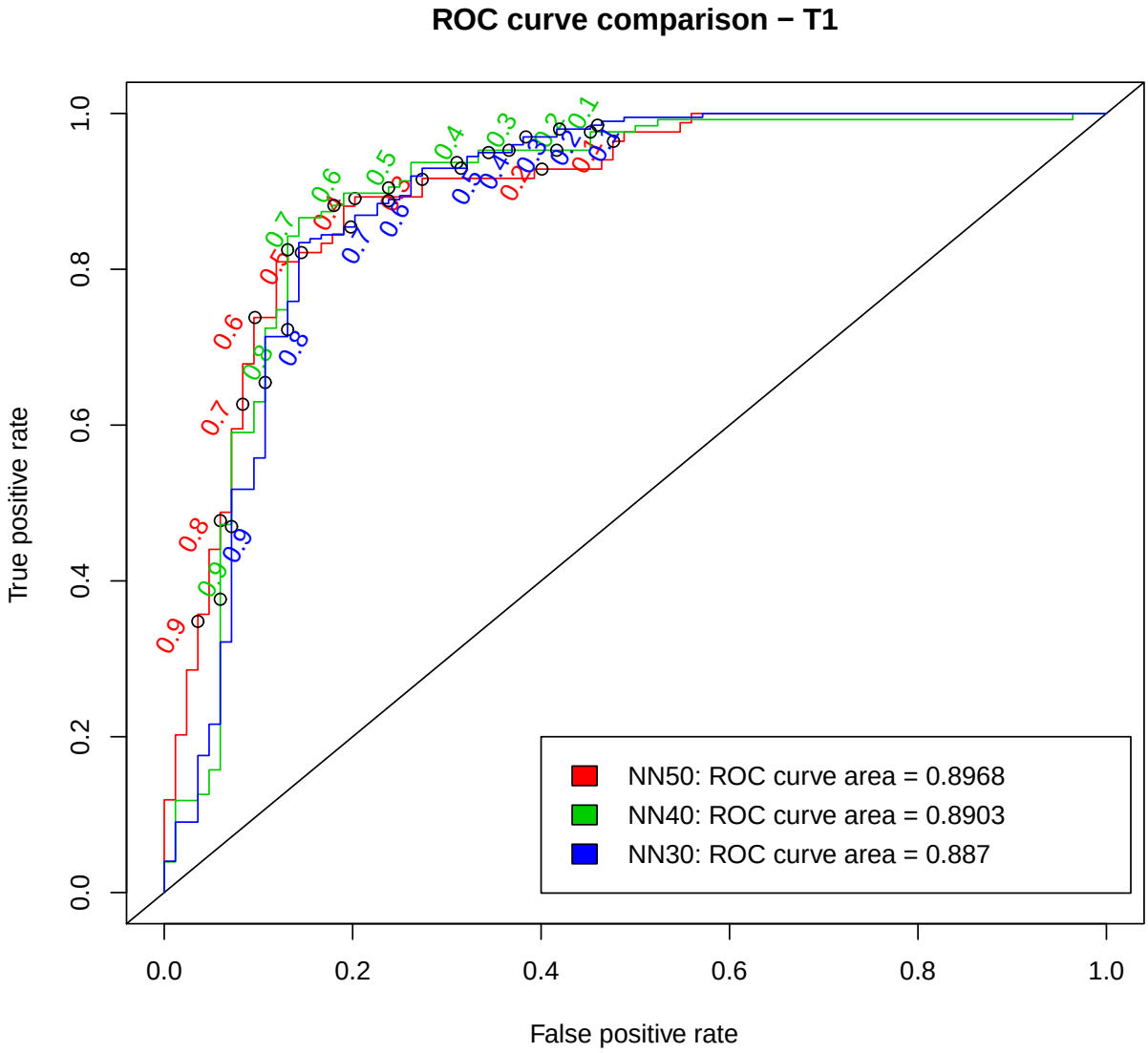

Figure 3.3: T1- ROC curve comparison between NN50 (3 hidden nodes and decay=0.005, balanced), NN40 (3 hidden nodes and decay=0.005, unbalance level=40% minority class) and NN30 (4 hidden nodes and decay=0.005, unbalance level=30% minority class). The cutoff values are indicated along the curve at the corresponding curve positions.

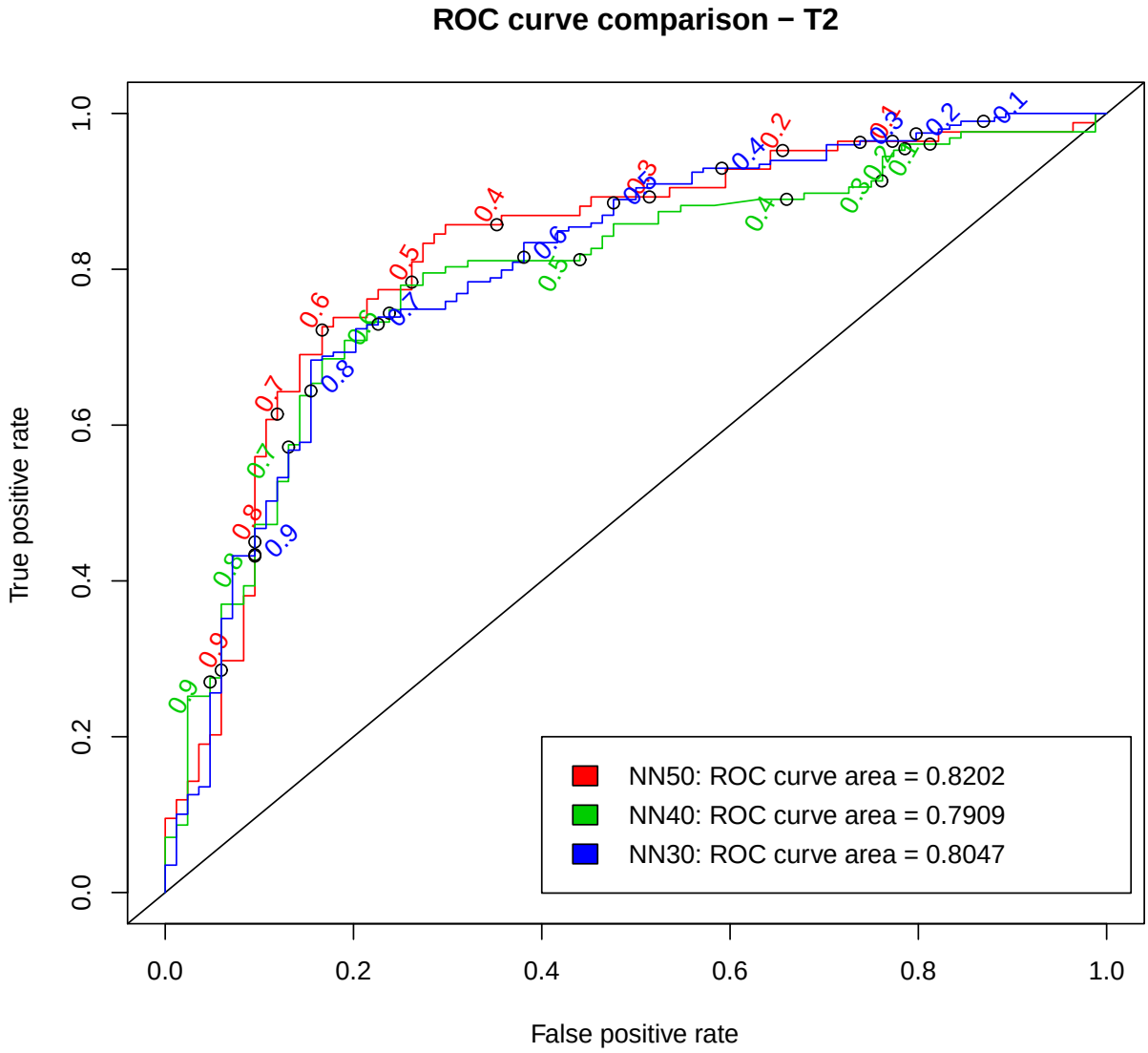

Figure 3.4: T2- ROC curve comparison between NN50 (4 hidden nodes and decay=0.005, balanced), NN40 (4 hidden nodes and decay=0.005, unbalance level=40% minority class) and NN30 (6 hidden nodes and decay=0.005, unbalance level=30% minority class). The cutoff values are indicated along the curve at the corresponding curve positions.

derline that, from the economic perspective, the information connected to more delayed lag-period (two-years prior to failure) are less worthy than the information immediately close to the "event" into consideration (one-year prior to failure). As a consequence, the results related to the second order lag could be less reliable and partly unstable. If we consider the curves in the ROC space, NN_{50} curve outperforms both the NN_{40} and NN_{30} curves in the majority of the north-west area associated to cut-off (thresholds) values up to 0.5, see Fig. 3.4. In few parts of the the north-west area, the superiority of one curve on the others is not clear, and the curves tend to overlap. The hierarchy between NN_{40} and NN_{30} curves, as regards the AUC values, results towards a slightly superiority of the NN_{30} (more unbalanced), probably influenced by the increasing number of elements in the training process.

The results in T2 (two-years prior to failure) are likely to confirm a tendency of the classifier accuracy to decrease when we move towards the natural distribution even if this case the counter-effect of the benefit of an augmented number of units in the training process seems to more affect the performance of the classifier.

3.7.4 Comments

In the work, we investigated the effect of the unbalance in the performance of a neural network model for distress prediction. The real-world dataset provided had a natural distribution of the minor class (failed companies) of about 2%; and the analysis processed aimed at finding a tendency (increase/decrease) on classifier performance when moving from a balanced benchmark, 50% of elements belonging to the minor class, to 40% and 30% minor class units. The results provided evidence of a tendency of the classifier accuracy to slightly decrease when we move towards the natural distribution even if this effect seems to be smoothed over by the counter-effect of the benefit of an augmented number of units in the training process. The balanced procedure superiority are likely to be verified more in T1 (one-year prior to failure) lag period than in T2 (two-years prior to failure) one. In T1 the results are confirmed both according to the AUC accuracy criterion than from the analysis of the curve in the ROC space. In T2, NN_{50} is outperforming

according to AUC criterion, but at certain points the ROC curves are overlapping. Similar results have been obtained by Weiss et al. (2001), [48].[38]

Further research would focus on investigating the effect of the natural unbalance degree on the performance of the classifier, aiming at finding the optimal distribution for the minor class in problem, like corporate distress, where the costs associated with the incorrect classification of the minor class elements are relevant.

[38]They investigated the effect of unbalance on a decision tree classifier's performance and concluded that "..the strategy of allocating half of the training examples to the minority, while it will not always yield optimal results, will generally lead to results that are no worse than, and often superior to, those which use the natural class distribution".

Chapter 4

Conclusion and further research

In the **first part** of the work, we focused on unsupervised methodologies for corporate analysis, in particular a classification method, model-based clustering, and its application on a industry sector segmentation.
In corporate analysis framework, this preliminary procedure may result relevant in a ”scenario” analysis because of the fragmentation of information[1] and moreover key-variables used for distress analysis differ from a sector to another.
In case-study A, we have followed a procedure by applying the model-based clustering to the scores of a PCA on a set of financial and economic ratios, a procedure usually used in literature. The results provided by our analysis have shown that a clustering procedure applied on a specific industrial sector reports a segmentation according to the financial and economic level of the companies. By applying this procedure, it would be possible to have ”financial level” information on the analysed sector and a segmentation of it, also providing correspondent key-indicators average values. Model based clustering is a more flexible methodology, compared to the other clustering methods, because every unit is assigned to every of the n-group with a posterior probability, so for each company it is provided a probability of belonging to a specific ”financial level” class (for example: weak, average, strong). Furthermore, we found that model based clustering tends to de-

[1] Often there are not only two groups of companies: failed and not failed, but also intermediate stages, but data about them are not provide or easily disclosed. Bankruptcy is a legal procedure, formally regulated, but sometimes delayed with respect to the solvency event occurrence.

tect more clusters than K-means. Similar findings have been provided by Atkinson et al. (2010), see [23]. According to the reported classification of the companies belonging to a specific level-class, further analysis could be conducted. For example, a supervised analysis could be applied taking into consideration the specific segmentation of the industrial sector to be analysed, by considering the presence of a specified number of classes to be predicted according to the financial-level classes. Moreover, if we consider the problem of class imbalance, and in specific the within-class unbalance, it would be possible to apply re-sampling methods on the clusters obtained, see [52]. Our intent is both to extend the analysis to other industrial sectors and to consider a different procedure, consisting in the simultaneous combination of dimensionality reduction and clustering operation, see [22]. Further analysis may focus on verifying misleading association, signalled by the presence of components with few elements, or units with not very high posterior probability of belonging and not very well separated groups, that could be connected with the merging problem of normal distributions or not Gaussian distributions. Moreover, both dispersed few elements group or very low probability of belonging of an element to a cohesive group could indicate the presence of potential outliers. We intend to proceed with further research in order to provide a more robust model based approach for clustering, by considering mixture of *t* distributions instead of Gaussian mixture, see [24] or other robust clustering methodology.

In the **second part**, we focused on the supervised methodologies for corporate distress analysis with the case-study B, aiming at comparing the forecasting accuracy of two classifiers, Logit and Neural Networks, and the case-study C, focused on the problem of class unbalance effect on classifier (Neural network) performance.

The comparative analysis carried on in case-study B, showed the superiority of the neural networks models over the logistic model, in term of prediction accuracy.

The neural networks over-performing is relevant for the costs of wrong prediction in credit scoring model, otherwise, the logistic regression offers a more readable economic interpretation of the input variables effect on the output vector. Leverage and Ebit/Sales variables resulted strongly significant, differently the other key-variables, maybe due to the Italian disclosure rules for the Financial statement items.

Further research may be conducted aiming at implementing a more stable predictors frame, also for other European countries.

Further investigations may focus on input variables sensitivity analysis to interpret the impact of predictors on the output vector in neural networks, [139] [32]. As concerns the methodological procedure, further developments may be conducted aiming at improving the tuning in the neural network optimization process and in managing with unbalanced data. As concern the case study C, the work focused on analysing the effect of the unbalance in the performance of a neural network model for distress prediction aiming at finding a tendency (increase/decrease) on classifier performance when considering unbalance levels differing from the balanced benchmark, 50% of elements belonging to the minor class. The unbalance levels considere have been: 40% and 30% minor class units. The results showed a tendency of the classifier accuracy to slightly decrease when we move towards the natural distribution even if this effect seems to be equilized by the benefit, in the training process, connected to an increase in the number of units for the larger class. Similar results have been obtained by Weiss et al. (2001), [48]. Further research may focus on finding an optimal distribution for the minor class in particular for real-world application, like credit scoring, because of the costs associated with an incorrect classification of the minor class elements.

Appendix A

MCLUST: an R function for model based clustering

MCLUST Version 3 contributed R statistical package by Chris Fraley and Adrian Raftery (2006), [109], provides functionality for model-based clustering. MCLUST provides iterative EM methods for parameter maximum likelihood estimation in parametrized, with a variety of covariance structures, Gaussian mixture models. At each EM iteration, the E-step computes a matrix $\mathbf{z}$ such that $\mathbf{z}_{ik}$ is an estimate of the conditional probability that observation *i* belongs to group *k* given the current parameter estimates, and the M-step computes parameter estimates given z. MCLUST functions *em* and *me* implement the EM algorithm for parametrized Gaussian mixtures. Functions *em* starts with E-step; besides the data and model specification, the model parameters (means, covariances and mixing proportions) proportions must be provided. Function *me* starts with the M-step; besides the data and model specification, the conditional probabilities $\mathbf{z}$ must be provided. The output for both are the maximum-likelihood estimates of the model parameters and $\mathbf{z}$. Below, an overview of what MCLUST function does is shown:

```
Mclust(data, G=NULL, modelNames=NULL, prior=NULL,
    control=emControl(), initialization=NULL, warn=FALSE, ···)
```

Such function allows:

- to specify the numbers of mixture components (clusters) for which the BIC is to be calculated, in the option G. The default is G=1:9.

- to specify the models to be fitted in the EM phase of clustering, in the option modelNames. The default is c(”E”, ”V”) for univariate data and mclustOptions() emModelNames for multivariate data (n>d), the spherical and diagonal models c(”EII”, ”VII”, ”EEI”, ”EVI”, ”VEI”, ”VVI”) for multivariate data ($n \leq d$);

- to specify a conjugate prior on the means and variances through the function priorControl. The default assumes no prior;

- to indicate a list of control parameters for EM. The defaults are set by the call emControl();

- to initialize the EM process. It consists of a list containing zero or more of the following components: a) hcPairs: A matrix of merge pairs for hierarchical clustering such as produced by function hc. For multivariate data, the default is to compute a hierarchical clustering tree by applying function hc with modelName = ”VVV” to the data or a subset as indicated by the subset argument. The hierarchical clustering results are to start EM. For univariate data, the default is to use quantiles to start EM; b)subset: A logical or numeric vector specifying a subset of the data to be used in the initial hierarchical clustering phase;

- to set a logical value indicating whether or not certain warnings (usually related to singularity) should be issued. The default is to suppress these warnings.

The function MCLUST is used to obtain the optimal model according to BIC for EM initialized by hierarchical clustering for parametrized Gaussian mixture models.
The function returns a list giving the optimal (according to BIC) parameters, conditional probabilities z, and log-likelihood, together with the associated classification and its uncertainty. In particular, the details of the output components are stored in specific objects: BIC, containing all BIC values referred to the different models, giving the possibility to evaluate the model selection according both to the BIC value and also to the more parsimonious model in terms of number of parameters/complexity of the model and number of components; z, a matrix whose [i,k]th entry is the probability that observation i in the test data belongs to the kth class; classification map(z), the classification corresponding to z; uncertainty, the uncertainty associated with the classification.
The function `plot()` returns the model-based clustering plots: BIC values used for choosing the number of clusters. For data in more than two dimensions, a pairs plot of the showing the classification, a coordinate projections of the data showing location of the mixture components, classification, and uncertainty. For one- and two- dimensional data, plots showing location of the mixture components, classification, uncertainty, and density.

Appendix B

NNET: an R function for neural networks

The NNET R statistical package by Brian Ripley (2009) implements a procedure for fitting feed-forward neural networks.
Below, an overview of what NNET function does is shown:

```
nnet(x, y, weights, size, Wts, mask, linout = FALSE,
    entropy = FALSE, softmax = FALSE, censored = FALSE,
    skip = FALSE, rang = 0.7, decay = 0, maxit = 100,
    Hess = FALSE, trace = TRUE, MaxNWts = 1000,
    abstol = 1.0e-4, reltol = 1.0e-8, ...)
```

Such function allows:

- to specify the number of units in the hidden layer, in the option size. It can be zero if there are skip-layer units;
- to specify the initial parameter vector, in the option Wts. If missing chosen at random;

- to switch for linear output units, in the option linout. Default logistic output units. The activation function in the hidden layer of nnet() function is logistic, it is possible to specify the activation function in the output layer (for example to be linear using linout=T, in case of continuous variable forecasting) but in the analysed case, for binary output variable where the expected value is a probability, it is set as by default logistic;

- to switch for entropy (maximum conditional likelihood) fitting, in the option entropy. Default by least-squares;

- to switch to add skip-layer connections from input to output, in the option skip;

- to set a range for the initial random weights on [-rang, rang], in the option rang;

- to specify a value for weight decay, in the option decay. Default is zero;

- to indicate a maximum number of iterations, in the option maxit. Default is 100.

Optimization is done via the BFGS, a quasi-Newton method.
The function returns an object of class ”nnet”, a structure containing several objects: wts, a vector of the best set of weights found; the value of fitting criterion plus weight decay term; the fitted values for the training data; convergence 1 if the maximum number of iterations was reached, otherwise 0; the residuals for the training data.
NNET is a wrapper for other functions so it is possible to construct a net, where specifying more than one layer of neurons and setting other parameters.
If we consider a procedure in which we estimate different neural net, in terms of number of hidden layers and weight decay values, we proceed to compare them according to the selection criteria BIC-AIC-GCV computing from the residuals obtained. Once we select the best neural net structure, we proceed to evaluate its accuracy on a test set.
In NNET package, it is not possible to specify a validation set as well as a training set, as the neural net methodology would consider: learning set (to estimate the net parameters), validation set (to tune the net parameters) and test set (to evaluate the net performance). NNET function just trains, so the procedure to test the fitted model (the nnet object) against

new data consists in predicting new data, using the function `predict.nnet()`.
It consists in using the fitted model on a test sample and computing, from the residuals, the related performance criteria to evaluate the generalization properties of the net (ROC curve and AUC). The ROC curve and AUC are computed by the function `performance()` in the R package ROCR by T. Sing, O. Sander, N. Beerenwinkel, T. Lengauer (2009).

Bibliography

[1] W. H. Beaver: Financial Ratios As Predictors of Failure. Journal of Accounting Research **4**, Empirical Research in Accounting: Selected Studies 71–111 (1966)

[2] E. Altman: Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy. The Journal of Finance **23**, No.4 589–609 (1968)

[3] R. J. Taffler: Forecasting Company Failure in the UK using Discriminant Analysis and Financial Ratio Data. Journal of the Royal Statistical Society. Series A, **145**, No.3 342–358 (1982)

[4] J. A. Ohlson: Financial Ratios and the Probabilistic Prediction of Bankruptcy. Journal of Accounting Research **18**, No.1 109–131 (1980)

[5] M. E. Zmijewski: Methodological Issues Related to the Estimation of Financial Distress Prediction Models. Journal of Accounting Research **22**, 59–82 (1984)

[6] K. Y. Tam, M. Y. Kiang: Application of Neural Networks: The Case of Bank Failure Predictions. Management Science **38**, No.7 926–947 (1992)

[7] W. S. Chen, Y. K. Du: Using neural networks and data mining techniques for the financial distress prediction models. Expert system with Application **36**, 4075–4086 (2009)

[8] A. I. Dimitras, R. Slowinski, R. Susmaga, C. Zopounidis: Business failure prediction using rough sets. European Journal of Operational Research **114**, 263–280 (1999)

[9] L. Sun, P. P. Shenoy: Using Bayesian Networks for Bankruptcy Prediction: Some Methodological Issues. European Journal of Operational Research **180**,2 738–753 (2007)

[10] T. Lensberg, A. Eilifsen, T. E. Mckeee: Bankruptcy theory development and classification via genetic programming: Some Methodological Issues. European Journal of Operational Research **169** 677–697 (2006)

[11] A. K. Jain, M. N. Murty, P. J. Flinn: Data Clustering: A review. ACM Computing Surveys **31**,No.3 (1999)

[12] A. Azadeh, S. F. Ghaderi, Y. P. Miran, V. Ebrahimipour, K. Suzuki: An integrated framework for continuous assessment and improvement of manufacturing system. Applied Mathematics and Computation **186** 1216–1233 (2007)

[13] K. Rezaie, M. Dehghanbaghi, V. Ebrahimipour: Performance evaluation of manufacturing systems based on dependability management indicators-case study:chemical industry. International Journal of Manufacturing Technology & Management **43** 608–619 (2009)

[14] H. Bensmail, R. De Gennaro: Analyzing Imputed Financial Data: A New Approach to Cluster Analysis. Federal Reserve Bank of Atlanta-Working Paper Series **20**, (2004)

[15] J. Sun, H. Li: Data Mining method for listed companies' financial distress prediction. Knowledge-Based Systems **21**, 1–5 (2008)

[16] M. C. Gupta, R. J. Huefner: A cluster Analysis Study of Financial Ratios and Industry Characteristics. Journal of Accounting Research **10**, No.1, 77–95 (1972)

[17] J. D. Banfield, A. E. Raftery: Model-based Gaussian and Non-Gaussian Clustering. Biometrics **49**, No.3, 803–821 (1993)

[18] S. Ingrassia, R. Rocci: Constrained Monotone EM Algorithms for finite mixtures of multivariate Gaussians. Computational Statistics and Data Analysis **51**, 5339–5351 (2007)

[19] C. Fraley, A. E. Raftery: Model-based Clustering, Discriminant Analysis and Density Estimation. Journal of the American Statistical Association **97**, No.458, 611–631 (2002)

[20] H. Jo, I. Han, H. Lee: Bankruptcy Prediction Using Case-Based Reasoning, Neural Networks, and Discriminant Analysis. Expert system with Application **13**, No.2, 97–108 (1997)

[21] K. Lee, D. Booth, P. Alam: A comparison of supervised and unsupervised neural networks in predicting bankruptcy of Korean firms. Expert system with Application **29**, 1–16 (2005)

[22] M. Vichi, R. Rocci, H. A. L. Kiers: Simultaneous Component and Clustering models for three-way data: within and between approaches. Journal of Classification **24**, 71–98 (2007)

[23] A. C. Atkinson, M. Riani, A. Cerioli: Robust Clustering for Performance Evaluation. In: F. Palumbo et al. (eds.), Data Analysis and Classification, 381–390. Springer-Verlag, Berlin Heidelberg (2010)

[24] D. Peel, G. J. McLachlan: Robust Mixture Modelling using the t distribution. Statistics and Computing **10**, 339–348 (2000)

[25] F.Tseng, H.Hu: Comparing four bankruptcy prediction models: Logit, quadratic interval logit, neural & fuzzy neural networks. Expert Systems with Applications **37**, 1846–1853 (2010)

[26] N. Japkowicz, C. Myers, M. Gluck: A novelty detection approach to classification. IJCAÍ95: Proceedings of the 14th international joint conference on Artificial intelligence, 518–523 (1995)

[27] N. Japkowicz, S. Stephen: The class imbalance problem: A systematic study. Intelligent Data Analysis **6**, 429–449 (2002)

[28] D. J. Hand: Construction and Assessment of Classification Rules. John Wiley & Sons Ltd, Chichester (1997)

[29] D. Alexander, C. Nobes: Financial Accounting. An international Introduction. Prentice Hall, England (2004)

[30] C. M. Bishop: Neural Networks for Pattern Recognition. Clarendon Press, Oxford (1995)

[31] B. D. Ripley: Pattern Recognition and Neural Networks. Cambridge University Press (1996)

[32] J. J. Montano, A. Palmer: Numeric sensitivity analysis applied to feedforward neural networks. Neural Computing and Applications **12**, 119–125 (2003)

[33] F. Provost, T. Fawcett: Robust classification for imprecise environments. Machine Learning **42**, 203–231 (2001)

[34] G. M. Weiss, F. Provost: Learning when training data are costly: The effect of class distribution on tree induction. Journal of artificial intelligence research **19**, 315–354 (2003)

[35] R. Alaiz-Rodriguez, N. Japkowicz, P. Tischer: Visualizing Classifier Performance on Different Domains. Tools with Artificial Intelligence. ICTAI '08. 20th IEEE International Conference on **2** 3–10 (2008)

[36] G. M. Weiss: Mining with rarity: a unifying framework. SIGKDD Explor. Newsl. **6** 7–19 (2004)

[37] J. R. Quinlan: Improved estimates for the accuracy of small disjuncts. Machine Learning **6** 93–98 (1991)

[38] F. Hsieh, B. W. Turnbull: Non-parametric and Semi-parametric Estimation of the Receiver Operating Characteristic Curve. The Annals of Statistics **24**, 25–40 (1996)

[39] A. P. Bradley: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition **30**, 1145–1159 (1997)

[40] M. A. Maloof: Learning when data sets are imbalanced and when costs are unequal and unknown. ICML-2003 Workshop on Learning from Imbalanced Data Sets II, (2003)

[41] C. Drummond, R. C. Holte: Explicitly representing expected cost: an alternative to ROC representation. In Proceedings of the Sixth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining, (2000)

[42] R. Caruana, A. Niculescu-Mizil: Data mining in metric space: an empirical analysis of supervised learning performance criteria. KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 69–78 (2004)

[43] R. Alaiz-Rodríguez, N. Japkowicz, P. Tischer: Visualizing Classifier Performance on Different Domains. ICTAI '08: Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence, 3–10 (2008)

[44] M. Kubat, R. C. Holte, S. Matwin: Machine Learning for the Detection of Oil Spills in Satellite Radar Images. Machine Learning **30**, 195–215 (1998)

[45] N. V. Chawla: C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In Proceedings of the ICML03 Workshop on Class Imbalances, (2003)

[46] C.Elkan: The foundations of cost-sensitive learning. Proceedings of the 17th International Joint Conference on Artificial Intelligence, 973–978 (2001)

[47] P. Domingos: MetaCost: a general method for making classifiers cost-sensitive. Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 155–164 (1999)

[48] G. M. Weiss, F. Provost: The effect of class distribution on class learning: An empirical study. Technical Report, Rutgers University, (2001)

[49] S. Lawrence, I. Burns, A. Back, A. Tsoi, C. Giles: Neural network classification and prior class probabilities. Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, Springer Berlin, Heidelberg (1998)

[50] R. C. Holte, L. E. Acker, B. W. Porter: Concept learning and the problem of small disjuncts. IJCAÍ89: Proceedings of the 11th international joint conference on Artificial intelligence, 813–818 (1989)

[51] N. Japkowicz: Concept-Learning in the Presence of Between-Class and Within-Class Imbalances. Advances in Artificial Intelligence, Lecture Notes in Computer Science. Springer Berlin, Heidelberg 67–77 (2001)

[52] A. Nickerson, N. Japkowicz, and E. Millos: Using Unsupervised Learning to Guide Re-sampling in Imbalanced Data Sets. In Proceedings of the 8th International Workshop on AI and Statistics, 261–265 (2001)

[53] T. Jo, N. Japkowicz: Class imbalances versus small disjuncts. SIGKDD Explorations Newsletter **6**, 40–49 (2004)

[54] N. V. Chawla, N. Japkowicz, A. Kotcz: Editorial: special issue on learning from imbalanced data sets. SIGKDD Explorations Newsletter **6**, 1–6 (2004)

[55] Y. L. Murphey, H. Guo, L. A. Feldkamp: Neural learning from unbalanced data. Applied Intelligence **21**, 117–128 (2004)

[56] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer: SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16**, 321–357 (2002)

[57] H. Han, W. Wang, B. Mao: Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. Advances in Intelligent Computing, Lecture Notes in Computer Science, 878–887. Springer Berlin, Heidelberg (2005)

[58] G.E.A.P.A. Batista, R. C. Prati, M. C. Monard: Balancing Strategies and Class Overlapping. Advances in Intelligent Data Analysis VI- Lecture Notes in Computer Science, 24–35. Springer Berlin, Heidelberg (2005)

[59] M. Kubat, S. Matwin: Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In Proceedings of the Fourteenth International Conference on Machine Learning, 179–186 (1997)

[60] N. V. Chawla, D. Cieslak, L. Hall, A. Joshi: Automatically countering imbalance and its empirical relationship to cost. Data Mining and Knowledge Discovery, 225–252. Springer Netherlands, (2008)

[61] N. V. Chawla: Data Mining for Imbalanced Datasets: An Overview. Data Mining and Knowledge Discovery Handbook, 875–886. Springer US, (2010)

[62] N. Japkowicz. Class imbalance: Are we focusing on the right issue? In Proceedings of the ICML03 Workshop on Learning from Imbalanced Data Sets, (2003)

[63] B. Raskutti, A. Kowalczyk: Extreme re-balancing for SVMs: a case study. SIGKDD Explorations Newsletter, Special issue on learning from imbalanced datasets **6**, 60–69 (2004)

[64] P. Chan, S. Stolfo: Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection. Proceedings 4th International Conference on Knowledge Discovery and Data Mining, 164–168 (1998)

[65] R. Barandela, J. S. Sánchez, V. García, E. Rangel: Strategies for learning in class imbalance problems. Pattern Recognition **36**, 849–851 (2003)

[66] S. Visa, A. Ralescu: Learning Imbalanced and Overlapping Classes using Fuzzy Sets. In Proceedings of the International Conference of Machine Learning, Workshop on Learning from Imbalanced data Sets (II), 97–104 (2003)

[67] C. X. Ling, Q. Yang, J. Wang, S. Zhang: Decision trees with minimal costs. ICML'04: Proceedings of the twenty-first international conference on Machine learning**69**, (2004)

[68] R. Alejo, V. García, J. M. Sotoca, R. A. Mollineda, J. S. Sánchez: Improving the performance of the RBF neural networks trained with imbalanced samples. IWANN'07: Proceedings of the 9th international work conference on Artificial neural networks. 162–169, Springer-Verlag Berlin, Heidelberg (2007)

[69] Q. Tao, G. Wu, F. Wang, J. Wang: Posterior probability support vector Machines for unbalanced data. IEEE Transactions on Neural Networks, **16**, 1561–1573 (2005)

[70] C. Nguyen, T. Ho: An Imbalanced Data Rule Learner. Knowledge Discovery in Databases: PKDD 2005, Lecture Notes in Computer Science. 617–624, Springer Berlin, Heidelberg (2005)

[71] X. Guo, Y. Yin, C. Dong, G. Yang, G. Zhou: On the Class Imbalance Problem. Natural Computation, 2008. ICNC '08. Fourth International Conference on **4**, 192–201 (2008)

[72] Y. Freund, R. E. Schapire: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences **55**, 119–139 (1997)

[73] C. Wang, Y. Huang: Evolutionary-based feature selection approaches with new criteria for data mining: A case study of credit approval data, Expert Systems with Applications **36**, 5900–5908 (2009)

[74] G. Weiss, K. McCarthy , B. Zabar: Cost-sensitive learning versus sampling: which is best for handling unbalanced classes with unequal error costs? In: DMIN, 35–41 (2007)

[75] K. McCarthy, B. Zabar, G. Weiss: Does cost-sensitive learning beat sampling for classifying rare classes? In Proceedings of the 1st international Workshop on Utility-Based Data Mining, 69–77 (2005)

[76] A. Estabrooks, T. Jo, N. Japkowicz: A Multiple Resampling Method for Learning from Imbalanced Data Sets. Computational Intelligence **20**, 18–36 (2004)

[77] N. Japkowicz: Supervised Versus Unsupervised Binary-Learning by Feed-forward Neural Networks. Machine Learning **42**, 97–122 (2001)

[78] P. Juszczak, R. P. W. Duin: Uncertainty sampling methods for one-class classifiers. In Proceedings of the ICML'03 Workshop on Learning from Imbalanced Data Sets, (2003)

[79] C. Schaffer: Overfitting avoidance as bias. Machine Learning **10**, 153–178 (1993)

[80] I. Guyon, A. Elisseeff: An introduction to variable and feature selection. Journal of Machine Learning Research **3** , 1157–1182 (2003)

[81] M. D. del Castillo, J. I. Serrano: A multistrategy approach for digital text categorization from imbalanced documents. SIGKDD Explor. Newsl. **6**, 70–79 (2004)

[82] G.H. Nguyen, A. Bouzerdoum, S.L. Phung: A supervised learning approach for imbalanced data sets. Pattern Recognition, ICPR 2008. 19th International Conference on, 1–4 (2008)

[83] A. Estabrooks, N. Japkowicz: A Mixture-of-Experts Framework for Learning from Imbalanced Data Sets. Advances in Intelligent Data Analysis, Lecture Notes in Computer Science **2189**, 34–43 Springer Berlin, Heidelberg (2001)

[84] N. Japkowicz: The Class Imbalance Problem: Significance and Strategies. In Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI2000), (2000)

[85] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, C. Brunk: Reducing misclassification costs. Machine Learning, Proceedings of the Eleventh International Conference, (1994)

[86] M. Kubat, S. Matwin: Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. Proceedings of the Fourteenth International Conference on Machine Learning, 179–186 (1997)

[87] T. Eavis, N. Japkowicz: A Recognition-Based Alternative to Discrimination-Based Multi-layer Perceptrons. In Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of intelligence, Advances in Artificial intelligence. 280–292, Springer-Verlag, London (2000)

[88] M. Monard, G. Batista: Learning with skewed class distributions. In Advances in Logic Artificial Intelligence and Robotics, 173–180 (2002)

[89] C. Drummond, R.C. Holte: C4.5, Class Imbalance, and Cost Sensitivity: Why Under-Sampling beats Over-Sampling. Proceedings of the International Conference on Machine Learning (ICML 2003) Workshop on Learning from Imbalanced Data Sets II, (2003)

[90] T. Yu, T. Jan, S. Simoff, J. Debenham: A Hierarchical VQSVM for Imbalanced Data Sets. Neural Networks, IJCNN 2007. International Joint Conference on, 518–523 (2007)

[91] K. Yoon, S. Kwek: An unsupervised learning approach to resolving the data imbalanced issue in supervised learning problems in functional genomics. Hybrid Intelligent Systems, 2005. HIS ’05. Fifth International Conference on, 6–9 (2005)

[92] S. Ertekin, J. Huang, L. Bottou, L. Giles: Learning on the border: active learning in imbalanced data classification. CIKM ’07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, 127–136 (2007)

[93] L. Breiman: Bagging predictors. Machine Learning **24**, 123–140 (1996)

[94] H. Guo, H. L. Viktor: Learning from imbalanced data sets with boosting and data generation: the DataBoost-IM approach. SIGKDD Explor. Newsl. **6**, 1, 30–39 (2004)

[95] M. V. Joshi, V. Kumar, R.C. Agarwal: Evaluating boosting algorithms to classify rare classes: comparison and improvements. Data Mining- ICDM 2001, Proceedings IEEE International Conference on, 257–264 (2001)

[96] T. M. Padmaja, P. R. Krishna, R. S. Bapi: Majority filter-based minority prediction (MFMP): An approach for unbalanced datasets. TENCON 2008- IEEE Region 10 Conference, 1–6 (2008)

[97] N. Japkowicz: Supervised Learning with Unsupervised Output Separation. Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC'2002), 321–325 (2002)

[98] S. Ingrassia, I. Morlini: Neural network modeling for small datasets. Technometrics **47**, 297-311 (2005)

[99] G. J. McLachlan, T. Krishnan: The EM Algorithm and Extensions. John Wiley & Sons Inc., New York (1997)

[100] G. J. McLachlan, D. Peel: Finite mixture models. John Wiley & Sons Inc., New York (2000)

[101] K. V. Mardia, J. T. Kent, J. M. Bibby: Multivariate Analysis. Academic Press Limited, London. Eighth Ed. (1992)

[102] S. Ingrassia, C. Davino: Reti neuronali e metodi statistici. Franco Angeli, Milano (2002)

[103] E. Stanghellini: Introduzione ai metodi statistici per il credit scoring. Springer-Verlag, Milano (2009)

[104] D. W. Hosmer, S. Lemeshow: Applied logistic regression. John Wiley & Sons Inc., New York (2000)

[105] S. Zani, A. Cerioli: Analisi dei dati e data mining per le decisioni aziendali. Giuffrè Editore, Milano (2007)

[106] R. E. Kass, A. E. Raftery: Bayes Factors. Journal of the American Statistical Association **90**, 430, 773–795 (1995)

[107] G. Schwarz: Estimating the Dimension of a Model. The Annals of Statistics **6**, 2, 461–464 (1978)

[108] C. Fraley, A. E. Raftery: How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. The Computer Journal **41**, 8, 578–588 (1998)

[109] C. Fraley, A. E. Raftery: Mclust version 3 for R: Normal mixture modeling and Model-based clustering. Technical Report No. 504, Department of Statistics- University of Washington (2006)

[110] A. E. Raftery: Bayesian Model Selection in Social Research. Sociological Methodology **25**, 111–163 (1995)

[111] C. Fraley, A. E. Raftery: MCLUST: Software for Model-Based Cluster Analysis. Journal of Classification **16**, 2, 297–306 (1999)

[112] G. Celeux, G. Govaert: Gaussian parsimonious clustering models. Pattern Recognition **28**, 5, 781–793 (1995)

[113] X. Meng, D. B. Rubin: Maximum likelihood estimation via the ECM algorithm: A general framework. Biometrika **80**, 2, 267–278 (1993)

[114] C. Liu, D. B. Rubin: The ECME algorithm: A simple extension of EM and ECM with faster monotone convergence. Biometrika **81**, 4, 633–648 (1994)

[115] A. Dasgupta, A. E. Raftery: Detecting features in spatial point processes with cluster via model-based clustering. Journal of the American Statistical Association **93**, 441, 294–302 (1998)

[116] F. Murtagh, A. E. Raftery: Fitting straight lines to point patterns. Pattern Recognition **17**, 5, 479–483 (1984)

[117] H. H. Bock: Probabilistic models in cluster analysis. Computational Statistics & Data Analysis **23**, 1, 5–28 (1996)

[118] J. H. Wolfe: Pattern Clustering by Multivariate Mixture Analysis. Multivariate Behavioral Research **5** 329–350 (1970)

[119] A. W. F. Edwards, L. Cavalli-Sforza: A method for cluster analysis. Biometrics **21**, 2, 362–375 (1965)

[120] N. E. Day: Estimating the components of a mixture of normal distributions. Biometrika **56**, 3, 463–474 (1969)

[121] A. J. Scott, M. J. Symons: Clustering Methods Based on Likelihood Ratio Criteria. Biometrics **27**, 2, 387–397 (1971)

[122] D. A. Binder: Bayesian cluster analysis. Biometrika **65**, 1, 31–38 (1978)

[123] R. B. Avery, P. S. Calem, G. B. Canner: Consumer credit scoring: Do situational circumstances matter? Journal of Banking & Finance **28**, 4, 835–856 (2004)

[124] E. I. Altman: Revisiting Credit Scoring Models in a Basel 2 Environment. NYU Working Paper No. S-FI-02-11 (2002)

[125] E. I. Altman, G. Marco, F. Varetto: Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the Italian experience). Journal of Banking & Finance **18**, 3, 505–529 (1994)

[126] E. Altman, G. Sabato: Modelling Credit Risk for SMEs: Evidence from the U.S. Market. Abacus **43**, 3, 332–357 (2007)

[127] E. Altman, G. Sabato: Effects of the New Basel Capital Accord on Bank Capital Requirements for SMEs Journal of Financial Services Research. **28**, 1, 15–42 (2005)

[128] D. J. Hand: Modelling consumer credit risk. IMA Journal of Management Mathematics **12**, 2, 139–155 (2001)

[129] D. J. Hand: Classifier Technology and the Illusion of Progress. Statistical Science **21**, 1, 1–14 (2006)

[130] Basel Committee on Banking Supervision: International Convergence of Capital Measurement and Capital Standards. Bank for International Settlements (2004)

[131] D. J. Hand: Reject inference in credit operations. Handbook of Credit Scoring, 225–240, Glenlake E. Mays Editions, Chicago (2001)

[132] M. G. Kelly, D. J. Hand, N. M. Adams: Choosing good predictive models for consumer credit data. Technical Report TR-00-15. Department of Mathematics, Imperial College, London (2000)

[133] F. Rosenblatt: Principle of neurodynamics: perceptrons and the theory of brain mechanism. Spartan, Washington (1962)

[134] W. S. Mc Culloch, W. Pitts: A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics **5**, 4, 115–123 (1943)

[135] A.B.J. Novikoff: On convergence proofs on perceptrons. Proceedings of the symposium on the mathematical theory of automata, Polytechnic Institute of Brooklin **12**, 615–622 (1962)

[136] V. Vapnik: Estimation of dependencies based on empirical data. Springer-Verlag, New York (1982)

[137] V. Vapnik, A. Ya. Chervonenkis: The necessary and sufficient conditions of the method of empirical risk minimization. Pattern Recognition and Image Analysis **1**, 3, 284–305 (1991)

[138] G. D. Garson: Interpreting neural-network connection weights. AI Expert **6**, 4, 46–51 (1991)

[139] J. M. Zurada, A. Malinowski, S. Usui: Perturbation method for deleting redundant inputs of perceptron networks. Neurocomputing **14**, 2, 177–193 (1997)

[140] W. Wang, P. Jones, D. Partridge: Assessing the Impact of Input Features in a Feed-forward Neural Network. Neural Computing and Applications **9**, 2, 101–112 (2000)

[141] T. Tchaban, M. J. Taylor, J. P. Griffin: Establishing impacts of the inputs in a feed-forward neural network. Neural Computing and Applications **7**, 4, 309–317 (1998)