Sharpening the Accuracy of Credit Scoring Models with Machine Learning Algorithms

Massimo Guidolin and Manuela Pedio

Abstract The big data revolution and recent advancements in computing power have increased the interest in credit scoring techniques based on artificial intelligence. This has found easy leverage in the fact that the accuracy of credit scoring models has a crucial impact on the profitability of lending institutions. In this chapter, we survey the most popular supervised credit scoring classification methods (and their combinations through ensemble methods) in an attempt to identify a superior classification technique in the light of the applied literature. There are at least three key insights that emerge from surveying the literature. First, as far as individual classifiers are concerned, linear classification methods often display a performance that is at least as good as that of machine learning methods. Second, ensemble methods tend to outperform individual classifiers. However, a dominant ensemble method cannot be easily identified in the empirical literature. Third, despite the possibility that machine learning techniques could fail to outperform linear classification methods when standard accuracy measures are considered, in the end they lead to significant cost savings compared to the financial implications of using different scoring models.

1 Introduction

Credit scoring consists of a set of risk management techniques that help lenders to decide whether to grant a loan to a given applicant [42]. More precisely, financial institutions use credit scoring models to make two types of credit decisions. First, a lender should decide whether to grant a loan to a new customer. The process

M. Guidolin
Bocconi University, Milan, Italy
e-mail: massimo.guidolin@unibocconi.it

M. Pedio (✉)
University of Bristol, Accounting and Finance Department, Bristol, UK
Bocconi University, Milan, Italy
e-mail: manuela.pedio@unibocconi.it

S. Consoli et al. (eds.), *Data Science for Economics and Finance*,
https://doi.org/10.1007/978-3-030-66891-4_5

that leads to this decision is called *application scoring*. Second, a lender may want to monitor the risk associated with existing customers (the so-called behavioral scoring). In the field of retail lending, credit scoring typically consists of a binary classification problem, where the objective is to predict whether an applicant will be a "good" one (i.e., she will repay her liabilities within a certain period of time) or a "bad" one (i.e., she will default in part or fully on her obligations) based on a set of observed characteristics (*features*) of the borrower.[1] A feature can be of two types: continuous, when the value of the feature is a real number (an example can be the income of the applicant) or categorical, when the feature takes a value from a predefined set of categories (an example can be the rental status of the applicant, e.g., "owner," "living with parents," "renting," or "other"). Notably, besides traditional categories, new predictive variables, such as those based on "soft" information have been proposed in the literature to improve the accuracy of the credit score forecasts. For instance, Wang et al. [44] use text mining techniques to exploit the content of descriptive loan texts submitted by borrowers to support credit decisions in peer-to-peer lending.

Credit scoring plays a crucial role in lending decisions, considering that the cost of an error is relatively high. Starting in the 1990s, most financial institutions have been making lending decisions with the help of automated credit scoring models [17]. However, according to the Federal Reserve Board [15] the average delinquency rate on consumer loans has been increasing again since 2016 and has reached 2.28% in the first quarter of 2018, thus indicating that wide margins for improvement in the accuracy of credit scoring models remain. Given the size of the retail credit industry, even a small reduction in the hazard rate may yield significant savings for financial institutions in the future [45].

Credit scoring also carries considerable regulatory importance. Since the Basel Committee on Banking Supervision released the Basel Accords, especially the second accord in 2004, the use of credit scoring has grown considerably, not only for credit granting decisions but also for risk management purposes. Basel III, released in 2013, enforced increasingly accurate calculations of default risk, especially in consideration of the limitations that external rating agencies have shown during the 2008–2009 financial crisis [38]. As a result, over the past decades, the problem of developing superior credit scoring models has attracted significant attention in the academic literature. More recently, thanks to the increase in the availability of data and the progress in computing power the attention has moved towards the application of Artificial Intelligence (AI) and, in particular, Machine Learning (ML) algorithms to credit scoring, when machines may learn and make predictions without being explicitly assigned program instructions.

[1] There are also applications in which the outcome variable is not binary; for instance, multinomial models are used to predict the probability that an applicant will move from one class of risk to another. For example, Sirignano et al. [40] propose a nonlinear model of the performance of a pool of mortgage loans over their life; they use neural networks to model the conditional probability that a loan will transition to a different state (e.g., pre-payment or default).

There are four major ML paradigms: *supervised learning*, *semi-supervised learning*, *unsupervised learning*, and *reinforcement learning* [32]. In supervised learning, a training dataset should consist of both input data and their corresponding output target values (also called labels). Then, the algorithm is trained on the data to find relationships between the input variables and selected output labels. If only some target output values are available in a training dataset, then such a problem is known as semi-supervised learning. Unsupervised learning requires only the input data to be available. Finally, reinforcement learning does not need labelled inputs/outputs but focuses instead on agents making optimal decisions in a certain environment; a feedback is provided to the algorithm in terms of "reward" and "punishment" so that the final goal is to maximize the agent's cumulative reward. Typically, lending institutions store both the input characteristics and the output historical data concerning their customers. As a result, supervised learning is the main focus of this chapter.

Simple linear classification models remain a popular choice among financial institutions, mainly due to their adequate accuracy and straightforward implementation [29]. Furthermore, the majority of advanced ML techniques lack the necessary transparency and are regarded as "black boxes", which means that one is not able to easily explain how the decision to grant a loan is made and on which parameters it is based. In the financial industry, however, transparency and simplicity play a crucial role, and that is the main reason why advanced ML techniques have not yet become widely adopted for credit scoring purposes.[2] However, Chui et al. [12] emphasize that the financial industry is one of the leading sectors in terms of current and prospective ML adoption, especially in credit scoring and lending applications as they document that more than 25% of the companies that provide financial services have adopted at least one advanced ML solution in their day-to-day business processes.

Even though the number of papers on advanced scoring techniques has increased dramatically, a consensus regarding the best-performing models has not yet been reached. Therefore, in this chapter, besides providing an overview of the most common classification methods adopted in the context of credit scoring, we will also try to answer three key questions:

- Which individual classifiers show the best performance both in terms of accuracy and of transparency?
- Do ensemble classifiers consistently outperform individual classification models and which (if any) is the superior ensemble method?

[2] Casual interpretations of "black box" ML models have attracted considerable attention. Zhao and Hastie [50] provide a summary and propose partial dependence plots (PDP) and individual conditional expectations (ICE) as tools to enhance the interpretation of ML models. Dorie et al. [13] report interesting results of a data analysis competition where different strategies for causal inference—including "black box" models—are compared.

- Do one-class classification models score higher accuracy compared to the best individual classifiers when tested on imbalanced datasets (i.e., datasets where one class is underrepresented)?

Our survey shows that, despite that ML techniques rarely significantly outperform simple linear methods as far as individual classifiers are concerned, ensemble methods tend to show a considerably better classification performance than individual methods, especially when the financial costs of misclassification are accounted for.

2 Preliminaries and Linear Methods for Classification

A (supervised) learning problem is an attempt to predict a certain output using a set of variables (*features* in ML jargon) that are believed to exercise some influence on the output. More specifically, what we are trying to learn is the function $h(\mathbf{x})$ that best describes the relationship between the predictors (the features) and the output. Technically, we are looking for the function $h \in H$ that minimizes a *loss function*.

When the outcome is a categorical variable C (a label), the problem is said to be a classification problem and the function that maps the inputs $\mathbf{x}$ into the output is called *classifier*. The estimate $\hat{C}$ of C takes values in $\mathscr{C}$, the set of all possible classes. As discussed in Sect. 1, credit scoring is usually a classification problem where only two classes are possible, either the applicant is of the "good" (G) or of the "bad" (B) type. In a binary classification problem, the loss function can be represented by a 2 × 2 matrix L with zeros on the main diagonal and nonnegative values elsewhere. $L(k, l)$ is the cost of classifying an observation belonging to class $\mathscr{C}_k$ as $\mathscr{C}_l$. The expected prediction error (EPE) is

$$EPE = E[L(C, \hat{C}(X))] = E_X \sum_{k=1}^{2} L[\mathscr{C}_k, \hat{C}(X)]p(\mathscr{C}_k|X), \tag{1}$$

where $\hat{C}(X)$ is the predicted class C based on X (the matrix of the observed features), $\mathscr{C}_k$ represents the class with label *k*, and $p(\mathscr{C}_k|X)$ is the probability that the actual class has label *k* conditional to the observed values of the features. Accordingly, the optimal prediction $\hat{C}(X)$ is the one that minimizes the EPE pointwise, i.e.,

$$\hat{C}(x) = \arg\min_{c\in\mathscr{C}} \sum_{k=1}^{2} L(\mathscr{C}_k, c)p(\mathscr{C}_k|X = x), \tag{2}$$

where x is a realization of the features. Notably, when the loss function is of the 0–1 type, i.e., all misclassifications are charged a unit cost, the problem simplifies to

$$\hat{C}(x) = \arg\min_{c\in\mathscr{C}} [1 - p(c|X = x)], \tag{3}$$

which means that

$$\hat{C}(x) = \mathscr{C}_k \ \ if \ \ p(\mathscr{C}_k|X = x) = \max_{c \in \mathscr{C}} p(c|X = x). \tag{4}$$

In this section, we shall discuss two popular classification approaches that result in linear *decision boundaries*: logistic regressions (LR) and linear discriminant analysis (LDA). In addition, we also introduce the Naïve Bayes method, which is related to LR and LDA as it also considers a *log-odds* scoring function.

2.1 Logistic Regression

Because of its simplicity, LR is still one of the most popular approaches used in the industry for the classification of applicants (see, e.g., [23]). This approach allows one to model the posterior probabilities of K different applicant classes using a linear function of the features, while at the same time ensuring that the probabilities sum to one and that their value ranges between zero and one. More specifically, when there are only two classes (coded via y, a dummy variable that takes a value of 0 if the applicant is "good" and of 1 if she is "bad"), the posterior probabilities are modeled as

$$\begin{aligned} p(C = G|X = x) &= \frac{exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})}{1 + exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})} \\ p(C = B|X = x) &= \frac{1}{1 + exp(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})}. \end{aligned} \tag{5}$$

Applying the *logit* transformation, one obtains the log of the probability odds (the log-odds ratio) as

$$\log \frac{p(C = G|X = x)}{p(C = B|X = x)} = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}. \tag{6}$$

The input space is optimally divided by the set of points for which the log-odds ratio is zero, meaning that the posterior probability of being in one class or in the other is the same. Therefore, the decision boundary is the hyperplane defined by $\left\{x|\beta_0 + \boldsymbol{\beta}^T \mathbf{x} = 0\right\}$. Logistic regression models are usually estimated by maximum likelihood, assuming that all the observations in the sample are independently Bernoulli distributed, such that the log-likelihood functions is

$$\mathscr{L}(\theta|x) = p(y|x;\theta) = \sum_{i=1}^{T_0} \log p_{C_i}(\mathbf{x}_i;\theta), \tag{7}$$

where T_0 are the observations in the training sample, θ is the vector of parameters, and $p_k(\mathbf{x}_i;\theta) = p(C = k|X = \mathbf{x}_i;\theta)$. Because in our case there are only two classes coded via a binary response variable y_i that can take a value of either zero or one, $\hat{\beta}$ is found by maximizing

$$\mathscr{L}(\beta) = \sum_{i=1}^{T_0}(y_i \boldsymbol{\beta}^T \mathbf{x}_i - \log(1 + exp(\boldsymbol{\beta}^T \mathbf{x}_i)). \tag{8}$$

2.2 Linear Discriminant Analysis

A second popular approach used to separate "good" and "bad" applicants that lead to linear decision boundaries is LDA. The LDA method approaches the problem of separating two classes based on a set of observed characteristics $\mathbf{x}$ by modeling the class densities $f_G(\mathbf{x})$ and $f_B(\mathbf{x})$ as multivariate normal distributions with means $\boldsymbol{\mu}_G$, and $\boldsymbol{\mu}_B$ and the same covariance matrix Σ, i.e.,

$$f_G(\mathbf{x}) = (2\pi)^{-K/2}\,(|\Sigma|)^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{G})^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{G})\right)$$
$$f_B(\mathbf{x}) = (2\pi)^{-K/2}\,(|\Sigma|)^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{B})^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{B})\right). \tag{9}$$

To compare the two classes ("good" and "bad" applicants), one has then to compute and investigate the log-ratio

$$\log\frac{p(C = G|X = x)}{p(C = B|X = x)} = \log\frac{f_G(\mathbf{x})}{f_B(\mathbf{x})} + \log\frac{\pi_G}{\pi_B}$$
$$= log\frac{\pi_G}{\pi_B} - \frac{1}{2}(\boldsymbol{\mu}_\mathbf{B} + \boldsymbol{\mu}_\mathbf{G})^T\Sigma^{-1}(\boldsymbol{\mu}_\mathbf{B} + \boldsymbol{\mu}_\mathbf{G}) + \mathbf{x}^T\Sigma^{-1}(\boldsymbol{\mu}_\mathbf{B} + \boldsymbol{\mu}_\mathbf{G}), \tag{10}$$

which is linear in $\mathbf{x}$. Therefore, the decision boundary, which is the set where $p(C = G|X = x) = p(C = B|X = x)$, is also linear in $\mathbf{x}$. Clearly the Gaussian parameters $\boldsymbol{\mu}_G$, $\boldsymbol{\mu}_B$, and Σ are not known and should be estimated using the training sample as well as the prior probabilities π_G and π_B (set to be equal to the proportions of good and bad applicants in the training sample). Rearranging Eq. (10), it appears evident that the Bayesian optimal solution is to predict a point to belong to the "bad" class if

$$\mathbf{x}^T\hat{\Sigma}^{-1}(\hat{\boldsymbol{\mu}}_B - \hat{\boldsymbol{\mu}}_G) > \frac{1}{2}\hat{\boldsymbol{\mu}}_B^T\hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_B - \frac{1}{2}\hat{\boldsymbol{\mu}}_G^T\hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_G + \log\hat{\pi}_G - log\hat{\pi}_G, \tag{11}$$

which can be rewritten as

$$\mathbf{x}^T\mathbf{w} > z \tag{12}$$

where $\mathbf{w} = \hat{\Sigma}^{-1}(\hat{\boldsymbol{\mu}}_B - \hat{\boldsymbol{\mu}}_G)$ and $z = \frac{1}{2}\hat{\boldsymbol{\mu}}_B^T\hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_B - \frac{1}{2}\hat{\boldsymbol{\mu}}_G^T\hat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_G + \log\hat{\pi}_G - log\hat{\pi}_G$.

Another way to approach the problem, which leads to the same coefficients **w** is to look for the linear combination of the features that gives the maximum separation between the means of the classes and the minimum variation within the classes, which is equivalent to maximizing the separating distance M

$$M = \boldsymbol{\omega}^T \frac{\hat{\boldsymbol{\mu}}_G - \hat{\boldsymbol{\mu}}_B}{(\boldsymbol{\omega}^T \hat{\Sigma} \boldsymbol{\omega})^{1/2}}. \tag{13}$$

Notably, the derivation of the coefficients **w** does not require that $f_G(\mathbf{x})$ and $f_B(\mathbf{x})$ follow a multivariate normal as postulated in Eq. (9), but only that $\Sigma_G = \Sigma_B = \Sigma$. However, the choice of z as a cut-off point in Eq. (12) requires normality. An alternative is to use a cut-off point that minimizes the training error for a given dataset.

2.3 Naïve Bayes

The Naïve Bayes (NB) approach is a probabilistic classifier that assumes that given a class (G or B), the applicant's attributes are independent. Let π_G denote the prior probability that an applicant is "good" and π_B the prior probability that an applicant is "bad." Then, because of the assumption that each attribute x_i is conditionally independent from any other attribute x_j for $i \neq j$, the following holds:

$$p\left(\mathbf{x} \mid G\right) = p\left(x_1 | G\right) p\left(x_2 | G\right) \ldots p\left(x_n | G\right), \tag{14}$$

where $p(\mathbf{x} \mid G)$ is the probability that a "good" applicant has attributes **x**. The probability of an applicant being "good" if she is characterized by the attributes **x** can now be found by applying Bayes' theorem:

$$p\left(G \mid \mathbf{x}\right) = \frac{p\left(\mathbf{x} \mid G\right)\pi_G}{p(\mathbf{x})}. \tag{15}$$

The probability of an applicant being "bad" if she is characterized by the attributes **x** is

$$p\left(B \mid \mathbf{x}\right) = \frac{p\left(\mathbf{x} \mid B\right)\pi_B}{p(\mathbf{x})}. \tag{16}$$

The attributes **x** are typically converted into a score, $s(\mathbf{x})$, which is such that $p\left(G \mid \mathbf{x}\right) = p\left(G \mid s(\mathbf{x})\right)$. A popular score function is the log-odds score [42]:

$$\begin{aligned} s\left(\mathbf{x}\right) &= \log\left(\frac{p\left(G|\mathbf{x}\right)}{p\left(B|\mathbf{x}\right)}\right) = \log\left(\frac{\pi_G p\left(\mathbf{x}|G\right)}{\pi_B p\left(\mathbf{x}|B\right)}\right) = \\ &= \log\left(\frac{\pi_G}{\pi_B}\right) + \log\left(\frac{p\left(\mathbf{x}|G\right)}{p\left(\mathbf{x}|B\right)}\right) = s_{pop} + woe\left(\mathbf{x}\right), \end{aligned} \tag{17}$$

where s_{pop} is the log of the relative proportion of “good” and “bad” applicants in the population and $woe\,(\mathbf{x})$ is the weight of evidence of the attribute combination $\mathbf{x}$. Because of the conditional independence of the attributes, we can rewrite Eq. (17) as

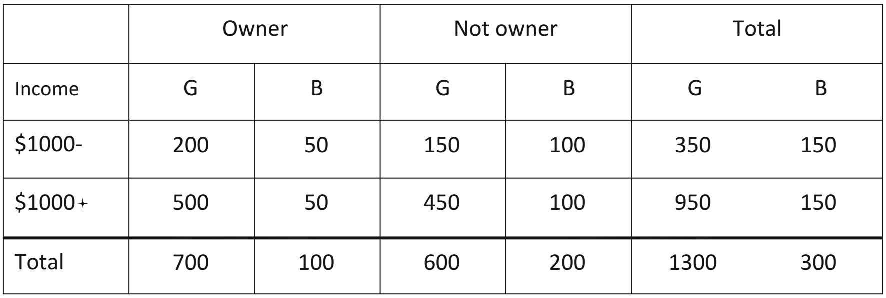
$$\begin{aligned} s\,(\mathbf{x}) &= \ln\left(\frac{\pi_G}{\pi_B}\right) + \ln\left(\frac{p\,(x_1|G)}{p\,(x_1|B)}\right) + \ldots + \ln\left(\frac{p\,(x_n|G)}{p\,(x_n|B)}\right) \\ &= s_{pop} + woe\,(x_1) + woe\,(x_2) + \ldots + woe\,(x_n)\,. \end{aligned} \tag{18}$$

If $woe\,(x_i)$ is equal to 0, then this attribute does not affect the estimation of the status of an applicant. The prior probabilities π_G and π_B are estimated using the proportions of good and bad applicants in the training sample; the same applies to the weight of evidence of the attributes, as illustrated in the example below.

Example Let us assume that a bank makes a lending decision based on two attributes: the residential status and the monthly income of the applicant. The data belonging to the training sample are given in Fig. 1. An applicant who has a monthly income of USD 2000 and owns a flat, will receive a score of:

$$s\,(\mathbf{x}) = \ln\left(\frac{1300}{300}\right) + \ln\left(\frac{950/1300}{150/300}\right) + \ln\left(\frac{700/1300}{100/300}\right) = 2.32.$$

If $p\,(G \mid s(\mathbf{x}) = 2.32)$, the conditional probability of being “good” when the score is 2.32, is higher than $p\,(B \mid s(\mathbf{x}) = 2.32)$, i.e., the conditional probability of being “bad,” this applicant is classified as “good” (and vice versa).



	Owner		Not owner		Total	
Income	G	B	G	B	G	B
$1000-	200	50	150	100	350	150
$1000+	500	50	450	100	950	150
Total	700	100	600	200	1300	300

Fig. 1 This figure provides the number of individuals in each cluster in a fictional training sample used to illustrate the NB approach. Two binary attributes are considered: the residential status (either “owner” or “not owner”) and monthly income (either more than USD 1000 or less than USD 1000). Source: Thomas et al. [42]

A lender can therefore define a cutoff score, below which applicants are automatically rejected as “bad.” Usually, the score $s(\mathbf{x})$ is linearly transformed so that its interpretation is more straightforward. The NB classifier performs relatively well in many applications but, according to Thomas et al. [42], it shows poor performance in the field of credit scoring. However, its most significant advantage is that it is easy to interpret, which is a property of growing importance in the industry.

3 Nonlinear Methods for Classification

Although simple linear methods are still fairly popular with practitioners, because of their simplicity and their satisfactory accuracy [29], more than 25% of the financial companies have recently adopted at least one advanced ML solution in their day-to-day business processes [12], as emphasized in Sect. 1. Indeed, these models have the advantage of being much more flexible and they may be able to uncover complex, nonlinear relationships in the data. For instance, the popular LDA approach postulates that an applicant will be “bad” if her/his score exceeds a given threshold; however, the path to default may be highly nonlinear in the mapping between scores and probability of default (see [39]).

Therefore, in this section, we review several popular ML techniques for classification, such as Decision Trees (DT), Neural Network (NN), Support Vector Machines (SVM), k-Nearest Neighbor (k-NN), and Genetic Algorithms (GA). Even if GA are not exactly classification methods, evolutionary computing techniques that help to find the “fittest” solution, we cover them in our chapter as this method is widely used in credit scoring applications (see, e.g., [49, 35, 1]). Finally, we discuss ensemble methods that combine different classifiers to obtain better classification accuracy. For the sake of brevity, we do not cover deep learning techniques, which are also employed for credit scoring purposes; the interested reader can find useful references in [36].

3.1 Decision Trees

Decision Trees (also known as Classification Trees) are a classification method that uses the training dataset to construct decision rules organized into tree-like structures, where each branch represents an association between the input values and the output label. Although different algorithms exist (such as classification and regression trees, also known as CART), we focus on the popular C4.5 algorithm developed by Quinlan [37]. At each node, the C4.5 algorithm splits the training dataset according to the most influential feature through an iterative process. The most influential feature is the one with the lowest entropy (or, similarly, with the highest information gain). Let $\hat{\pi}_G$ be the proportion of “good” applicants and $\hat{\pi}_B$ the proportion of “bad” applicants in the sample S. The entropy of S is then defined as in Baesens et al. [5]:

$$\text{Entropy}\,(S) = -\hat{\pi}_G \log_2\left(\hat{\pi}_G\right) - \hat{\pi}_B \log_2\left(\hat{\pi}_B\right). \tag{19}$$

According to this formula, the maximum value of the entropy is equal to 1 when $\hat{\pi}_G = \hat{\pi}_B = 0.5$ and it is minimal at 0, which happens when either $\hat{\pi}_G = 0$ or $\hat{\pi}_B = 0$. In other words, an entropy of 0 means that we have been able to identify the characteristics that lead to a group of good (bad) applicants. In order to split the sample, we compute the gain ratio:

$$\text{Gain ratio}\,(S, x_i) = \frac{\text{Gain}\,(S, x_i)}{\text{Split Information}(S, x_i)}. \tag{20}$$

$Gain\,(S, x_i)$ is the expected reduction in entropy due to splitting the sample according to feature x_i and it is calculated as

$$\text{Gain}\,(S, x_i) = \text{Entropy}\,(S) - \sum_{\upsilon} \frac{|S_\upsilon|}{|S|} \text{Entropy}\,(S_\upsilon)\,, \tag{21}$$

where $\upsilon \in \text{values}(x_i)$, S_υ is a subset of the individuals in S that share the same value of the feature x_i, and

$$\text{Split Information}\,(S,\ x_i) = -\sum_{k} \frac{|S_k|}{|S|} \log_2 \frac{|S_k|}{|S|} \tag{22}$$

where $k \in \text{values}(x_i)$ and S_k is a subset of the individuals in S that share the same value of the feature x_i. The latter term represents the entropy of S relative to the feature x_i. Once such a tree has been constructed, we can predict the probability that a new applicant will be a "bad" one using the proportion of "bad" customers in the leaf that corresponds to the applicant's characteristics.

3.2 Neural Networks

NN models were initially inspired by studies of the human brain [8, 9]. A NN model consists of input, hidden, and output layers of interconnected neurons. Neurons in one layer are combined through a set of weights and fed to the next layer. In its simplest single-layer form, a NN consists of an input layer (containing the applicants' characteristics) and an output layer. More precisely, a single-layer NN is modeled as follows:

$$\begin{aligned} u_k &= \omega_{k0} + \sum_{i=1}^{n} \omega_{ki} x_i \\ y_k &= f\,(u_k)\,, \end{aligned} \tag{23}$$

where $x_1, \ldots, x_n$ are the applicant's characteristics, which in a NN are typically referred to as *signals*, $\omega_{k1}, \ldots, \omega_{kn}$ are the weights connecting characteristic i to the layer k (also called *synaptic weights*), and ω_{k0} is the "bias" (which plays a similar role to the intercept term in a linear regression). Eq. (23) describes a single-layer NN, so that $k = 1$. A positive weight is called *excitatory* because it increases the effect of the corresponding characteristic, while a negative weight is called *inhibitory* because it decreases the effect of a positive characteristic [42]. The function f that transform the inputs into the output is called *activation function* and may take a number of specifications. However, in binary classification problems, it may be convenient to use a logistic function, as it produces an output value in the range [0, 1]. A cut-off value is applied to y_k to decide whether the applicant should be classified as good or bad. Figure 2 illustrates how a single-layer NN works.

A single-layer NN model shows a satisfactory performance only if the classes can be linearly separated. However, if the classes are not linearly separable, a multilayer model could be used [33]. Therefore, in the rest of this section, we describe multilayer perceptron (MLP) models, which are the most popular NN models in classification problems [5]. According to Bishop [9], even though multiple hidden layers may be used, a considerable number of papers have shown that MLP NN models with one hidden layer are universal nonlinear discriminant functions that can approximate arbitrarily well any continuous function. An MLP model with one hidden layer, which is also called a three-layer NN, is shown in Fig. 3. This model can be represented algebraically as

$$y_k = f^{(1)}(\sum_{i=0}^{n} \omega_{ki} x_i), \tag{24}$$

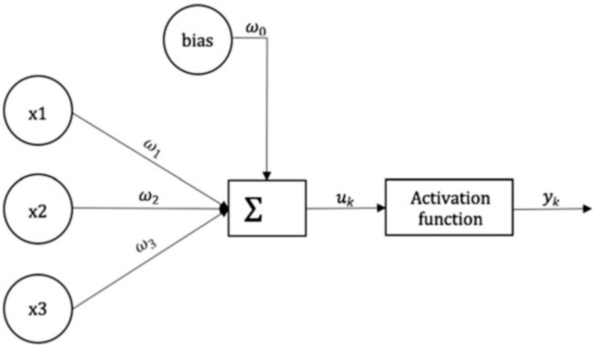


Fig. 2 The figure illustrates a single-layer NN with one output neuron. The applicant's attributes are denoted by $x_1, \ldots, x_n$, the weights are denoted by $\omega_1, \ldots, \omega_n$, and ω_0 is the "bias." The function f is called activation function and it transforms the sum of the weighted applicant's attributes to a final value. Source: Thomas et al. [42]

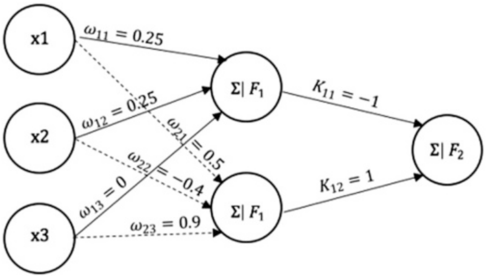


Fig. 3 The figure shows the weights of a three-layer MLP NN model, where the input characteristics are the following dummy variables: x_1 is equal to one if the monthly income is low; x_2 takes the value of one if the client has no credit history with the bank; x_3 represents the applicant's residential status

where $f^{(1)}$ is the activation function on the second (hidden) layer and y_k for $k = 1 \ldots, r$ are the outputs from the hidden layer that simultaneously represent the inputs to the third layer. Therefore, the final output values z_v can be written as

$$z_v = f^{(2)}\left(\sum_{k=1}^{r} K_{vk} y_k\right) = f^{(2)}\left(\sum_{k=1}^{r} K_{vk} f^{(1)}\left(\sum_{i=0}^{n} \omega_{ki} x_i)\right)\right) \tag{25}$$

where $f^{(2)}$ is the activation function of the third (output) layer, z_v for $v = 1, \ldots, s$ are the final outputs, and K_{vk} are the weights applied to the y_k values. The estimation of the weights is called *training* of the model and to this purpose the most popular method is the back-propagation algorithm, in which the pairs of input values and output values are presented to the model many times with the goal of finding the weights that minimize an error function [42].

3.3 Support Vector Machines

The SVM method was initially developed by Vapnik [43]. The idea of this method is to transform the input space into a high-dimensional feature space by using a nonlinear function $\varphi(\bullet)$. Then, a linear classifier can be used to distinguish between "good" and "bad" applicants. Given a training dataset of N pairs of observations $(\mathbf{x}_i, y_i)_{i=1}^{N}$, where $\mathbf{x}_i$ are the attributes of customer i and y_i is the corresponding binary label, such that $y_i \in [-1, +1]$, the SVM model should satisfy the following conditions:

$$\begin{cases} \mathbf{w}^T \varphi(\mathbf{x}_i) + b \geq +1 & \text{if } \ y_i = +1 \\ \mathbf{w}^T \varphi(\mathbf{x}_i) + b \leq -1 & \text{if } \ y_i = -1, \end{cases}$$

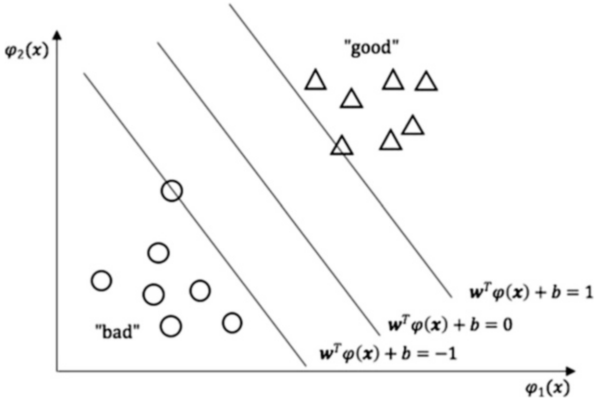


Fig. 4 This figure illustrates the main concept of an SVM model. The idea is to maximize the perpendicular distance between the support vectors and the separating hyperplane. Source: Baesens et al. [5]

which is equivalent to

$$y_i\left[\mathbf{w}^T\varphi(\mathbf{x}_i)+b\right]\geq 1,\quad i=1,\ldots,N. \tag{26}$$

The above inequalities construct a hyperplane in the feature space, defined by $\left\{x|\mathbf{w}^T\varphi(\mathbf{x}_i)+b=0\right\}$, which distinguishes between two classes (see Fig. 4 for the illustration of a simple two-dimensional case). The observations on the lines $\mathbf{w}^T\varphi(x_i)+b=1$ and $\mathbf{w}^T\varphi(x_i)+b=-1$ are called the *support vectors*. The parameters of the separating hyperplane are estimated by maximizing the perpendicular distance (called the *margin*), between the closest support vector and the separating hyperplane while at the same time minimizing the misclassification error.

The optimization problem is defined as:

$$\begin{cases}\min_{w,b,\xi}\ \mathscr{J}(\mathbf{w},\ b,\ \xi)=\frac{1}{2}\mathbf{w}^T\mathbf{w}+C\sum_{i=1}^{N}\xi_i,\\ \text{subject to:}\\ y_i\left[\mathbf{w}^T\ \varphi(x_i)+b\right]\geq 1-\xi_i,\quad i=1,\ \ldots,\ N\\ \xi_i\geq 0,\quad i=1,\ldots,N,\end{cases} \tag{27}$$

where the variables ξ_i are slack variables and C is a positive tuning parameter [5]. The Lagrangian to this optimization problem is defined as follows:

$$\mathscr{L}(\mathbf{w},b,\xi;\boldsymbol{\alpha},\ \boldsymbol{\nu})=\mathscr{J}(\mathbf{w},b,\xi)-\sum_{i=1}^{N}\alpha_i\left\{y_i\left[\mathbf{w}^T\ \varphi(\mathbf{x_i})+b\right]-1+\xi_i\right\}-\sum_{i=1}^{N}\nu_i\xi_i. \tag{28}$$

The classifier is obtained by minimizing $\mathscr{L}(\mathbf{w},b,\xi;\boldsymbol{\alpha},\boldsymbol{\nu})$ with respect to $\mathbf{w},b,\xi$ and maximizing it with respect to $\boldsymbol{\alpha},\ \boldsymbol{\nu}$. In the first step, by taking the derivatives with respect to $\mathbf{w}, b, \xi$, setting them to zero, and exploiting the results, one may represent the classifier as

$$y\left(\mathbf{x}\right) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K\left(\mathbf{x}_i,\ \mathbf{x}\right) + B\right) \tag{29}$$

where $K\left(\mathbf{x}_i, \mathbf{x}\right) = \varphi\left(\mathbf{x}_i\right)^T \varphi\left(\mathbf{x}_i\right)$ is computed using a positive-definite kernel function. Some possible kernel functions are the radial basis function $K\left(\mathbf{x}_i, \mathbf{x}\right) = exp(-||x_i - x_j||_2^2/\sigma^2)$ and the linear function $K\left(\mathbf{x}_i, \mathbf{x}\right) = \mathbf{x}_i^T \mathbf{x}_j$. At this point, the Lagrange multipliers α_i can be found by solving:

$$\begin{cases} \max_{\alpha_i} -\frac{1}{2}\sum_{i,j=1}^{N} y_i y_j K\left(\mathbf{x}_i,\ \mathbf{x}_j\right)\alpha_i\alpha_j + \sum_{i=1}^{N}\alpha_i \\ \text{subject to:} \\ \sum_{i=1}^{N}\alpha_i y_i = 0 \\ 0 \le \alpha_i \le C, \quad i = 1, \ldots, N. \end{cases}$$

3.4 k-Nearest Neighbor

In the k-NN method, any new applicant is classified based on a comparison with the training sample using a distance metric. The approach consists of calculating the distances between the new instance that needs to be classified and each instance in the training sample that has been already classified and selecting the set of the k-nearest observations. Then, the class label is assigned according to the most common class among k-nearest neighbors using a majority voting scheme or a distance-weighted voting scheme [41]. One major drawback of the k-NN method is that it is extremely sensitive to the choice of the parameter k, as illustrated in Fig. 5. Given the same dataset, if k=1 the new instance is classified as “bad,” while if k=3 the neighborhood contains one “bad” and two “good” applicants, thus, the new instance will be classified as “good.” In general, using a small k leads to overfitting (i.e., excessive adaptation to the training dataset), while using a large k reduces accuracy by including data points that are too far from the new case [41].

The most common choice of a distance metric is the Euclidean distance, which can be computed as:

$$d\left(\mathbf{x}_i, \mathbf{x}_j\right) = ||\mathbf{x}_i - \mathbf{x}_j|| = \left[\left(\mathbf{x}_i - \mathbf{x}_j\right)^T\left(\mathbf{x}_i - \mathbf{x}_j\right)\right]^{\frac{1}{2}} \tag{30}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the vectors of the input data of instances i and j, respectively. Once the distances between the newest and every instance in the training sample are calculated, the new instance can be classified based on the information available

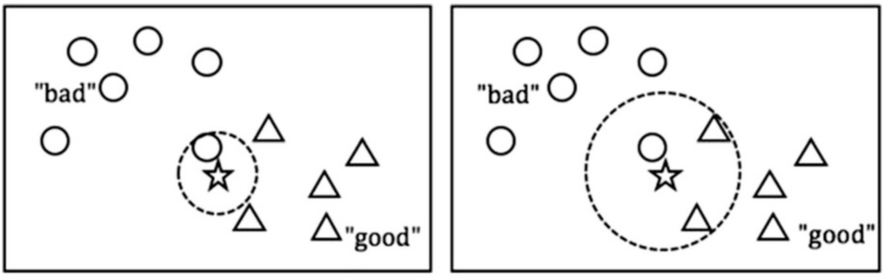


Fig. 5 The figure illustrates the main problem of a k-NN method with the majority voting approach: its sensitivity to the choice of k. On the left side of the figure, a model with k=1 is shown. Based on such a model, the new client (marked by a star symbol) would be classified as "bad." However, on the right side of the figure, a model with k=3 classifies the same new client as "good." Source: Tan et al. [41]

from its k-nearest neighbors. As seen above, the most common approach is to use the majority class of k-nearest examples, the so-called majority voting approach

$$y^{new} = \arg\max_{\nu} \sum_{(x_i, y_i) \in \mathbf{S}_k} I(\nu = y_i), \tag{31}$$

where y^{new} is the class of the new instance, ν is a class label, $\mathbf{S}_k$ is the set containing k-closest training instances, y_i is the class label of one of the k-nearest observations, and $I(\bullet)$ is a standard indicator function.

The major drawback of the majority voting approach is that it gives the same weight to every k-nearest neighbor. This makes the method very sensitive to the choice of k, as discussed previously. However, this problem might be overcome by attaching to each neighbor a weight based on its distance from the new instance, i.e.,

$$\omega_i = \frac{1}{d\left(\mathbf{x}_i, \mathbf{x}_j\right)^2} \tag{32}$$

This approach is known as the distance-weighted voting scheme, and the class label of the new instance can be found in the following way:

$$y^{new} = \arg\max_{\nu} \sum_{(x_i, y_i) \in \mathbf{S}_k} \omega_i I(\nu = y_i), \tag{33}$$

One of the main advantages of k-NN is its simplicity. Indeed, its logic is similar to the process of traditional credit decisions, which were made by comparing a new applicant with similar applicants [10]. However, because estimation needs to be performed afresh when one is to classify a new instance, the classification speed may be slow, especially with large training samples.

3.5 Genetic Algorithms

GA are heuristic, combinatorial optimization search techniques employed to determine automatically the adequate discriminant functions and the valid attributes [35]. The search for the optimal solution to a problem with GA imitates the evolutionary process of biological organisms, as in Darwin's natural selection theory. In order to understand how a GA works in the context of credit scoring, let us suppose that $(x_1, \ldots, x_n)$ is a set of attributes used to decide whether an applicant is good or bad according to a simple linear function:

$$y = \beta_0 + \sum_{i=1}^{N} \beta_i x_i. \tag{34}$$

Each solution is represented by the vector $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_N)$ whose elements are the coefficients assigned to each attribute. The initial step of the process is the generation of a random population of solutions $\boldsymbol{\beta}_J^0$ and the evaluation of their fitness using a fitness function. Then, the following algorithms are applied:

1. **Selection**: a genetic operator selects the solutions that survive (the fittest solutions)
2. **Crossover**: a genetic operator recombines the survived solutions
3. **Mutation**: a genetic operator allows for random mutations in the survived solutions, with a low probability

The application of these algorithms results in the generation of a new population of solutions $\boldsymbol{\beta}_J^0$. The algorithms selection-crossover-mutation are applied recursively until an (approximate) optimal solution $\boldsymbol{\beta}_J^*$ is converged to.

Compared to traditional statistical approaches and NN, GA offers the advantage of not being limited in its effectiveness by the form of functions and parameter estimation [11]. Furthermore, GA is a nonparametric tool that can perform well even in small datasets [34].

3.6 Ensemble Methods

In order to improve the accuracy of the individual (or *base*) classifiers illustrated above, ensemble (or classifier combination) methods are often used [41]. Ensemble methods are based on the idea of training multiple models to solve the same problem and then combine them to get better results. The main hypothesis is that when weak models are correctly combined, we can obtain more accurate and/or robust models. In order to understand why ensemble classifiers may reduce the error rate of individual models, it may be useful to consider the following example.

Example Suppose that an ensemble classifier is created by using 25 different base classifiers and that each classifier has an error rate $\epsilon_i = 0.25$. If the final credit decision is taken through a majority vote (i.e., if the majority of the classifiers suggests that the customer is a "good" one, then the credit is granted), the error rate of the ensemble model is

$$\epsilon_{ensemble} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.003, \tag{35}$$

where $i = 13, \ldots, 25$, which is much less than the individual rate of 0.25, because the ensemble model would make a wrong decision only if more than half of the base classifiers yield a wrong estimate.

It is easy to understand that ensemble classifiers perform especially well when they are uncorrelated. Although in real-world applications it is difficult to obtain base classifiers that are totally uncorrelated, considerable improvements in the performance of ensemble classifiers are observed even when some correlations exists but are low [17]. Ensemble models can be split into homogeneous and heterogeneous. Homogeneous ensemble models use only one type of classifier and rely on resampling techniques to generate k different classifiers that are then aggregated according to some rule (e.g., majority voting). Examples of homogeneous ensemble models are *bagging* and *boosting* methods. More precisely, the bagging algorithm creates k bootstrapped samples of the same size as the original one by drawing with replacement from the dataset. All the samples are created in parallel and the estimated classifiers are aggregated according to majority voting. Boosting algorithms work in the same spirit as bagging but the models are not fitted in parallel: a sequential approach is used and at each step of the algorithm the model is fitted, giving more importance to the observations in the training dataset that were badly handled in the previous iteration. Although different boosting algorithms are possible, one of the most popular is AdaBoost. AdaBoost was first introduced by Freund and Schapire [19]. This algorithm starts by calculating the error of a base classifier h_t:

$$\epsilon_t = \frac{1}{N}\left[\sum_{j=1}^{N} \omega_j I\left(h_t\left(x_j\right) \neq y_j\right)\right]. \tag{36}$$

Then, the importance of the base classifier h_t is calculated as:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right). \tag{37}$$

The parameter α_t is used to update the weights assigned to the training instances. Let $\omega_i^{(t)}$ be the weight assigned to the training instance i in the t^{th} boosting round. Then, the updated weight is calculated as:

$$\omega_i^{(t+1)} = \frac{\omega_i^{(t)}}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\mathbf{x}_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(\mathbf{x}_i) \neq y_i, \end{cases} \tag{38}$$

where Z_t is the normalization factor, such that $\sum_i \omega_i^{(t+1)} = 1$. Finally, the AdaBoost algorithm decision is based on

$$h\left(\mathbf{x}\right) = sign\left(\sum_{t=1}^{T} \alpha_t h_t\left(\mathbf{x}\right)\right). \tag{39}$$

In contrast to homogeneous ensemble methods, heterogeneous ensemble methods combine different types of classifiers. The main idea behind these methods is that different algorithms might have different views on the data and thus combining them helps to achieve remarkable improvements in predictive performance [47]. An example of heterogeneous ensemble method can be the following:

1. Create a set of different classifiers $H = \{h_t, t = 1, \ldots, T\}$ that map an instance in the training sample to a class y_i: $h_t\left(\mathbf{x}_i\right) = y_i$.
2. Start with an empty ensemble ($S = \emptyset$).
3. Add to the ensemble the model from the initial set that maximizes the performance of the ensemble on the validation dataset according to the error metric.
4. Repeat Step 3 for k iterations, where k is usually less than the number of models in the initial set.

A comparative evaluation of alternative ensemble methods is provided in Sect. 4.2.

4 Comparison of Classifiers in Credit Scoring Applications

The selection of the best classification algorithm among all methods that have been proposed in the literature has always been a challenging research area. Although many studies have examined the performance of different classifiers, most of these papers have traditionally focused only on a few novel algorithms at the time and, thus, have generally failed to provide a comprehensive overview of pros and cons of alternative methods. Moreover, in most of these papers, a relatively small number of datasets were used, which limited the practical applicability of the empirical results reported. One of the most comprehensive studies that attempts to overcome these issues and to apply thorough statistical tests to compare different algorithms has been published by Stefan Lessmann and his coauthors [29]. By combining their results with other, earlier studies, this section seeks to isolate the best classification algorithms for the purposes of credit scoring.

4.1 Comparison of Individual Classifiers

In the first decade of the 2000s, the focus of most papers had been on performing comparisons among individual classifiers. Understandably, the question of whether advanced methods of classification, such as NN and SVM, might outperform LR and LDA had attracted much attention. While some authors have since then concluded that NN classifiers are superior to both LR and LDA (see, e.g., [2]), generally, it has been shown that simple linear classifiers lead to a satisfactory performance and, in most cases, that the differences between NN and LR are not statistically significant [5]. This section compares the findings of twelve papers concerning individual classifiers in the field of credit scoring. Papers were selected based on two features: first, the number of citations, and, second, the publishing date. The sample combines well-known papers (i.e., [45, 5]) with recent work (e.g., [29, 3]) in an attempt to provide a well-rounded overview.

One of the first comprehensive comparisons of linear methods with more advanced classifiers was West [45]. He tested five NN models, two parametric models (LR, LDA), and three nonparametric models (k-NN, kernel density, and DT) on two real-world datasets. He found that in the case of both datasets, LR led to the lowest credit scoring error, followed by the NN models. He also found that the differences in performance scores of the superior models (LR and three different way to implement NN) vs. the outperformed models were not statistically significant. Overall, he concluded that LR was the best choice among individual classifiers he tested. However, his methodology presented a few drawbacks that made some of his findings potentially questionable. First, West [45] used only one method of performance evaluation and ranking, namely, average scoring accuracy. Furthermore, the size of his datasets was small, containing approximately 1700 observations in total (1000 German credit applicants, 700 of which were creditworthy, and 690 Australian applicants, 307 of which were creditworthy).

Baesens et al. [5] remains one of the most comprehensive comparisons of different individual classification methods. This paper overcame the limitations in West [45] by using eight extensive datasets (for a total of 4875 observations) and multiple evaluation methods, such as the percentage of correctly classified cases, sensitivity, specificity, and the area under the receiver operating curve (henceforth, AUC, an accuracy metric that is widely used when evaluating different classifiers).[3] However, the results reported by Baesens et al. [5] were similar to West's [45]: NN

[3] A detailed description of the performance measurement metrics that are generally used to evaluate the accuracy of different classification methods can be found in the previous chapter by Bargagli-Stoffi et al. [6].

and SVM classifiers had the best average results; however, also LR and LDA showed a very good performance, suggesting that most of the credit datasets are only weakly nonlinear. These results have found further support in the work of Lessmann et al. [29], who updated the findings in [5] and showed that NN models perform better than LR model, but only slightly.[4]

These early papers did not contain any evidence on the performance of GA. One of the earliest papers comparing genetic algorithms with other credit scoring models is Yobas et al. [49], who compared the predictive performance of LDA with three computational intelligence techniques (a NN, a decision tree, and a genetic algorithm) using a small sample (1001 individuals) of credit scoring data. They found that LDA was superior to genetic algorithms and NN. Fritz and Hosemann [20] also reached a similar conclusion even though doubts existed on their use of the same training and test sets for different techniques. Recently, these early results have been overthrown. Ong et al. [35] compared the performance of genetic algorithms to MLP, decision trees (CART and C4.5), and LR using two real-world datasets, which included 1690 observations. Genetic algorithms turned out to outperform other methods, showing a solid performance even on relatively small datasets. Huang et al. [26] compared the performance of GA against NN, SVM, and decision tree models in a credit scoring application using the Australian and German benchmark data (for a total of almost 1700 credit applicants). Their study revealed superior classification accuracy from GA than under other techniques, although differences are marginal. Abdou [1] has investigated the relative performance of GA using data from Egyptian public sector banks, comparing this technique with probit analysis, reporting that GA achieved the highest accuracy rate and also the lowest type-I and type-II errors when compared with other techniques.

One more recent and comprehensive study is that of Finlay [16], who evaluated the performance of five alternative classifiers, namely, LR, LDA, CART, NN, and k-NN, using the rather large dataset of Experian UK on credit applications (including a total of 88,789 applications, 13,261 of which were classified as “bad”). He found that the individual model with the best performance is NN; however, he also showed that the overperformance of nonlinear models over their linear counterparts is rather limited (in line with [5]).

Starting in 2010, most papers have shifted their focus to comparisons of the performance of ensemble classifiers, which are covered in the next section. However, some recent studies exist that evaluate the performance of individual classifiers. For instance, Ala’raj and Abbod [2] (who used five real-world datasets for a total of 3620 credit applications) and Bequé and Lessmann [7] (who used three real-world credit datasets for a total of 2915 applications) have found that LR has the best performance among the range of individual classifiers they considered.

[4] Importantly, compared to Baesens et al. [5], Lessmann et al. [29] used the more robust H-measure instead of the AUC as a key performance indicator for their analysis. Indeed, as emphasized by Hand [21], the AUC has an important drawback as it uses different misclassification cost distributions for different classifiers (see also Hand and Anagnostopoulos [22]).

Although ML approaches are better at capturing nonlinear relationships, similarly to what is typical in credit risk applications (see [4]), it could be concluded that, in general, a simple LR model remains a solid choice among individual classifiers.

4.2 Comparison of Ensemble Classifiers

According to Lessmann et al. [29], the new methods that have appeared in ML have led to superior performance when compared to individual classifiers. However, only a few papers concerning credit scoring have examined the potential of ensemble methods, and most papers have focused on simple approaches. This section attempts to determine whether ensemble classifiers offer significant improvements in performance when compared to the best available individual classifiers and examines the issue of uncovering which ensemble methods may provide the most promising results. To succeed in this objective, we have selected and surveyed ten key papers concerning ensemble classifiers in the field of credit scoring.

West et al. [46] were among the first researchers to test the relative performance of ensemble methods in credit scoring. They selected three ensemble strategies, namely, cross-validation, bagging, and boosting, and compared them to the MLP NN as a base classifier on two datasets.[5] West and coauthors concluded that among the three chosen ensemble classifiers, boosting was the most unstable and had a mean error higher than their baseline model. The remaining two ensemble methods showed statistically significant improvements in performance compared to MLP NN; however, they were not able to single out which ensemble strategy performed the best since they obtained contrasting results on the two test datasets. One of the main limitations of this seminal study is that only one metric of performance evaluation was employed. Another extensive paper on the comparative performance of ensemble classifiers is Zhou et al.'s [51]. They compared six ensemble methods based on LS-SVM to 19 individual classifiers, with applications to two different real-world datasets (for a total of 1113 observations). The results were evaluated using three different performance measures, i.e., sensitivity, the percentage of correctly classified cases, and AUC. They reported that the ensemble methods assessed in their paper could not lead to results that would be statistically superior to an LR individual classifier. Even though the differences in performance were not large, the ensemble models based on the LS-SVM provided promising solutions to the classification problem that was not worse than linear methods. Similarly, Louzada et al. [30] have recently used three famous and publicly available datasets (the Australian, the German, and the Japanese credit data) to perform simulations under both balanced (p = 0:5, 50% of bad payers) and imbalanced cases (p = 0:1,

[5] While bagging and boosting methods work as described in Sect. 3, the cross-validation ensemble, also known as CV, has been introduced by Hansen and Salamon [24] and it consists of an ensemble of similar networks, trained on the same dataset.

10% of bad payers). They report that two methods, SVM and fuzzy complex systems offer a superior and statistically significant predictive performance. However, they also notice that in most cases there is a shift in predictive performance when the method is applied to imbalanced data. Huang and Wu [25] report that the use of boosted GA methods improves the performance of underlying classifiers and appears to be more robust than single prediction methods. Marqués et al. [31] have evaluated the performance of seven individual classifier techniques when used as members of five different ensemble methods (among them, bagging and AdaBoost) on six real-world credit datasets using a fivefold cross-validation method (each original dataset was randomly divided into five stratified parts of equal size; for each fold, four blocks were pooled as the training data, and the remaining part was employed as the hold out sample). Their statistical tests show that decision trees constitute the best solution for most ensemble methods, closely followed by the MLP NN and LR, whereas the k-NN and the NB classifiers appear to be significantly the worst.

All the papers discussed so far did not offer a comprehensive comparison of different ensemble methods, but rather they focused on a few techniques and compared them on a small number of datasets. Furthermore, they did not always adopt appropriate statistical tests of equal classification performance. The first comprehensive study that has attempted to overcome these issues is Lessmann et al. [29], who have compared 16 individual classifiers with 25 ensemble algorithms over 8 datasets. The selected classifiers include both homogeneous (including bagging and boosting) and heterogeneous ensembles. The models were evaluated using six different performance metrics. Their results show that the best individual classifiers, namely, NN and LR, had average ranks of 14 and 16 respectively, being systematically dominated by ensemble methods. Based on the modest performance of individual classifiers, Lessmann et al. [29] conclude that ML techniques have progressed notably since the first decade of the 2000s. Furthermore, they report that heterogeneous ensemble classifiers provide the best predictive performance.

Lessmann et al. [29] have also examined the potential financial implications of using ensemble scoring methods. They considered 25 different cost ratios based on the assumption that accepting a “bad” application always costs more than denying a “good” application [42]. After testing three models (NN, RF, and HCES-Bag) against LR, Lessmann et al. [29] conclude that for all cost ratios, the more advanced classifiers led to significant cost savings. However, the most accurate ensemble classifier, HCES-Bag, on average achieved lower cost savings than the radial basis function NN method, 4.8 percent and 5.7 percent, respectively. Based on these results, they suggested that the most statistically accurate classifier may not always be the best choice for improving the profitability of the credit lending business.

Two additional studies, Florez-Lopez and Ramon-Jeronimo [18] and Xia et al. [48], have focused on the interpretability of ensemble methods, constructing ensemble models that can be used to support managerial decisions. Their empirical results confirmed the findings of Lessmann et al. [29] that ensemble methods consistently lead to better performances than individual scoring. Furthermore, they concluded that it is possible to build an ensemble model that has both high interpretability and a high accuracy rate. Overall, based on the papers considered in this section, it is evident that ensemble models offer higher accuracy compared to the best individual models. However, it is impossible to select one ensemble approach that will have the best performance over all datasets and error costs. We expect that scores of future papers will appear with new, more advanced methods and that the search for "the silver bullet" in the field of credit scoring will not end soon.

4.3 One-Class Classification Methods

Another promising development in credit scoring concerns one-class classification methods (OCC), i.e., ML methods that try to learn from one class only. One of the biggest practical obstacles to applying scoring methods is the class imbalance feature of most (all) datasets, the so-called low-default portfolio problem. Because financial institutions only store historical data concerning the accepted applicants, the characteristics of "bad" applicants present in their data bases may not be statistically reliable to provide a basis for future predictions ([27]. Empirical and theoretical work has demonstrated that the accuracy rate may be strongly biased with respect to imbalance in class distribution and that it may ignore a range of misclassification costs [14], as in applied work it is generally believed that the costs associated with type-II errors (bad customers misclassified as good) are much higher than the misclassification costs associated with type-I errors (good customers mispredicted as bad). OCC attempts to differentiate a set of target instances from all the others. The distinguishing feature of OCC is that it requires labeled instances in the training sample for the target class only, which in the case of credit scoring are "good" applicants (as the number of "good" applicants is larger than that of "bad" applicants). This section surveys whether OCC methods can offer a comparable performance to the best two-class classifiers in the presence of imbalanced data features.

The literature on this topic is still limited. One of the most comprehensive studies is a paper by Kennedy [27], in which he compared eight OCC methods, in which models are separately trained over different classes of datasets, with eight two-class individual classifiers (e.g., k-NN, NB, LR) over three datasets. Two important conclusions emerged. First, the performance of two-class classifiers deteriorates significantly with an increasing class imbalance. However, the performance of some classifiers, namely, LR and NB, remains relatively robust even for imbalanced datasets, while the performance of NN, SVM, and k-NN deteriorates rapidly. Second, one-class classifiers show superior performance compared to two-class classifiers only at high levels of imbalance (starting at 99% of "good" and 1% of "bad" applicants). However, the differences in performance between OCC models and LR model were not statistically significant in most cases. Kennedy [27] concluded that OCC methods failed to show statistically significant improvements in performance compared to the best two-class classification methods. Using a proprietary dataset from a major US commercial bank from January 2005 to April

2009, Khandani et al. [28] showed that conditioning on certain changes in a consumer's bank account activity can lead to considerably more accurate forecasts of credit card delinquencies by analyzing subtle nonlinear patterns in consumer expenditures, savings, and debt payments using CART and SVM compared to simple regression and logit approaches. Importantly, their trees are "boosted" to deal with the imbalanced features of the data: instead of equally weighting all the observations in the training set, they weight the scarcer observations more heavily than the more populous ones.

These findings are in line with studies in other fields. Overall, the conclusion that can be drawn is that OCC methods should not be used for classification problems in credit scoring. Two-class individual classifiers show superior or comparable performance for all cases, except for cases of extreme imbalances.

5 Conclusion

The field of credit scoring represents an excellent example of how the application of novel ML techniques (including deep learning and GA) is in the process of revolutionizing both the computational landscape and the perception by practitioners and end-users of the relative merits of traditional vs. new, advanced techniques. On the one hand, in spite of their logical appeal, the available empirical evidence shows that ML methods often struggle to outperform simpler, traditional methods, such as LDA, especially when adequate tests of equal predictive accuracy are deployed. Although some of these findings may be driven by the fact that some of the datasets used by the researchers (especially in early studies) were rather small (as in the case, for instance, of West [45]), linear methods show a performance that is often comparable to that of ML methods also when larger datasets are employed (see, e.g., Finlay [17]). On the other hand, there is mounting experimental and on-the-field evidence that ensemble methods, especially those that involve ML-based individual classifiers, perform well, especially when realistic cost functions of erroneous classifications are taken into account. In fact, it appears that the issues of ranking and assessing alternative methods under adequate loss functions, and the dependence of such rankings on the cost structure specifications, may turn into a fertile ground for research development.

References

1. Abdou, H. A. (2009). Genetic programming for credit scoring: The case of Egyptian public sector banks. *Expert Systems with Applications, 36*(9), 11402–11417.
2. Abdou, H., Pointon, J., & El-Masry, A. (2008). Neural nets versus conventional techniques in credit scoring in Egyptian banking. *Expert Systems with Applications, 35*(3), 1275–1292.
3. Ala'raj, M., & Abbod, M. F. (2016). Classifiers consensus system approach for credit scoring. *Knowledge-Based Systems, 104*, 89–105.

4. Bacham, D., & Zhao, J. (2017). Machine learning: challenges, lessons, and opportunities in credit risk modelling. *Moody's Analytics Risk Perspectives/Managing Disruptions, IX*, 1–5.
5. Baesens, B., Gestel, T. V., Viaene, S., Stepanova, M., Suykens, J., & Vanthienen, J. (2003) Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society, 54*, 627–635.
6. Bargagli-Stoffi, F. J., Niederreiter, J., & Riccaboni, M. (2021). Supervised learning for the prediction of firm dynamics. In S. Consoli, D. Reforgiato Recupero, & M. Saisana (Eds.) *Data Science for Economics and Finance: Methodologies and Applications* (pp. 19–41). Switzerland: Springer-Nature.
7. Bequé, A., & Lessmann, S. (2017). Extreme learning machines for credit scoring: An empirical evaluation. *Expert Systems with Applications, 86*, 42–53.
8. Bishop, C. (1994). Novelty detection and neural network validation. *IEE Proceedings on Vision, Image and Signal Processing, 141*, 217–222.
9. Bishop, C. M. (1999). *Neural Networks for Pattern Recognition*. Oxford, United Kingdom: Oxford University.
10. Bunker, R., Naeem, A., & Zhang, W. (2016). *Improving a credit scoring model by incorporating bank statement derived features*. Working paper, Auckland University of Technology. arXiv, CoRR abs/1611.00252.
11. Chi, B., & Hsu, C. (2011). A hybrid approach to integrate genetic algorithm into dual scoring model in enhancing the performance of credit scoring model. *Expert Systems with Applications, 39*, 2650–2661.
12. Chui, M., Manyika, J., & Miremadi, M. (2018). *What AI can and can't do (yet) for your business*. https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/what-ai-can-and-cant-do-yet-for-your-business.
13. Dorie, V., Hill, J., Shalit, U., Scott, M., & Cervone, D. (2019). Automated versus do-it-yourself methods for causal inference: Lessons learned from a data analysis competition? *Statistical Science, 34*, 43–68.
14. Fawcett, T. & Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery, 1*(3), 291–316.
15. Federal Reserve Bank of New York (2020). *Household debt and credit report (Q4 2020), Center FRO Microeconomic data*. https://www.newyorkfed.org/microeconomics/hhdc.
16. Finlay, S. M. (2009). Are we modelling the right thing? The impact of incorrect problem specification in credit scoring. *Expert Systems with Applications, 36*(5), 9065–9071.
17. Finlay, S. (2011). Multiple classifier architectures and their application to credit risk assessment. *European Journal of Operational Research, 210*, 368–378.
18. Florez-Lopez, R., & Ramon-Jeronimo, J. M. (2015). Enhancing accuracy and interpretability of ensemble strategies in credit risk assessment. A correlated-adjusted decision forest proposal. *Expert Systems with Applications, 42*, 5737–5753.
19. Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*, 119–139.
20. Fritz, S., & Hosemann, D. (2000). Restructuring the credit process: Behaviour scoring for German corporates. *Intelligent Systems in Accounting, Finance & Management, 9*(1), 9–21.
21. Hand, D. J. (2009). Measuring classifier performance: A coherent alternative to the area under the roc curve. *Machine Learning, 77*(1), 103–123.
22. Hand, D. J., & Anagnostopoulos, C. (2014). A better beta for the h measure of classification performance. *Pattern Recognition Letters, 40*, 41–46.
23. Hand, D. J., & Zhou, F. (2010). Evaluating models for classifying customers in retail banking collections. *Journal of the Operational Research Society, 61*, 1540–1547.
24. Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*, 993–1001.
25. Huang, S. C. & Wu, C. F. (2011). Customer credit quality assessments using data mining methods for banking industries. *African Journal of Business Management, 5*(11), 4438–4445.
26. Huang, C. L., Chen, M. C., & Wang, C. J. (2007). Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications, 33*(4), 847–856.

27. Kennedy, K. (2013). Credit scoring using machine learning. Doctoral thesis, Technological University Dublin. https://doi.org/10.21427/D7NC7J.
28. Khandani, A. E., Kim, A. J., & Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance, 34*(11), 2767–2787.
29. Lessmann, S., Baesens, B., Seow, H., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.
30. Louzada, F., Ara, A., & Fernandes, G. B. (2016). Classification methods applied to credit scoring: Systematic review & overall comparison. *Surveys in Operations Research and Management Science, 21*(2), 117–134.
31. Marqués, A. I., García, V., & Sánchez, J. S. (2012). Exploring the behaviour of base classifiers in credit scoring ensembles. *Expert Systems with Applications, 39*(11), 10244–10250.
32. McCarthy, B., Chui, M., & Kamalnath, V. (2018). *An executive's guide to AI*. https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/an-executives-guide-to-ai.
33. Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. Cambridge, MA: MIT Press.
34. Nath, R., Rajagopalan, B., & Ryker, R. (1997). Determining the saliency of input variables in neural network classifiers. *Computers and Operations Researches, 24*, 767–773.
35. Ong, C., Huang, J., & Tzeng, G. (2005). Building credit scoring models using genetic programming. *Expert Systems with Applications, 29*, 41–47.
36. Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Applied Soft Computing, 93*, 106384.
37. Quinlan, J. R. (1993) *C4.5—Programs for machine learning*. San Francisco, CA, United States: Morgan Kaufmann Publishers.
38. Rohit, V. M., Kumar, S., Kumar, J. (2013). Basel II to Basel III the way forward. In *Infosys White Paper*. https://srinath-keshavan-naj7.squarespace.com/s/Basel-III_Basel-II-to-III.pdf.
39. Saunders, A., Allen, L. (2002). *Credit risk measurement: New approaches to value at risk and other paradigms*. New York: Wiley.
40. Sirignano, J., Sadhwani, A., Giesecke, K. (2018). Deep learning for mortgage risk. Technical report, Working paper available at SSRN. https://ssrn.com/abstract=2799443.
41. Tan, P., Steinbach, M., & Kumar, V. (2006). *Introduction to Data Mining*. New York, US: Pearson Educatio.
42. Thomas, L., Crook, J., & Edelman, D. (2017). Credit scoring and its applications. In *Society for Industrial and Applied Mathematics (SIAM), Philadelphia, US*. https://doi.org/10.1137/1.9781611974560.
43. Vapnik, N. (1998). *Statistical learning theory*. New York: Wiley.
44. Wang, Z., Jiang, C., Zhao, H., & Ding, Y. (2020). Mining semantic soft factors for credit risk evaluation in Peer-to-Peer lending. *Journal of Management Information Systems, 37*(1), 282–308.
45. West, D. (2000). Neural network credit scoring models. *Computers and Operations Research, 27*, 1131–1152.
46. West, D., Dellana, S., & Qian, J. (2005). Neural network ensemble strategies for financial decision applications. *Computers and Operations Research, 32*, 2543–2559.
47. Whalen, S., & Pandey, G. (2013). A comparative analysis of ensemble classifiers: Case studies in genomics. In *Data Mining (ICDM), 2013 IEEE 13th International Conference* (pp. 807–816). New Jersey: IEEE.
48. Xia, Y., Liu, C., Li, Y., & Liu, N. (2017). A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. *Expert Systems with Applications, 78*, 225–241.
49. Yobas, M. B., Crook, J. N. & Ross, P. (2000). Credit scoring using neural and evolutionary techniques. *IMA Journal of Mathematics Applied in Business and Industry, 11*(4), 111–125.

50. Zhao, Q., & Hastie, T. (2019). Causal interpretations of black-box models. *Journal of Business & Economic Statistics, 39*(1), 1–10.
51. Zhou, L., Lai, K. K., & Yu, L. (2010). Least squares support vector machines ensemble models for credit scoring. *Expert Systems with Applications, 37*, 127–133.