Credit Risk in Banking

Marek Teller, Václav Kozmík

January 16, 2022

Acknowledgments

We would like to thank Pavel Charamza, who prepared first notes for this lecture and covered most of the topics. Moreover, we would also like to thank Miloš Kopa for corrections and extension of the text and Michal Rychnovský for providing the section about scoring models.

Contents

Chapter 1

Logistic regression

Logistic regression belongs to the family of generalized linear models (GLM). It is often used for models with binary dependent variable $Y(x)$. We will start with quick revision of linear regression model, followed by introduction of it's generalization to GLM and finally we will discuss logistic regression as the most common method used for retail credit risk assessment during last decades. It is fair to say, that nowadays machine learning (ML) algorithms begin to slowly replace logistic regression. An example of ML algorithm will be introduced during the course Credit Risk in Banking.

1.1 Linear regression

Denote $\boldsymbol{x}_i,\ i = 1, ..., n$ vectors of the k-dimensional Euclidean space $\mathbb{R}^k$. We assume that in these points we make observations, which are random variables $Y_i,\ i = 1, ..., n$ with Gaussian distribution, i.e. $Y_i \sim \mathcal{N}(\mu_i, \sigma^2)$.

Linear regression model assumes linear relationship between dependent variable Y_i and independent explanatory variables $\boldsymbol{x}_i$ and is defined by equation:

$$Y_i = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} + e_i \tag{1.1}$$

where e_i represents random errors. Usually ordinary least square method (OLS) is used to estimate coefficients of the model. In order to ensure that OLS method gives us the best linear unbiased estimates (BLUE), Gauss-Markov assumptions must be met:

1. $\mathbb{E}[e_i] = 0$
2. $\mathrm{Var(e_i)} = \mathbb{E}[\mathrm{e_i^2}] = \sigma^2 = \text{constant}$

3. $\mathrm{Cov}(\mathrm{e_i}, \mathrm{e_j}) = \mathbb{E}[\mathrm{e_i e_j}] = 0, \mathrm{i} \neq \mathrm{j}$
4. $\mathrm{Cov}(\boldsymbol{x}_\mathrm{i}, \mathrm{e_i}) = 0$
5. $\mathrm{rank}(\boldsymbol{x}) = \mathrm{k}$

Using Gauss-Markov assumptions we can apply expected value on (1.1) to obtain:

$$\mathbb{E}[Y_i] = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} \tag{1.2}$$

1.2 Generalized linear model

Linear regression models can be used if $Y_i \sim \mathcal{N}(\mu_i, \sigma^2)$. But what if our dependent variable is not drawn from normal distribution. Then we can use generalized linear model (GLM) instead:

$$g\left(\mathbb{E}[Y_i]\right) = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} \tag{1.3}$$

Function $g(x)$ is called the link function. Generalization of linear model brought us following additional features:

- We don't need Y_i to be normally distributed random variable. In fact, Y_i can have any distribution from exponential family (will be defined later). Discrete distributions, such as alternative distribution, belong to this family.
- By using GLM we can model not only $\mathbb{E}[Y_i]$, but also non-linear transformation of $\mathbb{E}[Y_i]$.

1.2.1 Exponential family distributions

Definition 1. *Let $\boldsymbol{y} \subset \mathbb{R}^d$ and $\boldsymbol{\theta} \subset \mathbb{R}^m$. We say that probability density function $f(\boldsymbol{y}; \boldsymbol{\theta})$ belongs to the exponential family if:*

$$f(\boldsymbol{y}; \boldsymbol{\theta}) = s(\boldsymbol{y}) \cdot r(\boldsymbol{\theta}) \cdot \exp\left(\sum_{j=1}^{m} t_j(\boldsymbol{y}) q_j(\boldsymbol{\theta})\right) \tag{1.4}$$

where $s(\boldsymbol{y})$, $r(\boldsymbol{\theta})$, $t_j(\boldsymbol{y})$ and $q_j(\boldsymbol{\theta})$ are real function.

Let us verify that normal distribution belongs to family of distributions defined by (1.4). $f(y;\mu,\sigma^2)$ is probability density function of normal distribution. Therefore $\boldsymbol{\theta} = (\mu, \sigma)$.

$$\begin{aligned} f(y;\mu,\sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left\{-\frac{(y-\mu)^2}{2\sigma^2}\right\} = \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{y^2}{2\sigma^2} + \frac{y\mu}{\sigma^2}\right\} \exp\left\{-\frac{\mu^2}{2\sigma^2}\right\} \end{aligned} \tag{1.5}$$

and we can assign:

$$\begin{aligned} s(y) &= 1, \\ r(\boldsymbol{\theta}) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{\mu^2}{2\sigma^2}\right\} \\ t_1(y) &= y^2, \\ q_1(\boldsymbol{\theta}) &= -\frac{1}{2\sigma^2} \\ t_2(y) &= y, \\ q_2(\boldsymbol{\theta}) &= -\frac{\mu}{\sigma^2} \end{aligned}$$

We have proven that normal distribution belongs to the family of exponential distributions. Therefore linear regression models are subset of GLM.

1.2.2 Models for binary dependent variable

Binary dependent variable is often used in financial institutions for estimation of client's credibility through statistical model. Such dependent variable is usually called target, for instance it can represent client's ability to repay his or her obligations towards the bank. Proper definition of target variable will be discussed in following chapters.

Let Y_i be a random variable with alternative distribution (also sometimes called Bernoulli distribution):

$$\mathbb{P}[Y(x) = 1] = 1 - \mathbb{P}[Y(x) = 0] = \pi(x) \tag{1.6}$$

Equation (1.3) can be rewritten to:

$$\mathbb{E}[Y_i] = g^{-1}\left(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}\right) = \pi(\boldsymbol{x}_i) \tag{1.7}$$

Because expected value of Y_i is restricted to the interval $[0;1]$, we should choose our link function $g(y)$ so that $g^{-1}(x) \in [0;1]$. Also we would like

$g^{-1}(x)$ to be monotonic since its output will represent probability of Y being equal to 1. There is a set of functions that has such properties. Cumulative distribution function (CDF) of every distribution has range in $[0; 1]$ and is monotonic, therefore CDF functions are a good candidates for inverse link function.

1.2.2.1 Logistic link function

CDF for logistic distribution with parameters μ and σ is given by:

$$F_{\mu,\sigma}(y) = \frac{\exp\left(\frac{y-\mu}{\sigma}\right)}{1 + \exp\left(\frac{y-\mu}{\sigma}\right)} \tag{1.8}$$

Let us focus on case when $\mu = 0$ and $\sigma = 1$. We will use this function as inverse link function for our model with Y_i drawn from the Alternative distribution (binary target variable). By combining (1.7) and (1.8) we can easily derive:

$$g(\pi(\boldsymbol{x}_i)) = \ln\left(\frac{\pi(\boldsymbol{x}_i)}{1 - \pi(\boldsymbol{x}_i)}\right) \tag{1.9}$$

Function $g(\pi)$ is called logit function and is the most frequent choice for the link function in GLM model with a binary target. Inverse function g^{-1} is called logistic function, commonly also denoted as σ since it belongs to family of sigmoid fuctions. Ratio inside the logarithm on the right side of equation (1.9) is usually called odds:

$$\text{odds}(x) = \frac{\pi(x)}{1 - \pi(x)} \tag{1.10}$$

Using (1.9), (1.7) and (1.3) we can derive:

$$\pi(\boldsymbol{x}_i) = \frac{\exp\{\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}\}}{1 + \exp\{\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}\}} = \frac{1}{1 + \exp\{-(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})\}} \tag{1.11}$$

As we can see, range of $\pi(\boldsymbol{x}_i)$ belongs to the interval $[0, 1]$ as we required.

1.2.2.2 Other link functions

For binary dependent variable, other link functions than logit can be used. Here we will mention two more examples:

- Probit function is inverse function to CDF of standard normal distribution.

  $$g_2(\pi) = \text{probit}(\pi) = \phi^{-1}(\pi) \tag{1.12}$$

  where ϕ denotes CDF of $\mathcal{N}(0, 1)$.

- If we use for inverse link function CDF of Gumbel distribution (distribution with heavy tails - extreme value distribution), then we will have complementary loglog (cloglog) fuction as our link function:

$$g_3(\pi) = \text{cloglog}(\pi) = \log(-\log(1-\pi)) \tag{1.13}$$

Note that GLM can be also used for non-binary target. The link function will then be different from those mentioned in this section. For instance Poisson regression has dependent variable with Poisson distribution and can be used to model number of incoming events in time.

1.3 Parameter estimates

In this section we will discuss how to estimate parameters of logistic regression model and what are the properties of our estimates.

First, let us shortly revise theory of maximum likelihood estimates (MLE). Likelihood function is defined as:

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} f_{Y_i}(y_i; \boldsymbol{\theta}) \tag{1.14}$$

for continuous Y. For discrete Y we have:

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} \mathbb{P}[Y_i = y_i \mid \boldsymbol{\theta}] \tag{1.15}$$

Note that likelihood function is considered as function of parameter $\boldsymbol{\theta}$ given realization y_i of random variable Y_i. $\widehat{\boldsymbol{\theta}}$ is maximum likelihood estimate of parameter $\boldsymbol{\theta}$, if

$$L(\widehat{\boldsymbol{\theta}}) \geq L(\boldsymbol{\theta}), \forall \boldsymbol{\theta} \in \Theta \tag{1.16}$$

where Θ is a space of all possible values of $\boldsymbol{\theta}$. MLE estimates have following properties:

- MLE doesn't have to be unbiased.
- MLE are consistent - asymptotically MLE estimate converges in probability to real value being estimated.
- $\widehat{\boldsymbol{\theta}}_{MLE}$ is MLE for $\boldsymbol{\theta} \Rightarrow g(\widehat{\boldsymbol{\theta}}_{MLE})$ is MLE for $g(\boldsymbol{\theta})$.
- By certain assumptions $\sqrt{n}(\widehat{\boldsymbol{\theta}}_{MLE} - \boldsymbol{\theta})$ converges in distribution to $\mathcal{N}(0, J^{-1})$ where J is Fisher information matrix.

1.3.1 Fisher information matrix

Fisher information matrix is useful for deriving confidential intervals of our estimations and it is also used in algorithms for numerical solution of the likelihood equations as we will see in following sections.

Let us calculate expected value of logarithm of likelihood function.

$$\mathbb{E}\left[\frac{\partial}{\partial \boldsymbol{\theta}}\ln(f(Y;\boldsymbol{\theta}))\right] = \int_{\mathbb{R}} \frac{\frac{\partial}{\partial \boldsymbol{\theta}} f(y;\boldsymbol{\theta})}{f(y;\boldsymbol{\theta})} f(y;\boldsymbol{\theta})\mathrm{d}y = \frac{\partial}{\partial \boldsymbol{\theta}} \int_{\mathbb{R}} f(y;\boldsymbol{\theta})\mathrm{d}y = 0 \quad (1.17)$$

We might be also interested in covariance matrix of logarithm of likelihood function. It describes how hard it is to locate correct value of $\boldsymbol{\theta}$ and it is called Fisher information matrix.

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[\left(\frac{\partial}{\partial \boldsymbol{\theta}}\ln(f(Y;\boldsymbol{\theta}))\right)^2\right] = \int_{\mathbb{R}} \left(\frac{\partial}{\partial \boldsymbol{\theta}}\ln(f(y;\boldsymbol{\theta}))\right)^2 f(y;\boldsymbol{\theta})\mathrm{d}y \quad (1.18)$$

Since we have

$$\frac{\partial^2}{\partial \boldsymbol{\theta}^2}\ln f(Y;\boldsymbol{\theta}) = \frac{\frac{\partial^2}{\partial \boldsymbol{\theta}^2} f(Y;\boldsymbol{\theta})}{f(Y;\boldsymbol{\theta})} - \left(\frac{\frac{\partial}{\partial \boldsymbol{\theta}} f(Y;\boldsymbol{\theta})}{f(Y;\boldsymbol{\theta})}\right)^2 = \frac{\frac{\partial^2}{\partial \boldsymbol{\theta}^2} f(Y;\boldsymbol{\theta})}{f(Y;\boldsymbol{\theta})} - \left(\frac{\partial}{\partial \boldsymbol{\theta}}\ln(f(Y;\boldsymbol{\theta}))\right)^2 \quad (1.19)$$

and

$$\mathbb{E}\left[\frac{\frac{\partial^2}{\partial \boldsymbol{\theta}^2} f(Y;\boldsymbol{\theta})}{f(Y;\boldsymbol{\theta})}\right] = \frac{\partial^2}{\partial \boldsymbol{\theta}^2} \int_{\mathbb{R}} f(y;\boldsymbol{\theta})\mathrm{d}y = 0 \quad (1.20)$$

If we put 1.18, 1.19 and 1.20 together, we arrive at:

$$J(\boldsymbol{\theta}) = -\mathbb{E}\left[\frac{\partial^2}{\partial \boldsymbol{\theta}^2}\ln f(Y;\boldsymbol{\theta})\right] \quad (1.21)$$

Thus, Fisher information matrix at position (i, j) is

$$J(\boldsymbol{\theta})_{i,j} = -\mathbb{E}\left[\frac{\partial^2 \ln(f(Y;\boldsymbol{\theta}))}{\partial \theta_i \partial \theta_j}\right] \quad (1.22)$$

1.3.2 Parameters of logistic regression

Assume that we observe random variable $Y_i, i = 1, ..., n$ in points $\boldsymbol{x}_i \in \mathbb{R}^k$. Random variable Y_i has alternative distribution with parameter $\pi(\boldsymbol{x}_i)$ for every i and function $\pi(x)$ follows equation (1.6). We assume that random

variables $Y_i, i = 1, ..., n$ are independent. In this section we will derive maximum likelihood estimations for beta coefficients of logistic model. We start with a transformation:

$$\mathbb{P}[Y_i = y_i] = \pi(\boldsymbol{x}_i)^{y_i}(1 - \pi(\boldsymbol{x}_i))^{1-y_i} \tag{1.23}$$

Due to the independence of the random variable, we have:

$$\mathbb{P}[Y_1 = y_1, ..., Y_n = y_n] = \prod_{i=1}^{n} \pi(\boldsymbol{x}_i)^{y_i}(1 - \pi(\boldsymbol{x}_i))^{1-y_i} \tag{1.24}$$

Using previous formula, maximum likelihood function is:

$$L(\beta_0, \beta_1, ..., \beta_k) = \prod_{i=1}^{n} \pi(\boldsymbol{x}_i)^{Y_i}(1 - \pi(\boldsymbol{x}_i))^{1-Y_i} \tag{1.25}$$

and its logarithm:

$$\begin{aligned} l(\beta_0, \beta_1, ..., \beta_k) &= \sum_{i=1}^{n} Y_i \ln\{\pi(\boldsymbol{x}_i)\} + (1 - Y_i)\ln\{1 - \pi(\boldsymbol{x}_i)\} = \\ &= \sum_{i=1}^{n} Y_i \ln\left\{\frac{\pi(\boldsymbol{x}_i)}{1 - \pi(\boldsymbol{x}_i)}\right\} + \ln\{1 - \pi(\boldsymbol{x}_i)\} = \\ &= \sum_{i=1}^{n} Y_i \left(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}\right) + \ln\left\{\frac{1}{1 + \exp\{\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}\}}\right\} = \\ &= \sum_{i=1}^{n} Y_i \left(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}\right) - \ln\{1 + \exp\{\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}\}\} \end{aligned} \tag{1.26}$$

Now we can write maximum likelihood equations for beta coefficients:

$$\begin{aligned} &\frac{\partial l(\beta_0, ..., \beta_k)}{\partial \beta_0} = 0 \\ &\frac{\partial l(\beta_0, ..., \beta_k)}{\partial \beta_j} = 0, \forall j \in 1, ..., k \end{aligned} \tag{1.27}$$

After simple evaluation:

$$\begin{aligned}
\frac{\partial l(\beta_0,...,\beta_k)}{\partial \beta_0} &= \sum_{i=1}^{n} Y_i - \frac{1}{1+\exp\{\beta_0+\sum_{j=1}^{k}\beta_j x_{ij}\}}\exp\{\beta_0+\sum_{j=1}^{k}\beta_j x_{ij}\} = \\
&= \sum_{i=1}^{n} Y_i - \pi(\boldsymbol{x}_i) \\
\frac{\partial l(\beta_0,...,\beta_k)}{\partial \beta_l} &= \sum_{i=1}^{n} Y_i x_{il} - \frac{1}{1+\exp\{\beta_0+\sum_{j=1}^{k}\beta_j x_{ij}\}}\exp\{\beta_0+\sum_{j=1}^{k}\beta_j x_{ij}\}x_{il} = \\
&= \sum_{i=1}^{n} \left(Y_i - \pi(\boldsymbol{x}_i)\right) x_{il}, \forall j \in 1,...,k
\end{aligned} \tag{1.28}$$

Therefore, likelihood equations take the following form:

$$\begin{aligned}
\sum_{i=1}^{n} Y_i - \pi(\boldsymbol{x}_i) &= 0 \\
\sum_{i=1}^{n} x_{ij}(Y_i - \pi(\boldsymbol{x}_i)) &= 0, \forall j \in 1,...,k
\end{aligned} \tag{1.29}$$

Solution of these equations is a vector denoted by $\hat{\boldsymbol{\beta}} = \left(\hat{\beta}_0,...,\hat{\beta}_k\right)$ and it will be called the maximum likelihood estimator of $\boldsymbol{\beta} = (\beta_0,...,\beta_k)$.

1.4 Basic properties of parameter estimates

Maximum likelihood estimator of the value $\pi(\boldsymbol{x}_i),\ i = 1,...,n$ is accordingly

$$\widehat{\pi}(\boldsymbol{x}_i) = \frac{\exp\{\widehat{\beta}_0+\sum_{j=1}^{k}\widehat{\beta}_j x_{ij}\}}{1+\exp\{\widehat{\beta}_0+\sum_{j=1}^{k}\widehat{\beta}_j x_{ij}\}} \tag{1.30}$$

With respect to equation (1.29), interesting relation occurs

$$\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} \widehat{\pi}(\boldsymbol{x}_i) \tag{1.31}$$

therefore, the sum of observed values Y_i is equal to the sum of predicted probabilities of Y_i being equal to 1.

Let's now calculate the variance of our estimators. According to the theory of maximum likelihood estimators, the vector $\widehat{\boldsymbol{\beta}}$ is a consistent estimator of the true value $\boldsymbol{\beta}$ and the term $\sqrt{n}\left(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta}\right)$ follows asymptotically normal distribution with the expected value equal to zero vector and variance matrix given by the inverse of Fisher information matrix. Using (1.22) we have:

$$J(\boldsymbol{\beta})=\begin{bmatrix}\sum_{i=1}^{n}\pi(\boldsymbol{x}_i)(1-\pi(\boldsymbol{x}_i)) & \cdots & \sum_{i=1}^{n}x_{ik}\pi(\boldsymbol{x}_i)(1-\pi(\boldsymbol{x}_i))\\ \sum_{i=1}^{n}x_{i1}\pi(\boldsymbol{x}_i)(1-\pi(\boldsymbol{x}_i)) & \cdots & \sum_{i=1}^{n}x_{ik}x_{i1}\pi(\boldsymbol{x}_i)(1-\pi(\boldsymbol{x}_i))\\ \vdots & \ddots & \vdots\\ \sum_{i=1}^{n}x_{ik}\pi(\boldsymbol{x}_i)(1-\pi(\boldsymbol{x}_i)) & \cdots & \sum_{i=1}^{n}x_{ik}^2\pi(\boldsymbol{x}_i)(1-\pi(\boldsymbol{x}_i))\end{bmatrix} \tag{1.32}$$

since the second derivatives of $l(\beta_0,...,\beta_k)$ does not depend on the values of $Y_i,\ i=1,...,n$. Plugging in the estimate $\widehat{\pi}(x_i)$ and inverting equation (1.22) leads to variance matrix of the estimator $\widehat{\boldsymbol{\beta}}$.

The knowledge of Fisher information matrix therefore allows an evaluation of confidence interval for β_j. Denote $\mathrm{SE}_i=\frac{1}{\sqrt{nJ(\boldsymbol{\beta})_{ii}}}$ standard error in the estimator of β_i, then its confidence interval with confidence level α is

$$\left(\widehat{\beta}_i-u_{1-\frac{\alpha}{2}}\mathrm{SE}_i,\widehat{\beta}_i+u_{1-\frac{\alpha}{2}}\mathrm{SE}_i\right) \tag{1.33}$$

where $u_{1-\frac{\alpha}{2}}$ is an appropriate quantile of normal distribution.

1.5 Numerical solution of the likelihood equation

Maximum likelihood equations (1.29) are usually solved numerically. One of the basic algorithms is Newton-Raphson algorithm. Denote $\boldsymbol{\beta}$ the vector of parameters $(\beta_0,...,\beta_k)$. Newton-Raphson algorithm constructs a series $\boldsymbol{\beta}^{(t)},\ t=0,\dots,m$, that converges to the true solution. $\boldsymbol{\beta}^{(t+1)}$ is given by the following equation, with any real vector $\boldsymbol{r}$ used as the initial estimate:

$$\begin{aligned}\boldsymbol{\beta}^{(0)}&=\boldsymbol{r}\\ \boldsymbol{\beta}^{(t+1)}&=\boldsymbol{\beta}^{(t)}-(\boldsymbol{H}^{(t)})^{-1}\boldsymbol{q}^{(t)}\end{aligned} \tag{1.34}$$

where $\boldsymbol{q}^{(t)}=\nabla l(\boldsymbol{\beta}^{(t)})$ and $\boldsymbol{H}^{(t)}$ is a kxk matrix given by

$$H_{i,j}^{(t)}=\frac{\partial^2 l(\boldsymbol{\beta}^{(t)})}{\partial\beta_i\partial\beta_j} \tag{1.35}$$

If we denote the value of $\pi^t(x)$:

$$\pi^{(t)}(\boldsymbol{x}) = \frac{\exp\left\{\beta_0^{(t)} + \sum_{j=1}^k \beta_j^{(t)} x_j\right\}}{1 + \exp\left\{\beta_0^{(t)} + \sum_{j=1}^k \beta_j^{(t)} x_j\right\}} \tag{1.36}$$

we have

$$\boldsymbol{q}^{(t)} = \begin{pmatrix} \sum_{i=1}^n Y_i - \pi^{(t)}(\boldsymbol{x}_i) \\ \sum_{i=1}^n x_{1i}(Y_i - \pi^{(t)}(\boldsymbol{x}_i)) \\ \vdots \\ \sum_{i=1}^n x_{ki}(Y_i - \pi^{(t)}(\boldsymbol{x}_i)) \end{pmatrix} \tag{1.37}$$

and

$$\boldsymbol{H}^{(t)} = -\begin{bmatrix} \sum_{i=1}^n \pi^{(t)}(\boldsymbol{x}_i)(1-\pi^{(t)}(\boldsymbol{x}_i)) & \cdots & \sum_{i=1}^n x_{ik}\pi^{(t)}(\boldsymbol{x}_i)(1-\pi^{(t)}(\boldsymbol{x}_i)) \\ \sum_{i=1}^n x_{i1}\pi^{(t)}(\boldsymbol{x}_i)(1-\pi^{(t)}(\boldsymbol{x}_i)) & \cdots & \sum_{i=1}^n x_{ik}x_{i1}\pi^{(t)}(\boldsymbol{x}_i)(1-\pi^{(t)}(\boldsymbol{x}_i)) \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^n x_{ik}\pi^{(t)}(\boldsymbol{x}_i)(1-\pi^{(t)}(\boldsymbol{x}_i)) & \cdots & \sum_{i=1}^n x_{ik}^2\pi^{(t)}(\boldsymbol{x}_i)(1-\pi^{(t)}(\boldsymbol{x}_i)) \end{bmatrix} \tag{1.38}$$

Finally, let's note that the matrix $\boldsymbol{H}^{(t)}$ does not depend on the value of observations and therefore $-\boldsymbol{H}^{(t)}$ represents Fisher information matrix computed at t-th iteration.

Explanation of Newton-Rhapson algorithm is based on Taylor expansion. Let $\boldsymbol{\beta}^*$ be true solution of MLE equations (1.29). Then

$$\boldsymbol{q}(\boldsymbol{\beta}^*) = \begin{pmatrix} \frac{\partial l(\beta_0,\ldots,\beta_k)}{\partial \beta_1} \\ \vdots \\ \frac{\partial l(\beta_0,\ldots,\beta_k)}{\partial \beta_k} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{1.39}$$

For any other value of $\boldsymbol{\beta}$ (let's denote it as $\boldsymbol{\beta}^{(0)}$), based on Taylor expansion we can write:

$$\mathbf{0} = \boldsymbol{q}(\boldsymbol{\beta}^*) \cong q(\boldsymbol{\beta}^{(0)}) + \boldsymbol{H}(\boldsymbol{\beta}^{(0)})(\boldsymbol{\beta}^* - \boldsymbol{\beta}^{(0)}) \tag{1.40}$$

and thus:

$$\boldsymbol{\beta}^* \cong \boldsymbol{\beta}^{(0)} - (\boldsymbol{H}(\boldsymbol{\beta}^{(0)}))^{-1} \cdot \boldsymbol{q}(\boldsymbol{\beta}^{(0)}) \tag{1.41}$$

1.6 Test of the parameter significance

In this section, we will derive a significance test of parameters $\beta_1, \ldots, \beta_k$, which is the key factor of the dependence on variable $\boldsymbol{x}$. The null hypothesis

will be that all parameters are equal to zero, e.g. $\beta_j = 0,\ j = 1, \ldots, k$. If we do not reject this hypothesis, it is equivalent to not rejecting the hypothesis that $Y_i,\ i = 1, \ldots, n$ are identically distributed. In such case, we can see that for $\beta_1 = 0$:

$$\pi(\boldsymbol{x}) = \mathbb{P}\left[Y(\boldsymbol{x}) = 1\right] = \frac{\exp\left\{\beta_0\right\}}{1 + \exp\left\{\beta_0\right\}} \tag{1.42}$$

Therefore

$$\text{odds}(\boldsymbol{x}) = \frac{\pi(\boldsymbol{x})}{1 - \pi(\boldsymbol{x})} = \exp\left\{\beta_0\right\} \tag{1.43}$$

and

$$\beta_0 = \ln\left\{\text{odds}(\boldsymbol{x})\right\} = \text{logit}(\boldsymbol{x}) \tag{1.44}$$

is equal to logit. From equation (1.29) we have:

$$\begin{aligned}
\sum_{i=1}^{n} Y_i - \pi(\boldsymbol{x}_i) &= 0 \\
\sum_{i=1}^{n} Y_i - \frac{\exp\left\{\beta_0\right\}}{1 + \exp\left\{\beta_0\right\}} &= 0 \\
\left(1 + \exp\left\{\beta_0\right\}\right) \sum_{i=1}^{n} Y_i &= \sum_{i=1}^{n} \exp\left\{\beta_0\right\} \\
\sum_{i=1}^{n} Y_i &= \exp\left\{\beta_0\right\} \left(n - \sum_{i=1}^{n} Y_i\right) \\
\beta_0 &= \ln\left\{\frac{\sum_{i=1}^{n} Y_i}{n - \sum_{i=1}^{n} Y_i}\right\}
\end{aligned} \tag{1.45}$$

Consequently, if we denote n_0 number of performing loans and by n_1 number of defaulted loans, then $\widehat{\beta_0} = \ln\left\{\frac{n_1}{n_0}\right\}$, the ratio of performing and non-performing loans. Moreover, we have:

$$\pi(x) = \frac{\frac{n_1}{n_0}}{1 + \frac{n_1}{n_0}} = \frac{\frac{n_1}{n_0}}{\frac{n_0 + n_1}{n_0}} = \frac{n_1}{n} \tag{1.46}$$

As expected, with only one constant predictor, our prediction exhibits the same default rate as the sample used for training.

We turn our attention back to the hypothesis testing with the test which comes from the general principle of testing by likelihood ratio. This ratio can be used to judge the predictive power of the model itself as well.

In linear regression, we measure the deviation of predicted values from the real ones by residual sum of squares, denoted by SSE. It is equal to:

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \widehat{y}_i)^2 \tag{1.47}$$

where $\widehat{y}_i$ are maximum likelihood estimators in classical linear regression model. A generalization of SSE is called deviance and it is driven by the likelihood of models. Deviance is equivalent with SSE in the linear model. An ideal model is called saturated model: consider having same number of predictors as the number of observations with perfect prediction of $\widehat{y}_i = y_i,\ i = 1, \ldots, n$. We have following equation for the deviance:

$$D = -2 \ln \left\{ \frac{\text{likelihood of the linear regression model}}{\text{likelihood of the saturated model}} \right\} \tag{1.48}$$

We now run the calculation for logistic regression model.

$$\begin{aligned} D &= -2 \ln \left\{ \frac{\prod_{i=1}^{n} \widehat{\pi}(\boldsymbol{x}_i)^{y_i} (1 - \widehat{\pi}(\boldsymbol{x}_i))^{1-y_i}}{\prod_{i=1}^{n} y_i^{y_i} (1 - y_i)^{1-y_i}} \right\} \\ &= -2 \ln \left\{ \prod_{i=1}^{n} \left(\frac{\widehat{\pi}(\boldsymbol{x}_i)}{y_i} \right)^{y_i} \left(\frac{1 - \widehat{\pi}(\boldsymbol{x}_i)}{1 - y_i} \right)^{1-y_i} \right\} \\ &= -2 \sum_{i=1}^{n} y_i \ln \left\{ \frac{\widehat{\pi}(\boldsymbol{x}_i)}{y_i} \right\} + (1 - y_i) \ln \left\{ \frac{1 - \widehat{\pi}(\boldsymbol{x}_i)}{1 - y_i} \right\} \end{aligned} \tag{1.49}$$

Since the likelihood of the saturated model is equal to one, we can also write simply that $D = -2l$.

If we compare the values of D for models with and without parameter β_j, we can statistically test significance of β_j. This test is called likelihood-ratio test and it has asymptotically chi-squared distribution with one degree of freedom under hypothesis of $\beta_j = 0$. Test statistic has following form:

$$G = D(\text{model without } \beta_j) - D(\text{model with } \beta_j) \sim \chi_1^2 \tag{1.50}$$

We reject the null hypothesis in the case of high values of the statistic G. We note that since we know the distribution of β_j estimator is asymptotically normal with known variance, we can also apply a test:

$$Z = \frac{\widehat{\beta}_j}{\widehat{\sigma}(\widehat{\beta}_j)} \sim \mathcal{N}(0, 1) \tag{1.51}$$

Note 2. *Since n should be large enough for asymptotic test, we know that Student distribution with $n-2$ degrees of freedom is approximately the same as normal.*

In literature, the likelihood-ratio test is reported to be more powerful.

We can test the hypothesis stating that some sub-vector of $\boldsymbol{\beta}$ is zero by performing likelihood ratio test. Suppose that parts $i_1, \ldots, i_l$ of vector $\boldsymbol{\beta}$ are equal to zero, then we compute:

$$D = -2 \ln \left\{ \frac{\text{likelihood of the simplified model}}{\text{likelihood of the full model}} \right\}, \tag{1.52}$$

which has under the null hypothesis χ^2 distribution with l degrees of freedom. We reject the null hypothesis if the value of statistic (1.52) is greater than corresponding quantile of χ^2_l distribution.

A special case is a choice of such vectors $\boldsymbol{x}_i,\ i = 1, \ldots, n$ that the first component is always equal to 1. Basically, the regression is then performed with respect to $k-1$ dimensional space and the first component of independent variables plays a role of intercept. The test of significance on at least one of the components of $\boldsymbol{x}$ is then a special case of the test above, with $i_1 = 2, \ldots, i_{k-1} = k$ and $l = k-1$. In this case, the test statistic has χ^2_{k-1} distribution.

A common procedure to construct a model starts with a full model and tests if at least one parameter is nonzero, according to the test mentioned above. If the null hypothesis is rejected, an indicative test of parameter significance is computed for each parameter $\beta_2, \ldots, \beta_k$:

$$S_j = \frac{\widehat{\beta}_j}{\widehat{\sigma}(\widehat{\beta}_j)},\ j = 2, \ldots, k, \tag{1.53}$$

where $\widehat{\sigma}(\widehat{\beta}_j)$ is given by a square root of the j-th diagonal item of the inverse of Fisher information matrix multiplied by square root of n. Statistics $S_j, j = 2, \ldots, k$ have asymptotically normal distribution with zero mean and unit variance. Based on these tests, we find out candidates to be tested altogether by the test (1.52). However, this does not give us further hint what to do if the final test rejects null hypothesis of these parameters being zero, even though each single univariate test supports this hypothesis. Therefore, stepwise selection procedures are very popular in practical applications.

1.7 Stepwise regression

In this section, we will suppose that vectors $\boldsymbol{x}_i,\ i = 1, \ldots, n$ have the first component equal to 1 and we will try to find out, which of the parameters

$\beta_2, \ldots, \beta_k$ are significant to the final model. Generally, we have two possible approaches to construct the model - backward and forward regression. In the backward regression, we start with the full model, containing all parameters $\boldsymbol{\beta}$, and reduce the count of parameters in the model. In forward selection, we start with a model containing just parameter β_1 and keep adding parameter with greatest significance.

Algorithm 3 (Stepwise regression)**.**

1. *Denote L_M, $M \subset \{2, \ldots, k\}$ values of the log likelihood function assuming that only parameters from the set M are included in the model. Assume that β_1 is always included in the model. We will choose the next candidate based on the statistic $G_j^{(0)} = 2(l_{\{1,j\}} - l_{\{1\}})$. This statistic has χ_1^2 distribution under the hypothesis of $\beta_j = 0$. Denote $p_j^{(0)} = \mathbb{P}\left[X > G_j^{(0)}\right]$, where $X \sim \chi_1^2$. The lower this probability is, the greater is the probability that the model $L_{\{1,j\}}$ is more appropriate. Denote $e_0 = \underset{j=2,\ldots,k}{\arg\min}\, p_j^{(0)}$, index which corresponds to the lowest value of $p_j^{(0)}$. Suppose that a level p_E, which controls the significance of new variables entering the model, was already specified. If $p_{e_0} \geq p_E$, algorithm ends. Otherwise set iteration index $s = 0$ and continue with the next step. We will discuss the appropriate choice of parameter p_E later.*

2. *Denote E_s index set of parameters after iteration s, for example $E_1 = \{1, e_0\}$. Set $s = s + 1$ and choose the next variable to be added to the model. Compute $G_j^{(s)} = 2(l_{E_{s-1} \cup \{j\}} - l_{E_{s-1}})$, $j \notin E_{s-1}$ and $p_j^{(s)} = \mathbb{P}\left[X > G_j^{(s)}\right]$, where $X \sim \chi_1^2$. Select $e_s = \underset{j \notin E_{s-1}}{\arg\min}\, p_j^{(s)}$, if $p_{e_s} \geq p_E$, algorithm ends, otherwise set $E_s = E_{s-1} \cup \{e_s\}$ and continue with the next step.*

3. *In this step, we will check whether the newly added variable caused that some of the already present variable could be removed due to dependence with the new one. This can be done again by using statistics $G_j^{(s_B)} = 2(l_{E_s} - l_{E_s \setminus \{j\}})$, $j \in E_s \setminus \{1, e_s\}$, where s_B denotes that we are in the "backward" part of the algorithm. We compute $p_j^{(s_B)} = \mathbb{P}\left[X > G_j^{(s_B)}\right]$, where $X \sim \chi_1^2$ and this time we want to find high values of $p_j^{(s_B)}$ which support the hypothesis of β_j being zero and therefore can be removed from the model. Denote $r_s = \underset{j \in E_s \setminus \{1, e_s\}}{\arg\max}\, p_j^{(s_B)}$. If the value r_s is higher than fixed level p_R, we remove variable r_s from the model, setting $E_s =$*

$E_s \backslash \{r_s\}$. Next, iterate again with step 2. The value of p_R has to be greater than value of p_E, which ensures that the algorithm does not cycle. If we add in some step variable e_s and remove variable r_s it holds that $e_{s+1} \neq r_s$, because $p_R > p_E$ guarantees that $G_{e_{s+1}}^{(s+1)} > G_{r_s}^{(s_B)}$.

Under the assumption of $p_R > p_E$ the algorithm does not cycle and since the number of variables is finite, it ends in a finite number of steps. We shall now discuss the choice of p_E and p_R. Even though these parameters emerge in the context of hypothesis testing, their purpose is to indicate if a variable is suitable to enter or leave the model, not to perform actual statistical test. If we would use the levels common in statistical testing, the resulting models would be too small in the terms of number of variables. Generally, toughness of the rule to enter the model shall be proportional to the toughness of the rule to leave, meaning that lower p_E should lead to lower p_R, however, having in mind that $p_R > p_E$. Suggested value of p_E are between 0.15 and 0.2, while the value of p_R should be greater by 0.05 or 0.1. If we want to ensure that small number of variables are removed, we can choose greater value of p_R.

Due to the fact that the values of p_E do not have a meaning of significance levels for statistical testing, we run one more procedure on the final model from the previous algorithm. In this procedure, we determine which variables will be present in the final model. There are two possibilities, the first is based on the statistical test of significance of gradually added variables.

Algorithm 4. *Let $i_q, q = 1, \ldots, Q$ denote the sequence of variables gradually added to the model, where $Q + 1$ is the number of variables in the model, including the intercept β_1. We note that this sequence is a sub-sequence of the sequence e_s, because some of the variables could be removed in subsequent steps. We introduce the variable sets E_q and statistics $G_q = 2(L_{E_q} - L_{E_{q-1}})$ in the same manner as before. Let $X \sim \chi_1^2$, if*

$$\mathbb{P}\left[X > G_q\right] < \alpha \tag{1.54}$$

we continue with $q + 1$, otherwise we end and select the last model where (1.54) was satisfied.

The second procedure is based on test which examines if all variables in model are zero, if it is rejected, we consider the first variable as arguably present and continue by increasing the number of variables in the model.

Algorithm 5. *Set $q = 1$ and compute $G_q = 2(l_{E_{Q+1}} - l_{E_q})$. Let $X \sim \chi_{Q+1-q}^2$, if $\mathbb{P}\left[X > G_q\right] < \alpha$ we continue with $q = q + 1$. If the previous inequality is not satisfied for some $\hat{q}$, the final model contains variables from $E_{\hat{q}-1}$.*

1.8 Goodness of fit tests

Classical goodness of fit tests based on Pearson's statistic are designed for k categories, with O_i observed count in category i and E_i expected count in this category. Recall that for n independent clients with probability of default of p_i we would have $E_i = np_i$.

$$\chi^2 = \sum_{k=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{1.55}$$

Such tests are useful in the case where we have the multiple vectors $\boldsymbol{x}_i$ of the same value and therefore multiple observations for this point. This is usually not the case in credit risk and therefore we will focus on different procedures.

Hosher and Lemeshow proposed in the article [2] following goodness of fit test. Suppose for the sake of simplicity distinct values $\boldsymbol{x}_i$ for all i.The test divides the observations into 10 groups, $C_1, \ldots, C_{10}$. There are two ways how to do this - in the first one we put into C_1 such $\boldsymbol{x}_i$ that their estimated probabilities $\widehat{\pi}(\boldsymbol{x}_i)$ belong to the first tenth of all estimated probabilities. The second group then contains vectors $\boldsymbol{x}_i$ whose estimated probabilities are between first and second decile and so on. An alternative approach could put into the first group such $\boldsymbol{x}_i$ that $\widehat{\pi}(\boldsymbol{x}_i) \leq \frac{1}{10}$, the probabilities in the second group would be between $\frac{1}{10}$ and $\frac{2}{10}$ and so on. Let $n_k = |C_k|$ denote the number of observations in the group C_k, o_k the number of events (defaults) in the group C_k: $o_k = \sum_{i:\ \boldsymbol{x}_i \in C_k} Y_i$. Finally, we denote by $\bar{\pi}_k$ the average of probabilites in the group C_k: $\bar{\pi}_k = \frac{1}{n_k} \sum_{\boldsymbol{x}_i \in C_k} \widehat{\pi}(\boldsymbol{x}_i)$. The Hosmer-Lemeshow statistic is based on value:

$$\hat{C} = \sum_{k=1}^{10} \frac{(o_k - n_k \bar{\pi}_k)^2}{n_k \bar{\pi}_k (1 - \bar{\pi}_k)} \tag{1.56}$$

This statistic has χ^2_{g-2} distribution, where g is the number of groups. In our case, $g = 10$ and therefore we have 8 degrees of freedom.

Other approaches measure the quality of fit by statistics with analogous meaning as R^2 in linear regression. Denote the log-likelihood function of our model l_M and log-likelihood of the model with just intercept as l_0. Then, the McFadden "pseudo" R^2 is:

$$R^2_{\text{McFadden}} = 1 - \frac{l_M}{l_0} \tag{1.57}$$

As reported in the literature, values of McFadden R^2 between 0.2 and 0.4 already represent models with a very good fit.

Second approach introduced by Tjur [3] computes the difference between predicted probabilities between events and non-events. Recall that n_1 denotes the number of defaults and n_0 the number of performing loans:

$$R^2_{\text{Tjur}} = \frac{1}{n_1} \sum_{i:\ Y_i=1} \widehat{\pi}(\boldsymbol{x}_i) - \frac{1}{n_0} \sum_{i:\ Y_i=0} \widehat{\pi}(\boldsymbol{x}_i) \tag{1.58}$$

1.9 Diversification power

In previous sections, we have discovered models and procedures to estimate parameters in the logistic regression. Namely, we have estimated regression function:

$$\mathbb{P}\left[Y(\boldsymbol{x}) = 1\right] = 1 - \mathbb{P}\left[Y(\boldsymbol{x}) = 0\right] = \pi(\boldsymbol{x}) \tag{1.59}$$

It is important to realize that even though we could have a very nice estimate and fit of the regression function, it does not have to mean much for the task of differentiation between good and bad client. Based on outputs of the logistic regression we have to build a decision rule which will select good and bad clients. For some data sets, constant prediction using average bad rate of the sample can give pretty nice value of the goodness of fit. However, having the same prediction for all clients means that we have no information to make decision which client to approve and which client to reject.

Taking reasons above into consideration, we shall focus on measuring the quality of decisioning. Our decisions will be of a good quality if they differentiate well between good and bad clients. To put this in a more formal setup, denote K random client who comes to a bank. $\boldsymbol{x}_K$ denotes the information which is known about the client at the time of application for a loan. Next, denote by Y_K a random variable, which determines if the client is good ($Y_K = 0$) or bad ($Y_K = 1$). We denote the respective probabilities $p_G = \mathbb{P}\left[Y_K = 0\right]$ and $p_B = \mathbb{P}\left[Y_K = 1\right]$. We will explain later what could be the meaning of good and bad client. Moreover, let $s(\boldsymbol{x})$ denote the decisioning function based on which we will decide about the client. Intentionally, we do not keep the notation of $\pi(\boldsymbol{x})$, because our decisions do not have to be based on logistic regression. We will suppose that range of $s(\boldsymbol{x})$ is $[0, 1]$ and that greater values mean worse client. To assess the discriminating power of the function, we need to analyse its distribution for good and bad clients. The cumulative distribution function of good clients is following:

$$F^G(p) = \mathbb{P}\left[s(\boldsymbol{x}) < p | Y_K = 0\right]. \tag{1.60}$$

Similarly, for the bad clients:

$$F^B(p) = \mathbb{P}\left[s(\boldsymbol{x}) < p | Y_K = 1\right]. \tag{1.61}$$

Now we define True Positive Rate (TPR, sensitivity, recall or hit rate) and False Positive Rate (FPR, fall-out):

$$\begin{aligned} \mathrm{TPR}(p) &= F^G(p) \\ \mathrm{FPR}(p) &= F^B(p) \end{aligned} \tag{1.62}$$

If we put values $[\mathrm{FPR}(p), \mathrm{TPR}(p)]\,,\ p \in [0,1]$ into a chart, we get so-called Receiver operating characteristic curve (ROC curve). Basically, if the ROC curve would be close to identity, our discriminating power between good and bad clients is not very efficient. On the other hand, it the ROC curve would be very close to the upper left angle (maximum of x and y axis), our decisions would be strong.

Definition 6. *Integral criterion of decisioning function quality is defined by:*

$$\mathrm{IK}(s) = \int_0^1 |F^G(p) - F^B(p)| \ \mathrm{d}p. \tag{1.63}$$

Kolmogorov-Smirnov statistic of decisioning function quality is defined by:

$$\mathrm{KS}(s) = \sup_{p \in [0,1]} |F^G(p) - F^B(p)|. \tag{1.64}$$

*Area Under ROC curve (*AUC *s) is given by:*

$$\mathrm{AUC}(s) = \int_0^1 \mathrm{TPR}(p) \ \mathrm{d}\,\mathrm{FPR}(p) = \int_0^1 F^G(p) \ \mathrm{d}F^B(p). \tag{1.65}$$

Gini coefficient is defined by:

$$\begin{aligned} \mathrm{Gini}(s) &= 2 \int_0^1 (\mathrm{TPR}(p) - \mathrm{FPR}(p))\, \mathrm{d}\,\mathrm{FPR}(p) \\ &= 2 \int_0^1 \left(F^G(p) - F^B(p)\right) \mathrm{d}F^B(p). \end{aligned} \tag{1.66}$$

Let K_1 and K_2 be independent, c*-statistic is the value of*

$$\mathrm{c}(s) = \mathbb{P}\left[s(\boldsymbol{x_{K_1}}) \leq s(\boldsymbol{x_{K_2}}) | Y_{K_1} = 0,\ Y_{K_2} = 1\right] \tag{1.67}$$

We will usually drop the argument s from $\mathrm{IK}(s)$ and write just IK for simplicity. Let us now discuss basic properties of these criteria.

Lemma 7. *Let F^G, F^B be continuous distribution functions and let $F^G(p) \geq F^B(p)$ for all $p \in [0,1]$. Then it holds:*

1. $\text{Gini} = 2\,\text{AUC} - 1$
2. $\text{AUC} = 1 - \int_0^1 F^B(p)\ \mathrm{d}F^G(p)$
3. $\text{c} = \text{AUC}$

Proof. By evaluation of integrals:

1. With $\int_0^1 F^B(p)\mathrm{d}F^B(p) = \left[F^B(p)F^B(p)\right]_0^1 - \int_0^1 F^B(p)\mathrm{d}F^B(p) = \frac{1}{2}$:
$$\begin{aligned}\text{Gini}(s) &= 2\int_0^1 \left(F^G(p) - F^B(p)\right)\mathrm{d}F^B(p)\\ &= 2\int_0^1 F^G(p)\mathrm{d}F^B(p) - 2\int_0^1 F^B(p)\mathrm{d}F^B(p)\\ &= 2\,\text{AUC} - 1\end{aligned}$$
2. Using integration by parts:
$$\begin{aligned}\text{AUC} &= \int_0^1 F^G(p)\ \mathrm{d}F^B(p)\\ &= \left[F^G(p)F^B(p)\right]_0^1 - \int_0^1 F^B(p)\ \mathrm{d}F^G(p)\\ &= 1 - \int_0^1 F^B(p)\ \mathrm{d}F^G(p)\end{aligned}$$
3. Due to the independence we have:
$$\begin{aligned}&\text{c} = \mathbb{P}\left[s(\boldsymbol{x_{K_1}}) \leq s(\boldsymbol{x_{K_2}}) | Y_{K_1} = 0,\ Y_{K_2} = 1\right] = \int_0^1\int_p^1 \mathrm{d}F^B(t)\ \mathrm{d}F^G(p) =\\ &= \int_0^1 \left(1 - F^B(p)\right)\mathrm{d}F^G(p) = 1 - \int_0^1 F^B(p)\mathrm{d}F^G(p) =\\ &= 1 + \text{AUC} - 1 = \text{AUC}\end{aligned}$$

□

Note 8. *Originally Gini coefficient was defined to measure wealth inequality in population using area above so called Lorenz curve. However, Lorenz curve uses distribution of whole population on the x axis, not only distribution of bads or goods as we do in ROC curve. We therefore suggest to use ROC curve only, since by using Lorenz curve the resulting Gini coefficient might have slightly different value.*

The equations above have to be adjusted for practical application. Suppose we have $K_1, \ldots, K_n$ randomly chosen clients, representing the structure of the bank clients. For each of them we have the value $s(\boldsymbol{x}_i),\ i = 1, \ldots, n$, which is based on the input information $\boldsymbol{x}_i$. We will estimate the distribution functions F^G and F^B by the empirical distribution functions. Suppose now that the clients are ordered in a way that first n_G clients are good and the rest $n_B = n - n_G$ are bad:

$$\begin{aligned} \hat{F}^G(p) &= \frac{|i : i \in \{1, \ldots, n_G\},\ s(\boldsymbol{x}_i) \leq p|}{n_G} \\ \hat{F}^B(p) &= \frac{|i : i \in \{n_G + 1, \ldots, n\},\ s(\boldsymbol{x}_i) \leq p|}{n_B} \end{aligned} \tag{1.68}$$

These estimates can be used to construct ROC curve. The estimate of Lorentz curve can be based on piece-wise linear function which runs through these points. The computation of Gini coefficient or other criteria can be then performed using the values from this piece-wise linear curve. The computation of the c-statistic can be run directly by combinatorial argument: every chosen couple of a bad and good client has the same probability. We compute the number of couples where $s(\boldsymbol{x}_{K_G}) \leq s(\boldsymbol{x}_{K_B})$ and estimate c by the ratio of this number to the number of all couples.

In practice, other statistics are used to estimate Gini coefficient. Suppose again we have scores $s(\boldsymbol{x}_i),\ i = 1, \ldots, n$ of n clients. Denote:

- a the number of couples $(i, j),\ i < j$ for which $\operatorname{sgn}(s(\boldsymbol{x}_i) - s(\boldsymbol{x}_j)) = \operatorname{sgn}(Y_i - Y_j) \neq 0$, e.g. the number of cases when the good client received better score than the bad client.

- b the number of couples $(i, j),\ i < j$ for which $\operatorname{sgn}(s(\boldsymbol{x}_i) - s(\boldsymbol{x}_j)) = -\operatorname{sgn}(Y_i - Y_j) \neq 0$, e.g. the number of cases when the good client received worse score than the bad client.

- c the number of couples $(i, j),\ i < j$ for which $s(\boldsymbol{x}_i) = s(\boldsymbol{x}_j)$ and $Y_i \neq Y_j$, e.g. the number of cases when the good client received same score as the bad client.

Then the Sommers d-statistic is given by:

$$\mathrm{d} = \frac{a - b}{a + b + c} \tag{1.69}$$

It can be shown that the Sommers-d equals to the Gini coefficient. Sometimes, Gini coefficient is also calculated based on the number of interchanges needed to reach the ideal state, when all the bad clients have worse score than

the good clients. In such case, no interchanges in ordering are necessary. In random model, we have to do $c_r = \frac{1}{2} n_G n_B$ interchanges to reach the optimal ordering. Suppose number of interchanges for our model is c_m, then:

$$\text{Gini} = \frac{c_r - c_m}{c_r} \tag{1.70}$$

Another useful statistic is Lift, which indicates the ratio of the proportion of bad clients with a score of greater than p, $p \in [0, 1]$, to the proportion of bad clients in the general population. Usually, lift is computed for the worst α-percent of population, where it indicates how many times the scoring performs better than just random selection. Suppose that the clients are ordered by score, e.g. $s(\boldsymbol{x}_1) \leq \ldots \leq s(\boldsymbol{x}_n)$, then:

$$\text{Lift}_\alpha = \frac{\frac{1}{\lfloor \alpha n \rfloor} \sum_{i=n-\lfloor \alpha n \rfloor}^{n} Y_i}{\frac{1}{n} \sum_{i=1}^{n} Y_i} \tag{1.71}$$

The level α should be set to estimated rejection percentage.

Example 9. *In the population with default rate $d_t = \frac{1}{3}$ we would like to use a scoring function with* $\text{Lift}_{20\%}$ *of 2 and reject $r = 20\%$ worst clients. What would be the default rate of the approved population?*

The total number of defaulters is $d_t n$ and the total number of rejected defaulters is given by multiplication with rejection share and lift. Therefore, the resulting default rate is

$$d_a = \frac{d_t n - r d_t n \, \text{Lift}_{20\%}}{(1-r)n} = \frac{d_t (1 - r \, \text{Lift}_{20\%})}{1-r} = \frac{\frac{1}{3}(1 - \frac{2}{5})}{\frac{4}{5}} = \frac{1}{4}$$

1.10 Scoring Models

In this section we would like to describe the basic principle of the most common scoring models used in finance. First, we start with some important definitions.

1.10.1 Odds Ratio Definition

Suppose that our database contains s explanatory categorical variables (predictors) for each client, where the i-th variable consists of p_i categories. Scoring models in practice consist often of categorical predictors only We put

$$Z = \big\{(i, j) : i \in \{1, \ldots, s\}, j \in \{1, \ldots p_i, \}\big\} \tag{1.72}$$

the set of all ordered pairs (i, j) of variables i and their categories j.

Then for each client k we have the vector

$$\boldsymbol{x}_k = \big((x_j^i)_k : (i,j) \in Z\big) \tag{1.73}$$

of dummy variables (i.e. $(x_j^i)_k = 1$ if the client k lies in the category j of the variable i, and $(x_j^i)_k = 0$ otherwise). Then we denote by B the index set of all defaulted clients (we also call them *bad* clients) and G the index set of all non-defaulted clients (*good* clients), and in the same spirit we define

$$B_j^i = \big\{k : k \in B, (x_j^i)_k = 1\big\} \tag{1.74}$$

as the index set of all defaulted clients k lying in the category j of the variable i, and

$$G_j^i = \big\{k : k \in G, (x_j^i)_k = 1\big\} \tag{1.75}$$

as the index set of all non-defaulted clients k lying in the category j of the variable i.

Now, based on the database of the observed clients, we define the total odds as the ratio of the number of defaulted vs. non-defaulted clients in the sample,

$$\text{odds} = \frac{|B|}{|G|}, \tag{1.76}$$

and also for individual categories j of variable i the odds_j^i for the certain category,

$$\text{odds}_j^i = \frac{|B_j^i|}{|G_j^i|}. \tag{1.77}$$

Finally we define the *odds ratio* (OR_j^i) as the ratio of categorical odds_j^i and total odds,

$$\text{OR}_j^i = \frac{\text{odds}_j^i}{\text{odds}}. \tag{1.78}$$

1.10.2 Fundamental of Scoring Models

In the beginning we remark that the notation of the variable odds from (1.76) is consistent with the function $\text{odds}(\boldsymbol{x})$ from (1.10) because with the usual estimates of algebraic probabilities we can write

$$odds = \frac{|B|}{|G|} = \frac{\frac{|B|}{|B \cup G|}}{\frac{|G|}{|B \cup G|}} = \frac{\mathbb{P}\left[Y = 1\right]}{\mathbb{P}\left[Y = 0\right]}.$$

Now estimate the value of the theoretical function odds($\boldsymbol{x}$) from definition (1.10) based on the empirical values for the introduced variables. We write

$$\text{odds}(\boldsymbol{x}) = \frac{\mathbb{P}\left[Y_{\boldsymbol{x}} = 1\right]}{\mathbb{P}\left[Y_{\boldsymbol{x}} = 0\right]} = \frac{\frac{|B_{\boldsymbol{x}}|}{|B_{\boldsymbol{x}} \cup G_{\boldsymbol{x}}|}}{\frac{|G_{\boldsymbol{x}}|}{|B_{\boldsymbol{x}} \cup G_{\boldsymbol{x}}|}} = \frac{|B_{\boldsymbol{x}}|}{|G_{\boldsymbol{x}}|}, \tag{1.79}$$

where we denote

$$B_{\boldsymbol{x}} = \left\{k : k \in B, \boldsymbol{x}_k = \boldsymbol{x}\right\} \tag{1.80}$$

as the index set of all defaulted clients with the vector of characteristics $\boldsymbol{x}$, and

$$G_{\boldsymbol{x}} = \left\{k : k \in G, \boldsymbol{x}_k = \boldsymbol{x}\right\} \tag{1.81}$$

as the index set of all non-defaulted clients with the vector of characteristics $\boldsymbol{x}$.

As the values $|B_{\boldsymbol{x}}|$ and $|G_{\boldsymbol{x}}|$ are dependent on specific combinations of values of the vector $\boldsymbol{x}$, it is usually not convenient to estimate odds($\boldsymbol{x}$) using (1.79). There are $\prod_{i=1}^{p} p_i$ of those combinations which leads generally to very few observations in the single segments; and thus, the estimation of the probabilities would be very inaccurate. Therefore, we alter this expression

$$\text{odds}(\boldsymbol{x}) = \frac{|B_{\boldsymbol{x}}|}{|G_{\boldsymbol{x}}|} = \frac{|B|}{|G|} \frac{\frac{|B_{\boldsymbol{x}}|}{|B|}}{\frac{|G_{\boldsymbol{x}}|}{|G|}}. \tag{1.82}$$

As the expression $\frac{|B_{\boldsymbol{x}}|}{|B|}$ can be interpreted as an empirical estimate of the probability that a defaulted client will have the vector of characteristics $\boldsymbol{x}$, we can under the assumption of independence of predictors rewrite this probability as the product of individual probabilities for individual predictors. This assumption can be in practice sometimes difficult to fulfill (especially with a higher number of predictors); and therefore, we often choose more robust models instead. We can write:

$$\frac{|B_{\boldsymbol{x}}|}{|B|} = \prod_{(i,j) \in Z} \left(\frac{|B_j^i|}{|B|}\right)^{x_j^i}. \tag{1.83}$$

This means that we compute the product for all categories where $x_j^i = 1$ (i.e. all relevant categories). In the same spirit we can express $\frac{|G_{\boldsymbol{x}}|}{|G|}$ and substitute into (1.82). Thus we get

$$\text{odds}(\boldsymbol{x}) = \frac{|B|}{|G|} \frac{\prod_{(i,j) \in Z} \left(\frac{|B_j^i|}{|B|}\right)^{x_j^i}}{\prod_{(i,j) \in Z} \left(\frac{|G_j^i|}{|G|}\right)^{x_j^i}} = \frac{|B|}{|G|} \prod_{(i,j) \in Z} \left(\frac{\frac{|B_j^i|}{|G_j^i|}}{\frac{|B|}{|G|}}\right)^{x_j^i}. \tag{1.84}$$

Finally, using the introduced notation we get the estimate of odds($\boldsymbol{x}$) in the form

$$\text{odds}(\boldsymbol{x}) = \text{odds} \prod_{(i,j)\in Z} \left(\frac{\text{odds}_j^i}{\text{odds}} \right)^{x_j^i} = \text{odds} \prod_{(i,j)\in Z} (\text{OR}_j^i)^{x_j^i}. \tag{1.85}$$

This expression, together with the assumption of independence of predictors, forms the basics of the *Independence model*.

1.10.3 Independence Model

The *Independence model* is the simplest from the three introduced scoring models. The scoring function is based only on the computed values of odds and OR_j^i and can be according to the previous theory represented in the following way

$$S^{IM}(\boldsymbol{x}) = \text{odds} \prod_{(i,j)\in Z} (\text{OR}_j^i)^{x_j^i}, \tag{1.86}$$

where $\boldsymbol{x} = \left(x_j^i : (i,j) \in Z\right)$ is the set of dummy variables representing the client. Sometimes also a logarithm of this function is used as a scoring function – in this case the score value corresponds to logit($\boldsymbol{x}$),

$$\ln\left\{S^{IM}(x)\right\} = \ln\{\text{odds}\} + \sum_{(i,j)\in Z} x_j^i \ln\left\{\text{OR}_j^i\right\}. \tag{1.87}$$

The main disadvantage of this model is the assumption of independence and the fact that all categories have the same weights. Even though the assumption of independence is in practice seldom completely fulfilled, this model is for its simplicity often used (especially with a low number of predictors which are not much dependent). There are two more models generalizing this approach by adding a non-negative weight to each variable or even to each category.

1.10.4 WOE Model

Another approach to model the probability of default is the *WOE model* (for *Weight of Evidence*) as a generalization of the function (1.86), where to all variables we assign a weight according to their statistical importance in contribution to the final fit. This way we get the scoring function in the form

$$S^{WOE}(\boldsymbol{x}, \boldsymbol{\lambda}) = \text{odds} \prod_{(i,j)\in Z} (\text{OR}_j^i)^{\lambda^i x_j^i}, \tag{1.88}$$

where $\boldsymbol{x} = \left(x_j^i : (i,j) \in Z\right)$ is again the set of predictors and $\boldsymbol{\lambda} = \left(\lambda^i : i \in \{1,\dots,s\}\right)$ is a vector of parameters (weights) for individual predictors.

Again, the scoring function (1.88) is the estimation of the function odds($\boldsymbol{x}$). Thus, for logit($\boldsymbol{x}$) we get a logarithm of the scoring function in the form

$$\ln\left\{S^{WOE}(\boldsymbol{x},\boldsymbol{\lambda})\right\} = \ln\{\text{odds}\} + \sum_{(i,j)\in Z} \lambda^i x_j^i \ln\left\{\text{OR}_j^i\right\}. \tag{1.89}$$

From this form of logit($\boldsymbol{x}$) we can then estimate the vector of parameters $\boldsymbol{\lambda} = \left(\lambda^i : i \in \{1,\dots,s\}\right)$ using the logistic regression introduced in previous chapters.

This model is computationally more difficult than the Independence model but especially for the databases with higher number of defaults it can provide more precise estimates of probabilities of defaults. the WOE model is suitable for databases with at least 150 defaults.

1.10.5 Full Logistic Model

Finally, in the *Full logistic model* we put a certain weight to each category of the categorical variables (i.e. to each dummy variable). The scoring function is then in the form

$$S^{FLM}(\boldsymbol{x},\boldsymbol{\lambda}) = \text{odds} \prod_{(i,j)\in Z} (\text{OR}_j^i)^{\lambda_j^i x_j^i}, \tag{1.90}$$

where $\boldsymbol{x} = \left(x_j^i : (i,j) \in Z\right)$ is again the set of predictors and $\boldsymbol{\lambda} = \left(\lambda_j^i : (i,j) \in Z\right)$ is a vector of parameters for all the individual categories of predictors.

These parameters we again estimate using the logistic regression from the form for logit($\boldsymbol{x}$), i.e. from the logarithm of the scoring function,

$$\ln\left\{S^{FLM}(\boldsymbol{x},\boldsymbol{\lambda})\right\} = \ln\{\text{odds}\} + \sum_{(i,j)\in Z} \lambda_j^i x_j^i \ln\left\{\text{OR}_j^i\right\}. \tag{1.91}$$

This model is the most flexible but also the most complicated from the three introduced models. The full logistic model is suitable for databases with at least 1200 defaults. From the form (1.91) we can see that due to the used logit function this approach is equivalent to the logistic regression model introduced in previous chapters.

Chapter 2

Client Scoring Procedure

2.1 Determination of Good and Bad Clients

The input set of clients entering the scoring function model consists of clients having information $X_k^r, k = 1, \ldots, n$ during period r and who are able to be determined as "good" or "bad" at the end of the period. The definition of a good and a bad client tends to differ across models and it is very important to take the definition into account and examine it thoroughly when constructing models. Too strict definition of a bad client resulting in too many borderline clients being included amongst them lessens the scoring models' effectiveness and means we are not making proper use of the tools they offer us. We are discarding clients as bad too soon, generalizing the whole group. On the other hand, less strict classification means we categorize even borderline clients as good during scoring function development, with the resulting good scores carrying the risk of making deals with these borderline clients. We can, of course, combine the bounds of the definition, use a stricter definition for the function development and softer bounds for the following deliberations, but in every case it is important to properly examine the bounds used, especially from the angle of a potential possibility of once a bad client becoming a good one again. A commonly used bound to define a good and a bad client is the inability to pay bank installments within an interval of thirty to ninety days.

2.2 Good Client at the Start of the Period

We have a set of clients $K = 1, \ldots, n$ with information X_k^r about them, and at the end of period r information Y_k^r exists, with $Y_k^r, k = 1, \ldots, n$ being alternatively distributed random variables with parameter $\pi(X_k^r)$. It is obvious, yet important, that while information X_k^r is tied to the start of the

examined period, client's quality Y_k^r is evaluated at the end of the examined period. Only clients determined as good at the start of the period enter the model, with a caveat that the definition of a "good" and a "bad" client can differ between the start and the end of the examined period. Additionally, the goodness of a client is further assured nowadays by the credit register systems supplying online information about the "good" and "bad" definitions based on mutually shared bank information about non-paying subjects before the deal is sealed. With the help of these registers, client's affordability to pay installments as well as total credit amount of all loans shall be considered to prevent over-indebtedness.

2.3 Logistic Regression Tool for Scoring

Based on mentioned data, we construct a logistic regression, using the recursive or gradually widening regression model we estimate a suitable model and its parameter values, perform statistical checks for the model significance, outlying observations, calculate Hosmer-Lemeshow statistics, calculate R^2. Then we estimate the values of the distribution functions F^G and F^B, Gini coefficient and Lift. It is always a good idea to develop multiple models to compare their results and improve the safety of the final selection. It is advisable to consult the final model's parameters with financial analysis experts. The use of a logistic regression is a common and controversial problem, because statistically-wise, it offers better information concerning all the mentioned characteristics, but the variable coefficients' signs may have the opposite economic meaning than usual. This can be caused by the correlation between the individual variables, and such a function can really be statistically better. Moreover, such problems can arise from the so called reject inference scenarios, where the current population observable for scoring is determined by the previous scoring function. However, problems with interpretation for concrete clients having "weird" results for specific business cases shall trump the statistical advantage. This problem manifests mainly in cases of bank clients with history whose critical indicator improves, with trader and analytic taking this as a clear and objective sign of the client situation's improvement, but the scoring function valuing him less favorably. Because this problem can challenge the validity of a model that is generally very beneficial for the bank, we recommend using models that do not suffer from such problems.

2.4 Development and Reference Sample

Another very important moment in the development of a model is to develop on a development sample and test on a reference sample. We are talking mainly about selection of the variables. If we want to use the scoring function to predict clients' quality, we need to test its forecasting function. This is why we use the method of gradual model widening or a recursive regression to select the parameters on a randomly selected subset of clients (its size is usually half of all the clients). We estimate the model parameters and test the Gini coefficient values and other statistics for both the development and reference samples. In case of choosing between alternate models, we need to favor the models with high enough Gini coefficients on the reference sample. Moreover, out-of-time samples should be considered to evaluate scoring model quality evolution as time progresses forward.

2.5 Determining Risk Grades

Based on an arbitrarily constructed function s we need to set the bank's strategy towards closing deals. Available strategies are:

- We will offer deals to everyone, but good clients will cover the defaults of bad clients. This results in a reduction of the bank's competitiveness towards good clients. The bank offers too high rate and the competition steals those clients. This is why this model is not suitable even though it is very simple.

- We set a limit for the scoring function value, and if the examined client surpasses it, we will not offer him a credit. For granted credits we again assume that good clients have to cover the losses from bad deals. This results in the same problem as above, but with the difference that the final rate does not have to be as markedly unfavorable as in the previous model. This model is sometimes accepted for private persons loans.

- We set multiple risk grades. We will offer deals in all the grades, but within each the good clients have to cover for the bad clients of the same grade. This model sufficiently covers the problem of a competitive rate, especially towards the creditworthy clients. On the other hand, the worst zone has a problem that the risk rate meant to cover the losses is too high. This leads to the better clients of this zone either not accepting the rate or having problems with repayment, while the

worse clients in this zone accept the rate without a second thought, because they know they will have problems with paying the installments anyway, and the bank loan will only help them to postpone a problem and potentially carry out a credit fraud. This leads then to a so-called adverse selection, where the credits are more likely to be accepted by the worse clients and the real risk rate is therefore higher than the estimates based on the past.

- We set multiple risk grades with an assumption that we will not offer credits in the worst one. In the rest we set the risk premiums in a way so that the installments of the good clients cover the losses from the bad clients. This model is the most correct one and when being careful with setting the risk grades, it can offer the bank a substantial competitive edge while respecting the minimization of losses. Even in this model though we need to take into account the fact that the bank is losing a large percentage of potentially good clients in the worst zone, but their probability of being unable to pay installments is so high that the bank cannot risk this insolvency. This is a psychological barrier that the banks and especially the management of their business departments have to be able to overcome and respect.

When we are setting more than one risk grade, we need to specify their boundaries. Even though we can determine the probability of not paying the installments from the scoring function values, another approach is often used. All the clients that are good at the start of the period (i.e. they meet the condition of not being overdue at the start of the period) enter the estimation. We proceed with the increasing score of the individual clients and calculate the probability of not paying the installments $p(x)$ on intervals $[0, x], x \in [0, 1]$. $p(x)$ should be an increasing function in x, which is why the estimate is constructed using the so-called isotonic regression. We will now formalize this base thought. In accordance with the previous designations let us recall that we have K clients with score $s(X_k), k = 1, \ldots, K$. Without loss of generality, we will assume that $s(X_1) \leq s(X_2) \leq \ldots \leq s(X_K)$. The probability estimate of default for clients with score lower than x is reached based on:

$$\underset{p(x):[0,1]\to[0,1],\ \text{non-decreasing}}{\arg\min} \sum_{k=1}^{K} (Y_k - p(s(X_k)))^2 \tag{2.1}$$

The solution to problem (2.1) (let us denote it as p) is unambiguous in respect to points $s(X_k), k = 1, \ldots, K$, which is why we will assume that the function $p(x)$ is linear between those points and we will assume continuous functions.

The solution to problem (2.1) is easily algorithmized, the description of the algorithm is available for example in [1].

The values of x where the most significant turns of the function p happen are the subjective criterion for setting the risk grade bounds. At the same time we have to keep in mind that the number of clients in each grade should be roughly equal. The boundary of the last grade should be set in a way so that the probability of not paying installments in the last active grade does not lead to adverse selection. Here we always have to take into account the current credit market situation, marketability of rates and other business criteria. In case of applying the grade boundaries as an extension to already existing boundaries, we also need to think about the continuity of the probability of not paying installments, so that the probability in the respective grade does not change substantially, and the risk cost with it.

Chapter 3

Machine Learning in Credit Risk

The text in this chapter is mostly based on the tutorial to one of the popular machine learning models, XGBoost, [5]. XGBoost stands for "Extreme Gradient Boosting", where the term "Gradient Boosting" originates from the paper [6].

There are variety of tasks which can be solved by application of machine learning methods:

ML Task	Type of relation	Business task example
Regression	$\mathbb{R}^n \to \mathbb{R}$	Forecasting salary
Binary classification	$\mathbb{R}^n \to \{0, 1\}$	Credit scoring, spam detection
Multi-class classification	$\mathbb{R}^n \to \{0, \ldots, m\}$	Classify support incidents by types
Multi-label classification	$\mathbb{R}^n \to \{c \vert c \in \{0, \ldots, m\}\}$	Email categorization
Ranking	...	Ranking search query results
Clustering	...	Find typical group of payments
Anomaly detection	...	Find exceptional customers, intrusions
Dimension reduction	...	Text annotating
Collaborative filtering	...	Movie recommendation

ML algorithms achieve best results with large datasets (in terms of both number of observations and attributes) by extracting complex relationships from the data. This results in in high accuracy models, but also brings the danger of over-fitting that should be mitigated by applying proper regularization techniques.

In supervised learning problems, we use the training data with multiple features $\boldsymbol{x}_i$ to predict a target variable y_i. Before we learn about XGBoost specifically, let us start by reviewing the basic elements in supervised learning.

3.1 Model and Parameters

The *model* in supervised learning usually refers to the mathematical formula by which the prediction $\hat{y}_i$ is made based on the inputs $\boldsymbol{x}_i$. A common example is a *linear model*, where the prediction is given as $\hat{y}_i = \sum_j \theta_j x_{ij}$, a linear combination of weighted input features.

Different models are designed to solve different problems and based on the model, the prediction can have different interpretations. For example, prediction of logistic regression model, after applying logistic transformation, have the meaning of probability of positive class. Other models can be used for ranking categories or classify observations to multiple classes.

Trainable parameters are the undetermined part that we need to learn from data. In linear regression problems, the parameters are the coefficients $\boldsymbol{\theta}$. We will use $\boldsymbol{\theta}$ to denote the parameters of a model in general. Besides trainable parameters, ML usually contains another set of parameters that must be set before fitting the model. Those are called hyper-parameters. An example of hyper-parameter is maximal depth of decision tree model.

The task of *training* the model consists of finding such a set of trainable parameters $\boldsymbol{\theta}$ that best describes relationship between training data $\boldsymbol{x}_i$ and labels y_i. This is achieved through optimization of *objective function* that measures how well the model fit the training data.

A salient characteristic of objective functions is that they consist of two parts: *training loss* and *regularization term*:

$$\text{obj}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \Omega(\boldsymbol{\theta}) \tag{3.1}$$

where L is the training loss function, and Ω is the regularization term.

The training loss measures how predictive our model is with respect to the training data. A common choice of L in case of linear regression is the mean squared error, which is given by

$$L(\boldsymbol{\theta}) = \sum_i (y_i - \hat{y}_i)^2 \tag{3.2}$$

In logistic regression, commonly used loss function is logistic loss:

$$L(\boldsymbol{\theta}) = \sum_i -y_i \ln(\pi(\boldsymbol{x}_i)) - (1 - y_i) \ln(1 - \pi(\boldsymbol{x}_i)) \tag{3.3}$$

where

$$\begin{aligned} \boldsymbol{\theta} &= (\beta_0, \beta_1, ..., \beta_k) \\ \pi(\boldsymbol{x}_i) &= \frac{\exp\{\beta_0 + \sum_{j=1}^k \beta_j x_{ij}\}}{1 + \exp\{\beta_0 + \sum_{j=1}^k \beta_j x_{ij}\}} \end{aligned} \tag{3.4}$$

Logistic loss was derived in detain in a chapter devoted to logistic regression.

Another possible loss function for binary classification could be Hinge loss:

$$L(\boldsymbol{\theta}) = \sum_i y_i \max(0, 1 - \pi(\boldsymbol{x}_i)) + (1 - y_i) \max(0, \pi(\boldsymbol{x}_i)) \tag{3.5}$$

Note 10. *Sometimes, in machine learning models the binary target y_i is encoded in a form of $\{-1, 1\}$, which allows for simpler notation of loss functions. For instance, Hinge loss is then given by*

$$L(\boldsymbol{\theta}) = \sum_i \max(0, 1 - y_i \pi(\boldsymbol{x}_i)) \tag{3.6}$$

The *regularization term* is what people usually forget to add. The regularization term controls the complexity of the model, which helps us to avoid over-fitting. This might sounds a bit abstract, so let us consider the following problem. You are asked to fit visually a step function given the input data points on the upper left corner figure. Which solution among the three do you think is the best fit?

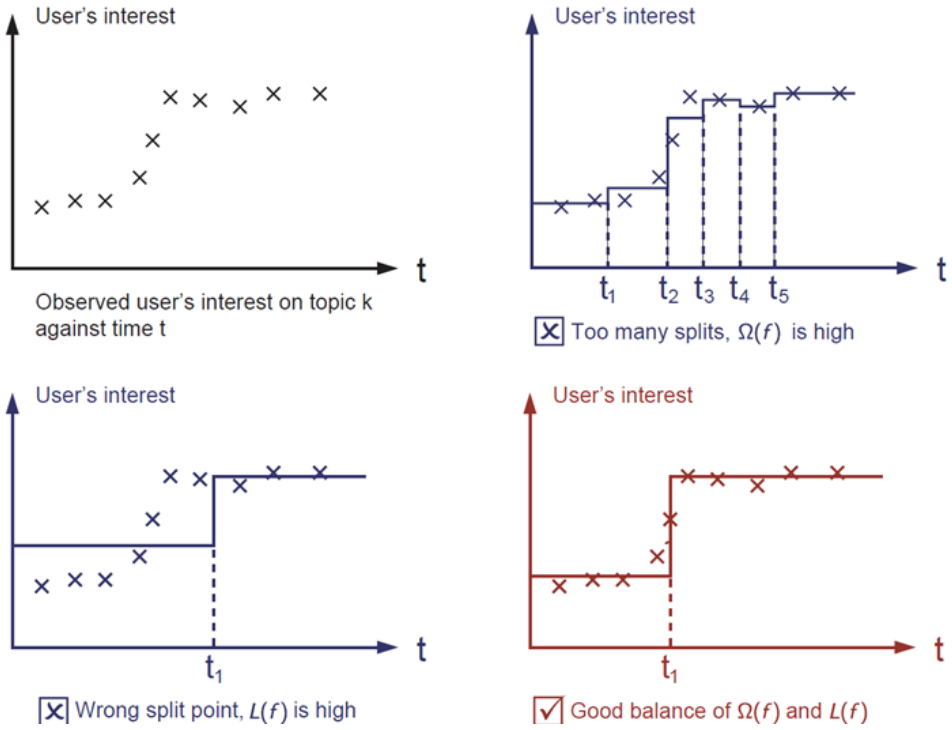

Figure 3.1: Step functions to fit data points, illustrating bias-variance trade-off

The correct answer is marked in red. Please consider if this visually seems a reasonable fit to you. The general principle is that we prefer simple model with good predictive power. Trade-off between predictive power and model simplicity is also referred as *bias-variance trade-off*.

Another problem that can be suppressed with proper regularization is correlated predictors. With correlated predictors, regression coefficients can be unstable when retraining the model on different subsets of your dataset. Suppose we have two predictors x_1 and x_2 with high correlation close to equality. Let's denote logistic function with f. Two following models can provide similar predictions and performance metrics:

$$\begin{aligned} y &\sim f(1x_1 + 3x_2 + \ldots) \\ y &\sim f(2x_1 + 2x_2 + \ldots) \end{aligned} \tag{3.7}$$

In such a case, proper solution would be to make both predictors influence the model in the same way, i.e. $\beta_1 = \beta 2$. Thus predictors' strength is distributed equally between two predictors which is correct solution since both predictors holds almost the same information about the target. In term of regularization it can be achieved by employing either $L1$ or $L2$ regularization.

$$\begin{aligned} \Omega_{L1}(\boldsymbol{\theta}) &= \sum_{i=1}^{n} |\beta_i| \\ \Omega_{L2}(\boldsymbol{\theta}) &= \sum_{i=1}^{n} \beta_i^2 \end{aligned} \tag{3.8}$$

Another possible approach to derive $L2$ regularization is to introduce prior information about the weights in the model. Let's suppose, that without any knowledge about the variables and target, we consider that the weights follow normal distribution. That way, average influence of each parameter is zero and it can have both positive and negative impact on the target. Using Bayes theorem:

$$\mathbb{P}\left[\boldsymbol{\theta}|\boldsymbol{x}, y\right] = \frac{\mathbb{P}\left[\boldsymbol{x}, y|\boldsymbol{\theta}\right]\mathbb{P}\left[\boldsymbol{\theta}\right]}{\mathbb{P}\left[\boldsymbol{x}, y\right]}$$

Now when we try to maximize a posteriori probability estimate, we can ignore the part $\mathbb{P}\left[\boldsymbol{x}, y\right]$ since it is constant with respect to $\boldsymbol{\theta}$. Let's take the prior $\boldsymbol{\theta} \sim N(0, \sigma^2 I)$:

$$\begin{aligned} &\max_{\boldsymbol{\beta}} \ \mathbb{P}\left[\boldsymbol{x}, y|\boldsymbol{\theta}\right]\mathbb{P}\left[\boldsymbol{\theta}\right] \\ &\max_{\boldsymbol{\beta}} \ \log\left\{\mathbb{P}\left[\boldsymbol{x}, y|\boldsymbol{\theta}\right]\mathbb{P}\left[\boldsymbol{\theta}\right]\right\} \\ &\max_{\boldsymbol{\beta}} \ \log\left\{\mathbb{P}\left[\boldsymbol{x}, y|\boldsymbol{\theta}\right]\right\} + \log\left\{\mathbb{P}\left[\boldsymbol{\theta}\right]\right\} \end{aligned} \tag{3.9}$$

The first term in the equation is a clasicall lost part same as we had in 1.3.2. For the second term, we arrive at $L2$ regularization:

$$\begin{aligned} \log\left\{\mathbb{P}\left[\boldsymbol{\theta}\right]\right\} &= \log\left\{\frac{\exp\left\{-\frac{1}{2}\boldsymbol{\beta}^{\top} I \boldsymbol{\beta}\right\}}{\sqrt{(2\pi)^{k}}}\right\} \\ \log\left\{\mathbb{P}\left[\boldsymbol{\theta}\right]\right\} &= \alpha \sum_{i=1}^{N} \beta_i^2 + const. \end{aligned} \tag{3.10}$$

If we would consider Laplacean prior, we would derive $L1$ regularization in a same way.

So, in a modern notation of model training task, we have a loss part which makes model universal (we can select different loss functions to solve different tasks) and regularizer which fights complexity and over-fitting. A regularization power is often controlled by a choice of the regularization coefficient $\alpha > 0$:

$$\text{obj}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + \alpha\Omega(\boldsymbol{\theta}) \tag{3.11}$$

Following table summarizes commonly used model names based on the type of loss and regularizer.

Loss	Regularizer	Model	Task
Logloss	-	Classical logistic regression	Binary classification
Logloss	L2	L2 logistic regression	Binary classification
Logloss	L1	L1 logistic regression	Binary classification
Hinge loss	L2	Support vector machine	Binary classification
Squared error	L2	Ridge	Regression
Squared error	L1	Lasso	Regression
Squared error	L1+L2	Elasticnet	Regression

Note 11. *Regularization term in objective function is not the only way how to regularize a model. Model can be regularized for instance by selection of proper architecture (number of layers in neural networks, maximum depth of decision tree) or by modifying dataset used in training (select randomized subset in each iteration, use only randomized subset of predictors in each iteration).*

3.2 Decision Trees

Before we deep dive in theory behind XGBoost model, let us first explore decision tree models. In XGBoost they are the most commonly used basic building blocks.

There are different types of decision tree models. They differ in number of branches each split can produce or algorithm used to determine optimal split value. We will describe classification and regression trees (CART). Here's a simple example of a CART that classifies whether someone will like a hypothetical computer game X.

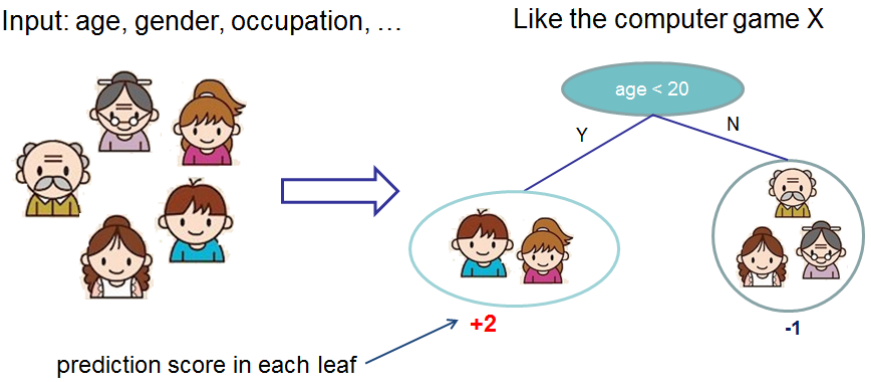

Figure 3.2: A toy example of CART model

We classify the members of a family into different leaves, and each leaf is then assigned with corresponding score. A CART is a bit different from decision trees, in which the leaf only contains decision values. In CART, a real score is associated with each of the leaves, which gives us richer interpretations that go beyond classification. This also allows for a principled, unified approach to optimization, as we will see in a later part of this text.

Definition 12 (Impurity function)**.** *In order to measure the quality of split, we introduce the impurity function:*

- *Becomes minimum at* $p_1 = 0$, $p_2 = 1$ *and* $p_2 = 0$, $p_1 = 1$.
- *Is symmetric:* $l(p_1, p_2) = l(p_2, p_1)$.
- *Becomes maximum at* $p_1 = p_2 = 0.5$.

Some examples of impurity functions include:

- Entropy impurity function: $l_{entropy}(p_1, p_2) = -p_1 \log p_1 - p_2 \log p_2$
- Gini impurity function $l_{gini}(p_1, p_2) = 1 - p_1^2 - p_2^2 = 2p_1p_2$.

We find the ideal splits in decision trees by using brute force, iterating over all possible variables and possible split values (or selected quantiles) and evaluating the "gain" from the split:

$$\text{gain} = l(\text{root}) - \frac{n_{\text{left}}}{n} l(\text{left}) - \frac{n_{\text{right}}}{n} l(\text{right}) \tag{3.12}$$

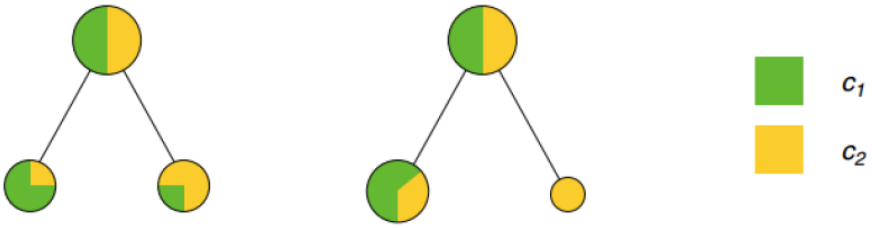

Figure 3.3: Two ways how to make a split in decision tree. Which one is better?

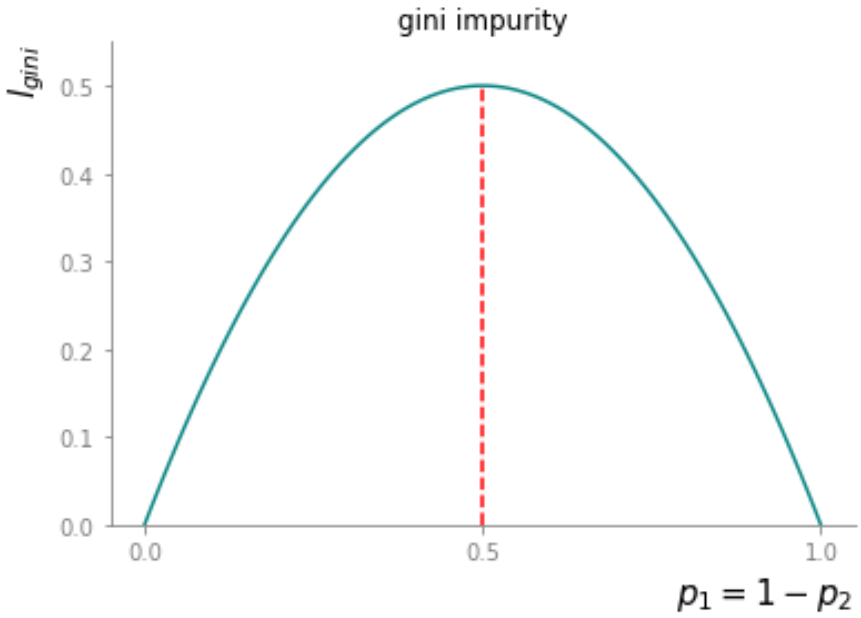

Figure 3.4: Gini impurity function.

It is hard to construct ideal decision tree, since they tend to under-fit or over-fit. On the other hand, decision trees can model non-linear dependencies and they are not sensitive to monotonic transformation of variables. Small decision trees can be also interpreted and well understood.

3.3 Tree ensembles

Usually, a single decision tree is not strong enough to be used in practice. What is actually used is the ensemble model, which sums the prediction of multiple trees together.

In the figure 3.5 is an example of an ensemble of two trees. The prediction

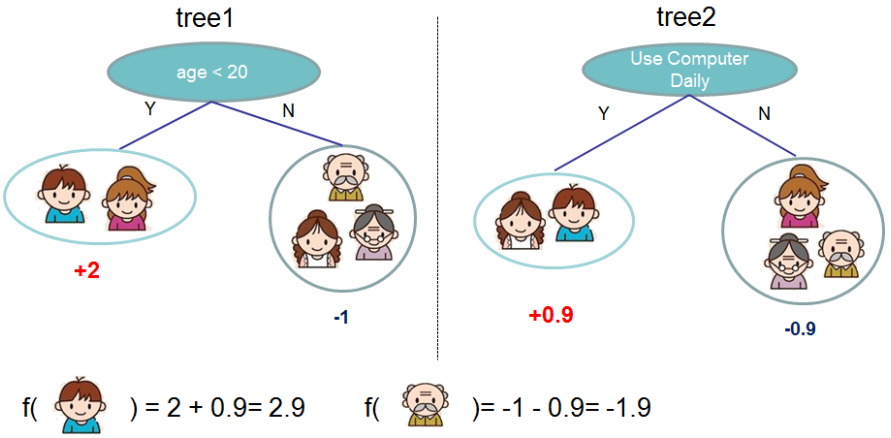

Figure 3.5: A toy example for tree ensemble, consisting of two CARTs

scores of each individual tree are summed up to get the final score. If you look at the example, an important fact is that the two trees try to complement each other.

Let us assume that we have 3 non-correlated binary classifiers and probability of correct answer of each of them is p. What is the probability of correct answer of "voting" classifier?

$$p^3 + 3p^2(1-p) = p^2(3-2p) \tag{3.13}$$

We see that for classifiers with good prediction quality ($p > 0.5$), the probability of correct answer by the ensemble is greater then for each single classifier.

Mathematically, we can write our model in the form

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i), f_k \in \mathcal{F} \tag{3.14}$$

where K is the number of trees, f is a function in the functional space $\mathcal{F}$, and $\mathcal{F}$ is the set of all possible CARTs. The objective function to be optimized is given by

$$\text{obj}(\boldsymbol{\theta}) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega_k(\boldsymbol{\theta}) \tag{3.15}$$

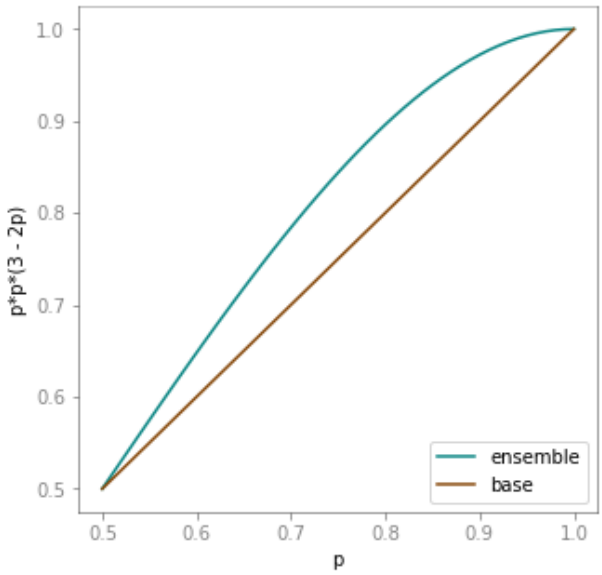

Figure 3.6: Precision of ensemble model with three binary classifiers, each of precision p.

3.4 Random Forest

The idea of training multiple trees independently and combining their prediction using ensembling was introduces in the article [7]. To make the trees more independent, two techniques are used - bagging and random subspaces. Averaging of probabilities or voting of class is used to combine decisions of all the trees. Trees are usually deep (not restricted by depth and minimum number of samples in leave).

Bagging is a technique to prevent over-fitting:

- In each iteration random subsample of given size is selected (Usually 60% for random forest).
- A new decision tree (base classifier) is trained on the subsample only.

Bagging mitigates influence of outliers since they are not present in every subsample - approach emphasizes main patterns.

Similar sampling approach is applied to select random subspaces of predictors to be used in each iteration:

- In each iteration randomly select predictors to be used for base classifier training.

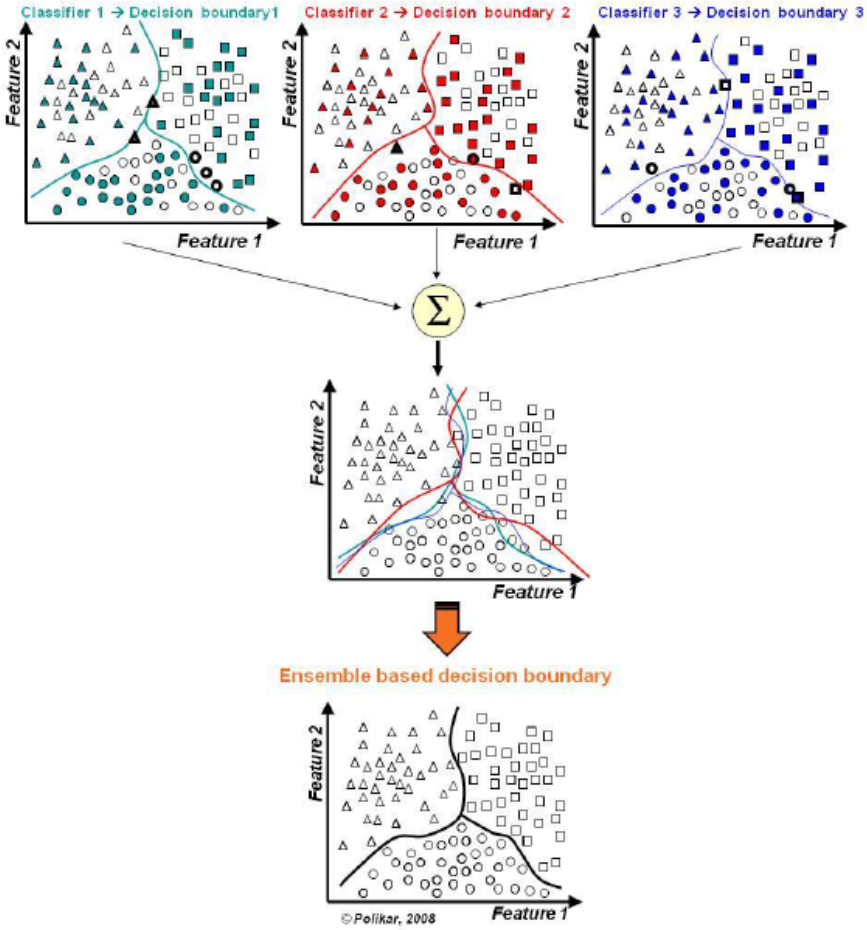

Figure 3.7: Illustration of bagging technique

- Predictive power of final ensemble is spread out amongst multiple predictors (single model would rely on lower number of predictors).

Using random subspaces, base classifiers are less correlated which means that the ensemble should be more efficient.

Random forests are often used for their good precision and robustness with respect to outliers. The training can be easily parallelized and increasing the number of trees does not lead to over-fitting. The method is easy to use, since there are only few hyper-parameters (fraction of predictors used in random subsamples or subsampling ratio in bagging). As we will see in the following section, precision of the prediction can be further improved by making each new decision tree reflects and build on already fitted decision trees.

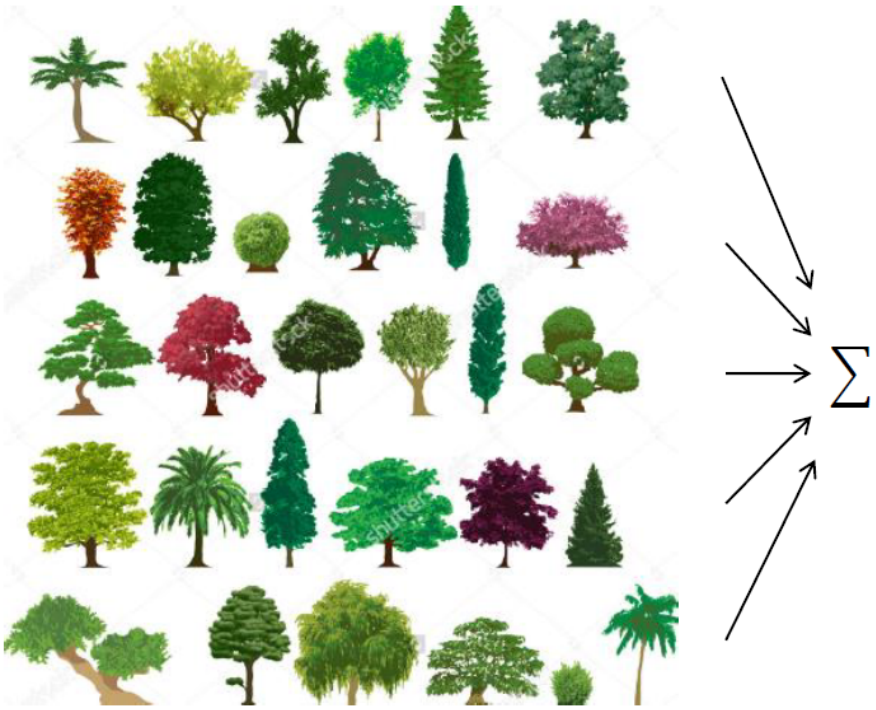

Figure 3.8: Illustration of random subspaces technique

3.5 Gradient Boosting

As introduced in the article [6], gradient boosting algorithms train a new decision tree (or any weak classifier) in each step with respect to the already existing ensemble. Probabilities (or predictions) from activated leaves are summed up for all trees to get the final prediction. Adding a new tree to the sum makes entire model more precise each step of the training process. In other words, in each iteration, boosting is trying to fix errors that remained from previous iterations while not breaking correct predictions of the model.

Gradient boosting models are usually more accurate than random forest and need less trees which are not so deep. However, the training process is not so intuitively parallelizable as random forest and number of trees should be accurately selected to prevent over-fitting. Multiple hyper-parameters should be also carefully selected (tree depth, minimum number of observations in leave, shrinkage, etc.), otherwise gradient boosting can easily over-fit.

Gradient descent is a method for finding minimum of complex functions:

- In each iteration we calculate the derivatives of the objective function with respect to model trainable parameters (gradient).

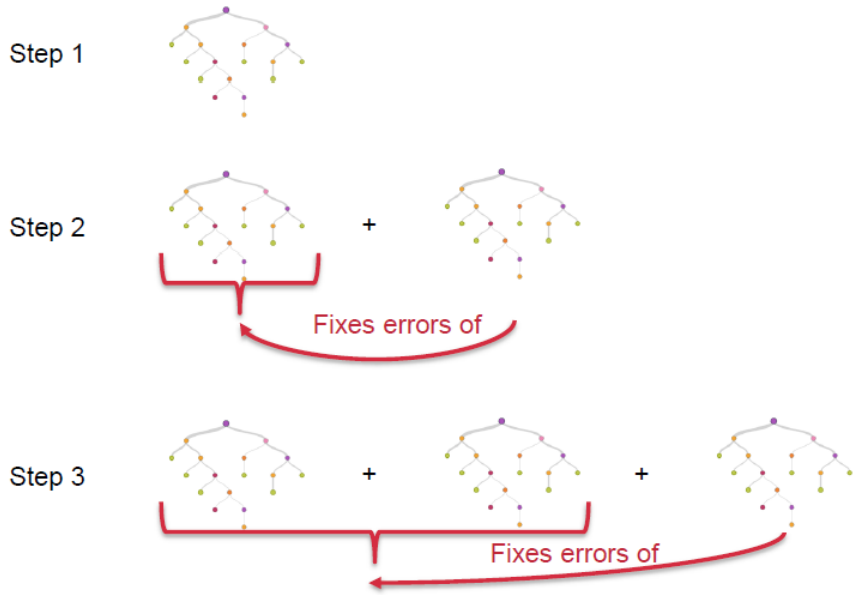

Figure 3.9: General boosting idea.

- Usually, learning rate lower than 1 is used. The gradient is multiplied by learning rate to be more conservative.
- Neural networks utilizes more advanced algorithms, that can modify learning rate in time.

Let's start with an example from [8]. We have following dataset and would like to predict target variable weight.

Height(m)	Favourite Color	Gender	Weight(kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73

Before we start gradient boosting, we usually consider average of all obervations as the first prediction. Let's take 73.5 as the average and calculate residuals:

Height(m)	Favourite Color	Gender	Weight(kg)	Prediction 1	Residual 1
1.6	Blue	Male	88	73.5	14.5
1.6	Green	Female	76	73.5	2.5
1.5	Blue	Female	56	73.5	-17.5
1.8	Red	Male	73	73.5	-0.5

Now, the residuals are the new target variable which we want to explain using a decison tree. Let's imagine a very simple decision tree with one split: if height > 1.5 then assign 5.5, else -17.5. In gradient boosting, the second prediction tries to fix the error of all previous predictions, therefore new residuals can be computed as *weight - (prediction 1 + prediciton 2)*.

Height(m)	Favourite Color	Gender	Weight(kg)	Prediction 1	Prediction 2	Residual 2
1.6	Blue	Male	88	73.5	5.5	9
1.6	Green	Female	76	73.5	5.5	-3
1.5	Blue	Female	56	73.5	-17.5	0
1.8	Red	Male	73	73.5	5.5	6

Now we can see that after first prediction, we had sum of absolute residuals of 35 while after the second one only 18, so the ensemble is improving. We can now take *residual 2* as a target variable and build another decision tree and so on.

Let's put this into more exact formulations - using residuals as a target variable is feasible only for continuous regression problems. In logistic regression, it would be tricky to even calculate residual, therefore we focus on general approach to minimize the loss function. In case of gradient boosting models, we apply the gradient descent to minimize the loss function $L = \sum_i^n l(y_i, \hat{y}_i)$. Let's consider finding the prediction iteratively, constructing a series of predictions $\hat{y}_i^{(m)}, n = 1, \ldots, M$. In m-th iteration of gradient descent, we have:

$$\begin{aligned} \hat{y}_i^{(m)} &= \hat{y}_i^{(m-1)} - b_m \nabla l \\ \nabla l &= \left[\frac{\partial l}{\partial \hat{y}_i^{(m-1)}} (\boldsymbol{x}_i) \right]_{i=1}^{N} = \left[\frac{\partial l \left(y_i, \hat{y}_i^{(m-1)} \right)}{\partial \hat{y}_i^{(m-1)}} (\boldsymbol{x}_i) \right]_{i=1}^{N} \end{aligned} \tag{3.16}$$

Similarly to our example above, we cannot use the gradient of the loss function directly. When using residuals, we have used them to define the next decision tree, which can be then used generally to any observations outside of the training set. With gradient of loss function, we just train a model $f_m(\boldsymbol{x}_i)$ with ∇l as a target. ∇l gives us the direction in which we should move. To determine the size of our step, we can use fastest descent through minimization of l via linear search:

$$b_m = \underset{b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{N} l \left(y_i, \hat{y}_i^{(m-1)} - b f_m(\boldsymbol{x}_i) \right) \tag{3.17}$$

Usually, to prevent bouncing, shrinkage parameter $\nu \in (0, 1]$ is added:

$$\hat{y}_i^{(m)} = \hat{y}_i^{(m-1)} - \nu b_m f_m(\boldsymbol{x}_i) \tag{3.18}$$

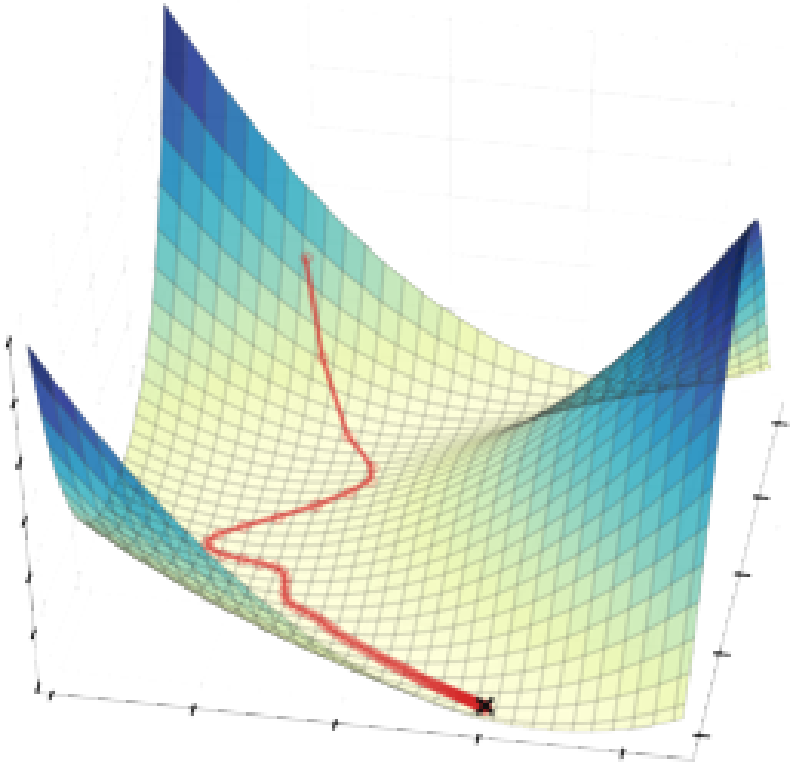

Figure 3.10: Gradient descent.

Gradient boosting of decision trees (GBDT) are widely used, functions f_m are decision trees. Usual hyper-parameters of each tree are: max tree depth, max children in leaf, etc. Few of common boosting algorithms:

- LS-boosting (regression): $l(y_i, \hat{y}_i) = \frac{(y_i - \hat{y}_i)^2}{2}$
- LAD-boosting (regression): $l(y_i, \hat{y}_i) = |y_i - \hat{y}_i|$
- AdaBoost (classification): $l(y_i, \hat{y}_i) = \exp\{-y_i\hat{y}_i - (1 - y_i)\hat{y}_i\}$
- LogitBoost (classification): $l(y_i, \hat{y}_i) = \log\{1 + \exp\{-2y_i\hat{y}_i - 2(1 - y_i)\hat{y}_i\}\}$

Let's now calculate the gradient for LS-boosting:

$$\frac{\partial l}{\partial \hat{y}_i} = \frac{\partial \frac{(y_i - \hat{y}_i)^2}{2}}{\partial \hat{y}_i} = y_i - \hat{y}_i$$

We can see that the general approach with loss function simplifies to the procedure with residuals shown in our simple example.

3.5.1 Extreme Gradient Boosting

XGBoost (Extreme Gradient Boosting) is one of mostly used boosting implementations today (the other very commonly used implementation is LightGBM). XGBoost comes with many regularization features (bagging, random subspaces, $L2$-regularization, etc.). XGBoost algorithm gained its fame mostly through Kaggle competitions. Features of XGBoost are:

- Using model ensembles to achieve best performance.
- Base classifiers can be either tree based models or linear models.
- XGBoost can be used to predict probability of event, number of events (Poisson), continuous target.
- By our experience, usually it is the best algorithm for structured (tabular-like) data.

Nowadays, Python is the most favourite programming language for data science. XGBoost has API in several other languages like R, C++ or Julia, but we will focus on working with XGBoost in Python. XGBoost documentation can be found at https://xgboost.readthedocs.io/en/latest/.

3.5.2 Training the model

Let's now focus on how to train the XGBoost model. Let the following be the objective function (remember it always needs to contain training loss and regularization):

$$\text{obj} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \tag{3.19}$$

In further text, base classifiers f_i will be decision trees (they are the most often choice).

Training a decision tree consists of two parts - deriving a tree structure and assigning a score to each leaf. Trainable parameters in XGBoost model are leaves' scores. Those can be set based on gradient of objective function. Structure of the tree is solved by brute force (as mentioned before). Final model prediction uses additive strategy, at step t the prediction $\hat{y}_i^{(t)}$ is given by:

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(\boldsymbol{x}_i) = \hat{y}_i^{(0)} + f_1(\boldsymbol{x}_i) \\
\hat{y}_i^{(2)} &= f_1(\boldsymbol{x}_i) + f_2(\boldsymbol{x}_i) = \hat{y}_i^{(1)} + f_2(\boldsymbol{x}_i) \\
&\cdots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(\boldsymbol{x}_i) = \hat{y}_i^{(t-1)} + f_t(\boldsymbol{x}_i)
\end{aligned}
\tag{3.20}
$$

It remains to ask: which tree do we want at each step? A natural thing is to add the one that optimizes our objective.

$$
\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\boldsymbol{x}_i)) + \Omega(f_t) + \text{constant}
\end{aligned}
\tag{3.21}
$$

If we consider using mean squared error (MSE) as our loss function, the objective becomes

$$
\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(\boldsymbol{x}_i)))^2 + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} [2(\hat{y}_i^{(t-1)} - y_i) f_t(\boldsymbol{x}_i) + f_t(\boldsymbol{x}_i)^2] + \Omega(f_t) + \text{constant}
\end{aligned}
\tag{3.22}
$$

The form of MSE is friendly, with a first order term (usually called the residual) and a quadratic term. For other losses of interest (for example, logistic loss), it is not so easy to get such a nice form. So in the general case, we take the Taylor expansion of the loss function up to the second order:

$$
\text{obj}^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i)] + \Omega(f_t) + \text{constant} \tag{3.23}
$$

where the g_i and h_i are defined as

$$
\begin{aligned}
g_i &= \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \\
h_i &= \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})
\end{aligned}
\tag{3.24}
$$

After we remove all the constants, the remaining part of objective function that enters minimization at step t becomes

$$\sum_{i=1}^{n}[g_i f_t(\boldsymbol{x}_i) + \frac{1}{2}h_i f_t^2(\boldsymbol{x}_i)] + \Omega(f_t) \tag{3.25}$$

This becomes our optimization goal for the new tree. One important advantage of this definition is that the value of the objective function only depends on g_i and h_i. This is how XGBoost supports custom loss functions - they need to provide gradient and Hessian. We can optimize every loss function, including logistic regression, using exactly the same solver that takes g_i and h_i as input!

3.5.3 Model Complexity

We have introduced the training step, but wait, there is one important thing, the *regularization term*! We need to define the complexity of the tree $\Omega(f)$. In order to do so, let us first refine the definition of the tree $f(x)$ as

$$f_t(x) = w_{q(x)}, w \in R^T, q : R^d \to \{1, 2, \cdots, T\}. \tag{3.26}$$

Here w is the vector of scores on leaves, q is a function assigning each data point to the corresponding leaf, and T is the number of leaves. In XGBoost, we define the complexity as

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.27}$$

Of course, there is more than one way to define the complexity, but this one works well in practice. The regularization is one part most tree packages treat less carefully, or simply ignore. This was because the traditional treatment of tree learning only emphasized improving impurity, while the complexity control was left to heuristics. By defining it formally, we can get a better idea of what we are learning and obtain models that perform well in the wild.

3.5.4 Structure score

Here is the magical part of the derivation. After re-formulating the tree model, we can write the objective value with the t-th tree as:

$$\begin{aligned}\text{obj}^{(t)} &\approx \sum_{i=1}^{n}[g_i w_{q(\boldsymbol{x}_i)} + \frac{1}{2}h_i w^2_{q(\boldsymbol{x}_i)}] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T\end{aligned} \tag{3.28}$$

where $I_j = \{i|q(\boldsymbol{x}_i) = j\}$ is the set of indices of data points assigned to the j-th leaf. Notice that in the second line we have changed the index of the summation because all the data points on the same leaf get the same score. We could further compress the expression by defining $G_j = \sum_{i\in I_j} g_i$ and $H_j = \sum_{i\in I_j} h_i$:

$$\text{obj}^{(t)} = \sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T \tag{3.29}$$

In this equation, w_j are independent with respect to each other (optimal value of w_j can be found without any regard to other weights $w_i, \forall i \neq j$), the form $G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2$ is quadratic and the best w_j for a given structure $q(x)$ and the best objective reduction we can get are:

$$\begin{aligned}w_j^* &= -\frac{G_j}{H_j + \lambda} \\ \text{obj}^* &= -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T\end{aligned} \tag{3.30}$$

The last equation measures how good a tree structure $q(x)$ is.

If all this sounds a bit complicated, let's take a look at the picture, and see how the scores can be calculated. Basically, for a given tree structure, we push the statistics g_i and h_i to the leaves they belong to, sum the statistics together, and use the formula to calculate how good the tree is. This score is like the impurity measure in a decision tree, except that it also takes the model complexity into account.

3.5.5 Learn the tree structure

Now that we have a way to measure how good a tree is, ideally we would evaluate all possible trees and pick the best one. In practice this is intractable,

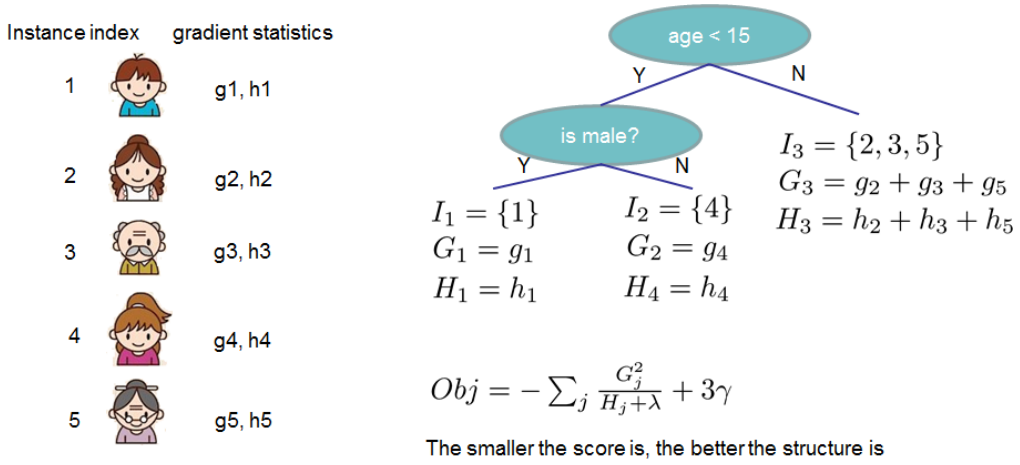

Figure 3.11: Illustration of structure score (fitness).

so we will try to optimize one level of the tree at a time. Specifically, we try to split a leaf into two leaves. Reduction in objective function is represented by gain:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma \tag{3.31}$$

This formula can be decomposed as 1) the score on the new left leaf, 2) the score on the new right leaf, 3) The score on the original leaf, 4) regularization on the additional leaf. We can see an important fact here - if the gain is smaller than γ, we would do better not to add that split. This is exactly the *pruning* technique in tree based models! By using the principles of supervised learning, we can naturally come up with the reason these techniques work.

For real valued data, we usually want to search for an optimal split. To efficiently do so, we place all the instances in sorted order, like at the Fig. 3.5.5 (persons are sorted by age).

A left to right scan is sufficient to calculate the structure score of all possible split solutions, and we can find the best split efficiently (once split position is changed, G_L and G_R can be updated only by affected observations that were moved from left leaf to right leaf, thus saving computational power).

Since it is intractable to enumerate all possible tree structures, we add one split at a time. This approach works well most of the time, but there are some edge cases that fail due to this approach. For those edge cases, training results in a degenerate model because we consider only one feature dimension at a time.

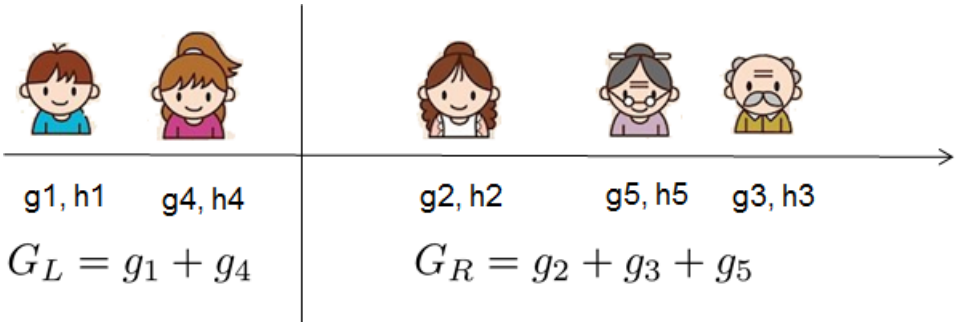

Figure 3.12: Schematic of choosing the best split

3.6 Data preprocessing

XGBoost expects to process numerical data and thus we need to encode categorical features with numerical values first. In fact, XGBoost implementation called catboost is capable of processing categorical data by encoding categorical features automatically, but other XGBoost implementations require numerical inputs.

3.6.1 Encoding categorical features

First option how to encode categorical features is label encoding, where for each category some number (integer) is assigned.

Obs ID	Color
1	Red
2	Green
3	Blue
4	Red

Label encoding →

Obs ID	Color
1	1
2	2
3	3
4	1

This encoding is useful when the categories can be sorted from the best to worst, for example company rating ('AA', 'B+', etc.). If there is no clear ordering, then by assigning a numerical value to each category we might influence the splitting process in negative way - it might be impossible to find split value that separate good and bad categories.

Alternative approach for encoding categorical features is dummy encoding sometimes also called one-hot-encoding. For each category, we introduce a new variable and items in this category receive value of 1, others get 0.

Obs ID	Color
1	Red
2	Green
3	Blue
4	Red

Dummy encoding →

Obs ID	Red	Green	Blue
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0

Another interesting technique is called mean target encoding. It is especially useful for variables with high number of categories. In this case, the number assigned to each category represents the default (target) rate of the category itself. To make the calculation more stable with respect to categories with low number of observations, coefficient α is introduced which balances the averaging between category default rate and common default rate. A rule of thumb could be that for variables with less than 10 categories dummy encoding can be used, mean target encoding for the rest.

$$MTE_{cat} = \frac{f_{cat} \cdot avg_{cat}(target) + \alpha \cdot avg(target)}{f_{cat} + \alpha} \tag{3.32}$$

f_{cat} stands for relative share of observations in given category. Note that if $f_{cat} = \alpha$, mean target encoding of given category is just simple average of category default rate ($avg_{cat}(target)$) and average default rate in the whole data sample ($avg(target)$), because metrics enter weighted average with the same weight. As category frequency increases, mean target encoding is closer to category default rate. For α, reasonable value based on our experience might be 0.01. This means the same weights in 3.32 in case category is populated with 1% of all observations.

3.6.2 Treating missing values

In case of logistic regression, missing values need to be replaced with proper value. When continuous features are binned to several groups (grouping is often utilized in credit risk models), null value can form a separate group. Groups are then usually encoded, for instance, with weight of evidence approach. In case of using continuous features in a model, we need to deal with nulls other way.

In XGBoost algorithm missing values are treated automatically:

1. Make a split using non-missing values only.
2. Calculate target rate in right and left branch using non-missing values only.

3. Calculate target rate for population with missing values.

4. Assign missing values population to right or left based on lower difference in target rates.

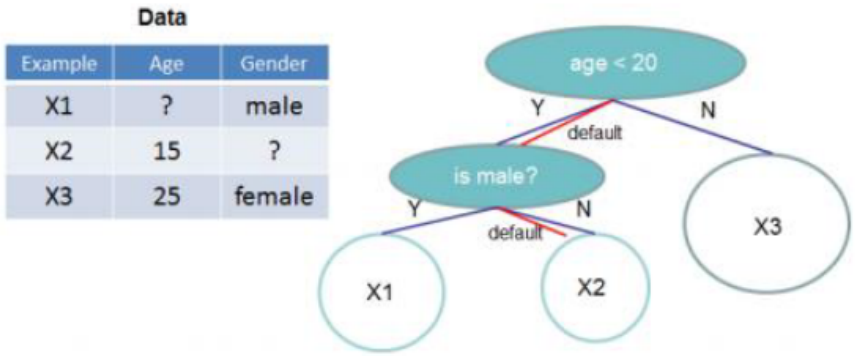

Figure 3.13: Treating missing values.

3.7 Over-fitting

As the XGBoost model training progresses, loss function on training set is improving. However, at some point the algorithm starts to learn to explain a noise on training sample and learnt patterns are not transferable to independent samples (test, valid). Since then performance on independent sample is decreasing - the model is over-fitting.

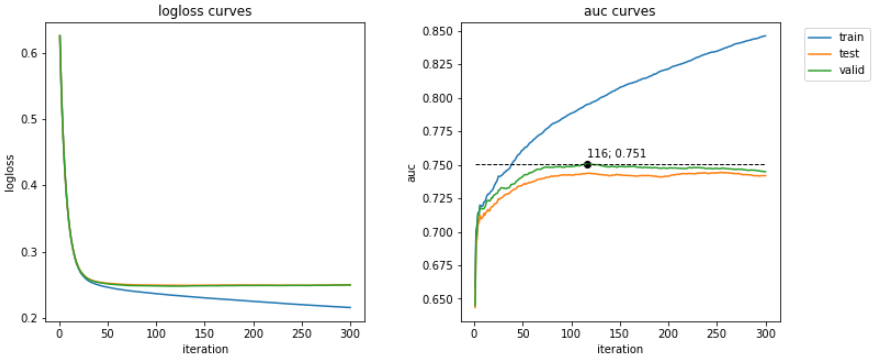

Figure 3.14: Training curves for XGBoost.

Since validation sample is usually used for deciding when to stop training (once performance on validation sample stops improving), it is actually not an

independent sample. Therefore, another sample (test) is often introduced in ML model development. When producing the samples, *stratification* is a good technique to ensure that the data in each sample have similar distribution.

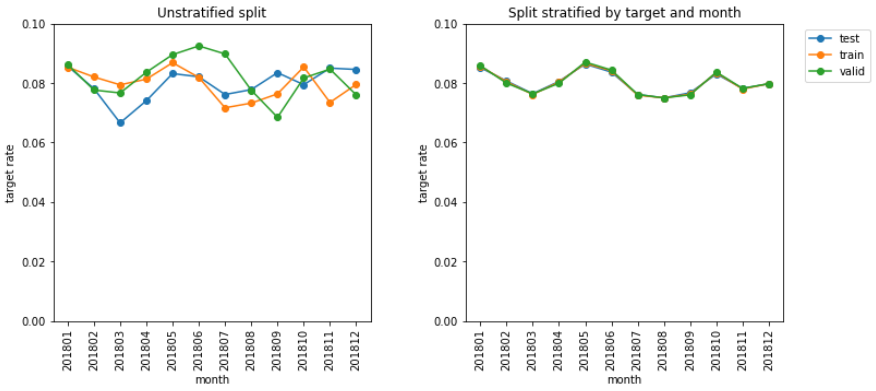

Figure 3.15: Sample stratification

Measured performance on training sample is misleading, because of overfitting. But if we have smaller dataset, we might want to use as muchobservations for training as possible. *Cross-validation* is a technique that allows us to use all the data for training and to effectively assess expected performance of the model.

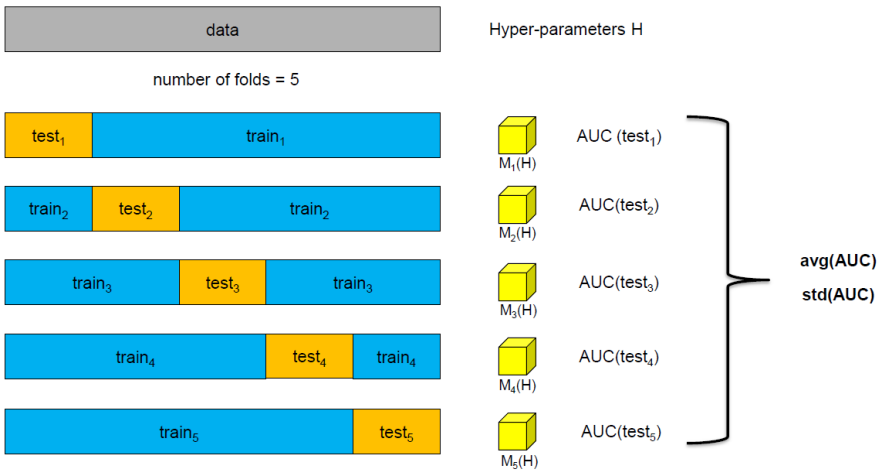

Figure 3.16: Cross validation technique.

N-fold cross-validation divides the data set randomly into N parts. Each of these parts is then selected as a test set for one iteration and rest of the data is marked as train set. The algorithm trains N-models and returns

performance on test fold on each of them. Average performance on test fold (for example, AUC) gives the estimation of the overall performance. Standard deviation of the quality metric gives use information about how robust and reliable this value is.

Cross-validation score is a way how to extract not over-fitted prediction for the whole dataset (even though the data was used for training), we use only the scores produced on the test sets by N different models and combine them together.

Since we have built N models, usually there is a question which one of them should be deployed. Common solution is to train a single new model on the whole data set and deploy it. Under the assumption that performance metrics are stable between the folds and that score distributions are similar for all the N models, it is reasonable to suppose that the single model on all data should provide similar or better performance. Note that the expected performance of the model is given by mean performance on test folds from cross-validation.

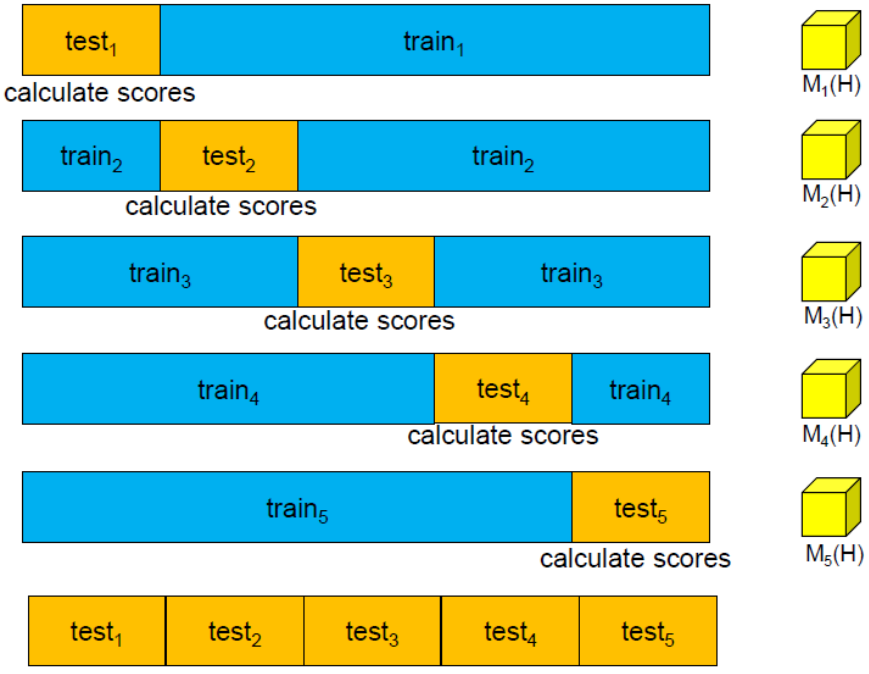

Figure 3.17: Cross validation score

Bootstrapping is an alternative way to obtain confidence interval for performance metric. Bootstrapping uses random sampling with replacement. Let us assume that our data sample has N observations.

1. Split data for training and testing sample $N = N_{train} + N_{test}$.
2. Re-sample training dataset using random sample with replacement (sample N_{train} cases).
3. Train a model on re-sampled training dataset.
4. Measure performance on testing sample.
5. Repeat the procedure M-times.

Now we can estimate average performance on test sample with confidence interval. Such a procedure helps us to understand how the expected model performance is sensitive for selection of data sample used for training.

3.8 Objective functions & hyper-parameters

XGBoost algorithm can handle many more tasks than the prediction of probability of default. Below we will list objective functions used for different modelling tasks.

Binary classification

- binary:logistic

$$\text{obj} = -\frac{1}{N}\sum_{i=1}^{N}[label_i \ln(pred_i) + (1 - label_i)\ln(1 - pred_i)] + \Omega \quad (3.33)$$

prediction: probability of target

- binary:logitraw

$$\text{obj} = -\frac{1}{N}\sum_{i=1}^{N}[label_i \ln(pred_i) + (1 - label_i)\ln(1 - pred_i)] + \Omega \quad (3.34)$$

prediction: logit of probability of target (score)

- binary:hinge

$$\text{obj} = \sum_{i=1}^{N} max(0, 1 - label_i \cdot pred_i) + \Omega \quad (3.35)$$

prediction: binary classification

Regression

- reg:squaredederror

$$\text{obj} = \frac{1}{\sqrt{N}}\sqrt{\sum_{i=1}^{N}(pred_i - label_i)^2} + \Omega \tag{3.36}$$

- reg:squaredlogerror

$$\text{obj} = \frac{1}{\sqrt{N}}\sqrt{\sum_{i=1}^{N}\left(\ln(pred_i + 1) - \ln(label_i + 1)\right)^2} + \Omega \tag{3.37}$$

note: All input labels must be greater than -1.

Multi-class classification

- multi:softmax

$$\text{obj} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} label_{ij} \cdot \ln pred_{ij} + \Omega \tag{3.38}$$

prediction: class with highest probability
note: M models with binary target (denoting each class) are fitted.

- multi:softprob

$$\text{obj} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} label_{ij} \cdot \ln pred_{ij} + \Omega \tag{3.39}$$

prediction: probability of each class
note: M models with binary target (denoting each class) are fitted.

Ranking

- rank:pairwise
- rank:ndcg

Number of occurrences

- count:poisson

$$\text{obj} = \frac{1}{N}\sum_{i=1}^{N}\left(\ln\Gamma(label_i + 1) + pred_i + label_i \cdot \ln pred_i\right) + \Omega \tag{3.40}$$

3.8.1 XGBoost hyper-parameters

When invoking the training process of XGBoost, there are many hyper-parameters that can be set up by the user. Some of them correspond to the nature of the task being solved and some of them can influence the complexity and over-fitting of the model. While many of these parameters can be set expertly, for some of them hyper-parameter optimization techniques are useful. We will discuss such techniques later.

General hyper-parameters

- booster [default=gbtree]
  - Which booster to use. Can be gbtree, gblinear or dart; gbtree and dart use tree based models while gblinear uses linear functions.
- verbosity [default=1]
  - Verbosity of printing messages. Valid values are 0 (silent), 1 (warning), 2 (info), 3 (debug).
- nthread [default to maximum number of threads available if not set]
  - Number of parallel threads used to run XGBoost.

Tree booster hyper-parameters

- eta [default=0.3, alias: learning_rate]
  - Step size shrinkage used in update to prevents over-fitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.
  - range: $[0, 1]$
- gamma [default=0, alias: min_split_loss]
  - Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be.
  - range: $[0, \infty)$
- max_depth [default=6]

- Maximum depth of a tree. Increasing this value will make the model more complex and more likely to over-fit. 0 is only accepted in loss guided growing policy when tree_method is set as hist or gpu_hist and it indicates no limit on depth.
- Beware that XGBoost aggressively consumes memory when training a deep tree.
- range: $[0, \infty)$

- min_child_weight [default=1]
  - Minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning.
  - In linear regression task, this simply corresponds to minimum number of instances needed to be in each node. The larger min_child_weight is, the more conservative the algorithm will be.
  - range: $[0, \infty)$

- max_delta_step [default=0
  - Maximum delta step we allow each leaf output to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative.
  - Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced. Set it to value of 1-10 might help control the update.
  - range: $[0, \infty)$

- subsample [default=0]
  - Subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees. and this will prevent over-fitting. Subsampling will occur once in every boosting iteration.
  - range: $(0, 1]$

- colsample_bytree [default=1]
  - Subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed.

  - range: $(0, 1]$

- colsample_bylevel [default=1]
  - Subsample ratio of columns for each level. Subsampling occurs once for every new depth level reached in a tree. Columns are subsampled from the set of columns chosen for the current tree.
  - range: $(0, 1]$

- colsample_bynode [default=1]
  - Subsample ratio of columns for each node (split). Subsampling occurs once every time a new split is evaluated. Columns are subsampled from the set of columns chosen for the current level.
  - range: $(0, 1]$

- lambda [default=1, alias: reg_lambda]
  - L2 regularization term on weights. Increasing this value will make model more conservative.

- alpha [default=1, alias: reg_alpha]
  - L1 regularization term on weights. Increasing this value will make model more conservative.
  - range: $[0, \infty)$

- tree_method [default=auto]
  - The tree construction algorithm used in XGBoost.
  - available values: [auto, exact, approx, hist, gpu_hist]

- monotone_constraint [default=auto]
  - Constraint of variable monotonicity.

Learning task hyper-parameters

- objective [default=reg:squarederror]
  - Objective function to be minimized.

- base_score [default=0.5]

- The initial prediction score of all instances, global bias

- eval_metric [default according to objective]
  - Evaluation metrics for validation data, a default metric will be assigned according to objective (rmse for regression, and logloss for classification, mean average precision for ranking).
  - User can add multiple evaluation metrics.

- seed [default=0]
  - Random number seed.

One additional and very useful hyper-parameter - early stopping rule. We have discussed that we should stop adding new trees once the performance on validation sample stops improving. However, performance could be quite volatile and it is not reasonable to stop after every drop in AUC on the test sample. Instead, we measure if in the last K iterations there was some improvement in the AUC on the test sample and if not, we stop at that iteration with best AUC.

3.9 Hyper-parameters optimization

If we are not sure about the ideal value of some hyper-parameters, following algorithms can be used for hyper-parameters tuning:

- Grid search.
- Randomized grid search.
- Grid search / randomized grid search with halving.
- Bayesian optimization.
- Gradient based optimization.
- Evolutionary optimization.

Always remember to use cross validation during the optimization process of hyper-parameters.

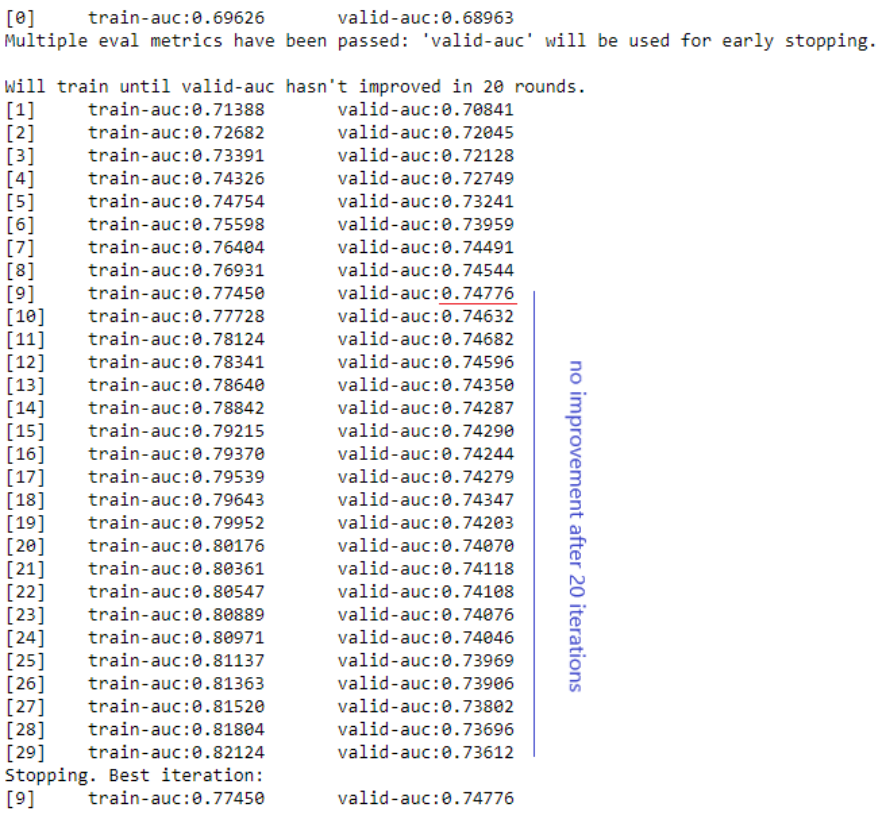

Figure 3.18: Early stopping - algorithm output.

3.9.1 Grid search

Grid search is the basic optimization technique, which searches parameter space by brute force in selected points.

Advantages

+ Non-convex cases are handled well - no convergence to local minima.

Disadvantages

- Time consuming.
- Number of combinations grow exponentially with increasing number of parameters.
- Cannot allocate amount of resources.

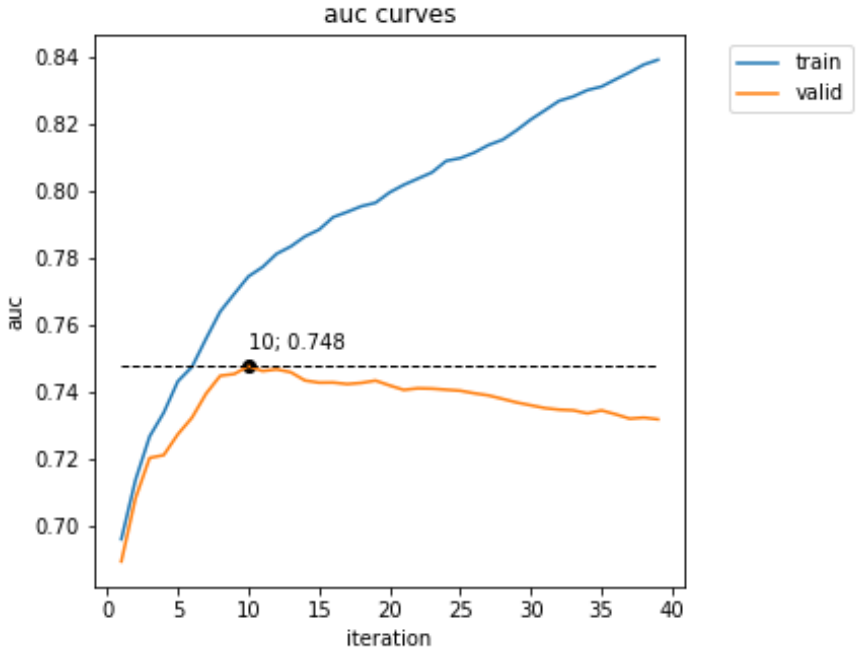

Figure 3.19: Early stopping - evolution of AUC

- Unimportant parameters consume same amount of resources as important parameters.
- Does not reflect already learnt information.

3.9.2 Randomized Grid Search

Randomized Grid search overcomes the curse of dimensionality by randomly selecting points from search space to be investigated.

Advantages

+ Non-convex cases are handled well - no convergence to local minima.
+ Budget can be chosen independently on number of parameters.
+ Unimportant parameters do not negatively effect search process.

Disadvantages

- Time consuming.
- Does not reflect already learnt information.

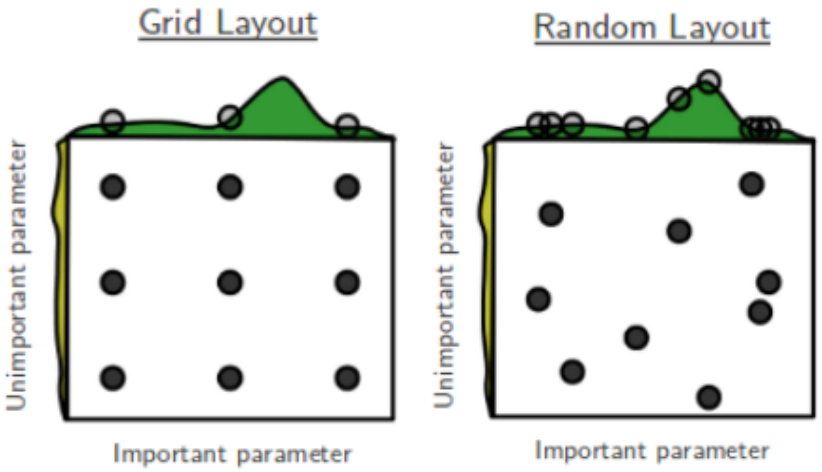

Figure 3.20: Grid search and Randomized grid search

3.9.3 Grid search with halving

Grid search with halving tackles the problem of big computation cost of hyper-parameters optimization process. It tries to start with coarse selection using limited number of data points and limited resources and as fewer candidate solutions remain, it increases the precision of the evaluation process by increasing resources. As a resource, we can select number of observations, but also for instance number of decision trees used in random forest model.

Algorithm:

1. Select initial candidates (hyper-parameter combinations) and use limited resources (number of observations) to evaluate objective function at each candidate point.

2. Reduce number of candidates by *factor* and increase number of resources by *factor*.

3. Continue process until all resources are used or only one candidate remains.

3.9.4 Bayesian optimization

Bayesian optimization is an advanced approach to hyper-parameters tuning. Our prior believes about an unknown objective function are updated based on observed data. Proposition of next test point is trade-off between **exploration** and **exploitation**:

- Exploitation seeks to sample where the surrogate model predicts a good objective.
- Exploration seeks to sample in locations where the uncertainty is high.

Next test point is given by **acquisition function**. The simplest acquisition function is probability of improvement (PI):

$$PI(\boldsymbol{x}) = \phi\left(\frac{\mu(\boldsymbol{x}) - \mu^{+}}{\sigma(\boldsymbol{x})}\right) \tag{3.41}$$

ϕ is a CDF of standard normal distribution, μ^{+} is the best achieved performance (amongst already measured points), $\mu(\boldsymbol{x})$ represents expected performance at point $\boldsymbol{x}$ and $\sigma(\boldsymbol{x})$ is expected standard deviation of $\mu(\boldsymbol{x})$.

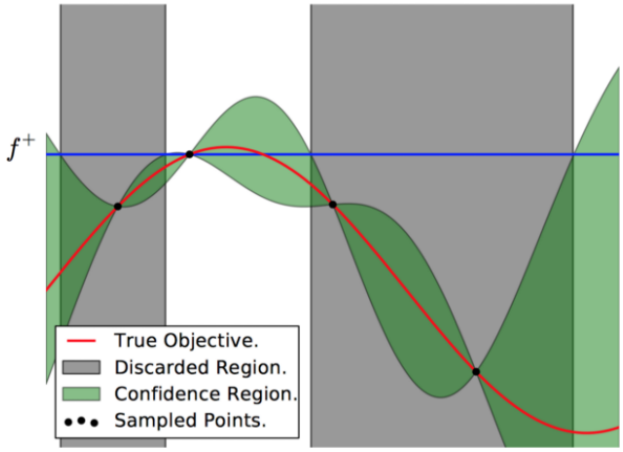

Figure 3.21: Bayes optimization.

3.9.5 Evolutionary optimization

In evolutionary optimization, generative algorithms can be used for hyperparameters optimization:

1. Initialization - create initial population with N parameter vectors. Each vector parameter is selected randomly from search space. For each vector calculate objective function.
2. For each vector in population:

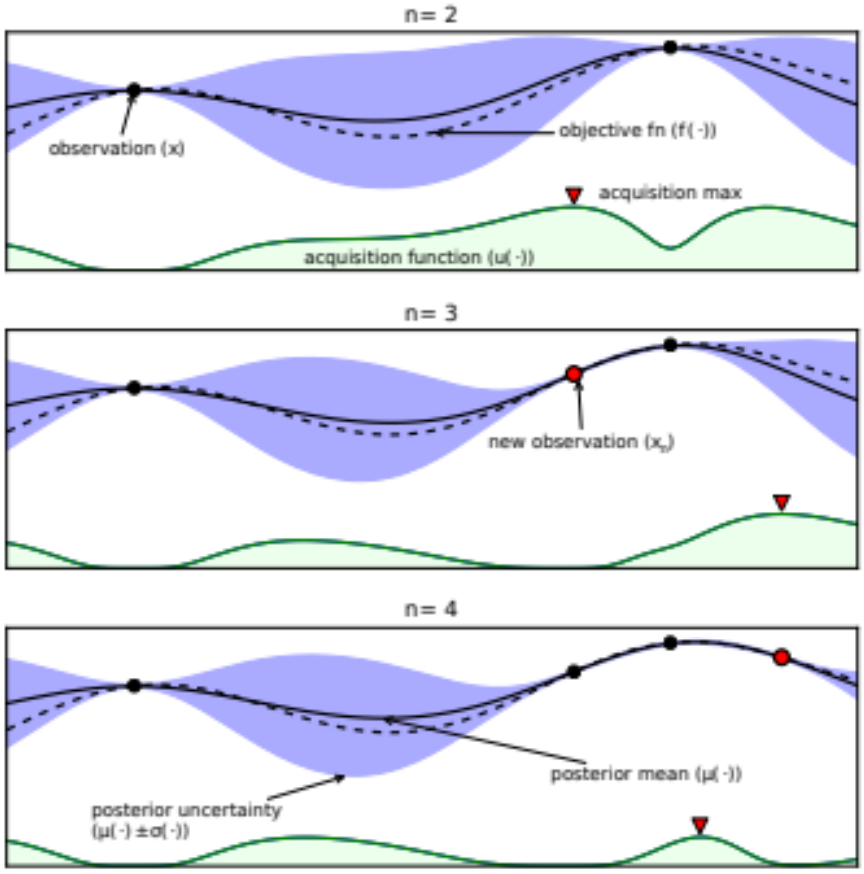

Figure 3.22: Bayes optimization - next point selection with acquisition function.

- Mutation

$$p_i^{mut} = p_i^{best} + F \cdot (p_i^{r_1} - p_i^{r_2}) \tag{3.42}$$

  where F is a **mutation rate** and $p_i^{r_1}$ and $p_i^{r_2}$ are parameter values from randomly chosen vectors r_1 and r_2.

- Recombination - new generation vector is created by selecting each of its items as either value from current vector or value from mutated vector. To choose between current or mutated value, random number is drawn from $[0; 1]$. If this number is lower than **recombination rate**, then current value is used, otherwise mutated value is selected.

- Replacement - if objective function for new generation vector is lower then for current vector, then replace current vector with new

generation vector.

3. Stop condition - stop evolution if standard deviation of population is lower then tolerance. Maximum number of iterations can be also selected.

3.9.6 Summary

Few general notes and summary about the algorithms from practice:

- Grid search is probably most common technique and will usually work fine.
- Randomized grid search allows you to control time of evaluation.
- Bayesian optimization can reduce consumed resources.
- Use training sample to search for hyper-parameters. Otherwise your test set won't be independent.

Always think about the problem at hand. Try to avoid meaningless combinations. Few examples:

- Target rate is 1% - restricting decision tree to have 100 observations in each leaf does not make sense.
- Having low amount of observations, decision trees with large depth will over-fit.
- Make sure, you are using proper metric. For instance, accuracy is not the best choice for binary target problems with unbalanced sample.

3.10 Model interpretation

Understanding how the trained model works might be crucial in practice. Black box ML models might have superior performance over classic regression algorithms, but if they do not respect the common sense of the business users, trust in the decisions can be broken. It is therefore very important to spend time to understand your model and validate if all the predictors influence the final probabilities in the way that you would expect. Nowadays, interpretability of modern ML algorithms like XGBoost is pretty good.

3.10.1 Feature importance

XGBoost calculates several metrics that can be used to assess feature importance:

- **weight:** The number of times a feature is used to split the data across all trees.
- **gain:** The average gain across all splits the feature is used in.
- **cover:** The average coverage across all splits the feature is used in. Cover is defined in xgboost package as the sum of second order gradient of training data classified to the leaf. If objective function uses mean square error loss, this simply corresponds to the number of instances in that branch. Deeper in the tree a node is, lower this metric will be.
- **total_gain:** The total gain across all splits the feature is used in.
- **total_cover:** The total coverage across all splits the feature is used in.

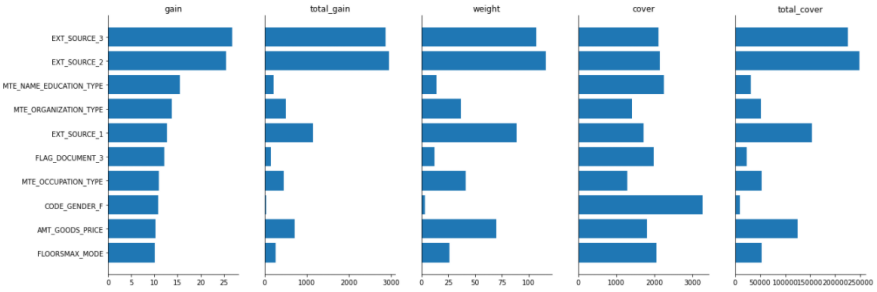

Figure 3.23: Various metrics of feature importance.

3.10.2 Marginal contribution

Marginal contribution works the same for simple regression models and for ML models.

- Assume we have a model with given strength and K predictors.
- For each predictor:
  - Fit new model with given predictor excluded.
  - Measure performance of new model.

- Drop in performance is called marginal contribution.

Marginal contribution requires fitting of K models. If we use cross validation, we must fit $K \times n\_folds$ models.

3.10.3 Permutation importance

Permutation importance (PI) assess what will happen if the predictor would be broken (start returning random values):

- For each predictor repeat n-times:
  - Randomly shuffle predictor values.
  - Calculate model performance using data with modified predictor.
  - Compare measured model strength with original model strength.

Comparison with marginal contribution:

- No need to fit new models.
- PI asses predictors importance in given model, not in general.

3.10.4 Partial dependence

Partial dependence describes marginal effect of a feature on the predicted outcome.

$$f(x_S) = \mathbb{E}_{x_C}\left[f(x_S, x_C)\right] = \int f(x_S, x_C)\mathrm{d}\mathbb{P}(x_C) \tag{3.43}$$

where f is a machine learning model, x_s is feature for which we calculate partial dependence and x_c are other feature included in model f. Function f can be estimated by calculating averages in training data:

$$f(x_S) = \frac{1}{N}\sum_{i=1}^{N} f(x_S, x_C^{(i)}) \tag{3.44}$$

3.10.5 Individual Conditional Expectation

Individual Conditional Expectation (ICE) plots are extension of partial dependence plots. They basically calculates the same, but rather than plotting only averaged information, ICE plots display one line per instance - for given

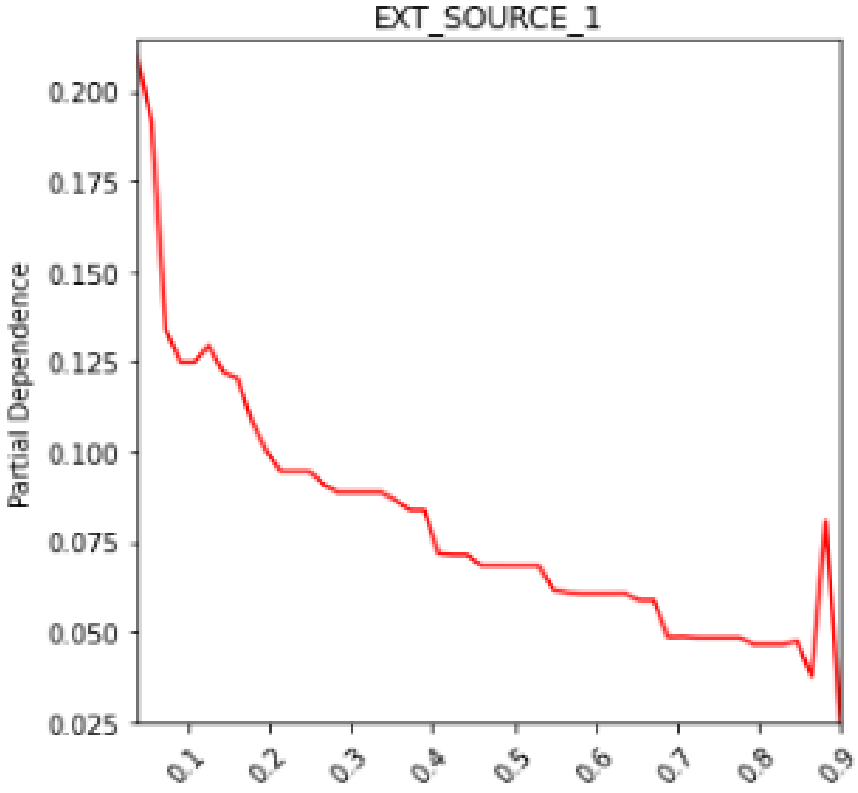

Figure 3.24: Partial dependence plot.

observation, try different values of explored predictor and plot the prediction based on value of predictor.

If we would average all lines plotted in ICE plot, we obtain partial dependence curve.

Partial dependence plots can obscure a heterogeneous relationship created by interactions. ICE plot provides more insight.

3.10.6 Shapley Value

What is the feature effect on model prediction in case of linear regression? Model prediction is given by:

$$f(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p \tag{3.45}$$

Contribution of j-th feature is:

$$\phi_j(f) = \beta_j x_j - \mathbb{E}(\beta_j X_j) \tag{3.46}$$

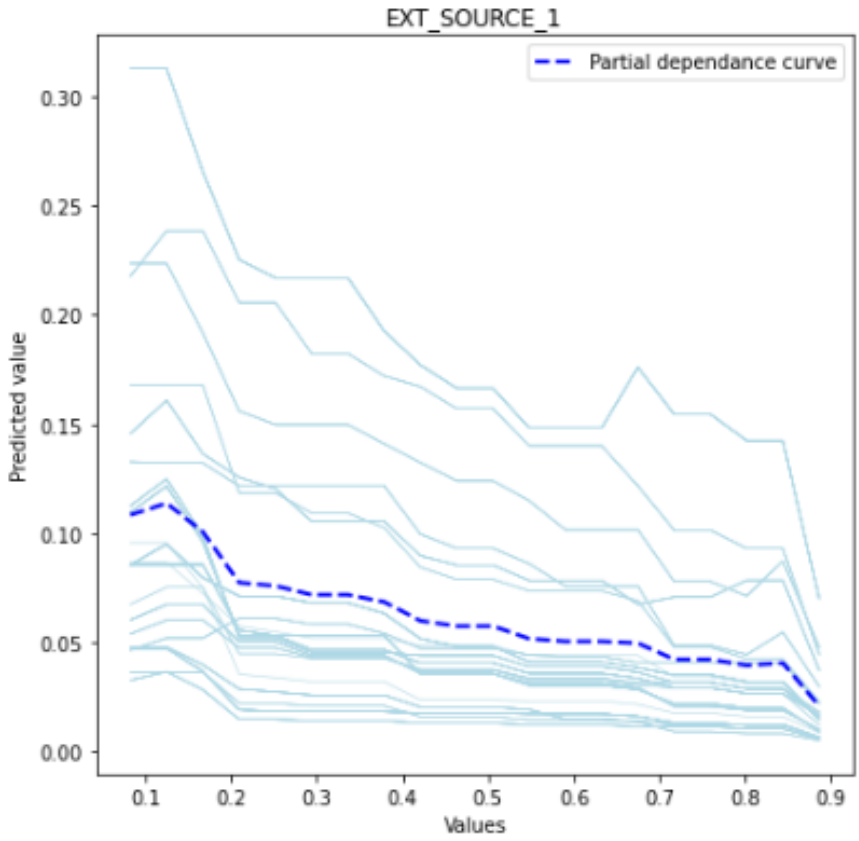

Figure 3.25: ICE plot.

where $\mathbb{E}(\beta_j X_j)$ is the mean effect estimate for feature j.

In case of ML model we can calculate feature effect using Shapley value. Shapley value concept has its origin in cooperative game theory.

Let $X_1, \ldots, X_p$ be predictors of ML model f and S is a subset of features used in the model. We define value function val as:

$$val_x(S) = \int f(x_1, \ldots, x_p) \mathrm{d}\mathbb{P}_{x \notin S} - \mathbb{E}_X(f(X)) \tag{3.47}$$

For instance, if ML model includes four features x_1, x_2, x_3 and x_4, then value function of coalition S consisting of feature values x_1 and x_3 is:

$$val_x(S) = val_x(\{x_1, x_3\}) = \int_{\mathbb{R}} \int_{\mathbb{R}} f(x_1, X_2, x_3, X_4) \mathrm{d}\mathbb{P}_{X_2 X_4} - \mathbb{E}_X(f(X)) \tag{3.48}$$

The Shapley value of a feature value is:

$$\phi_j(val) = \sum_{S \subseteq \{x_1,\ldots,x_p\} \setminus \{x_j\}} \frac{|S|!(p-|S|-1)!}{p!} \left(val(S \cup \{x_j\}) - val(S)\right) \quad (3.49)$$

Note that Shapley value ϕ_j of predictor x_j is calculated for a single observation. If our dataset contains N observations, for predictor x_j we have N different Shapley values.

Shapley value satisfies following properties:

- Efficiency
- Symmetry
- Dummy
- Additivity

Efficiency The feature contributions must add up to the difference of prediction for x and the average.

$$\sum_{j=1}^{p} \phi_j = f(x) - \mathbb{E}_X(f(x)) \quad (3.50)$$

Symmetry The contributions of two feature values j and k should be the same if they contribute equally to all possible coalitions. If

$$val(S \cup \{x_j\}) = val(S \cup \{x_k\}) \quad (3.51)$$

for all $S \subseteq \{x_1, \ldots, x_p\} \setminus \{x_j, x_k\}$, then

$$\phi_j = \phi_k \quad (3.52)$$

Dummy A feature j that does not change the predicted value – regardless of which coalition of feature values it is added to – should have a Shapley value of 0. If

$$val(S \cup \{x_j\}) = val(S) \quad (3.53)$$

for all $S \subseteq \{x_1, \ldots, x_p\}$, then

$$\phi_j = 0 \quad (3.54)$$

Additivity For a game with combined payouts $val + val^+$ the respective Shapley values are as follows:

$$\phi_j + \phi_j^+ \quad (3.55)$$

Number of possible coalitions grows exponentially with number of features. Thus Shapley values are estimated using following approximation:

$$\hat{\phi}_j = \frac{1}{M} \sum_{m=1}^{M} \left(f(x_{+j}^m) - f(x_{-j}^m) \right) \tag{3.56}$$

where $f(x_{+j}^m)$ is the prediction for x with random number of feature values replaced by feature values from random data sample. Feature x_j is not replaced though. x_{-j}^m is the same as x_{+j}^m, but with x_j also replaced with randomized data.

There are multiple plots provided in the XGBoost implementation for SHAP. Summary plot (Figure 3.26) is especially useful for understanding the model as a whole. The variables are sorted by their total importance from top to bottom. For each feature, red color represents high values of this feature for selected observation, blue represents low value (imagine sorting by quantiles). On the x axis, the effect on the final prediction of the model is displayed. The bigger concentration of dots for the same values on x axis represents bigger number of observations which belongs to this place.

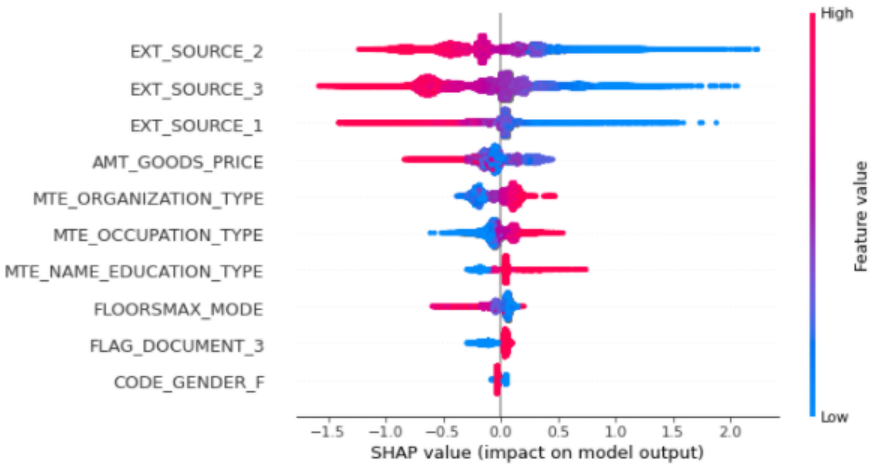

Figure 3.26: SHAP summary plot.

SHAP interaction plot (Figure 3.27) is useful for examining the interaction of two predictors. If there is no strong interaction, the values would lie close to a direct line. If you can find some parts of the chart with different behaviour, there could be some interesting interaction between the two selected predictors.

SHAP force plot (Figure 3.28) displays effect of each feature for one selected observation. You can see which features had the biggest positive and negative effect on the final prediction.

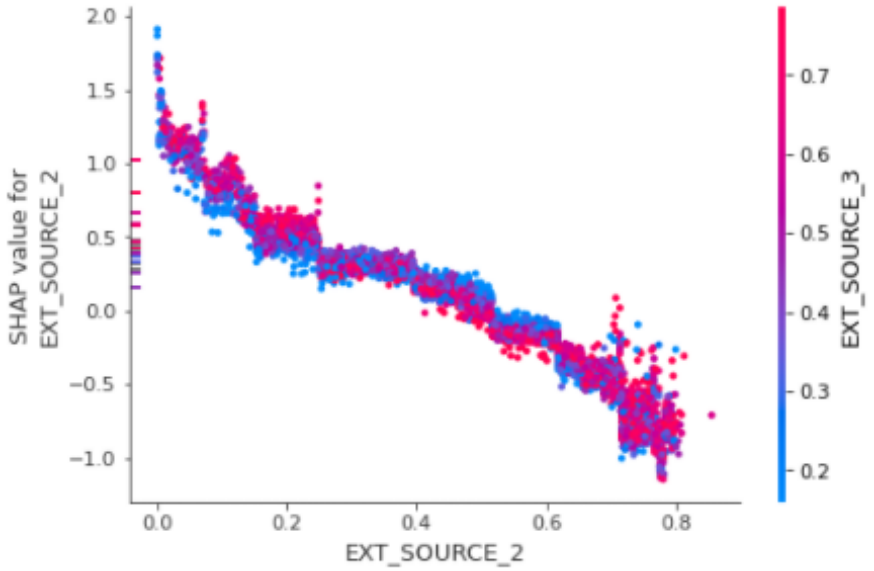

Figure 3.27: SHAP interaction plot.

Figure 3.28: SHAP force plot.

Chapter 4

Expected & Unexpected Credit Loss

Each financial institution is exposed to different types of risk such as credit risk, market risk, liquidity risk, etc. Those risks need to be properly monitored and covered in a form of reserves. Amount of required reserves is governed by regulator and in the most of the developed countries Basel regulatory framework is adopted. First version of Basel norms (Basel I) unifying capital regulations across countries was introduced in 1988. Since then Basel framework evolved reflecting various real encountered situations, mainly Global Financial Crisis of 2007-09. Nowadays (in 2019), third version of Basel regulations (Basel III) is being used.

It cannot be expected that all the bank's claims will be repaid. There will be always some losses. We can distinguish between expected and unexpected losses. Expected losses can be reasonably predicted based on historical observations, while unexpected losses cannot. However, we can still estimate maximum amount for unexpected loss (Fig. 4.1).

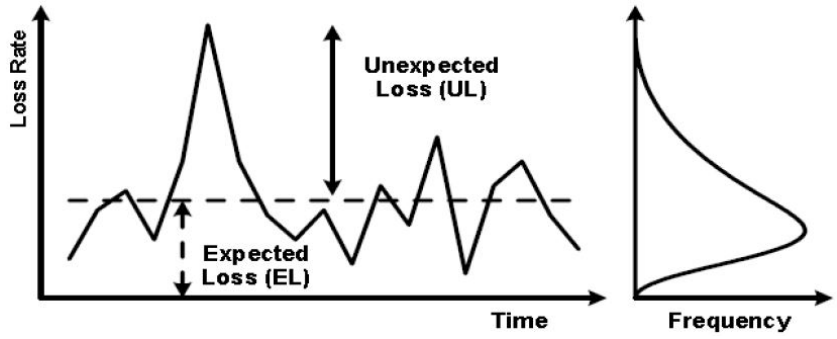

Figure 4.1: Bank's expected and unexpected losses.

Expected losses are covered by loss provisions that are to be made from current earnings, while unexpected losses should be mitigated by capital maintained by the bank. As visible on figure 4.2, only losses up to some confidence level are covered, since having covered all possible losses would be very costly and inefficient.

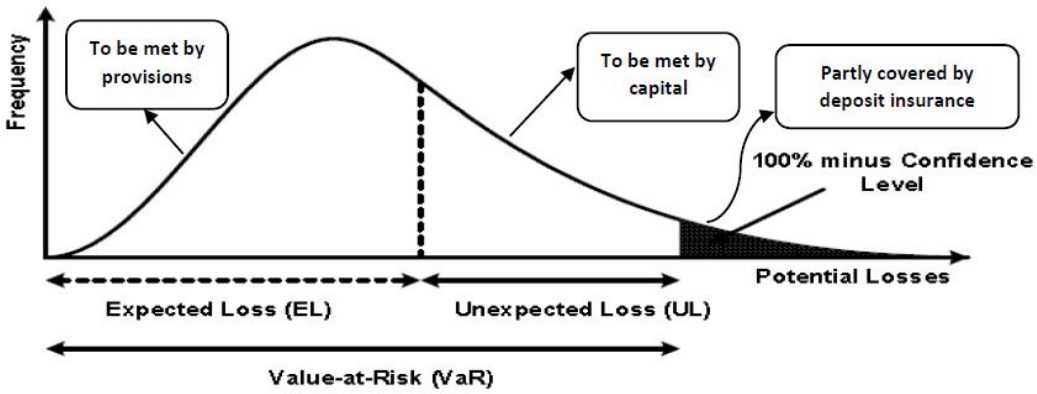

Figure 4.2: Bank's potential loss.

4.1 Unexpected losses

The amount of necessary capital to cover unexpected losses is established through capital adequacy ratio (CAR) defined as:

$$\text{CAR} = \frac{\text{Bank's capital}}{\text{Risk Weighted Assets}} = \frac{\text{Tier 1 capital} + \text{Tier 2 capital}}{\text{Risk Weighted Assets}} \tag{4.1}$$

Bank's capital is divided into Tier 1 and Tier 2 capital. Tier 1 capital is the core capital consisting of shareholder's equity and retained earnings (earnings disclosed on financial statement). Tier 2 capital is supplementary, less reliable, and it consists of subordinated term debt, undisclosed reserves, revaluation reserves, general provision and hybrid instruments. Sometimes Tier 3 capital is also introduced. Tier 3 can be used to support market risk, commodities risk and foreign currency risk and may include a greater number of subordinated issues, undisclosed reserves and general loss reserves compared to tier 2 capital.

Not all bank's assets carry the same risk. Mortgage loans secured by collateral are less risky than unsecured loans and therefore should consume less capital. This idea is realized through concept of Risk Weighted Assets (RWA). Basel regulation prescribes that each bank must group its assets

together by their underlying risk so that amount of required capital can reflect risk level of each asset group. Calculation of risk weights further depends on whether bank has incorporated standardized or internal rating based (IRB) approach. In standardized approach banks are required to use ratings from external credit rating agencies. Each rating class then has assigned risk weight by Basel regulation. In IRB approach banks are allowed to assess risk parameters of assets on their own.

Under Basel III, required capital adequacy ratio is 8%. Out of this, 4.5% must constitute of common equity capital, Tier 1 capital together must be at least 6% and remaining 2% can be produced by Tier 2 capital. Additional 2.5% of CAR is required for conservation buffer (in a form of common equity capital) and national regulator can prescribe up to 2.5% additional CAR for countercyclical capital buffer. This buffer should be decreased in times of crisis to help banks overcome hard times without additional burden in a form of required capital. Furthermore national regulator can assign additional CAR requirements for SIFI (systematically important financial institutions). SIFI companies are sometimes referred as "to big to fail". Finally, Basel III states one more non-risk related condition called leverage ratio. It is defined by following formula:

$$\frac{\text{Tier 1 capital}}{\text{Total exposure}} \geq 3\% \tag{4.2}$$

4.2 Expected losses

As mentioned in the previous section, Expected Credit Losses (ECL) are covered by bank in a form of loss provisions. Evaluation of ECL is guided by International Financial Reporting Standards (IFRS) that came into effect since 1st of January 2018 by replacing International Accounting Standards (IAS39). IFRS is divided into 17 parts. Each part is devoted to different topic and for credit risk purposes, the most important part is IFRS9 devoted to the financial instruments. Specifically, section 5.5 of IFRS9 describes instruments for evaluation of ECL.

When the financial crisis in 2008 arrived, lot of companies got into problem because of increasing demands for loss provisions. This was caused by a fact that IAS39 standard recognized a credit problem only after impairment of the loan. Amount of loss provisions was based solely on days past due (DPD), thus when the crisis started and institution's portfolio was suddenly more riskier, the institutions suffered from lower interest revenues (because clients did not pay their obligations) and moreover more provisions had to be created.

The idea implemented in IFRS9 is that institutions should be prepared for "bad times", so that in times of crisis the demand for additional loss provisions is minimal, actually loss provisions might be even lowering during the crisis. This should be accomplished using future-looking models for probability of default (PD), exposure at default (EAD) and loss given default (LGD).

$$ECL = PD \cdot EAD \cdot LGD \tag{4.3}$$

Before we will discuss PD, EAD and LGD models more closely, we will mention one other feature of ECL computation defined in IFRS 9 - Staging.

4.2.1 Staging

IFRS derives methodology for ECL computation based on three stages. The most important practical impact of different stages (at least for ECL) is that while in Stage 2 and Stage 3 possible defaults during lifetime of the loans must be recognized, for Stage 1 only default events expected within 12 months after the reporting date are considered.

Stage 1

Financial instruments that have not had a significant increase in credit risk since initial recognition or that have low credit risk at reporting date.

Stage 2

Financial instruments that have had significant increase in credit risk since initial recognition.

Stage 3

Financial instruments that are credit impaired at reporting date.

Exact definition of significant increase in credit risk is not provided by IFRS 9, each financial institution must derive its own rules for detection of credit risk deterioration.

Definition of credit impaired asset is very similar to the definition of default as given by Basel regulation. Financial asset is considered impaired based on observing at least one of the following events:

(a) significant financial difficulty of the issuer or the borrower;

(b) a breach of contract, such as a default or past due event;

(c) the lender(s) of the borrower, for economic or contractual reasons relating to the borrower's financial difficulty, having granted to the borrower a concession(s) that the lender(s) would not otherwise consider;

(d) it is becoming probable that the borrower will enter bankruptcy or other financial reorganisation;

(e) the disappearance of an active market for that financial asset because of financial difficulties; or

(f) the purchase or origination of a financial asset at a deep discount that reflects the incurred credit losses.

Note that not only days past due can trigger asset impairment, but also other negative events such as client entering insolvency or client having some of his assets restructured (even assets in different financial institution).

4.2.2 PD model

Survival models are often used to estimate probability of default, though it is not necessarily the only choice (regulation does not prescribe what techniques should be used). Unlike tools like logistic regression, survival models are capable of estimating probability of default in dependence on time, meaning that we do not have the same probability of default for let's say first and fifth instalment. Furthermore, survival models can utilize censored data, which allows us to extend our sample for model development with more recent data.

There are several survival models from which Kaplan-Meier estimator and Cox regression are the most popular. Generally, survival models were developed for use in medicine to assess impact of medical treatment. Patients were observed until some event happened, let it be death of the patient, or until they left observed population without knowing the exact time of patient's death (for instance the patient had moved out and did not visit the same doctor any more). Because of patients leaving the studied population before experiment end, it was necessary to incorporate censoring into these models. The output of the survival models is survival curve that gives us probability of surviving up to certain time.

4.2.2.1 Survival models

Let T be a non-negative random variable representing the waiting time until the occurrence of an event. Let's assume for now that T is continuous

random variable with probability density function (PDF) $f(t)$ and cumulative distribution function (CDF) $F(t) = \mathbb{P}[T < t] = \int_{-\infty}^{t} f(x)\mathrm{d}x$. Survival function is then complement of $F(t)$:

$$S(t) = 1 - F(t) \tag{4.4}$$

Alternatively, we can describe distribution of T with hazard function:

$$\lambda(t) = \lim_{\mathrm{d}t \to 0} \frac{\mathbb{P}[t \leq T \leq t + \mathrm{d}t | T \geq t]}{\mathrm{d}t} \tag{4.5}$$

Using Bayes' theorem about conditional probability that can be mathematically expressed as:

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[B|A]\,\mathbb{P}[A]}{\mathbb{P}[B]} \tag{4.6}$$

we can rewrite (4.5) to:

$$\lambda(t) = \lim_{\mathrm{d}t \to 0} \frac{\mathbb{P}[T \geq t | t \leq T \leq t + \mathrm{d}t]\,\mathbb{P}[t \leq T \leq t + \mathrm{d}t]}{\mathbb{P}[T \geq t]\,\mathrm{d}t} = \frac{\mathbb{P}[t \leq T \leq t + \mathrm{d}t]}{\mathbb{P}[T \geq t]\,\mathrm{d}t} \tag{4.7}$$

Since $\mathbb{P}[t \leq T \leq t + \mathrm{d}t] = f(t)\mathrm{d}t$ for $\mathrm{d}t \to 0$, we have:

$$\lambda(t) = \frac{f(t)}{S(t)} \tag{4.8}$$

Hazard function can be also written as:

$$\lambda(t) = -\frac{\mathrm{d}}{\mathrm{d}t} \log S(t) \tag{4.9}$$

which gives us:

$$S(t) = \exp\left\{ -\int_0^t \lambda(x)\mathrm{d}x \right\} \tag{4.10}$$

Cumulative hazard function is defined as:

$$\Lambda(t) = \int_0^t \lambda(x)\mathrm{d}x \tag{4.11}$$

Kaplan-Meier estimator

Let's assume that we don't have any censored observations in our data sample. In that case the simplest estimator of survival function would be $1 - F_n(t)$, where $F_n(t)$ is empirical cumulative distribution function based on

observed times of event $T_1, ..., T_n$. We will denote empirical survival function by $S_n(t)$.

$$S_n(t) = \frac{\text{number of } T_i > t}{n} \tag{4.12}$$

When censored data are present, $S(t)$ can be estimated by Kaplan-Meier product-limit estimator. This estimator is based on chain rule for conditional probabilities, which for two random events A and B can be written as:

$$\mathbb{P}[A \cap B] = \mathbb{P}[B|A]\,\mathbb{P}[A] \tag{4.13}$$

Let $t_0 = 0 < t_1 < t_2 < ... < t_{k-1} < t_k = t$. Then survival function takes the following form:

$$\begin{aligned} S(t) = \mathbb{P}[T \geq t] = \mathbb{P}[T \geq t_{k-1} \cap T \notin [t_{k-1}; t_k)] = \\ = \mathbb{P}[T \notin [t_{k-1}; t_k) | T \geq t_{k-1}]\,\mathbb{P}[T \geq t_{k-1}] \end{aligned} \tag{4.14}$$

We could apply the same trick on $\mathbb{P}[T \geq t_{k-1}]$ in the next step and eventually we would obtain survival function in a form:

$$S(t) = \prod_{i=1}^{k} (1 - \mathbb{P}[t_{i-1} \leq T < t_i | T \geq t_{i-1}]) \tag{4.15}$$

In case of uncensored data, empirical probability of event being realized in interval $[t_{i-1}; t_i)$, i.e. $\mathbb{P}[t_{i-1} \leq T < t_i | T > t_{i-1}]$, is given by number of events realized during given time interval, divided by number of observations observable at the beginning of the interval (event did not happen before t_{i-1}). In case of censored data, we will adjust estimation of events happening in $[t_{i-1}; t_i)$ by taking the number of events in this interval and dividing it by number of observable cases at the beginning of the interval minus number of censored data during this interval. We can write Kaplan-Meier estimator of survival function as:

$$\hat{S}(t) = \prod_{i:t_i \leq t} \left(1 - \frac{d_i}{n_i}\right) \tag{4.16}$$

with t_i a time when at least one event happened, d_i the number of events at time t_i and n_i the number of still observable cases at time t_i. Note that probability of default at time t can be extracted using (4.4).

Example 13. *Let's have 10 observations. For each observation we know either time of event T or censoring time C as given in following table. Then survival function given by Kaplan-Meier estimator is $S(T)$.*

ID	*T*	*C*	*S(T)*
1	*1*	-	$1 - \frac{1}{10} = 0.90$
2	*2*	-	$0.9 \cdot \left(1 - \frac{1}{9}\right) = 0.80$
3	-	*3*	-
4	*5*	-	$0.8 \cdot \left(1 - \frac{2}{7}\right) = 0.57$
5	*5*	-	$0.8 \cdot \left(1 - \frac{2}{7}\right) = 0.57$
6	*10*	-	$0.57 \cdot \left(1 - \frac{1}{5}\right) = 0.46$
7	-	*13*	-
8	*14*	-	$0.46 \cdot \left(1 - \frac{1}{3}\right) = 0.30$
9	-	*16*	-
10	*18*	-	$0.30 \cdot \left(1 - \frac{1}{1}\right) = 0.0$

Cox regression

Cox regression is a member of proportional hazard models. Hazard function in time t is defined as:

$$\lambda(t|\boldsymbol{x}_i) = \lambda_0(t)\exp\left\{\boldsymbol{x}_i^T\boldsymbol{\beta}\right\} \tag{4.17}$$

$\boldsymbol{x}_i$ is a vector of covariates and $\lambda_0(t)$ is called baseline or underlying hazard function. Note that unlike Kaplan-Meier estimator, Cox regression is a parametric model, we need to estimate parameters $\boldsymbol{\beta}$. Cumulative hazard rate is:

$$\Lambda(t|\boldsymbol{x}_i) = \Lambda_0(t)\exp\left\{\boldsymbol{x}_i^T\boldsymbol{\beta}\right\} = \exp\left\{\boldsymbol{x}_i^T\boldsymbol{\beta}\right\}\int_0^t \lambda_0(z)\mathrm{d}z \tag{4.18}$$

and survival function is defined as:

$$S(t|\boldsymbol{x}_i) = \exp\left\{-\Lambda(t|\boldsymbol{x}_i)\right\} = \exp\left\{-\Lambda_0(t)\right\}^{\exp\left\{\boldsymbol{x}_i^T\boldsymbol{\beta}\right\}} \tag{4.19}$$

Probability of default in given time can be again derived from cumulative distribution function:

$$F(t|\boldsymbol{x}_i) = 1 - S(t|\boldsymbol{x}_i) = 1 - \exp\left\{-\int_0^t \lambda_0(z)\exp\left\{\boldsymbol{x}_i^T\boldsymbol{\beta}\right\}\mathrm{d}z\right\} \tag{4.20}$$

In proportional hazard model, baseline hazard function describes hazard rate of population where all the covariates are equal to zero. The proportional part $\exp\{\boldsymbol{x}_i^T\beta\}$ then modifies this hazard rate multiplicatively with respect to baseline hazard. In other words, the time structure is included in baseline hazard while covariates contribution is focused in proportional part.

β coefficients in Cox regression have a nice interpretation. Let's assume that we have only one covariate x in our model. Then for two different cases,

we can compute ratio of their hazard functions:

$$\frac{\lambda(t|x_1)}{\lambda(t|x_2)} = \exp\{\beta(x_1 - x_2)\} \tag{4.21}$$

Therefore $\exp\{\beta\}$ has a meaning of proportional difference in hazard when value of x is increased by 1. This property of β remains valid even in case of more than one covariate included in the model.

Cox regression model can use stratification to distinguish different baseline hazard evolution in different sub-populations. This consists of learning separate baseline hazard curves for given sub-populations. However, note that the coefficients in proportional part are trained using all available data without distinguishing between populations and therefore this approach is not the same as building two separate models for two separate sub-populations.

Using both Cox regression and Kaplan-Meier estimator we are able to estimate cumulative distribution function $F(t)$ of time to default T. Probability of default for n-th payment is then:

$$\mathbb{P}[T = n] = F(n) - F(n-1) \tag{4.22}$$

At the beginning of this section it is mentioned that not only survival models are used for modelling PD. Institutions can employ any techniques for PD estimation as long as their approach is approved by auditor. Markov chains, for instance, can be used as an alternative. We will briefly introduce Markov chains in next section.

4.2.2.2 Finite Markov chains

We will consider a discrete-time stochastic process $X_k, k = 0, 1, 2, ...$, where allowed states of X_k are given by finite set $S = \{1, ..., N\}$. Possible values of X_k defined by S will be called states of the system. For purpose of ECL evaluation, those states will usually be delinquency buckets. One bucket will consist of clients currently without any unpaid obligations, next bucket will include clients currently having an obligation between 1 and 30 days past due and so on. Furthermore we will consider only processes fulfilling Markov property:

$$\mathbb{P}[X_t|X_1, ..., X_{t-1}] = \mathbb{P}[X_t|X_{t-1}] \tag{4.23}$$

In other words, Markov property states that to predict future state of our system, it is enough to consider only present state and not its history.

Furthermore, we will restrict ourself to time-homogeneous system, meaning that probability of transition between states of the system is constant in

time. Mathematically it can be formulated as:

$$p_{k,s} = \mathbb{P}\left[X_t = s | X_{t-1} = k\right] = \mathbb{P}\left[X_{t+u} = s | X_{t+u-1} = k\right], \forall u \in \{-t+1, -t+2, ...\} \tag{4.24}$$

Note that $p_{k,s}$ is probability of transition from state k to state s. We will extend our set S by two more absorbing states. Once the system enters absorption state, it stays in this state forever, i.e. $p_{a,a} = 1$ for any absorption state a and $p_{a,s} = 0$ for any state of the system other than a. As mentioned before, we will need two absorption states - default d and repaid r. Once the client defaults, he stays defaulted for good. Repaid means that all the obligations of the client are repaid and the loan is closed.

To describe probability distribution of our system, we need to know initial probability distribution $\phi(i) = \mathbb{P}\left[X_0 = i\right]$ and transition matrix P.

$$P = \left(p_{k,s}\right)_{k,s \in S \cup \{d,r\}} \tag{4.25}$$

Since in each step the process must make a transition to some of the states from S, including current state, the transition matrix is a stochastic matrix:

$$0 \leq p_{k,s} \leq 1, \qquad \forall k \in S \cup \{d, r\}, \forall s \in S \cup \{d, r\} \tag{4.26a}$$

$$\sum_{s \in S \cup \{d,r\}} p_{k,s} = 1, \qquad \forall k \in S \cup \{d, r\} \tag{4.26b}$$

Transition matrix of the system with absorption states can be divided in several subsections:

$$P = \left(\begin{array}{c|c} P_0 & R \\ \hline 0 & I \end{array}\right) \tag{4.27}$$

0 is a matrix of zeros by the shape $h \times |S|$, where h is a number of absorbing states, i.e. in our case $h = 2$, I is identity matrix with dimensions $h \times h$ and R is a matrix of shape $|S| \times h$ containing probabilities of transition from states of S to absorbing states d and r.

Having initial state $\boldsymbol{\phi}_0$, after one time step distribution of states will be given by:

$$\boldsymbol{\phi}_1 = \boldsymbol{\phi}_0 P \tag{4.28}$$

After k steps our system will be described by:

$$\boldsymbol{\phi}_k = \boldsymbol{\phi}_0 P^k \tag{4.29}$$

where

$$P^k = \begin{pmatrix} P_0^k & \sum_{t=1}^{k} P_0^{t-1} R \\ 0 & I \end{pmatrix} \tag{4.30}$$

Let's verify that this formula holds by computing one step further:

$$\begin{pmatrix} P_0^k & \sum_{t=1}^{k} P_0^{t-1} R \\ 0 & I \end{pmatrix} \begin{pmatrix} P_0 & R \\ 0 & I \end{pmatrix} = \begin{pmatrix} P_0^{k+1} & P_0^k R + \sum_{t=1}^{k} P_0^{t-1} R \\ 0 & I \end{pmatrix} \tag{4.31}$$

Since we described how to progress the system in time, we can discuss the long term behaviour. Suppose $\boldsymbol{\psi}$ is limiting state vector (state of our system in infinite time):

$$\boldsymbol{\psi} = \lim_{n\to\infty} \boldsymbol{\phi} P^n = \left(\lim_{n\to\infty} \boldsymbol{\phi} P^{n-1} \right) P = \boldsymbol{\psi} P \tag{4.32}$$

From (4.32) it is visible that $\boldsymbol{\psi}$ is stationary state of our system, also referred to as equilibrium or steady state. Let us now define the fundamental matrix.

Definition 14. *For an absorbing Markov chain with transition matrix P, the matrix $N = (I - P_0)^{-1}$ is called the fundamental matrix for P. The entry n_{ij} of N gives the expected number of times that the process is in the transient state s_j if it has started in the transient state s_i.*

Fundamental matrix tells us how many times will be each transient state visited before absorption given state of origin. What if we would like to know what is the expected time to complete absorption? Since each transient state s_j will be visited n_{ij}-times if the chain starts in s_i, time to absorption is given by:

$$\boldsymbol{t} = N\boldsymbol{c} \tag{4.33}$$

where $\boldsymbol{c}$ is a vector of ones. And finally, we can evaluate absorption probabilities - what portion will be absorbed by each absorption state in the stationary state of our system.

$$R_\infty = (I - P_0)^{-1} R \tag{4.34}$$

$(R_\infty)_{i,j}$ gives us probability of final state being j (in our case $j \in \{d, r\}$) if initial state is $s_i \in S$.

We have described a bit the theory behind Markov processes. Now, how to use it for estimation of probability of default for ECL evaluation? In fact, using Markov process, we will be estimating directly $PD \cdot EAD$ instead of only PD. Based on the historical data we learn the transition matrix having delinquency buckets extended by default and repaid as our states of the system. Then we take current active exposure in our portfolio and divide it into the delinquency buckets. This is our initial state. And then we can can apply transition matrix to learn future state of our system and by using (4.34) we can estimate lifetime behaviour of our assets as well. Note that

by having repaid state in the system, we simulated time to default, because transition to default is being distributed across the steps and in each step we also repay something. Therefore Markov process is capable of simulating the fact that if the loan defaults after k-th step, it does not default with full initial exposure (some part of the loan was repaid in the previous $k-1$ steps).

4.2.3 EAD model

In previous section we have derived how to construct PD model, giving us probability of default for each future instalment of each client. To combine this information with information about expected remaining exposure at the time of default (EAD), we will first look at the concept of annuity instalment.

Annuity payment

Banks usually utilize compounded interest rate calculation. Accrued interest is computed as percentage (given by annual interest rate) of remaining principal part of the loan at the time of the instalment. However, it means that amount of accrued interest is changing across the time (across individual instalments). For the client it is much more convenient to pay the fixed amount each month, rather than having each month different amount to be paid. Annuity instalment is a concept that allows us to fix the instalment amount while using compounded interest rate calculation. To derive formula for annuity instalment, we first need to take a look at the concept of time value of the money.

Let us have 100 of currency unit (CU). We can lend the money today for an annual interest rate 10% in a form of a loan to be repaid in one instalment after one year. After one year we will be paid 110 CU. It means that our present value (PV) of 100 CU is equal to future value (FV) of 110 CU after one year. In other words, if we would have 100 CU after one year, in fact we generated a loss, because we could have invested the money. The relationship between future value and present value is given by:

$$FV = PV\left(1 + \frac{i}{n}\right)^{nd} \tag{4.35}$$

where i is annual interest rate, n is number of payments per year and d is number of years. Now, if we compute present value of each future incoming payment (given annual interest rate), sum of those amounts should be equal

to credit amount of the loan L:

$$L = \sum_{k=1}^{n \cdot d} FV \left(1 + \frac{i}{n}\right)^{-k} \tag{4.36}$$

We want to have constant payments in time, therefore FV will be a constant equal to our annuity payment A, i.e. $A = FV$. By summing up first $n \cdot d$ elements of geometric sequence and denoting $q = 1 + \frac{i}{n}$ we can get following relationship:

$$\begin{aligned} L = \sum_{k=0}^{n \cdot d} A \left(1 + \frac{i}{n}\right)^{-k} - A = A \left(\sum_{k=0}^{n \cdot d} \left(\frac{1}{q}\right)^{k} - 1\right) = \\ = A \left(\frac{\frac{1}{q^{n \cdot d+1}} - 1}{\frac{1}{q} - 1} - 1\right) = A \frac{q^{n \cdot d} - 1}{q^{n \cdot d}(q-1)} \end{aligned} \tag{4.37}$$

and therefore amount of annuity payment is given by:

$$A = L \frac{q^{n \cdot d}(q-1)}{q^{n \cdot d} - 1} \tag{4.38}$$

Part of the annuity payment is used to decrease remaining principal, while the other part covers accrued interests. Principal part of the annuity for k-th payment is given by:

$$A_p^{(k)} = A \frac{1}{(1 + \frac{IR}{n})^{nd+1-k}} \tag{4.39}$$

Exposure at default

Let's get back to EAD model. For standard annuity based products, the principal at time t under assumption that time of default T is greater then t is known (based on (4.38) and (4.39)). So what will be the exposure at default if the client fails to pay k-th future payment? First we take remaining principal at the time of k-th payment. Then we should add interests and fees that will be accrued until the loan is paid-off (after pay off exposure is fixed - no more fees and interest gains are being generated). Usually paid-off is applied once loan reaches 90 DPD, though this rule might differ for each company.

To compute final $PD \cdot EAD$ for given loan we can simply sum over all future instalments:

$$PD \cdot EAD = \sum_{k=1}^{n} \mathbb{P}\left[T = k\right] \cdot EAD_k \tag{4.40}$$

where n is number of remaining instalments on given loan and EAD_k is exposure at default for k-th instalment evaluated as described in the previous paragraph.

For revolving products is the situation a bit more complicated. Usually, we know what is the used part of the loan at the time of evaluation. However, we should estimate what will be the used part of the credit amount at the time of default. Therefore we need to create yet another model, that will evaluate how the client will use the revolving product in time. Note that predicted EAD should not be lower then currently utilized amount, since it makes sense that if the client gets into trouble, he won't be lowering the utilization of the revolving product, but rather he will be using bigger part of the credit limit than now.

4.2.4 LGD model

Loss given default is a metric whose purpose is to estimate real loss under the condition that the asset has defaulted. It can be also defined as 1 minus recovery rate (RR). By recovery we can understand any incoming payment after default. This includes payments directly from the client, revenue from selling the defaulted asset to the third party or, in case of secured loans, income from sold collateral. On the other hand, LGD should include information about collection costs, such as expenses related to legal proceeding and other. Therefore the formula for LGD could be written as:

$$LGD = 1 - RR + Collection\ Costs \tag{4.41}$$

In case of having collateral in different currency than the asset, there might be required additional haircut for currency mismatch by local central bank. Application of the haircut is usually done using multiplication of LGD by constant lower then 1.

Another modification consists of recommendation to compute "Downturn" LGD, which reflects LGD at the times of downturn business cycle. Since Downturn LGD can be interpreted by each institution differently, we will not describe Downturn LGD here in detail.

Because collection of recoveries can extend over long time period, it is necessary to discount post-default cash flows to the time of the default. It means that we need to apply (4.35) to adjust (or discount) the value of incoming cash flow to the time of the default.

Based on the previous paragraph, we can rewrite (4.41) as:

$$LGD = 1 - \frac{1}{EAD} \sum_{i:t_i \leq t_{max}} recovery_i \left(1 + \frac{i}{n}\right)^{-n \cdot t_i} + Collection\ Costs \tag{4.42}$$

where t_i is time of receiving recovery i, t_{max} is time of asset's write off (after asset is written off from the accounting books, no more recovery will be collected). Note that for i, effective interest rate (EIR) should be used. EIR is interest rate based on all future cash flows related to the loan, including instalments and fees. Full definition of EIR for ECL computation can be found in Appendix A of IFRS9 [4].

that encounters for any additional payments related to the loan, such as payments for insurance and fees.

4.2.5 Macroeconomics adjustment

According to IFRS, credit risk assessment should also incorporate forward-looking analysis of macroeconomics. Final adjustment of ECL by expected macroeconomic should be combined from different scenarios as their probability weighted average. For instance we can evaluate baseline scenario, upside scenario and adverse scenario and our final macroeconomic prediction will be based on weighted average of those three scenarios, where the weights should reflect probability of each given scenario. Number of scenarios is not restricted, we can have multiple different scenarios.

Chapter 5

Estimates of probability of defaults and price of risk

In this chapter we will describe how to compute price of risk for loans provided by financial institution. Knowledge of price of risk is essential for management of bank's pricing policy as well as for evaluation of necessary reserves (by regulation each bank needs to have reserves for expected losses).

We will assume following basic model, which could be used by the bank to determine interest rate:

$$\text{Interest} = \text{Cost of funding} + \text{Cost of risk} + \text{Price margin} \tag{5.1}$$

Cost of funding determines how much the bank has to pay for the money, which are then used to grant loans. The bank itself usually borrows this money, either in the form of client accounts or by getting a loan from other banks. Cost of funding is usually driven by the interest rates on the interbank lending market. We will not deal with the right setting of funding costs here, but rather focus on the price of risk part. We note that the value of price margin depends mostly on the marketing strategy of the bank. It is even possible to have negative margin on some products, given the fact that the total margin and profit would be greater than zero client-based. In such case, detailed calculations should be made to understand all incomes that the bank will get from the client, including fees and other transactions that would be processed by the bank.

The equation (5.1) depends on the term of the loan. Funding costs and price of risk grow when the length of the deal increases. In the next sections, we will discuss how to calculate price of risk based on the probability distribution of the time to default.

5.1 Price of risk based on time to default

Let's now assume that we know probability of default p for the given loan. We will start with evaluation of risk margin (interest rate that would cover for our cost of risk) for the simplest case - one year loan with one time payment. Then we will generalize the evaluation for gradually paid loans and for loans with terms different from one year. Eventually, we will discuss secured loans.

5.1.1 One-time one year loan

Suppose that the client borrows amount X, which shall be paid back after one year. Moreover, let's suppose that the probability of default is p. The price of risk can be determined by a simple equation, based on the fact that income from the risk margin shall be equal to the potential loss. Denoting the risk margin r, we have:

$$r(1-p)X = pX \tag{5.2}$$

And therefore:

$$r = \frac{p}{1-p} \tag{5.3}$$

Let's denote by T the random variable describing the time to default. More realistic models suppose that conditional distribution of T given that client defaults, $T \leq 1$ is uniform or exponential. Under the assumption that client pays all his interest liabilities until time of default (principal part is still paid one time after one year) we have expected income from risk margin equal to $(1-p)rX + prX\mathbb{E}\left[T|T \leq 1\right]$ while expected loss is still pX.

In the uniform case, we have:

$$\begin{aligned} (1-p)rX + prX\frac{1}{2} &= pX \\ r &= \frac{p}{1-\frac{p}{2}} \end{aligned} \tag{5.4}$$

Let's now suppose, that the time to default has exponential distribution with the density $f(x) = \lambda \exp\left\{-\lambda x\right\}$ and parameter λ is chosen such that the probability of default in one year should be equal to p. Cumulative distribution function is then $F(x) = 1 - \exp\left\{-\lambda x\right\}$ and we have:

$$\begin{aligned} \mathbb{P}\left[T \leq 1\right] = F(1) = 1 - \exp\left\{-\lambda\right\} &= p \\ 1 - p &= \exp\left\{-\lambda\right\} \\ \lambda &= -\ln\left\{1-p\right\} \end{aligned} \tag{5.5}$$

The expected income is then:

$$(1-p)rX + prX\mathbb{E}\left[T|T \leq 1\right] \tag{5.6}$$

with:

$$\begin{aligned}\mathbb{E}\left[T|T \leq 1\right] &= \frac{1}{1-\exp\left\{-\lambda\right\}} \int_0^1 t\lambda \exp\left\{-\lambda t\right\} \mathrm{d}t = \\ &= \frac{1}{1-\exp\left\{-\lambda\right\}} \left(\left[-t\exp\left\{-\lambda t\right\}\right]_0^1 - \int_0^1 -\exp\left\{-\lambda t\right\} \mathrm{d}t \right) \\ &= \frac{1}{1-\exp\left\{-\lambda\right\}} \left(-\exp\left\{-\lambda\right\} - \frac{\exp\left\{-\lambda\right\}-1}{\lambda} \right) \\ &= \frac{1}{\lambda} - \frac{\exp\left\{-\lambda\right\}}{1-\exp\left\{-\lambda\right\}}\end{aligned} \tag{5.7}$$

Combining the equations together, we get:

$$\begin{aligned}(1-p)rX + prX\mathbb{E}\left[T|T \leq 1\right] &= (1-p)rX + prX\left(\frac{1}{\lambda} - \frac{\exp\left\{-\lambda\right\}}{1-\exp\left\{-\lambda\right\}}\right) = \\ &= rX\left((1-p) + \frac{p}{\lambda} - \exp\left\{-\lambda\right\}\right) = rX\left(-\frac{p}{\ln\left\{1-p\right\}}\right)\end{aligned} \tag{5.8}$$

Therefore, the price of risk is:

$$r = -\ln\left\{1-p\right\} = \lambda \tag{5.9}$$

From equation (5.6) we also have a general formula to compute price of risk:

$$r = \frac{p}{1-p+p\mathbb{E}\left[T|T \leq 1\right]} \tag{5.10}$$

5.1.2 Gradually-repaid one-year loan

Assume for simplicity that gradually repaid loan is being paid uniformly, meaning that for one-year loan we have principal balance $X_t = (1-t)X$. If we denote again the time of default by T, then expected loss equals $\mathbb{E}\left[X(1-\min\left\{1,T\right\})\right]$. Under the assumption of continuous calculation of interest, we have expected income of $r\mathbb{E}\left[\int_0^{\min\{1,T\}}(1-t)X\mathrm{d}t\right]$.

Example 15. *Suppose the default at time $T = \frac{1}{2}$. Then the expected income is:*

$$r\mathbb{E}\left[\int_0^{\min\{1,T\}}(1-t)X\mathrm{d}t\right] = r\mathbb{E}\left[\int_0^{\frac{1}{2}}(1-t)X\mathrm{d}t\right] = rX\left[t-\frac{t^2}{2}\right]_0^{\frac{1}{2}} = \frac{3}{8}rX$$

General equation for price of risk of one-year gradually repaid loan is then:

$$r = \frac{X\left(1 - \mathbb{E}\left[\min\left\{1, T\right\}\right]\right)}{\mathbb{E}\left[\int_0^{\min\{1,T\}} (1-t)X\mathrm{d}t\right]} = \frac{1 - \mathbb{E}\left[\min\left\{1, T\right\}\right]}{\mathbb{E}\left[\int_0^{\min\{1,T\}} (1-t)\mathrm{d}t\right]} \tag{5.11}$$

For special cases when the distribution of T is specified, we can evaluate the formula above. If T is deterministically equal to 1, the losses are equal to zero and risk margin can be also considered zero. Suppose now that the conditional distribution of time to default given that the default happens till one 1 year is uniform. We have:

$$\mathbb{E}\left[\min\left\{1, T\right\}\right] = 1 - p + p\mathbb{E}\left[T|T \leq 1\right] = 1 - p + \frac{p}{2} = 1 - \frac{p}{2} \tag{5.12}$$

Therefore, expected loss is equal to $X\frac{p}{2}$. For expected income, let's calculate the integral first:

$$\int_0^{\min\{1,T\}} (1-t)\mathrm{d}t = \min\left\{1, T\right\} - \frac{\min\left\{1, T\right\}^2}{2} \tag{5.13}$$

And its expectation:

$$\begin{aligned}\mathbb{E}\left[\min\left\{1, T\right\} - \frac{\min\left\{1, T\right\}^2}{2}\right] &= \left(1 - \frac{p}{2}\right) - \frac{1-p}{2} - \frac{p}{2}\mathbb{E}\left[T^2|T \leq 1\right] = \\ &= \frac{1}{2} - \frac{p}{2}\left[\frac{t^3}{3}\right]_0^1 = \frac{1}{2} - \frac{p}{6}\end{aligned} \tag{5.14}$$

The risk price is then:

$$r = \frac{1 - \mathbb{E}\left[\min\left\{1, T\right\}\right]}{\mathbb{E}\left[\int_0^{\min\{1,T\}} (1-t)\mathrm{d}t\right]} = \frac{1 - \left(1 - \frac{p}{2}\right)}{\frac{1}{2} - \frac{p}{6}} = \frac{p}{1 - \frac{p}{3}} \tag{5.15}$$

5.1.3 k-year loans

We will extend our calculations in a natural way to handle k-year loans. Denote $p = \mathbb{P}\left[T \leq k\right]$. For one-time loans, we have:

$$(1-p)rkX + prX\mathbb{E}\left[T|T \leq k\right] = pX \tag{5.16}$$

And equation (5.10) extends to:

$$r = \frac{\mathbb{P}\left[T \leq k\right]}{k\mathbb{P}\left[T > k\right] + \mathbb{P}\left[T \leq k\right]\mathbb{E}\left[T|T \leq k\right]} \tag{5.17}$$

For continuous repayment, it holds $X_t = \frac{(k-t)}{k}X$ and our expected incomes from risk margin are $r\mathbb{E}\left[\int_0^{\min\{k,T\}} \frac{(k-t)}{k}X\mathrm{d}t\right]$. The risk price is then:

$$r = \frac{1 - \frac{\mathbb{E}[\min\{k,T\}]}{k}}{\mathbb{E}\left[\int_0^{\min\{k,T\}}(1-\frac{t}{k})\mathrm{d}t\right]} \tag{5.18}$$

5.1.4 Secured loans

We will now consider recoveries, which represent the ratio of the debt which will be collected by the bank after the client defaults. Usually, this is understood as NPV (net present value) of all financial inflows from the client from the time of default (say, 90 days past due) to his absolute payment default (bankruptcy) or, in the better case, back to being able to pay regularly as a good client. Alternatively, we can consider the recoveries as the proportional part of the loan which is secured, by buildings, other tangible assets, intangible assets, bills, etc. In this case, we consider value α defined as:

$$\alpha = \min\left\{\frac{\text{value of the security}}{\text{loan amount}}, 1\right\} \tag{5.19}$$

Let's further denote by $\xi(t)$ the amount of principal at time t, if the loan is being repaid at t. Special cases are:

- One-time loan, for which $\xi(t) = X,\ t \in (0,k)$, where X is the original loan amount and k is the term. Apparently, the function equals zero outside of $(0,k)$.
- Uniformly repaid loan, for which $\xi(t) = \frac{k-t}{k}X,\ t \in (0,k)$.

The loss incurred by the bank is again a random variable, which could be expressed as $\max\{\xi(T) - \alpha X, 0\}$. The income from risk margin does not depend on the security and is equal to $r_{\alpha,k}\int_0^{\min\{k,T\}}\xi(t)\mathrm{d}t$. The risk margin should depend on the value of security α and term k. We compute it again by equality of the expected income and expected loss:

$$r_{\alpha,k} = \frac{\mathbb{E}\left[\max\{\xi(T) - \alpha X, 0\}\right]}{\mathbb{E}\left[\int_0^{\min\{k,T\}}\xi(t)\mathrm{d}t\right]} \tag{5.20}$$

In the special case of one-time loan with security, we have:

$$r_{\alpha,k} = \frac{\mathbb{P}\left[T \le k\right](1-\alpha)}{k\mathbb{P}\left[T > k\right] + \mathbb{P}\left[T \le k\right]\mathbb{E}\left[T | T \le k\right]} = (1-\alpha)r_{0,k} \tag{5.21}$$

For uniformly repaid loan, the loss $\max\left\{\frac{k-T}{k}X-\alpha X,0\right\}$ is greater than zero for $T\leq(1-\alpha)k$. We get from the general equation (5.20):

$$\begin{aligned} r_{\alpha,k} &= \frac{\mathbb{P}\left[T\leq(1-\alpha)k\right]\frac{\mathbb{P}[T\leq k]}{\mathbb{P}[T\leq(1-\alpha)k]}\int_0^{(1-\alpha)k}\frac{(1-\alpha)k-t}{k}f(t)\mathrm{d}t}{\mathbb{P}\left[T>k\right]\int_0^k\frac{k-t}{k}\mathrm{d}t+\mathbb{P}\left[T\leq k\right]\int_0^k\int_0^t\frac{k-s}{k}f(t)\mathrm{d}s\mathrm{d}t} \\ &= \frac{\mathbb{P}\left[T\leq k\right]\int_0^{(1-\alpha)k}\frac{(1-\alpha)k-t}{k}f(t)\mathrm{d}t}{\mathbb{P}\left[T>k\right]\frac{k}{2}+\mathbb{P}\left[T\leq k\right]\int_0^k\left(t-\frac{t^2}{2k}\right)f(t)\mathrm{d}t} \end{aligned} \tag{5.22}$$

where $f(t)$ in the density of the random variable T. In the case when the distribution of T, given by the default up to time k, is uniform, we get:

$$\begin{aligned} \int_0^{(1-\alpha)k}\frac{(1-\alpha)k-t}{k}f(t)\mathrm{d}t &= \int_0^{(1-\alpha)k}\frac{(1-\alpha)k-t}{k}\frac{1}{k}\mathrm{d}t = \\ &= \left[\frac{(1-\alpha)t}{k}-\frac{t^2}{2k^2}\right]_0^{(1-\alpha)k} = \frac{(1-\alpha)^2}{2} \end{aligned} \tag{5.23}$$

Apparently, the equation $r_{\alpha,k}=(1-\alpha)^2 r_{0,k}$ holds. It does not hold in general. but it is often used for other distributions of T. Another generalization of our equations can be derived by including the price of money, or discount, as will be shown below.

5.2 Limit assignment

Suppose we have defined risk grades for our clients as discussed in the previous section. The question is how to assign limit to each client in order to keep risk under control and maximize profit of the bank. Usually, each risk grade is assigned different limit based on its expected risk behavior. Moreover, we should also consider other important drivers of risk, such as affordability, previous history of the client in our bank or credit bureau information. Following simple rule can be applied, calculate:

- Risk limit – based on the risk grade
- Affordability limit – based on the company's expected cash flows or client income and expected term of the loan
- Historical behavior limit - based on clients historical credit amount or annuity with our bank or credit bureau information about the same

We choose the minimum from above calculated limits and assign such limit to the customer. Proper limit setting can help to reduce the risk of the portfolio, as the least risky clients receive higher loans and therefore our weighted risk performance is better than in the case of flat limits.

5.3 Creditmetrics

In previous sections, we have dealt with the problem of loan pricing for a single loan. Now we will take a look at the whole portfolio at once. In the case of single loan, we examine the expectation of the random variable which corresponds to the loss coming from this loan. In the case of portfolio, we also have to calculate variability of this loss, so that we are able to predict evolution with some confidence. Following model is based on Creditmetrics methodology from JP Morgan, see www.creditmetrics.com.

There are various banking products covered by this methodology. The main products include:

- Bonds
- Loans
- Guarantees
- Letters of credit
- Discount loans

Apart from this basic products, there are more complicated derivates like forwards, options, etc. The most typical product with credit risk is a bond, which is an instrument with fixed principal value and so called coupon. Coupon is usually stated relatively to the principal value, in percents and it could be monthly, quarterly or yearly paid. There are also zero-coupon bonds, where interest is paid at the end together with principal. Coupons can have fixed interest rates or they can be bound to the market rates like 3 month LIBOR or PRIBOR. From the credit risk perspective, one-time loans are the same product as bonds, while coupon is represented by the interest payments.

5.3.1 Single-loan portfolio

Suppose we have a single bond with principal X, fixed coupon c and term T. This bond is rated on the scale $1, \ldots, S$ as s. We will try to estimate the value of this bond in one year from now and its variability. In the following year, the bond can migrate into another rating category or get into default. Suppose it goes to the category AAA, then the value of the bond is calculates as follows. We calculate discounted cash flows of coupon payments and principal payment based on zero forward one year curve, which denotes price of money for each rating category in various time horizons. This curve

is produced based on market behavior and it corresponds to a bond without profit, but captures risk. Let's denote the values of such curve for rating group r at time t as: $d_r(t), t = 1, \ldots,$. We get the value of our bond in one year in the case that it migrates to category r as:

$$\mu_r = c + \sum_{t=1}^{T} \frac{c}{(1 + d_r(t))^t} + \frac{X}{(1 + d_r(T))^T}. \tag{5.24}$$

In the case of default, we assume recoveries from the principal X of relative ration α and we denote this value as μ_d. The pricing of risk in the following year is done through the time discounting. The value of the bond in one year will therefore be a random variable with values $\mu_r, r \in \{1, \ldots, S\} \cup \{d\}$ and corresponding probabilities $p_{s,r}, r \in \{1, \ldots, S\} \cup \{d\}$. The expected value and variance are then given by:

$$\begin{aligned} \mu_1 &= \sum_{r \in \{1,\ldots,S\} \cup \{d\}} \mu_r p_{s,r} \\ \sigma_1^2 &= \sum_{r \in \{1,\ldots,S\} \cup \{d\}} \mu_r^2 p_{s,r} - \mu_1^2 \end{aligned} \tag{5.25}$$

5.3.2 Portfolio with two loans

Let's now consider the case of two bonds, first in rating category s_1 and second in rating category s_2. The procedure described above can be easily generalized in the case of independent bonds. However, if there is some dependence, we need to be cautious when calculating variance. We need to construct multivariate distribution of vector (X_1, X_2), representing the value of bonds after one year. The set of states is given by the Cartesian product $S^2 = \{1, \ldots, S\} \cup \{d\} \times \{1, \ldots, S\} \cup \{d\}$, but the transition probabilities are estimated through a special procedure. We will suppose that the quality of client depends on the value of his assets and his rating transition is given by the change of the value of assets after one year (Merton model 1974). If the value of assets remains unchanged, we do not expect change of rating, but a default might occur after significant drop of the asset value. We suppose that the asset value difference has a normal distribution with given expectation and variance. The transition probabilities then give margins, when such margins are crossed, the rating is changed. The correlation coefficient of the normal distribution drives the correlation of the rating categories. Let's recall that the correlation does not change with linear transformation, so we might consider only normal distribution with zero mean and unit variance and set the margins for this distribution. Then we will simulate bi-variate normal

distribution with given correlation and calculate the bi-variate distribution of rating categories from the simulation. Similarly as in the previous section, if we denote $\mu_{s,t}$ the value with transition to the states s and t and $p^*_{s,t}$ the transition probability from states s_1, s_2 we have:

$$\begin{aligned} \mu_1 &= \sum_{(s,t)\in S^2} \mu_{s,t} p^*_{s,t} \\ \sigma_1^2 &= \sum_{(s,t)\in S^2} \mu_{s,t}^2 p^*_{s,t} - \mu_1^2 \end{aligned} \tag{5.26}$$

Bibliography

[1] BARLOW, R. E., BARTHOLOMEW, D. J., BREMNER, J. M. and BRUNK, H. D. (1972): *Statistical inference under order restrictions; the theory and application of isotonic regression*. New York: Wiley. ISBN 0-471-04970-0.

[2] HOSMER D.W. and LEMESHOW S. (1980): *A goodness-of-fit test for the multiple logistic regression model*, Communications in Statistics vol. A10, pp. 1043–1069.

[3] TJUR, T. (2009): *Coefficients of Determination in Logistic Regression Models: A New Proposal: The Coefficient of Discrimination*, American Statistician vol. 63, pp. 366–372.

[4] IASB: *IFRS 9 Financial Instruments (replacement of IAS 39). Retrieved 20 November 2019, from* http://archive.ifrs.org/IFRSs/Pages/IFRS.aspx.

[5] XGBoost tutorial online: Available at: https://xgboost.readthedocs.io/en/latest/tutorials/model.html

[6] JEROME H. FRIEDMAN (2001): Greedy function approximation: A gradient boosting machine. *Ann. Statist. vol. 29 iss. 5, pp. 1189-1232. DOI: 10.1214/aos/1013203451*

[7] BREIMAN, L. (2001): Random Forests, *Machine Learning vol. 45, iss. 5–32. https://doi.org/10.1023/A:1010933404324*

[8] Gradient boosting example Available at: https://www.mygreatlearning.com/blog/gradient-boosting/