Chapter 11
Credit Scoring

Credit scoring provides a statistical assessment of a borrower's creditworthiness, helping financial institutions make informed decisions on e.g. loan applications. This chapter reviews the use of discriminant analysis and binomial logistic regression in credit scoring. We also discuss the properties of Receiver Operating Characteristic (ROC) curves.

11.1 Discriminant Analysis **273**
11.2 Decision Rule **277**
11.3 Logistic Regression **281**
11.4 ROC Curve **290**
Exercises **295**

11.1 Discriminant Analysis

Consider a set Ω of credit applicants which is partitioned as

$$\Omega = B \cup G$$

into a subset G of "good" (or solvent) applicants, and a subset B of "bad" (or insolvent) applicants, with $B \cap G = \emptyset$. Applicants are selected at random within the set Ω according to a probability distribution $\mathbb{P}$, so that

$$\mathbb{P}(B) + \mathbb{P}(G) = 1.$$

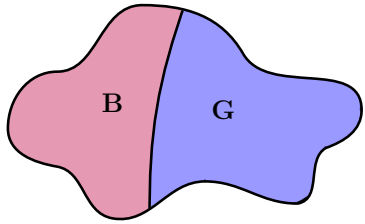

Fig. 11.1: Data set.

Here, $\mathbb{P}(B)$ represents the probability that an applicant chosen at random may default, while $\mathbb{P}(G)$ represents the probability that a randomly selected applicant is solvent. In addition, each applicant $\omega \in \Omega$ is assigned a real-valued rating (or score) $X(\omega)$ via a *random variable*[*]

$$\begin{aligned} X : \Omega &\longrightarrow \mathbb{R} \\ \omega &\longmapsto X(\omega) \end{aligned}$$

having the probability density function $f_X : \mathbb{R} \to \mathbb{R}_+$.

Definition 11.1. *1) The function*

$$\begin{aligned} \mathbb{R} &\longrightarrow [0,1] \\ z &\longmapsto \mathbb{P}(B \mid X = z) \end{aligned}$$

is called the probability default curve.

2) The function

$$\begin{aligned} \mathbb{R} &\longrightarrow [0,1] \\ z &\longmapsto \mathbb{P}(G \mid X = z) \end{aligned}$$

is called the probability acceptance *curve.*

Denoting by $f_X(z \mid B)$, resp. $f_X(z \mid G)$, the probability density function of X given B, resp. X given G, we have

$$f_X(z) = f_X(z \mid G)\mathbb{P}(G) + f_X(z \mid B)\mathbb{P}(B), \quad z \in \mathbb{R},$$

and the Bayes formula yields

$$\mathbb{P}(B \mid X = z) = f_X(z \mid B)\frac{\mathbb{P}(B)}{f_X(z)}$$

[*] See (MOE and UCLES 2022, page 14 lines 4-5) or (MOE and UCLES 2022, page 19 lines 4-5).

$$= \frac{\mathbb{P}(B) f_X(z \mid B)}{\mathbb{P}(G) f_X(z \mid G) + \mathbb{P}(B) f_X(z \mid B)}, \tag{11.1}$$

and

$$\mathbb{P}(G \mid X = z) = \frac{\mathbb{P}(G) f_X(z \mid G)}{\mathbb{P}(G) f_X(z \mid G) + \mathbb{P}(B) f_X(z \mid B)}.$$

The definitions of True Positive and False Positive rates are based on the conditional probabilities of acceptance of an applicant whose score X is above a lever z.

Definition 11.2. *1) The True Positive Rate (TPR) is given by the tail distribution function*

$$\overline{F}_G(z) := \mathbb{P}(X > z \mid G) = \int_z^\infty f_X(y \mid G) dy, \qquad z \in \mathbb{R}.$$

2) The False Positive Rate (FPR) is given by the tail distribution function

$$\overline{F}_B(z) := \mathbb{P}(X > z \mid B) = \int_z^\infty f_X(y \mid B) dy, \qquad z \in \mathbb{R}.$$

Proposition 11.3 will justify the use of logistic regression models in credit scoring.

Proposition 11.3. *Assume that the score X is Gaussian distributed given $\{G, B\}$, with the conditional densities*

$$f_X(z \mid G) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-(z-\mu_G)^2/(2\sigma^2)} \tag{11.2}$$

and

$$f_X(z \mid B) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-(z-\mu_B)^2/(2\sigma^2)}, \tag{11.3}$$

where $\mu_B < \mu_G$ and $\sigma > 0$. In this case, probability default and acceptance curves take the form

$$\mathbb{P}(B \mid X = z) = \frac{1}{1 + \mathrm{e}^{\gamma_0 + \gamma_1 z}} \quad \text{and} \quad \mathbb{P}(G \mid X = z) = \frac{\mathrm{e}^{\gamma_0 + \gamma_1 z}}{1 + \mathrm{e}^{\gamma_0 + \gamma_1 z}}, \tag{11.4}$$

$z \in \mathbb{R}$, where we let

$$\gamma_1 := \frac{\mu_G - \mu_B}{\sigma^2} > 0 \quad \text{and} \quad \gamma_0 := -\gamma_1 \frac{\mu_G + \mu_B}{2} + \log \frac{\mathbb{P}(G)}{\mathbb{P}(B)}.$$

Proof. We have

$$f_X(z) = \frac{\mathbb{P}(G)}{\sqrt{2\pi}\sigma} \mathrm{e}^{-(z-\mu_G)^2/(2\sigma^2)} + \frac{\mathbb{P}(B)}{\sqrt{2\pi}\sigma} \mathrm{e}^{-(z-\mu_B)^2/(2\sigma^2)}, \quad z \in \mathbb{R},$$

and by (11.1) we find the logistic probability default curve

$$\begin{aligned}\mathbb{P}(B \mid X=z) &= \frac{\mathbb{P}(B) f_X(z \mid B)}{\mathbb{P}(G) f_X(z \mid G) + \mathbb{P}(B) f_X(z \mid B)} \\ &= \frac{\mathbb{P}(B) \mathrm{e}^{-(z-\mu_B)^2/(2\sigma^2)}}{\mathbb{P}(G) \mathrm{e}^{-(z-\mu_G)^2/(2\sigma^2)} + \mathbb{P}(B) \mathrm{e}^{-(z-\mu_B)^2/(2\sigma^2)}} \\ &= \frac{1}{1+\mathrm{e}^{\gamma_0+\gamma_1 z}}, \qquad z \in \mathbb{R}.\end{aligned}$$

Similarly, the probability acceptance curve is given by

$$\begin{aligned}\mathbb{P}(G \mid X=z) &= \frac{\mathbb{P}(G) f_X(z \mid G)}{\mathbb{P}(G) f_X(z \mid G) + (1-\mathbb{P}(G)) f_X(z \mid B)} \\ &= \frac{\mathbb{P}(G) \mathrm{e}^{-(z-\mu_G)^2/(2\sigma^2)}}{\mathbb{P}(G) \mathrm{e}^{-(z-\mu_G)^2/(2\sigma^2)} + \mathbb{P}(B) \mathrm{e}^{-(z-\mu_B)^2/(2\sigma^2)}} \\ &= \frac{1}{1+\mathrm{e}^{-\gamma_0-\gamma_1 z}} \\ &= \frac{\mathrm{e}^{\gamma_0+\gamma_1 z}}{1+\mathrm{e}^{\gamma_0+\gamma_1 z}}, \qquad z \in \mathbb{R}. \end{aligned} \tag{11.5}$$

□

Figure 11.2 illustrates the probability default curve (11.2).

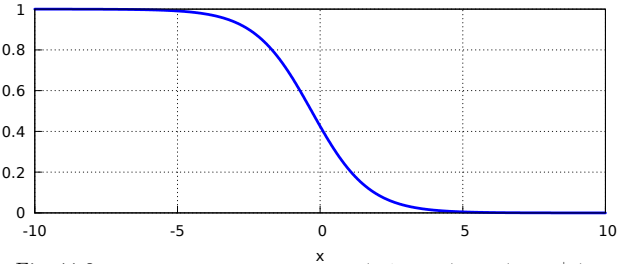

Fig. 11.2: Probability default curve $z \mapsto \mathbb{P}(B \mid X=z) = 1/(1+\mathrm{e}^{a+z})$.

Figure 11.3 illustrates the probability acceptance curve (11.3).

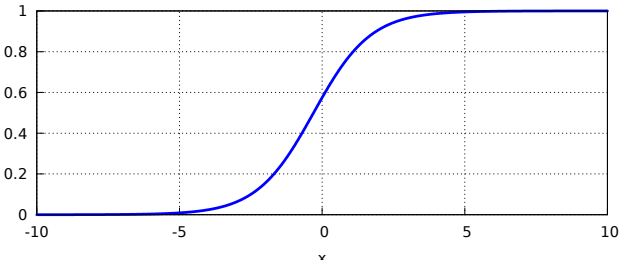

Fig. 11.3: Probability acceptance curve $z \mapsto \mathbb{P}(G \mid X = z) = \mathrm{e}^{a+z}/(1+\mathrm{e}^{a+z})$.

11.2 Decision Rule

We decide to accept the applicants whose score $X(\omega)$ belongs to a decision (or acceptance) set $\mathscr{A} \subset \mathbb{R}$, and to reject those whose score $X(\omega)$ belongs to the rejection set $\mathscr{A}^c = \mathbb{R} \setminus \mathscr{A}$ which is the complement of $\mathscr{A}$ in $\mathbb{R}$.

Definition 11.4. *Let*

- $D(z)$ *represents the cost incurred by the default of an applicant with score* $z \in \mathscr{A}$*, and*
- $L(z)$ *represents the loss (or missed earnings) incurred by the rejection of an applicant with score* $z \in \mathscr{A}^c$*.*

The cost associated to this decision rule is defined as

$$\underbrace{D(X)\mathbb{1}_{\{X\in\mathscr{A}\}\cap B}}_{\text{Cost of accepting a "bad" applicant}} + \underbrace{L(X)\mathbb{1}_{\{X\in\mathscr{A}^c\}\cap G}}_{\text{Loss from rejecting a "good" applicant}}$$

Theorem 11.5. *The optimal acceptance set* $\mathscr{A}^* \subset \mathbb{R}$ *that minimizes the expected cost*

$$\mathbb{E}\left[D(X)\mathbb{1}_{\{X\in\mathscr{A}\}\cap B} + L(X)\mathbb{1}_{\{X\in\mathscr{A}^c\}\cap G}\right]$$

is given by

$$\mathscr{A}^* = \left\{ z \in \mathbb{R} \;:\; \lambda(z) \geqslant \frac{D(z)}{L(z)} \frac{\mathbb{P}(B)}{\mathbb{P}(G)} \right\},$$

where $\lambda(z)$ *is the* likelihood ratio

$$\lambda(z) := \frac{f_X(z \mid G)}{f_X(z \mid B)} = \frac{\mathbb{P}(G \mid X = z)}{\mathbb{P}(B \mid X = z)} \frac{\mathbb{P}(B)}{\mathbb{P}(G)}, \qquad z \in \mathbb{R}.$$

Proof. The expected cost corresponding to an acceptance set $\mathscr{A} \subset \mathbb{R}$ can be written as

$$\begin{aligned}
&\mathbb{E}\left[D(X)\mathbb{1}_{\{X\in\mathscr{A}\}\cap B} + L(X)\mathbb{1}_{\{X\in\mathscr{A}^c\}\cap G}\right] \\
&= \int_{\mathscr{A}} D(z)\mathbb{P}(B\cap\{X\in dz\}) + \int_{\mathscr{A}^c} L(z)\mathbb{P}(G\cap\{X\in dz\}) \\
&= \int_{\mathscr{A}} D(z)\mathbb{P}(B \mid X=z)f_X(z)dz + \int_{\mathscr{A}^c} L(z)\mathbb{P}(G \mid X=z)f_X(z)dz \\
&= \int_{-\infty}^{\infty} \big(\mathbb{1}_{\mathscr{A}}(z)D(z)\mathbb{P}(B \mid X=z) + \mathbb{1}_{\mathscr{A}^c}(z)L(z)\mathbb{P}(G \mid X=z)\big)f_X(z)dz.
\end{aligned}$$

The expected cost can be minimized pointwise by finding the set $\mathscr{A}$ that minimizes its conditional value

$$\begin{aligned}
&\mathbb{E}\left[D(X)\mathbb{1}_{\{X\in\mathscr{A}\}\cap B} + L(X)\mathbb{1}_{\{X\in\mathscr{A}^c\}\cap G} \,\middle|\, X=z\right] \\
&\quad = \mathbb{1}_{\mathscr{A}}(z)D(z)\mathbb{P}(B \mid X=z) + \mathbb{1}_{\mathscr{A}^c}(z)L(z)\mathbb{P}(G \mid X=z)
\end{aligned}$$

given that $X = z$. For this, we note that

$$\begin{aligned}
&\mathbb{1}_{\mathscr{A}}(z)D(z)\mathbb{P}(B \mid X=z) + \mathbb{1}_{\mathscr{A}^c}(z)L(z)\mathbb{P}(G \mid X=z) \\
&\quad \geqslant \min(D(z)\mathbb{P}(B \mid X=z), L(z)\mathbb{P}(G \mid X=z)) \\
&\quad = D(z)\mathbb{P}(B \mid X=z)\mathbb{1}_{\{D(z)\mathbb{P}(B|X=z)\leqslant L(z)\mathbb{P}(G|X=z)\}} \\
&\qquad + L(z)\mathbb{P}(G \mid X=z)\mathbb{1}_{\{L(z)\mathbb{P}(G|X=z)<D(z)\mathbb{P}(B|X=z)\}} \\
&\quad = D(z)\mathbb{P}(B \mid X=z)\mathbb{1}_{\mathscr{A}^*}(z) + L(z)\mathbb{P}(G \mid X=z)\mathbb{1}_{(\mathscr{A}^*)^c}(z),
\end{aligned}$$

where the set $\mathscr{A}^*$ which achieves equality in the above inequality is given by

$$\begin{aligned}
\mathscr{A}^* &:= \{D(z)\mathbb{P}(B \mid X=z) \leqslant L(z)\mathbb{P}(G \mid X=z)\} \\
&= \left\{z\in\mathbb{R} \,:\, \frac{D(z)}{L(z)} \leqslant \frac{\mathbb{P}(G \mid X=z)}{\mathbb{P}(B \mid X=z)}\right\}.
\end{aligned}$$

In addition, the optimal acceptance set $\mathscr{A}^*$ can be rewritten in terms of the *likelihood ratio* function

$$\lambda(z) := \frac{f_X(z \mid G)}{f_X(z \mid B)} = \frac{\mathbb{P}(G \mid X=z)}{\mathbb{P}(B \mid X=z)}\frac{\mathbb{P}(B)}{\mathbb{P}(G)}, \qquad z\in\mathbb{R},$$

as

$$\mathscr{A}^* = \left\{z\in\mathbb{R} \,:\, \lambda(z) \geqslant \frac{D(z)}{L(z)}\frac{\mathbb{P}(B)}{\mathbb{P}(G)}\right\}.$$

□

For simplicity, in the sequel we assume that $D = D(z)$ and $L = L(z)$ are constant in $z \in \mathbb{R}$, in which case we have

$$\mathscr{A}^* = \left\{ z \in \mathbb{R} \ : \ \lambda(z) \geqslant \frac{D}{L} \frac{\mathbb{P}(B)}{\mathbb{P}(G)} \right\}.$$

Proposition 11.6. *In the Gaussian example (11.2)-(11.3) with $\mu_B < \mu_G$, the optimal acceptance set $\mathscr{A}^*$ is given by the interval*

$$\mathscr{A}^* = \left[\frac{\mu_G + \mu_B}{2} + \frac{1}{\gamma_1} \log \left(\frac{D}{L} \frac{\mathbb{P}(B)}{\mathbb{P}(G)} \right), \infty \right), \tag{11.6}$$

where

$$\gamma_1 := \frac{\mu_G - \mu_B}{\sigma^2} > 0,$$

under the condition

$$\mathbb{E}[X \mid B] = \mu_B < \mu_G = \mathbb{E}[X \mid G].$$

Proof. Using (11.2)-(11.3), the likelihood ratio is given by

$$\begin{aligned} \lambda(z) &= \frac{f_X(z \mid G)}{f_X(z \mid B)} \\ &= \mathrm{e}^{-(z-\mu_G)^2/(2\sigma^2)+(z-\mu_B)^2/(2\sigma^2)} \\ &= \mathrm{e}^{-(\mu_G^2-\mu_B^2-2(\mu_G-\mu_B)z)/(2\sigma^2)} \\ &= \mathrm{e}^{\gamma_1 z-(\mu_G^2-\mu_B^2)/(2\sigma^2)}, \qquad z \in \mathbb{R}, \end{aligned}$$

hence the condition

$$\lambda(z) = \mathrm{e}^{\gamma_1 z-(\mu_G^2-\mu_B^2)/(2\sigma^2)} \geqslant \frac{D}{L} \frac{\mathbb{P}(B)}{\mathbb{P}(G)}$$

is equivalent to

$$\begin{aligned} z &\geqslant \frac{\mu_G^2 - \mu_B^2}{2\gamma_1\sigma^2} + \frac{1}{\gamma_1} \log \left(\frac{D}{L} \frac{\mathbb{P}(B)}{\mathbb{P}(G)} \right) \\ &\geqslant \frac{\mu_G^2 - \mu_B^2}{2(\mu_G - \mu_B)} + \frac{1}{\gamma_1} \log \left(\frac{D}{L} \frac{\mathbb{P}(B)}{\mathbb{P}(G)} \right) \\ &\geqslant \frac{\mu_G + \mu_B}{2} + \frac{1}{\gamma_1} \log \left(\frac{D}{L} \frac{\mathbb{P}(B)}{\mathbb{P}(G)} \right) \\ &=: z^*, \end{aligned}$$

hence $\mathscr{A}^* = [z^*, \infty)$, provided that

$$\gamma_1 := \frac{\mu_G - \mu_B}{\sigma^2} > 0,$$

which yields (11.6). □

We note that the optimal boundary point z^* satisfies the relation

$$\lambda(z^*) = \frac{\mathbb{P}(X = z^* \mid G)}{\mathbb{P}(X = z^* \mid B)} = \frac{\mathbb{P}(G \mid X = z^*)}{\mathbb{P}(B \mid X = z^*)} \frac{\mathbb{P}(B)}{\mathbb{P}(G)} = \frac{D}{L} \frac{\mathbb{P}(B)}{\mathbb{P}(G)},$$

i.e.

$$\frac{\mathbb{P}(G \mid X = z^*)}{\mathbb{P}(B \mid X = z^*)} = \frac{D}{L}. \tag{11.7}$$

Figure 11.4 illustrates the optimal decision rule by taking $L = D = \$1$ and using the default and acceptance curves (11.4)-(11.5).

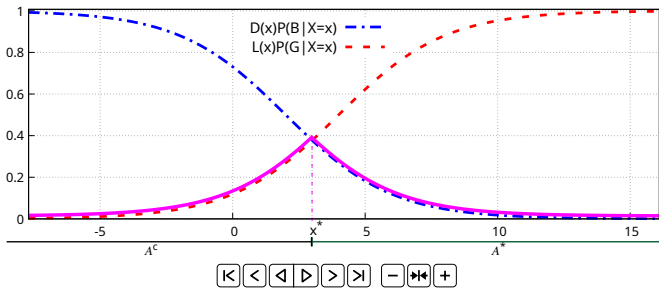

Fig. 11.4: Animated graph of optimal decision rule.*

In Figure 11.2, the conditional expected cost function

$$z \longmapsto \mathbb{1}_{\mathscr{A}}(z) D(z) \mathbb{P}(B \mid X = z) + \mathbb{1}_{\mathscr{A}^c}(z) L(z) \mathbb{P}(G \mid X = z)$$

is represented by the purple curve for a given set $\mathscr{A}$. Its uniform minimum over different threshold values is obtained for $\mathscr{A}^*$ of the form $\mathscr{A}^* = [z^*, \infty)$ where z^* lies at the intersection of the curves $z \mapsto D(z)\mathbb{P}(B \mid X = z)$ and $z \mapsto L(z)\mathbb{P}(G \mid X = z)$ as in (11.7).

The acceptance rate, or probability that an applicant is accepted according to the rule $\mathscr{A}$, is given by

$$\begin{aligned} \mathbb{P}(X \in \mathscr{A}) &= \mathbb{P}(\{X \in \mathscr{A}\} \cap G) + \mathbb{P}(\{X \in \mathscr{A}\} \cap B) \\ &= \mathbb{P}(X \in \mathscr{A} \mid G)\mathbb{P}(G) + \mathbb{P}(X \in \mathscr{A} \mid B)\mathbb{P}(B), \end{aligned}$$

where

$$\mathbb{P}(\{X \in \mathscr{A}\} \cap B) = \mathbb{P}(X \in \mathscr{A} \mid B)\mathbb{P}(B)$$

* The animation works in Acrobat Reader on the entire pdf file.

is the default rate, or probability that an applicant accepted according to the rule $\mathscr{A}$ will default. We can attempt to minimize the default rate $\mathbb{P}(\{X \in \mathscr{A}\} \cap B)$ subject to a given acceptance rate $\mathbb{P}(X \in \mathscr{A}) = a$.

11.3 Logistic Regression

In this section, we address the problem of constructing the random score variable $X : \Omega \to \mathbb{R}$ in a concrete setting in which $\Omega = \{1, 2, \ldots, n\}$ is a finite set. For this, consider a set of m financial criteria or indicators $(x_{\omega,j})_{j=1,2,\ldots,m}$ applying to each of n credit applicants $\omega = 1, 2, \ldots, n$, with $\mathbb{P}(G) = 0.7$ and $\mathbb{P}(B) = 0.3$ in this data set.

```
install.packages("caret"); library(caret); data(GermanCredit); head(GermanCredit)
ggplot(GermanCredit, aes(x= Class))+ geom_bar(aes(y= (..count..)/sum(..count..)),fill=c(
  "red","darkgreen"))+ labs(y= "prob.")+ theme_bw()
```

Listing 11.1: R code - Fetching credit data.

j	Age	ForeignWorker	Property.RealEstate	Housing.Own	CreditHistory	Class
$x_{\omega,j}$	$x_{\omega,1} = 67$	$x_{\omega,2} = 1$	$x_{\omega,3} = 0$	$x_{\omega,4} = 1$	$x_{\omega,5} = 0$	$c_\omega = 1$

```
import pandas as pd, matplotlib.pyplot as plt #
url= "https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data"
column_names= ['Status', 'Duration', 'CreditHistory', 'Purpose', 'CreditAmount', 'Savings',
  'Employment', 'InstallmentRate', 'PersonalStatus', 'OtherDebtors', 'Residence', 'Property',
  'Age', 'OtherPlans', 'Housing', 'NumCredits', 'Job', 'NumPeople', 'Telephone', 'Foreign', 'Class']
GermanCredit= pd.read_csv(url, delimiter=' ', names=column_names)
GermanCredit['Class']=GermanCredit['Class'].map({1:'Good',2:'Bad'});plt.figure(figsize=(8,6))
class_counts= GermanCredit['Class'].value_counts(normalize=True)
bars= plt.bar(class_counts.index, class_counts.values, color=['darkgreen','r'])
[plt.text(bar.get_x()+ bar.get_width()/2, bar.get_height()+ 0.01, f'{value:.1%}', ha='center',
  va='bottom') for bar, value in zip(bars, class_counts.values)]
plt.ylabel('Probability'); plt.grid(axis='y', alpha=.3); plt.tight_layout(); plt.show()
```

Listing 11.2: Python code - Fetching credit data.

Definition 11.7. *The credit scoring class ("good" or "bad") of applicant n°ω is denoted by $c_\omega \in \{0, 1\}$ depending on his status, i.e. $c_\omega = 1$ for "good" applicants and $c_\omega = 0$ for "bad" applicants.*

Linear regression

The score z_ω of a given credit applicant in row n^o ω is modeled as

$$z_\omega = F\left(\gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right), \qquad \omega = 1, 2, \ldots, n.$$

where $(\gamma_j)_{j=0,1,\ldots,m}$ is a family of linear coefficients, and $1 - z_\omega$ represents the probability that applicant n^o ω may default. In a linear regression model

we would take $F(z) := z$, hence the system of equations

$$c_\omega = \gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}, \qquad \omega = 1, 2, \ldots, n,$$

would be used to estimate the coefficients $(\gamma_j)_{j=0,1,\ldots,m}$.

Binomial logistic regression

The shortcoming of linear models is that $F(z_\omega)$, which is assumed to represent a probability value, may exit the interval $[0, 1]$. Logistic regression models address this issue by replacing $F(x) = x$ with the logistic CDF (or sigmoid function) F_L defined as

$$F_L(x) := \frac{\mathrm{e}^x}{1 + \mathrm{e}^x} = \frac{1}{1 + \mathrm{e}^{-x}}, \qquad x \in \mathbb{R}, \tag{11.8}$$

see also the Gaussian probability acceptance curve (11.5).

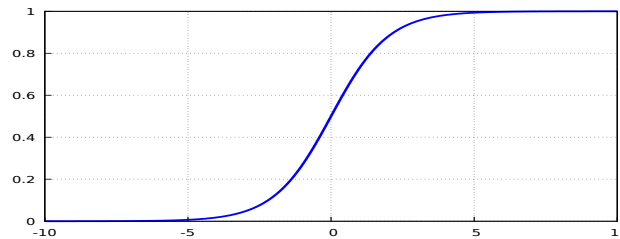

Fig. 11.5: Logistic CDF $x \overset{\times}{\mapsto} F_L(x) = \mathrm{e}^x/(1 + \mathrm{e}^x)$.

According to the Gaussian example considered in Proposition 11.3, we model the probability $p_\omega = \mathbb{P}(\omega \in G \mid X = z)$ that applicant ω is rated "good" given its score using the logistic CDF (11.8) as

$$\begin{aligned} p_\omega &= \mathbb{P}\left(\omega \in G \,\middle|\, X = \gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right) \\ &= F_L\left(\gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right) \end{aligned}$$

as in (11.5), $\omega = 1, \ldots, n$. Similarly, the probability that applicant ω is rated "bad" given its score is written as

$$\mathbb{P}\left(\omega \in B \,\middle|\, X = \gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right) = 1 - F_L\left(\gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right) = \overline{F}_L\left(\gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right).$$

The probability of sampling a given $\{0,1\}$-valued sequence $(c_\omega)_{1\leqslant\omega\leqslant n}$ of applicant classifications is

$$\prod_{\substack{1\leqslant\omega\leqslant n\\ c_\omega=0}} \mathbb{P}\left(\omega \in B \,\middle|\, X = \gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right) \prod_{\substack{1\leqslant\omega\leqslant n\\ c_\omega=1}} \mathbb{P}\left(\omega \in G \,\middle|\, X = \gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right)$$
$$= \prod_{\omega=1}^{n} \left(\mathbb{P}\left(\omega \in B \,\middle|\, X = \gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right)^{1-c_\omega} \mathbb{P}\left(\omega \in G \,\middle|\, X = \gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right)^{c_\omega}\right).$$

In order to estimate the sequence of coefficients $(\gamma_j)_{j=0,1,\ldots,m}$, we aim at maximizing the log-likelihood ratio

$$\log \prod_{\omega=1}^{n} \left(\mathbb{P}\left(\omega \in B \,\middle|\, X = \gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right)^{1-c_\omega} \mathbb{P}\left(\omega \in G \,\middle|\, X = \gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right)^{c_\omega}\right)$$
$$= \log \prod_{\omega=1}^{n} \left(\left(\overline{F}_L\left(\gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right)\right)^{1-c_\omega} \left(F_L\left(\gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right)\right)^{c_\omega}\right)$$
$$= \log \prod_{\omega=1}^{n} \left(\left(\frac{\mathrm{e}^{\gamma_0+\sum_{j=1}^{m}\gamma_j x_{\omega,j}}}{1+\mathrm{e}^{\gamma_0+\sum_{j=1}^{m}\gamma_j x_{\omega,j}}}\right)^{c_\omega} \left(\frac{1}{1+\mathrm{e}^{\gamma_0+\sum_{j=1}^{m}\gamma_j x_{\omega,j}}}\right)^{1-c_\omega}\right)$$
$$= \sum_{\omega=1}^{n} c_\omega \log \frac{\mathrm{e}^{\gamma_0+\sum_{j=1}^{m}\gamma_j x_{\omega,j}}}{1+\mathrm{e}^{\gamma_0+\sum_{j=1}^{m}\gamma_j x_{\omega,j}}} + \sum_{\omega=1}^{n} (1-c_\omega) \log \frac{1}{1+\mathrm{e}^{\gamma_0+\sum_{j=1}^{m}\gamma_j x_{\omega,j}}}$$

on the training set, over the sequence of parameters $(\gamma_0, \gamma_1, \ldots, \gamma_m)$. The default probabilities are then estimated on the testing set from

$$1 - p_\omega = \overline{F}_L\left(\gamma_0 + \sum_{j=1}^{m} \gamma_j x_{\omega,j}\right), \qquad \omega = 1, \ldots, n,$$

and the *logit*

$$\overline{F}_L^{-1}(p_\omega) = -\log\frac{p_\omega}{1-p_\omega} = \gamma_0 + \sum_{j=1}^{m}\gamma_j x_{\omega,j}, \qquad \omega = 1,\ldots,n,$$

or log-odds, represents probabilities on a logit scale. Two implementation examples of logistic regressions via Generalized Linear Models (GLM) are presented below using synthetic data with $n = 150$ and $m = 1$ for (γ_0, γ_1).

i) Logistic regression *vs.* logistic probabilities.

```
N=150; y= runif(N,-5,5); x= rbinom(N,1,prob=exp(y)/(1+exp(y))); y= data.frame(y)
glmreg= glm(x ~ y,data=y,family="binomial"); g0=as.numeric(glmreg$coefficients[1])
g1=as.numeric(glmreg$coefficients[2]); xx= seq(min(y),max(y),0.01); xx= data.frame(xx)
colnames(xx)= c('y'); pred= predict(glmreg,newdata=xx,type="response")
plot(y$y,x,col="blue"); lines(xx$y,pred,col="purple", lw=3); grid()
points(y$y,exp(g0+g1*y$y)/(1+exp(g0+g1*y$y)), lw=1)
points(y$y,exp(y$y)/(1+exp(y$y)),col="red",lw=2);legend("left",c("Data","GLM
 prediction","Est. probabilities","True probabilities"),col=c("blue","purple","black","red"),
 pch=c(1,NA,1,1), lty=c(NA,1,NA,NA), lwd=c(NA,3,1,2), pt.cex=1.2,
 bty="n");AIC(glmreg)
```

Listing 11.3: R code - GLM using logistic probabilities.

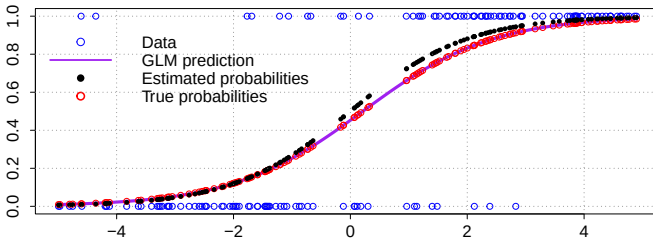

Fig. 11.6: GLM regression.

```
import numpy as np, pandas as pd; from sklearn.metrics import log_loss
from sklearn.linear_model import LogisticRegression; import matplotlib.pyplot as plt
N= 150; np.random.seed(42); y= np.random.uniform(-5, 5, N)
x= np.random.binomial(1, np.exp(y) / (1+ np.exp(y))); y_df= pd.DataFrame({'y': y})
glmreg= LogisticRegression(); glmreg.fit(y_df[['y']], x)
g0= glmreg.intercept_[0]; g1= glmreg.coef_[0][0]
xx= np.arange(y.min(), y.max(), 0.01); xx_df= pd.DataFrame({'y': xx})
pred=glmreg.predict_proba(xx_df[['y']])[:,1];m_gl=np.exp(g0+g1*y)/(1+np.exp(g0+g1*y))
true_pr= np.exp(y) / (1+ np.exp(y)); plt.figure(figsize=(10, 6))
plt.scatter(y, x, ec='b', alpha=.6, fc='none', label='Data')
plt.plot(xx, pred, c='m', lw=3, label='GLM prediction')
plt.scatter(y, m_gl, ec='k', s=30, fc='none', label='Est. probabilities')
plt.scatter(y, true_pr, ec='r', s=30, fc='none', label='True probabilities')
plt.xlabel('y'); plt.ylabel('x / Probability'); plt.legend(loc='upper left');
plt.grid(True, alpha=.3); plt.show(); n_par= 2;
log_l= -log_loss(x, glmreg.predict_proba(y_df[['y']]), normalize=False)
aic= 2 * n_par - 2 * log_l; print(f"AIC: {aic:.4f}")
```

Listing 11.4: Python code - GLM using logistic probabilities.

ii) Logistic regression *vs.* square probabilities.

```
N=150; y= runif(N,0,1); x= rbinom(N,1,prob=y*y); y= data.frame(y)
glmreg= glm(x ~ y,data=y,family="binomial")
g0=as.numeric(glmreg$coefficients[1]); g1=as.numeric(glmreg$coefficients[2])
xx= seq(min(y),max(y),0.01); xx= data.frame(xx); colnames(xx)= c('y')
pred= predict(glmreg,newdata=xx,type="response"); plot(y$y,x,col="blue")
lines(xx$y,pred,col="purple",lw=3);points(y$y,exp(g0+g1*y$y)/(1+exp(g0+g1*y$y)),lw=1)
points(y$y,((y$y-min(y$y))/(max(y$y)-min(y$y)))^2,col="red",lw=2); grid()
legend("left", legend= c("Data", "GLM prediction",
"Estimated probabilities", "True probabilities"), col= c("blue", "purple", "black", "red"),
pch=c(1,NA,1,1),lty=c(NA,1,NA,NA),lwd=c(NA,3,1,2),pt.cex=1.2,bty="n"); AIC(glmreg)
```

Listing 11.5: R code - GLM using square probabilities.

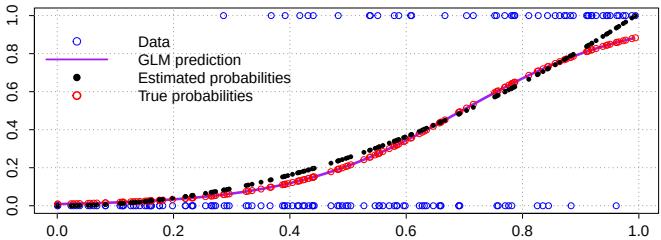

Fig. 11.7: GLM regression.

```
import numpy as np, pandas as pd; from sklearn.metrics import log_loss
from sklearn.linear_model import LogisticRegression; import matplotlib.pyplot as plt
N= 150; np.random.seed(42); y= np.random.uniform(0, 1, N);
x= np.random.binomial(1, y*y); y_df= pd.DataFrame({'y': y});
glmreg= LogisticRegression(); glmreg.fit(y_df[['y']], x); g0= glmreg.intercept_[0]
g1= glmreg.coef_[0][0]; xx= np.arange(y.min(), y.max(), 0.01)
xx_df= pd.DataFrame({'y': xx});pred= glmreg.predict_proba(xx_df[['y']])[:,1]
m_gl= np.exp(g0+ g1 * y) / (1+ np.exp(g0+ g1 * y))
y_norm= (y - y.min()) / (y.max() - y.min()); true_pr= y_norm ** 2
plt.figure(figsize=(10,6)); plt.scatter(y, x, ec='b', fc='none', alpha=.6, label='Data')
plt.plot(xx, pred, c='m', lw=3, label='GLM prediction')
plt.scatter(y, m_gl, ec='k', s=30, fc='none', label='Est. probabilities')
plt.scatter(y, true_pr, ec='r', s=30, fc='none', label='True probabilities')
plt.xlabel('y'); plt.ylabel('x / Probability');plt.grid(True, alpha=.3)
plt.legend(loc='upper left');plt.show();n_par=2;print(f"Intercept (g0): {g0:.4f}");
log_l=-log_loss(x,glmreg.predict_proba(y_df[['y']]),normalize=False)
aic=2*n_par-2*log_l;print(f"AIC: {aic:.4f}");print(f"Coefficient (g1): {g1:.4f}")
```

Listing 11.6: Python code - GLM using square probabilities.

Data partitioning

```
Train= createDataPartition(GermanCredit$Class, p=.6, list=FALSE)
training= GermanCredit[ Train, ];testing= GermanCredit[ -Train, ]; head(testing$Class);
testing= cbind(rownames(testing), testing); colnames(testing)[1]= "ID"
```

Listing 11.7: R code - Partitioning data into training and testing sets.

```
from sklearn.model_selection import train_test_split
Train, testing= train_test_split(GermanCredit, test_size=.4, random_state=42)
training= GermanCredit.loc[Train.index]
```

Listing 11.8: Python code - Partitioning data into training and testing sets.

The data is randomly split into a training set and a testing set. The training data set is used to fit the parameters of a generalized linear model, and the testing set is then used to estimate the corresponding default probabilities.

```
mod.glm= glm(Class ~ Age+ ForeignWorker+ Property.RealEstate+ Housing.Own+
  CreditHistory.Critical, data=training, family="binomial")
head(predict(mod.glm, newdata=testing, type="response"))
testing$Score5= predict(mod.glm, newdata=testing, type="response")
tsmp= head(testing,100);colnames(tsmp)= make.unique(names(tsmp))
ggplot(tsmp, aes(x=ID, y=Score5, fill=Class))+ geom_bar(stat="identity")+
 scale_fill_manual(values=c( "red","darkgreen"))+ scale_y_continuous(limits=c(0,1),expand=
 c(0,0))+ theme_bw(base_size= 18)+ xlab(NULL)+ theme(axis.text.x=
 element_blank(),aspect.ratio=.5)+ geom_hline(yintercept= 0.75,lwd=1.6)
```

Listing 11.9: R code - Logistic regression on five criteria.

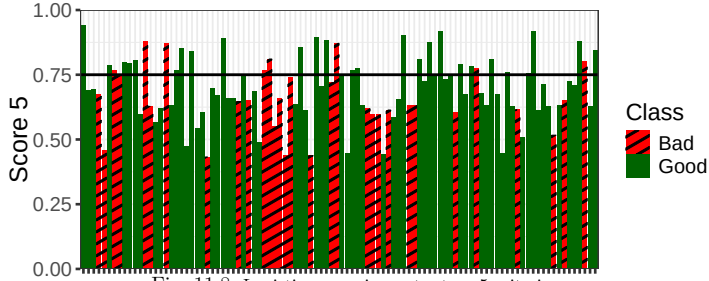

Fig. 11.8: Logistic regression output on 5 criteria.

This version: April 28, 2026
https://personal.ntu.edu.sg/nprivault/indext.html

```
import pandas as pd, matplotlib.pyplot as plt;from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
feature_cols= ['Age', 'Foreign', 'Property', 'Housing', 'CreditHistory']
X_tr= training[feature_cols]; y_tr= training['Class']
X_test= testing[feature_cols]; le= LabelEncoder()
[(X_tr.__setitem__(col, le.fit_transform(X_tr[col])), X_test.__setitem__(col,
  le.transform(X_test[col]))) for col in ['Foreign','Property','Housing','CreditHistory']]
m_g=LogisticRegression();m_g.fit(X_tr,y_tr);testing['Score5']=m_g.predict_proba(X_test)[:,1]
tsmp=testing.head(100).reset_index();plt.figure(figsize=(16,12));plt.tight_layout();plt.xticks([])
plt.bar(tsmp.index,tsmp['Score5'],color=tsmp['Class'].map({'Good':'darkgreen','Bad':'r'}))
plt.axhline(y=.75,c='k',lw=1.6);plt.ylim(0,1);plt.xlabel('');plt.ylabel('Score5');plt.show()
```

Listing 11.10: Python code - Logistic regression on five criteria.

Based on the above 100 samples, the next code identifies the True Positive Rate (TPR) $= 28/69 = 40.58\%$, and the False Positive Rate (FPR) $= 8/31 = 25.81\%$, at the threshold $p = 0.75$.

```
cat('TPR5=',length(tsmp$Score5[tsmp$Score5>0.75 &
  tsmp$Class=="Good"])/length(tsmp$Score5[tsmp$Class=="Good"]),'\n')
cat('FPR5=',length(tsmp$Score5[tsmp$Score5>0.75 &
  tsmp$Class=="Bad"])/length(tsmp$Score5[tsmp$Class=="Bad"]),'\n')
pred= ifelse(tsmp$Score5 > 0.75, "Good", "Bad");
  confusionMatrix(factor(noquote(pred)),factor(tsmp$Class))
```

Listing 11.11: R code - TPR and FPR estimation.

```
n_good= len(tsmp[tsmp['Class'] == 'Good']); n_bad= len(tsmp[tsmp['Class'] == 'Bad'])
tpr5=len(tsmp[(tsmp['Score5']>.75) & (tsmp['Class'] == 'Good')])/n_good
fpr5=len(tsmp[(tsmp['Score5']>.75) & (tsmp['Class'] == 'Bad')])/n_bad
print(f'TPR5= {tpr5:.4f}'); print(f'FPR5= {fpr5:.4f}')
from sklearn.metrics import confusion_matrix, classification_report
pred= ['Good' if score > 0.75 else 'Bad' for score in tsmp['Score5']]
print(confusion_matrix(tsmp['Class'],pred));print(classification_report(tsmp['Class'],pred))
```

Listing 11.12: Python code - TPR and FPR estimation.

```
mod.glm= glm(Class ~ Duration+ Amount+ InstallmentRatePercentage+ ResidenceDuration+
 Age+ NumberExistingCredits+ NumberPeopleMaintenance+ Telephone+ ForeignWorker+
 CheckingAccountStatus.lt.0+ CheckingAccountStatus.0.to.200+
 CheckingAccountStatus.gt.200+ CheckingAccountStatus.none+
 CreditHistory.NoCredit.AllPaid+ CreditHistory.ThisBank.AllPaid+ CreditHistory.PaidDuly+
 CreditHistory.Delay+ CreditHistory.Critical+ Purpose.NewCar+ Purpose.UsedCar+
 Purpose.Furniture.Equipment+ Purpose.Radio.Television+ Purpose.DomesticAppliance+
 Purpose.Repairs+ Purpose.Education+ Purpose.Vacation+ Purpose.Retraining+
 Purpose.Business+ Purpose.Other+ SavingsAccountBonds.lt.100+
 SavingsAccountBonds.100.to.500+ SavingsAccountBonds.500.to.1000+
 SavingsAccountBonds.gt.1000+ SavingsAccountBonds.Unknown+ EmploymentDuration.lt.1+
 EmploymentDuration.1.to.4+ EmploymentDuration.4.to.7+ EmploymentDuration.gt.7+
 EmploymentDuration.Unemployed+ Personal.Male.Divorced.Seperated+
 Personal.Female.NotSingle+ Personal.Male.Single+ Personal.Male.Married.Widowed+
 Personal.Female.Single+ OtherDebtorsGuarantors.None+
 OtherDebtorsGuarantors.CoApplicant+ OtherDebtorsGuarantors.Guarantor+
 Property.RealEstate+ Property.Insurance+ Property.CarOther+ Property.Unknown+
 OtherInstallmentPlans.Bank+ OtherInstallmentPlans.Stores+ OtherInstallmentPlans.None+
 Housing.Rent+ Housing.Own+ Housing.ForFree+ Job.UnemployedUnskilled+
 Job.UnskilledResident+ Job.SkilledEmployee+ Job.Management.SelfEmp.HighlyQualified,
 data=training, family="binomial")
testing$Score61= predict(mod.glm, newdata=testing, type="response")
tsmp= head(testing,100);colnames(tsmp)= make.unique(names(tsmp))
ggplot(tsmp, aes(x=ID, y=Score61, fill=Class))+ geom_bar(stat="identity")+
 scale_fill_manual(values=c( "red","darkgreen"))+ scale_y_continuous(limits=c(0,1),expand=
 c(0,0))+ theme_bw(base_size= 18)+ xlab(NULL)+ theme(axis.text.x=
 element_blank(),aspect.ratio=.5)+ geom_hline(yintercept= 0.75,lwd=1.6)
```

Listing 11.13: R code - Logistic regression on 61 criteria.

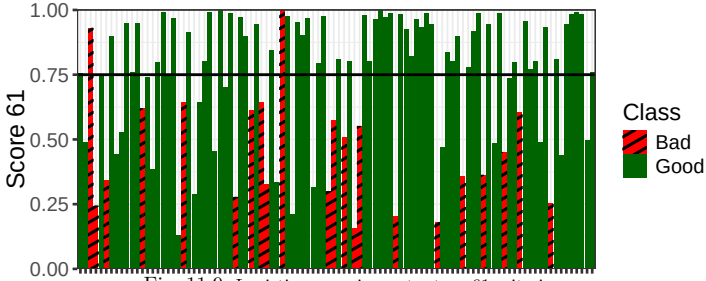

Fig. 11.9: Logistic regression output on 61 criteria.

```
import pandas as pd,matplotlib.pyplot as plt;from sklearn.preprocessing import LabelEncoder;
from sklearn.linear_model import LogisticRegression;
feature_cols= [col for col in training.columns if col != 'Class'];
X_tr= training[feature_cols].copy(); y_tr= training['Class'];
X_test= testing[feature_cols].copy(); le= LabelEncoder();
c_cols= X_tr.select_dtypes(include=['object']).columns;
([(X_tr.___setitem___(col, le.fit_transform(X_tr[col].astype(str))),
X_test.___setitem___(col, le.transform(X_test[col].astype(str)))) for col in c_cols])
m_g=LogisticRegression(max_iter=1000);m_g.fit(X_tr,y_tr);
testing['Score61']=m_g.predict_proba(X_test)[:,1];tsmp=testing.head(100).reset_index();
tsmp['ID']= tsmp.index; plt.figure(figsize=(16, 12));plt.xlabel(''); plt.ylabel('Score61');
plt.bar(tsmp['ID'], tsmp['Score61'], color=tsmp['Class'].map({'Good': 'darkgreen', 'Bad': 'r'}))
plt.axhline(y=.75, c='k', lw=1.6); plt.ylim(0, 1);plt.xticks([]); plt.tight_layout(); plt.show()
```

Listing 11.14: Python code - Logistic regression on 61 criteria.

This version: April 28, 2026
https://personal.ntu.edu.sg/nprivault/indext.html

In comparison with Figure 11.8, the 100 samples in Figure 11.9 above yield a higher True Positive Rate (TPR) $= 57/78 = 73.08\%$ and a lower False Positive Rate (FPR) $= 2/22 = 9.09\%$ at the level $p = 0.75$. In other words, the count of true positive samples has increased from 28 to 57, and the count of false positive samples has decreased from 8 to 2.

```
cat('TPR61=',length(tsmp$Score61[tsmp$Score61>0.75 &
  tsmp$Class=="Good"])/length(tsmp$Score61[tsmp$Class=="Good"]),'\n')
cat('FPR61=',length(tsmp$Score61[tsmp$Score61>0.75 &
  tsmp$Class=="Bad"])/length(tsmp$Score61[tsmp$Class=="Bad"]),'\n')
pred= ifelse(tsmp$Score61 > 0.75, "Good", "Bad")
confusionMatrix(factor(noquote(pred)),factor(tsmp$Class))
```

Listing 11.15: R code - TPR and FPR estimations.

```
n_good= len(tsmp[tsmp['Class'] == 'Good']); n_bad= len(tsmp[tsmp['Class'] == 'Bad'])
tpr61= len(tsmp[(tsmp['Score61'] > 0.75) & (tsmp['Class'] == 'Good')])/n_good
fpr61= len(tsmp[(tsmp['Score61'] > 0.75) & (tsmp['Class'] == 'Bad')])/n_bad
print(f'TPR61= {tpr61:.4f}'); print(f'FPR61= {fpr61:.4f}')
from sklearn.metrics import confusion_matrix, classification_report
pred= ['Good' if score > 0.75 else 'Bad' for score in tsmp['Score61']]
print(confusion_matrix(tsmp['Class'],pred));print(classification_report(tsmp['Class'],pred))
```

Listing 11.16: Python code - TPR and FPR estimations.

The above TPR and FPR can be plotted over several threshold values, to yield a sample of ROC curve considered in the next section.

```
levels=seq(0,1,length.out=10);TPR=numeric(10);FPR=numeric(10);dev.new(width=16,height=7)
par(mgp=c(2.5,.8,0)); for (i in seq_along(levels)) {l= levels[i]
  TPR[i]= sum(tsmp$Score61 > l & tsmp$Class == "Good") / sum(tsmp$Class == "Good")
  FPR[i]= sum(tsmp$Score61 > l & tsmp$Class == "Bad") / sum(tsmp$Class == "Bad")}
data.frame(Threshold = levels, TPR, FPR)
plot(FPR, TPR, type= "b", pch= 16, col= "blue", lwd=4, xaxs='i', yaxs='i', xlab= "FPR",
  ylab= "TPR", cex.lab=2, cex.axis=1.5, main= ""); abline(0, 1, lty= 2, lwd = 3, col= "black");
  grid(col='black',lwd=2)
```

Listing 11.17: R code - ROC curve with TPR and FPR estimates.

```
import numpy as np, pandas as pd, matplotlib.pyplot as plt
lvls= np.linspace(0, 1, 10); good_m= tsmp['Class'] == 'Good';
bad_m= tsmp['Class'] == 'Bad'; n_good= good_m.sum(); n_bad= bad_m.sum()
sc= tsmp['Score61'].values.reshape(-1, 1)
TPR=((sc>lvls).reshape(-1,len(lvls)) & good_m.values.reshape(-1,1)).sum(axis=0)/n_good
FPR=((sc>lvls).reshape(-1,len(lvls)) & bad_m.values.reshape(-1,1)).sum(axis=0)/n_bad
roc_df= pd.DataFrame({'Threshold': lvls,'TPR': TPR,'FPR': FPR}); print(roc_df);
plt.figure(figsize=(8, 8));plt.plot(FPR,TPR,'b-o',ms=8);plt.plot([0,1],[0,1],'k--',alpha=.5);
plt.xlabel('False Positive Rate (FPR)'); plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve (10 thresholds)');plt.grid(True,alpha=.3);plt.tight_layout(); plt.show()
```

Listing 11.18: Python code - ROC curve with TPR and FPR estimates.

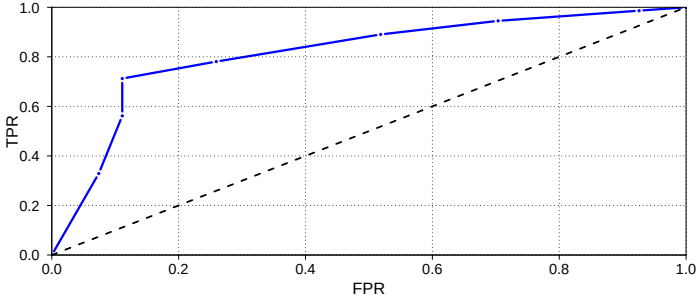

Fig. 11.10: ROC curve.

11.4 ROC Curve

The ROC curve is a plot of the True Positive Rate values

$$x \longmapsto \overline{F}_G(x)$$

against the False Positive Rate function

$$x \longmapsto \overline{F}_B(x).$$

Definition 11.8. *The Receiver Operating Characteristic (ROC) curve is the function of the threshold $p \in [0,1]$ defined as*

$$\begin{aligned} [0,1] &\longrightarrow [0,1] \\ p &\longmapsto \mathrm{ROC}(p) := \overline{F}_G\big(\overline{F}_B^{-1}(p)\big), \end{aligned}$$

where $\overline{F}_B^{-1}$ denotes the inverse of the tail distribution function $\overline{F}_B$.

The construction of Definition 11.8 proceeds in two steps.

a) Starting from a given FPR level p, compute the associated threshold x^* as $x^* := \overline{F}_B^{-1}(p)$, *i.e.*

$$p = \overline{F}_B(x^*) = \mathbb{P}(X > x^* \mid B).$$

b) From the threshold level, x^* estimate the corresponding TPR as

$$\mathbb{P}(X > x^* \mid G) = \overline{F}_G(x^*) = \overline{F}_G\big(\overline{F}_B^{-1}(p)\big) = \mathrm{ROC}(p).$$

Proposition 11.9. *The ROC function can be rewritten as the integral*

This version: April 28, 2026
https://personal.ntu.edu.sg/nprivault/indext.html

$$\mathrm{ROC}(p) = \int_0^p \lambda\big(\overline{F}_B^{-1}(q)\big)dq, \qquad 0 \leqslant p \leqslant 1,$$

where the likelihood ratio $\lambda(x)$ *is given by*

$$\lambda(x) = \frac{f_X(x \mid G)}{f_X(x \mid B)}, \qquad x \in \mathbb{R}.$$

Proof. The slope of the ROC curve at the point $p \in [0,1]$ is given by

$$\begin{aligned}
\frac{d}{dp}\overline{F}_G\big(\overline{F}_B^{-1}(p)\big) &= \overline{F}'_G\big(\overline{F}_B^{-1}(p)\big)\frac{d}{dp}\overline{F}_B^{-1}(p) \\
&= \frac{\overline{F}'_G\big(\overline{F}_B^{-1}(p)\big)}{\overline{F}'_B\big(\overline{F}_B^{-1}(p)\big)} \\
&= \frac{f_X\big(\overline{F}_B^{-1}(p) \mid G\big)}{f_X\big(\overline{F}_B^{-1}(p) \mid B\big)} \\
&= \lambda\big(\overline{F}_B^{-1}(p)\big),
\end{aligned}$$

hence we have

$$\overline{F}_G\big(\overline{F}_B^{-1}(p)\big) = \int_0^p \lambda\big(\overline{F}_B^{-1}(q)\big)dq, \qquad 0 \leqslant p \leqslant 1.$$

□

When X is Gaussian distributed given $\{G, B\}$ with the conditional densities

$$f_X(x \mid G) = \frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-(x-\mu_G)^2/(2\sigma^2)}$$

and

$$f_X(x \mid B) = \frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-(x-\mu_B)^2/(2\sigma^2)},$$

the likelihood ration is given by

$$\lambda(x) = \frac{f_X(x \mid G)}{f_X(x \mid B)} = \mathrm{e}^{\gamma_1 x - (\mu_G^2 - \mu_B^2)/(2\sigma^2)}, \qquad x \in \mathbb{R},$$

with $\gamma_1 := (\mu_G - \mu_B)/\sigma^2$.

Based on Definition 11.8, the ROC curve in this Gaussian setting can be computed using the following code.

```
FG= function(x, mug, sigma) {1 - pnorm((x - mug) / sigma)}
FBINV= function(x, mub, sigma) {mub+ sigma * qnorm(1 - x)}
ROC= function(x, mub, mug, sigma) {sapply(x, function(x_val) ifelse(x_val == 0, 0,
  FG(FBINV(x_val, mub, sigma), mug, sigma)))}
```

Listing 11.19: R code - Gaussian ROC curve.

```
import numpy as np; from scipy.stats import norm;
FG = lambda x, mug, sigma: 1 - norm.cdf((x - mug)/sigma);
FBINV = lambda x, mub, sigma: mub + sigma * norm.ppf(1 - x);
ROC = lambda x, mub, mug, sigma: np.array([0 if x__val==0 else FG(FBINV(x__val, mub,
  sigma), mug, sigma) for x__val in (x if hasattr(x, '___iter___') else [x])])
```

Listing 11.20: Python code - Gaussian ROC curve.

Figure 11.11 presents three examples of ROC curves in the Gaussian example with successively $(\mu_B, \mu_G) = (1, 4)$, $(\mu_B, \mu_G) = (1, 2)$, and $(\mu_B, \mu_G) = (1, 1)$.

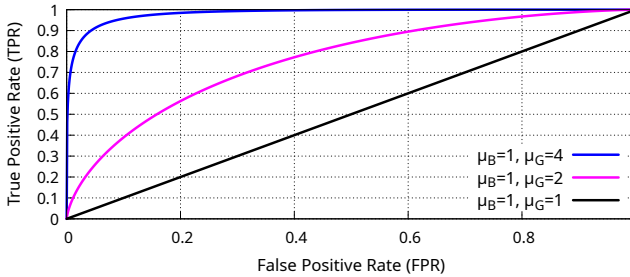

Fig. 11.11: Gaussian ROC curves.

We check that the classification is better when $\mu_B << \mu_G$. The ROC Curve shows the performance of the classification procedure, which is quantified by the Area Under The Curve (AUC). A perfect classification would correspond to a single point with coordinates $(0, 1)$ and AUC equal to 1, which corresponds to FPR=0% false negatives and TPR=100% true positives. The closer the AUC is to 1, the better the classification is performing.

On the other hand, a completely random guess would correspond to a point on the diagonal line. Points above the diagonal represent good classification results (better than random), while points below the line represent poor results (worse than random) (Wikipedia).

```
TPR5= length(testing$Score5[testing$Score5>0.75 &
  testing$Class=="Good"])/length(testing$Score5[testing$Class=="Good"])
FPR5= length(testing$Score5[testing$Score5>0.75 &
  testing$Class=="Bad"])/length(testing$Score5[testing$Class=="Bad"])
cat('TPR5=',TPR5,'\n'); cat('FPR5=',FPR5,'\n')
TPR61= length(testing$Score61[testing$Score61>0.75 &
  testing$Class=="Good"])/length(testing$Score61[testing$Class=="Good"])
FPR61= length(testing$Score61[testing$Score61>0.75 &
  testing$Class=="Bad"])/length(testing$Score61[testing$Class=="Bad"])
cat('TPR61=',TPR61,'\n'); cat('FPR61=',FPR61,'\n')
```

Listing 11.21: R code - ROC curve with TPR and FPR estimates.

```
TPR5= len(testing[(testing['Score5'] > 0.75)
& (testing['Class'] == 'Good')])/ len(testing[testing['Class'] == 'Good'])
FPR5= len(testing[(testing['Score5'] > 0.75)
& (testing['Class'] == 'Bad')])/ len(testing[testing['Class'] == 'Bad'])
print(f'TPR5= {TPR5:.4f}'); print(f'FPR5= {FPR5:.4f}')
TPR61= len(testing[(testing['Score61'] > 0.75) &
(testing['Class'] == 'Good')])/ len(testing[testing['Class'] == 'Good'])
FPR61= len(testing[(testing['Score61'] > 0.75) &
(testing['Class'] == 'Bad')])/ len(testing[testing['Class'] == 'Bad'])
print(f'TPR61= {TPR61:.4f}'); print(f'FPR61= {FPR61:.4f}')
```

Listing 11.22: Python code - Computation of TPR and FPR estimates.

The True Positive Rates (TPR) and False Positive Rates (FPR) based on 5 and 61 criteria at the level $p = 0.75$ are recomputed in the above code on the whole 400 data samples, and plotted on the ROC graphs of Figure 11.12.

```
install.packages("ROCR"); library(ROCR)
pred5= prediction(as.numeric(testing$Score5),as.numeric(testing$Class))
perf5= performance(pred5,"tpr","fpr"); dev.new(width=16,height=7); par(mar= c(4.5,4.5,2,2))
plot(perf5,col="purple",lwd=3, xaxs= "i", yaxs= "i",cex.lab=2,las=1)
segments(FPR5,0,FPR5,TPR5,col="purple",lwd=2);segments(0,TPR5,FPR5,TPR5,col="purple",
 lwd=2)
pred61= prediction(as.numeric(testing$Score61),as.numeric(testing$Class))
perf61= performance(pred61,"tpr","fpr"); par(new=TRUE)
plot(perf61,col="blue",lwd=3,main="", ann=FALSE, xaxs="i", yaxs="i")
legend("bottomright",legend=c("61 criteria","5 criteria"),col=c("blue","purple"),lwd=3,cex=3)
segments(0,0,1,1, lwd=3);segments(FPR61,0,FPR61,TPR61, col="blue", lwd=2);
segments(0,TPR61,FPR61,TPR61, col="blue", lwd=2)
```

Listing 11.23: R code - Computation of TPR and FPR estimates.

```
from sklearn.metrics import roc_curve, auc; import matplotlib.pyplot as plt
fpr5, tpr5, _ = roc_curve(testing['Class'], testing['Score5'], pos_label='Good')
fpr61, tpr61, _ = roc_curve(testing['Class'], testing['Score61'], pos_label='Good')
plt.figure(figsize=(16, 7)); plt.plot(fpr5, tpr5, c='m', lw=3, label='5 criteria')
plt.plot(fpr61, tpr61, c='b', lw=3, label='61 criteria')
plt.plot([FPR5,FPR5],[0,TPR5],c='m',lw=2);plt.plot([0,FPR5],[TPR5,TPR5],c='m',lw=2)
plt.plot([FPR61,FPR61],[0,TPR61],c='b',lw=2);plt.plot([0,FPR61],[TPR61,TPR61],c='b',lw=2)
plt.plot([0, 1], [0, 1], c='k', lw=3, ls='--');plt.xlabel('False Positive Rate', fontsize=14)
plt.ylabel('True Positive Rate', fontsize=14); plt.legend(loc='lower right', fontsize=12)
plt.grid(True, alpha=.3); plt.xlim(0, 1); plt.ylim(0, 1); plt.tight_layout(pad=0); plt.show()
```

Listing 11.24: Python code - ROC curve with TPR and FPR estimates.

The ROC graphs in Figure 11.12 confirm the improvement in classification reached when switching from 5 to 61 criteria.

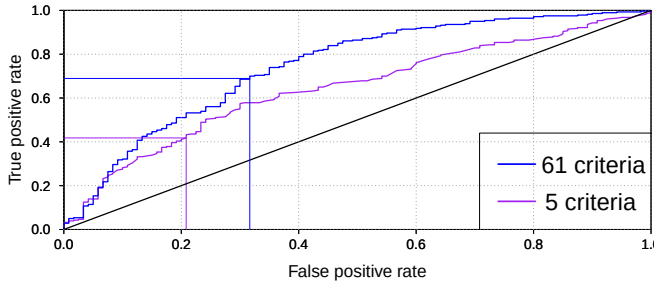

Fig. 11.12: ROC curves based on 5 criteria and 61 criteria.

Using a neural network in R

Note: Restarting R is *needed* before running the following code.

```
install.packages("neuralnet");
library(neuralnet);library(caret);data(GermanCredit);library(ROCR);
Train= createDataPartition(GermanCredit$Class, p=.6, list=FALSE)
training= GermanCredit[ Train, ];testing= GermanCredit[ -Train, ]
nn= neuralnet(Class ~ Age+ ForeignWorker+ Property.RealEstate+ Housing.Own+
  CreditHistory.Critical, data=training, hidden=c(3,1), linear.output=FALSE, threshold=.05)
nn$result.matrix; plot(nn,col.intercept = "blue")
temp_test= subset(testing, select = c("Age", "ForeignWorker", "Property.RealEstate",
  "Housing.Own", "CreditHistory.Critical"))
head(temp_test); nn.results= compute(nn, temp_test)
results= data.frame(actual = testing$Class, prediction = nn.results$net.result)
head(results); pred= ifelse(results[,3] > 0.75, "Good", "Bad")
confusionMatrix(factor(results$actual),factor(noquote(pred)))
pred2= prediction(as.numeric(results[,3]),as.numeric(results$actual))
perf= performance(pred2,"tpr","fpr"); plot(perf,col="purple",lwd=3, xaxs = "i", yaxs = "i")
```

Listing 11.25: R code - Neural network estimation.

Improved performance may be achieved by rescaling the binary variables to $\{-1, 1\}$.

```
training$ForeignWorker= ifelse(training$ForeignWorker == 0, -1, 1)
training$Property.RealEstate= ifelse(training$Property.RealEstate  == 0, -1, 1)
training$Housing.Own= ifelse(training$Housing.Own == 0, -1, 1)
training$CreditHistory.Critical= ifelse(training$CreditHistory.Critical  == 0, -1, 1)
nn2= neuralnet(Class ~ Age+ ForeignWorker+ Property.RealEstate+ Housing.Own+
  CreditHistory.Critical, data=training, hidden=c(3,1), linear.output=F, threshold=.05)
nn2.results= compute(nn2, temp_test)
results2= data.frame(actual = testing$Class, prediction = nn2.results$net.result)
pred3= ifelse(results2[,3] > 0.75, "Good", "Bad")
confusionMatrix(factor(results2$actual),factor(noquote(pred3)));dev.new(width=16,height=7)
pred4= prediction(as.numeric(results2[,3]),as.numeric(results$actual));par(mar= c(4.5,4.5,2,2))
plot(perf,col="purple",lwd=3, xaxs= "i", yaxs= "i",cex.lab=2,las=1)
perf3= performance(pred4,"tpr","fpr"); par(new=TRUE)
plot(perf3,col="blue",lwd=3,main="", ann=F, xaxs="i", yaxs="i")
legend("bottomright",c("Rescaled","Non rescaled"),col=c("blue","purple"),lwd=3,cex=3)
```

Listing 11.26: R code - Neural network estimation with rescaling.

Using a neural network in Python

Download the corresponding **IPython notebook*** that can be run **here** using this **data file**.

Using random forests in Python

Download the corresponding **IPython notebook** that can be run **here** using this **data file**.

Using XGBoost in R

Download the corresponding **R code**.

Support Vector Machines using scikit-learn in Python

Download the corresponding **IPython notebook** that can be run **here**.

Exercises

Exercise 11.1 Consider a set Ω of applicants decomposed into the partition $\Omega = G \cup B$, where each applicant ω is assigned a score $X(\omega)$ which is exponentially distributed given $\{G, B\}$ with the conditional densities

$$f_X(x \mid G) = \lambda_G \mathrm{e}^{-\lambda_G x} \mathbb{1}_{[0,\infty)}(x)$$

and

* Right-click to save as attachment (may not work on).

$$f_X(x \mid B) = \lambda_B \mathrm{e}^{-\lambda_B x} \mathbb{1}_{[0,\infty)}(x),$$

where $\lambda_B > \lambda_G > 0$.

a) Compute the conditional expected values $\mathbb{E}[X \mid G]$ and $\mathbb{E}[X \mid B]$.
b) Compute the *probability default curve*

$$x \longmapsto \mathbb{P}(B \mid X = x) = \frac{\mathbb{P}(B) f_X(x \mid B)}{\mathbb{P}(G) f_X(x \mid G) + \mathbb{P}(B) f_X(x \mid B)}$$

in terms of the likelihood ratio $\lambda(x) := f_G(x)/f_B(x)$, $x \in \mathbb{R}$.
c) Determine the acceptance set

$$\mathscr{A} := \big\{ x \in \mathbb{R} \ : \ D\mathbb{P}(B \mid X = x) \leqslant L\mathbb{P}(G \mid X = x) \big\},$$

where

- L represents the loss incurred by the rejection of an applicant, and
- D represents the loss incurred by the default of an applicant.

Exercise 11.2 Consider a set Ω of applicants decomposed into the partition $\Omega = G \cup B$, where each applicant ω is assigned a uniformly distributed score $X(\omega)$ given $\{G, B\}$, with the conditional densities

$$f_X(x \mid G) = \frac{1}{\lambda_G} \mathbb{1}_{[0,\lambda_G]}(x) dx$$

and

$$f_X(x \mid B) = \frac{1}{\lambda_B} \mathbb{1}_{[0,\lambda_B]}(x) dx,$$

where $0 < \lambda_B < \lambda_G$.

a) Compute the *probability default curve*

$$x \longmapsto \mathbb{P}(B \mid X = x) = \frac{\mathbb{P}(B) f_X(x \mid B)}{\mathbb{P}(G) f_X(x \mid G) + \mathbb{P}(B) f_X(x \mid B)}.$$

b) Compute the conditional expected values $\mathbb{E}[X \mid G]$ and $\mathbb{E}[X \mid B]$.
c) Determine the acceptance set

$$A := \big\{ x \in \mathbb{R}_+ \ : \ D\mathbb{P}(B \mid X = x) \leqslant L\mathbb{P}(G \mid X = x) \big\},$$

where

- L represents the missed earnings incurred by the rejection of applicant,

This version: April 28, 2026
https://personal.ntu.edu.sg/nprivault/indext.html

- D represents the loss incurred by the default of an applicant.

Exercise 11.3

a) Compute the ROC function $x \mapsto \overline{F}_G\big(\overline{F}_B^{-1}(x)\big)$ of Exercise 11.1, $x \in [0,1]$, and draw its graph for sample values of λ_G, λ_B.
b) Compute the ROC function $x \mapsto \overline{F}_G\big(\overline{F}_B^{-1}(x)\big)$ of Exercise 11.2, $x \in [0,1]$, and draw its graph for sample values of λ_G, λ_B.

Exercise 11.4 Consider a set Ω of customers decomposed as the partition $\Omega = G \cup B$, where each customer ω is assigned a uniformly distributed score $X(\omega)$ given $\{G, B\}$, with the conditional densities

$$f_X(x \mid G) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-(x-\mu_G)^2/(2\sigma^2)}$$

and

$$f_X(x \mid B) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-(x-\mu_B)^2/(2\sigma^2)},$$

with $\mu_B < \mu_G$.

a) Compute the *probability default curve*

$$x \longmapsto \mathbb{P}(B \mid X = x) := \frac{\mathbb{P}(B) f_X(x \mid B)}{\mathbb{P}(G) f_X(x \mid G) + \mathbb{P}(B) f_X(x \mid B)}.$$

b) Compute the likelihood ratio $\lambda(x)$ defined as $\lambda(x) := f_X(x \mid G)/f_X(x \mid B)$, $x \in \mathbb{R}$.
c) Letting

- $L(x) := \mathrm{e}^{-ax}$ represent the missed earnings incurred by rejecting an applicant with score $x \in \mathbb{R}$, with $a > 0$,
- $D(x) := \mathrm{e}^{bx}$ represent the loss incurred by the default of an applicant with score $x \in \mathbb{R}$, with $b > 0$,

determine the acceptance set

$$\mathcal{A} := \big\{x \in \mathbb{R}_+ \ : \ D(x)\mathbb{P}(B \mid X = x) \leqslant L(x)\mathbb{P}(G \mid X = x)\big\}.$$