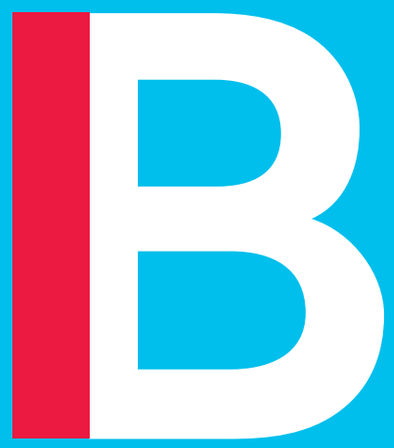
Imperial College
Business School

Imperial means
Intelligent Business

Machine Learning in Finance

Lecture 1
Machine Learning – Setting Up the scenes



Arnaud de Servigny & Hachem Madmoun

Outline:

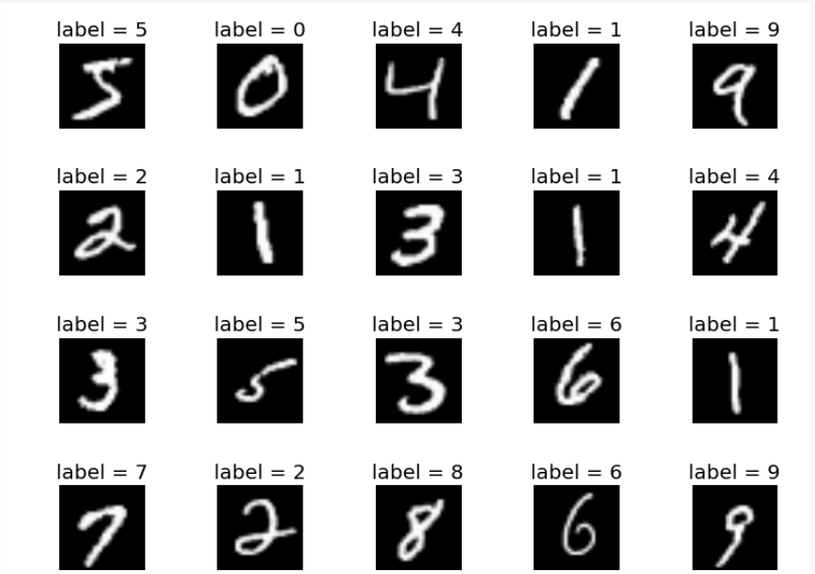
- The various typologies of Machine Learning approaches
- Best practices in terms of Data Handling
- Getting the best fit -A short review of optimisation concepts
- Evaluation Metrics for Classification
- Programming Session

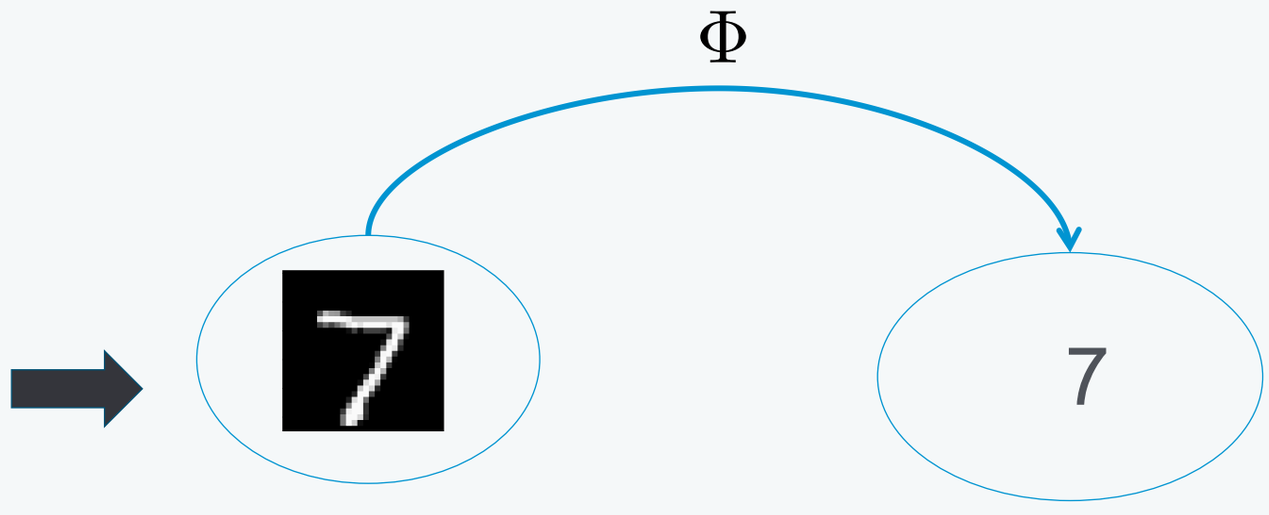
The various typologies of Machine Learning approaches

Supervised Learning – Unsupervised Learning

- Supervised Learning is the process of learning a function which maps features to an output based on several input-output pairs

MNIST Labeled dataset





- Unsupervised Learning is the process of identifying meaninful patterns in unlabeled data

MNIST Unlabeled dataset

2 dimensional representation of the MNIST dataset using a variational autoencoder (example of Unsupervised algorithm). Each color represents a number

Supervised Learning – Unsupervised Learning

Typology of machine-learning problems / Defining the problem

Generally there are two main types of machine learning problems: **supervised and unsupervised**.

Supervised machine learning problems

- There are problems where we want to make predictions using data (features) based on pre-defined targets (labels).
- In supervised machine learning, you feed the features and their corresponding labels into an algorithm in a process called training. During training, the algorithm gradually determines the relationship between features and their corresponding labels. This relationship is called the model.
- A useful locution to understand what is actually going on in practice is “Pattern Recognition”
- There are in fact two main types of supervised learning problems: classification which involves predicting a class label and regression which involves predicting a numerical value.
  - ✓ **Classification**: Supervised learning problem that involves predicting a class label.
  - ✓ **Regression**: Supervised learning problem that involves predicting a numerical label.

The wording” supervised learning” originates from the perspective of the target being provided by an “instructor” who teaches the machine learning algorithm what to do.

Supervised Learning – Unsupervised Learning

Unsupervised machine learning problems

- There are problems where the data does not have a defined set of categories, but instead we are looking for the machine-learning algorithms to help us organize it.
- In unsupervised machine learning, the goal is to identify meaningful patterns, i.e. describe or extract relationships in the data. To accomplish this, the machine must learn from an unlabelled data set. In other words, the model has no hints how to categorize each piece of data and must infer its own rules for doing so. It is all about making sense of the data.

In practice, we are talking about a variety of possible outcomes:

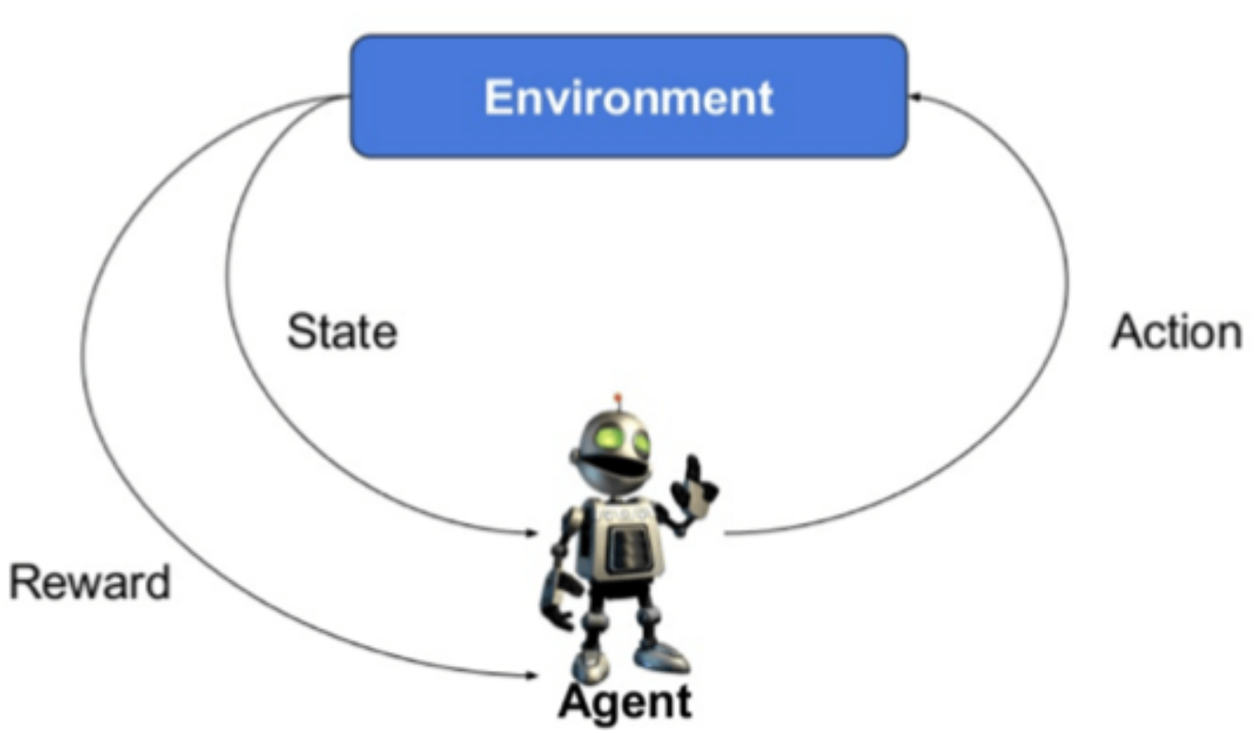
- **Clustering**: a problem which involves finding groups in the data.
- **Density Estimation**: a problem which involves summarizing the distribution of the data.
- **Visualization**: a problem which involves creating plots of data.
- **Projection**: a problem which involves creating lower-dimensional representations of the data.

This being said, it is more accurate to describe ML problems as falling along a spectrum of supervision between supervised and unsupervised learning. Let us not be too dependant on the above segmentation!

An example of a mixed approach: Reinforcement Learning

Reinforcement learning describes a class of problems where an agent operates in a game-like environment and must learn from a trial and error iterative process, using some reward feedback.

Reinforcement learning differs from the supervised learning approach in the sense that in supervised learning the training dataset provides the rule for the decision, i.e. a model is trained with the correct answer before being applied to unknown data. In reinforcement learning instead, there is no modelled answer but the reinforcement agent has to decide what to do to perform the given task. In the absence of a training dataset, it has to learn sequentially from experience.



Financial problems typically solved using Machine Learning

1. Classification: e.g. good / bad credit.
2. Risk prediction: likelihood to default type of scoring
3. Learning Associations: which type of client tends to feel comfortable with a kind of financial product (customer analytics).
4. Digital assistant: answering questions from clients
5. Extracting signals from a large spectrum of data repositories: removing noise (feature engineering)
6. Fraud detection: identifying anomalous behaviours
7. Investment prediction: inferring the dynamics of asset prices
8. Portfolio Management: making sensible asset allocation decisions
9. Document Analysis: extracting meaningful words / sentences; getting information on sentiment.
10. Model assessment and stress testing (Reinforcement Learning)

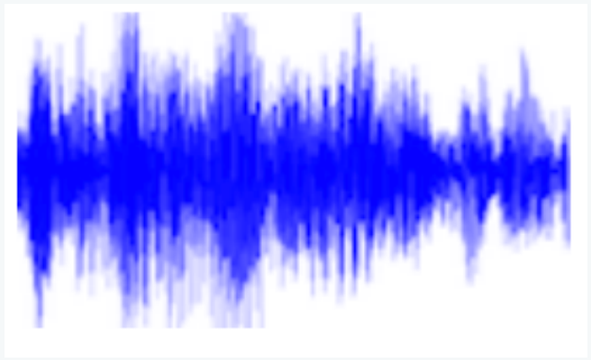
Best practices in terms of Data Handling

Best practices in terms of Data Handling – Preprocessing –

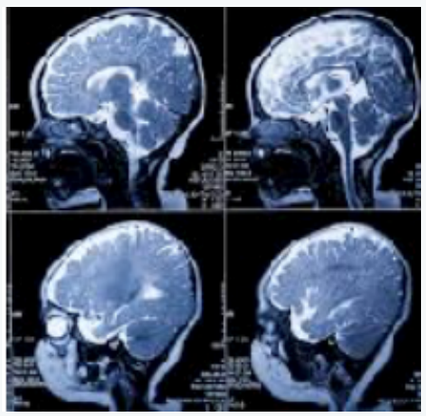
- Before diving into the algorithms of supervised and unsupervised learning, it is important to understand the best practices of building a good machine learning algorithm.

- The road map for building machine learning systems can be summarized as follows:

    - **Preprocessing :** Machine Learning algorithms can only be applied to numbers, but data can be of different forms : text, audio signal, images, etc. Moreover, it is usually important to transform the data (using unsupervised learning for instance) before feeding it to a supervised algorithm. We will illustrate it with several examples in future lectures.



Audio



Image

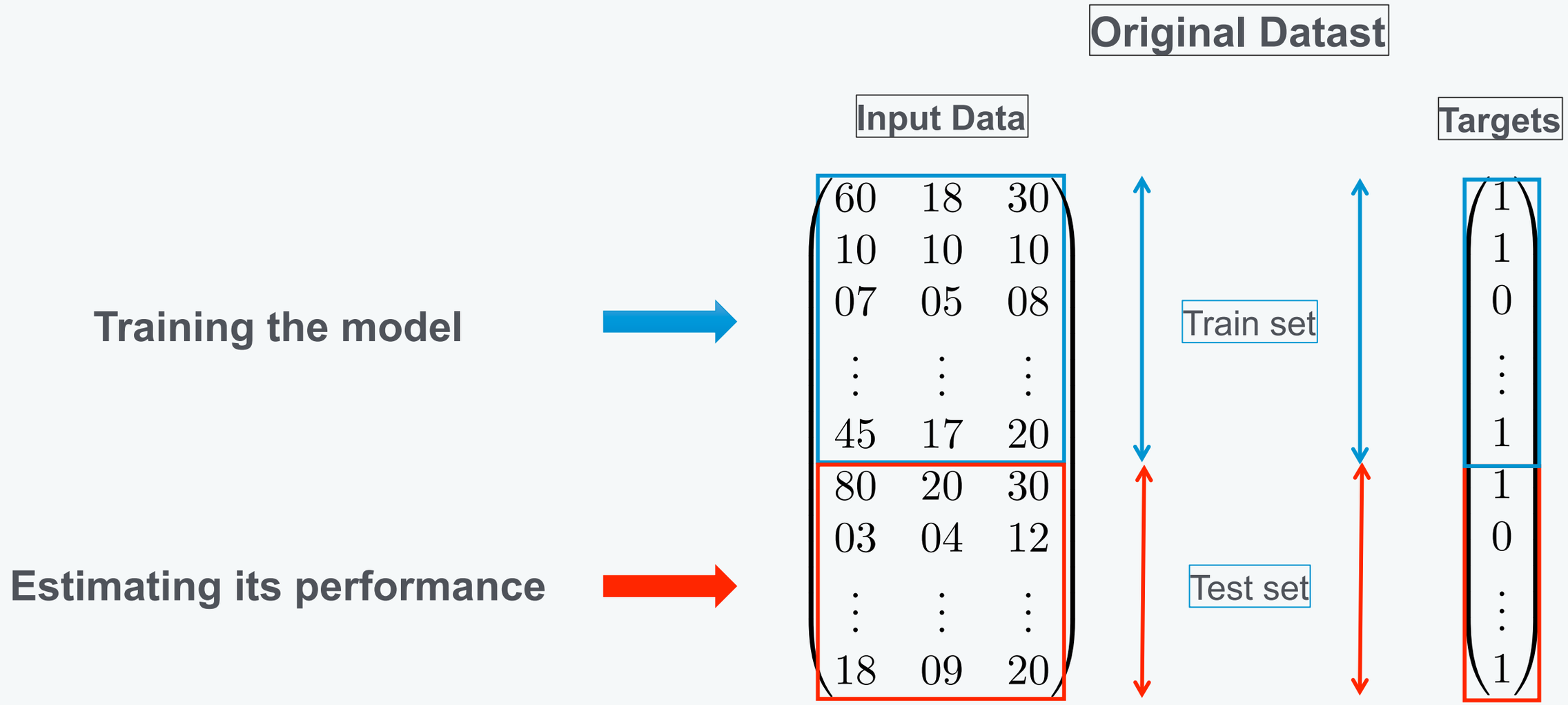
Texts

Social network

Best practices in terms of Data Handling – Splitting the dataset (The holdout method) –

- One of the key steps in building a machine learning model is to estimate its performance on data that the model hasn't seen before.

- To that end, a classic approach is to split the original dataset into a training set (used for training the model) and a test set (used to evaluate the generalization performance of the model).



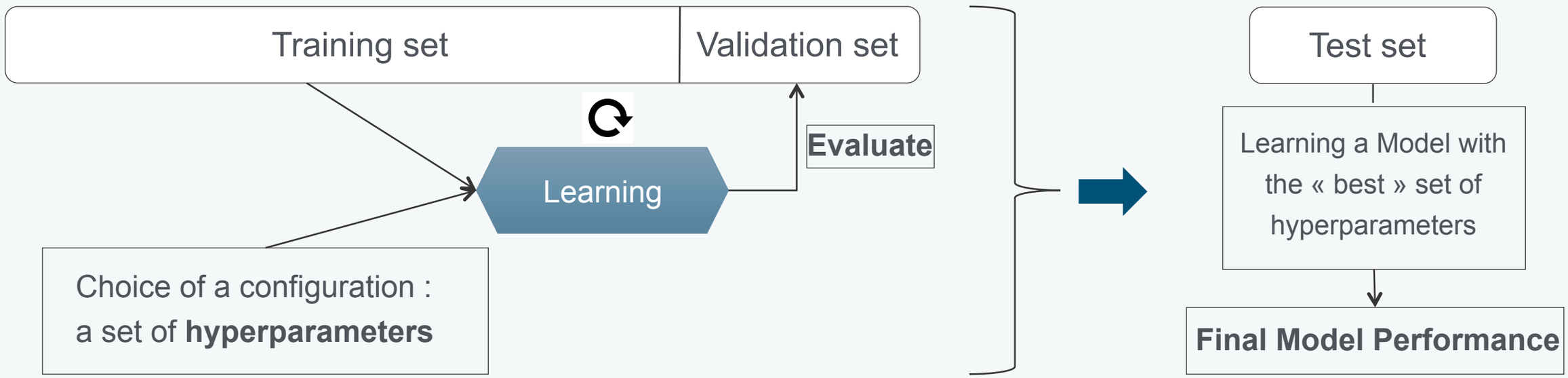
Best practices in terms of Data Handling – Splitting the dataset into Train / Validation / Test sets –

- This time, instead of training the model on the train set and testing it on the test set, the original dataset is partitioned into three sets : training, validation and test sets.

- The reason for adding a validation set is that deploying a model always involves tuning its configuration. For instance, we will see that creating a neural network requires choosing the number of layers, the number of neurons for each layer, etc. These choices are called hyperparameters.

- As a result, using the performance on the validation set to change the configuration results in information leaks: Indeed, information about the validation data leaks into the model.

- As we care about the performance on completely new data, we use the test set as a never-before-seen dataset to evaluate the model.



Best practices in terms of Data Handling – Splitting the dataset (The K-fold cross validation for small **non-sequential** datasets) –

- The previous splitting strategy suffers from one major flaw: if little data is available, then the validation and test set contain very few data points.

- As a result, for the same trained model, shuffling the data before splitting it ends up yielding different performance measures. K fold cross validation is a way of addressing this issue.

K fold cross validation:

- The training set is divided into K folds

- During K iterations, we use one fold for testing and we train the model on the K-1 other folds.

- We end up with a distribution of performance measures.

- By averaging the estimated performances on each test fold, we end up with the average performance of the model.

The Machine Learning Workflow – Learning and Predicting –

- **Learning and Model Evaluation**:
    - During this step, we first need to decide upon the metric to measure performance.
    - The choice of the most suitable evaluation metric will depend on the nature of the problem (classification or regression) and also on the nature of the dataset (balanced or imbalanced).
    - The next section will detail the different evaluation metrics for different contexts.
    - But how do we know which model to learn ? As David Wolpert's famous ***no free lunch theorem*** suggests, there is no model that works best for every problem, it is therefore essential to compare a handful of different models with different sets of hyperparameters.

- **Predicting new data**:
    - After we have selected the « best » model that has been fitted on the training set, we can use the test set to estimate how well it performs on the unseen data.
    - If we are satisfied with the generalization error, we can use the model to predict new data.

Getting the best fit

A short review of optimisation concepts

Maximum Likelihood Estimation

Position of the problem:

Tossing a Biased Coin

Given a biased coin that comes up heads with some probability greater than 1/2. How can we determine the bias parameter (i.e the probability of getting heads) using n flips ?

The Coronavirus Curve

Given the number of infection cases for n different days in a particular geography. How can we assess whether the distribution of the number of cases per day is flat enough to cope with the healthcare system capacity ?

Understanding the problem assuming the Statistical Model is known

- In the case of tossing a coin, we assume that this process is well characterised using a probabilistic model.

- Examples:

  - **Bernoulli model:** $X \sim \mathcal{B}(\theta)$ where $\theta \in [0,1]$:

$$\forall x \in \{0,1\} \quad p_\theta(x) = \mathbb{P}(X = x) = \theta^x (1-\theta)^{1-x}$$

The Bernoulli distribution models the outcome of a single binary output trial (success or failure), it typically models whether flipping a coin one time will result in heads or tails.

- **Binomial distribution**

$$\forall x \in \{0, \dots, n\} \quad p_\theta(x) = \mathbb{P}(X = x) = \binom{n}{x} \theta^x (1-\theta)^{1-x} = \frac{n!}{x!(n-x)!} \theta^x (1-\theta)^{1-x}$$

The Binomial distribution models the outcome of n trials. More specifically, it models the number of times the output is a success from performing n independent identically distributed Bernoulli trials.

Estimating the fitting parameters using a Maximum Likelihood approach

- We use a collection of observations to **learn** (or estimate) the parameters of a statistical model.

- These observations $X_1, \ldots, X_n$ are called samples in statistics or training set in Machine Learning.

- Usually, we make the assumption that the variables are i.i.d., which means independent identically distributed.

- Let $\mathcal{P}_\Theta = \{p(x;\theta) \mid \theta \in \Theta\}$ be the model and x the observation. We wish to estimate $\theta \in [0,1]$ from the training set.

- We should note here that again this measure of likelihood is not unique as it deals with the specific case where a good fit carries the same weight as a bad fit.

- In order to get there, we define a **likelihood function** $\mathcal{L}$, which gives an idea of the ability of the retained model to account for the observations:

$$\mathcal{L} : \Theta \to \mathbb{R}_+$$

$$\theta \mapsto p(x, \theta)$$

Maximum Likelihood Estimation

- The **maximum likelihood estimator** (MLE) is defined as the θ^* which maximizes the likelihood:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\ p(x; \theta)$$

- When the training set consists in n i.i.d. samples, we optimize the following **log-likelihood**:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}} \prod_{i=1}^{n} p(x_i; \theta) = \underset{\theta \in \Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log(p(x_i; \theta))$$

- The MLE:
  - does not always exist.
  - in not necessarily unique
  - It assumes a certain form of utility function

- We will see the example of the Bernoulli statistical model, where the MLE can be determined in a closed form.

Maximum Likelihood Estimation for the Bernoulli Model

Tossing a Biased Coin

Given a biased coin that comes up heads with some probability greater than one-half. How can we determine the bias parameter (i.e the probability of getting heads) using n flips ?

- Our training dataset consists in n observations $X_1, \ldots, X_n$ i.i.d. $\sim \mathcal{B}(\theta)$.

- We consider getting heads as being the success. Which means $X_i = 1$ if the i-th coin toss results in heads.

$$\mathbb{P}(X_i = 1) = \theta$$

- The log-likelihood is expressed as follows:

$$L(\theta) = \sum_{i=1}^{n} \log(p(x_i; \theta)$$

Maximum Likelihood Estimation for the Bernoulli Model

- The log-likelihood can be written as follows:

$$
\begin{aligned}
L(\theta) &= \sum_{i=1}^{n} \log(p(x_i;\theta) \\
&= \sum_{i=1}^{n} \log(\theta^{x_i}(1-\theta)^{1-x_i}) \\
&= \sum_{i=1}^{n} x_i \log(\theta) + (1-x_i)\log(1-\theta) \\
&= \left(\sum_{i=1}^{n} x_i\right)\log(\theta) + \left(n - \sum_{i=1}^{n} x_i\right)\log(1-\theta)
\end{aligned}
$$

- The log-likelihood is strongly concave, which implies the existence and uniqueness of the MLE.

Maximum Likelihood Estimation for the Bernoulli Model

- Since the log-likelihood is differentiable and strongly concave, its maximizer is its unique stationary point:

$$\frac{dL}{d\theta}(\theta) = 0 \iff \left(\sum_{i=1}^{n} x_i\right)\log(\theta) + \left(n - \sum_{i=1}^{n} x_i\right)\log(1-\theta)$$

$$\iff \frac{\sum_{i=1}^{n} x_i}{\theta} = \frac{n - \sum_{i=1}^{n} x_i}{1-\theta}$$

$$\iff \theta = \frac{\sum_{i=1}^{n} x_i}{n}$$

- The optimal parameter is then:

$$\theta^* = \frac{\sum_{i=1}^{n} x_i}{n}$$

Maximum Likelihood Estimation for a Categorical Model:

Interactive Session

Evaluation Metrics for classification

Classification Metrics:

- During this lecture, we will focus on **binary classification,** which consists in classifying the input data into two possible categories.

- For instance, predicting whether the market is going up or down is a binary classification task. So is predicting if a student is going to pass or fail an exam. In order to make a decision, we have to introduce a "cut-off point" to discriminate between the two predicted classes.

- By convention, one of the two classes is called the **positive class** (output = 1) and the other one is called the **negative class** (output = 0).

- In the next section, we will introduce some classification metrics used to assess the performance of a classifier:
  - The Accuracy Score.
  - The Confusion Matrix
  - The Receiver Operator Characteristic (ROC) graph and the Area Under the Curve (AUC).

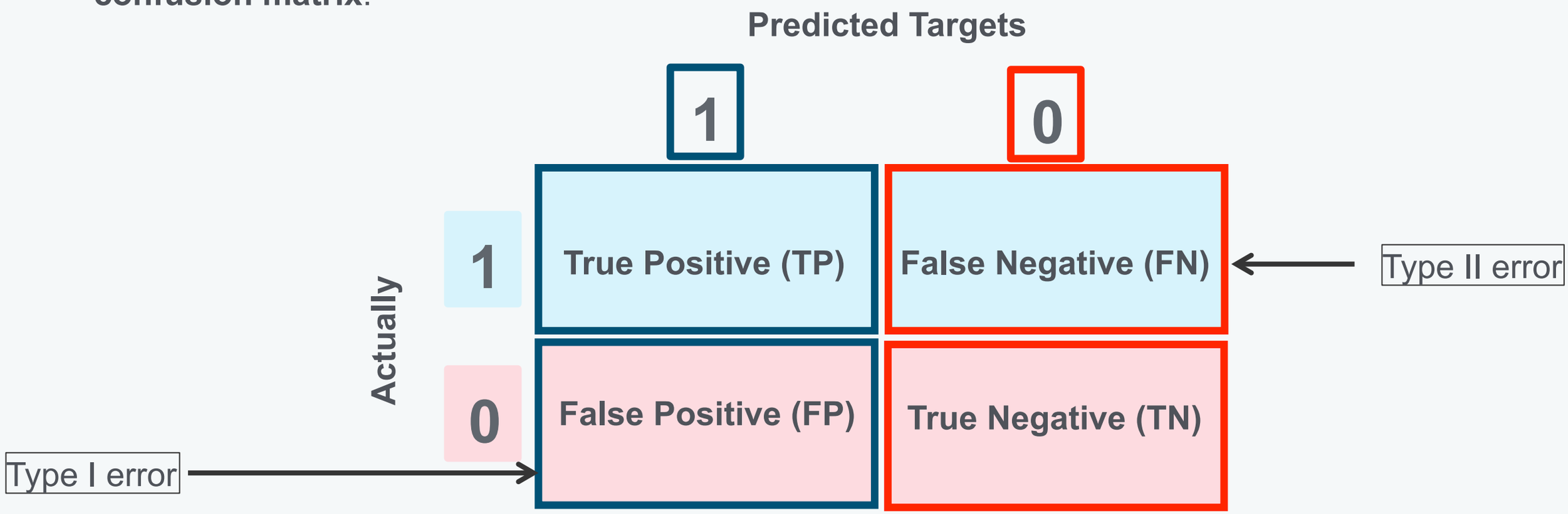
Classification Metrics for Binary Classification :

- The easiest way to evaluate a classifier is to determine the **Accuracy Score**, which is the fraction of the train test correctly classified.

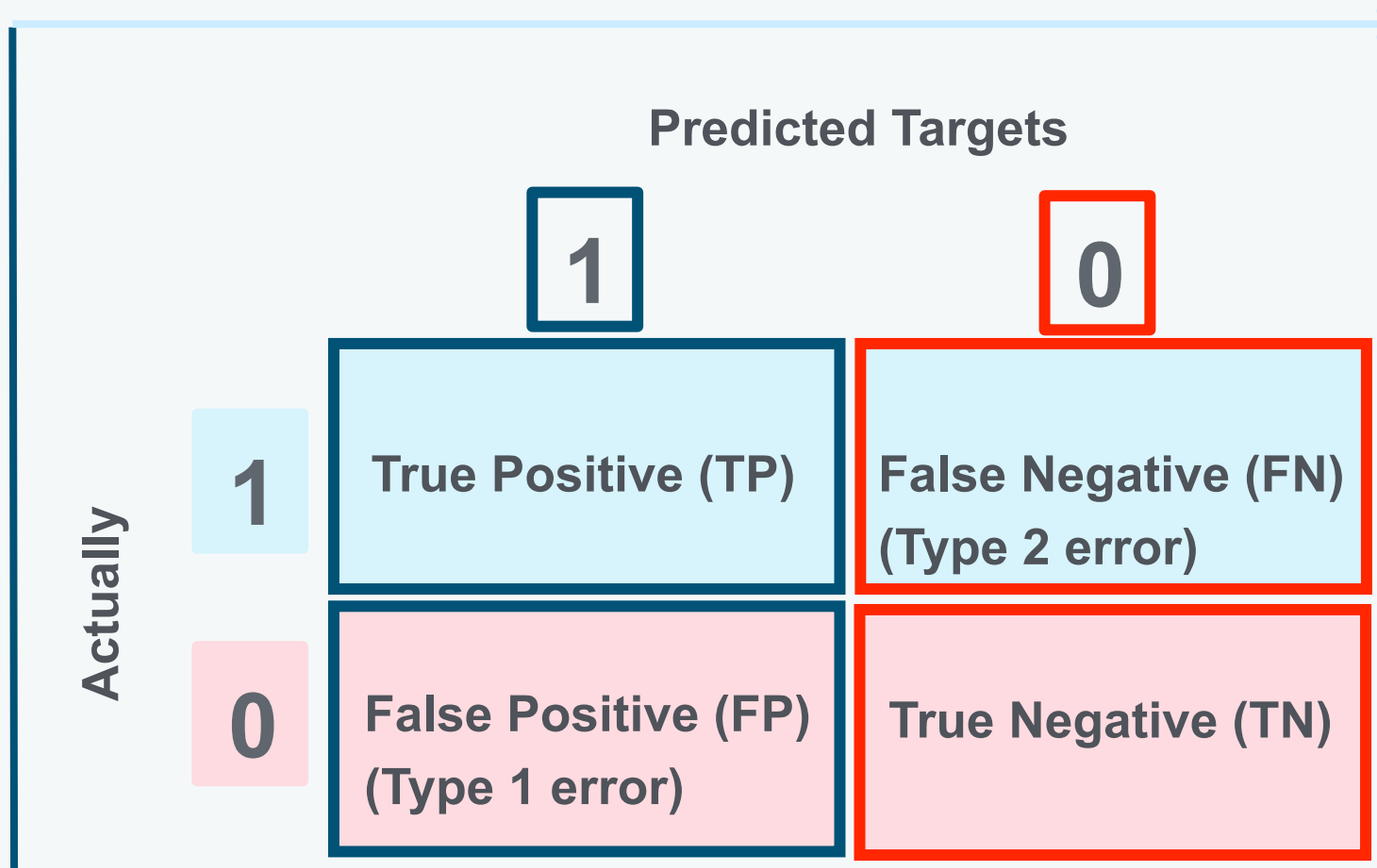
$$\text{Accuracy Score} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

- If the dataset contains 99% of positive labels and 1% of negative labels, a naive classifier which predicts always the positive label will have a very high accuracy score, which is problematic !

- So, the accuracy score should not be used for imbalanced datasets. Thus, we introduce the following **confusion matrix**:



Classification Metrics for Binary Classification :



Recall

- Given a class, will the classifier detect it ?

$$\textbf{Recall} = \frac{\text{TP}}{\text{Actual Positives}}$$

$$= \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Recall considers FN as the worst errors

Precision

- Given a class prediction from the classifier, how likely is it to be correct ?

$$\textbf{Precision} = \frac{\text{TP}}{\text{Predited Positives}}$$

$$= \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision considers FP as the worst errors

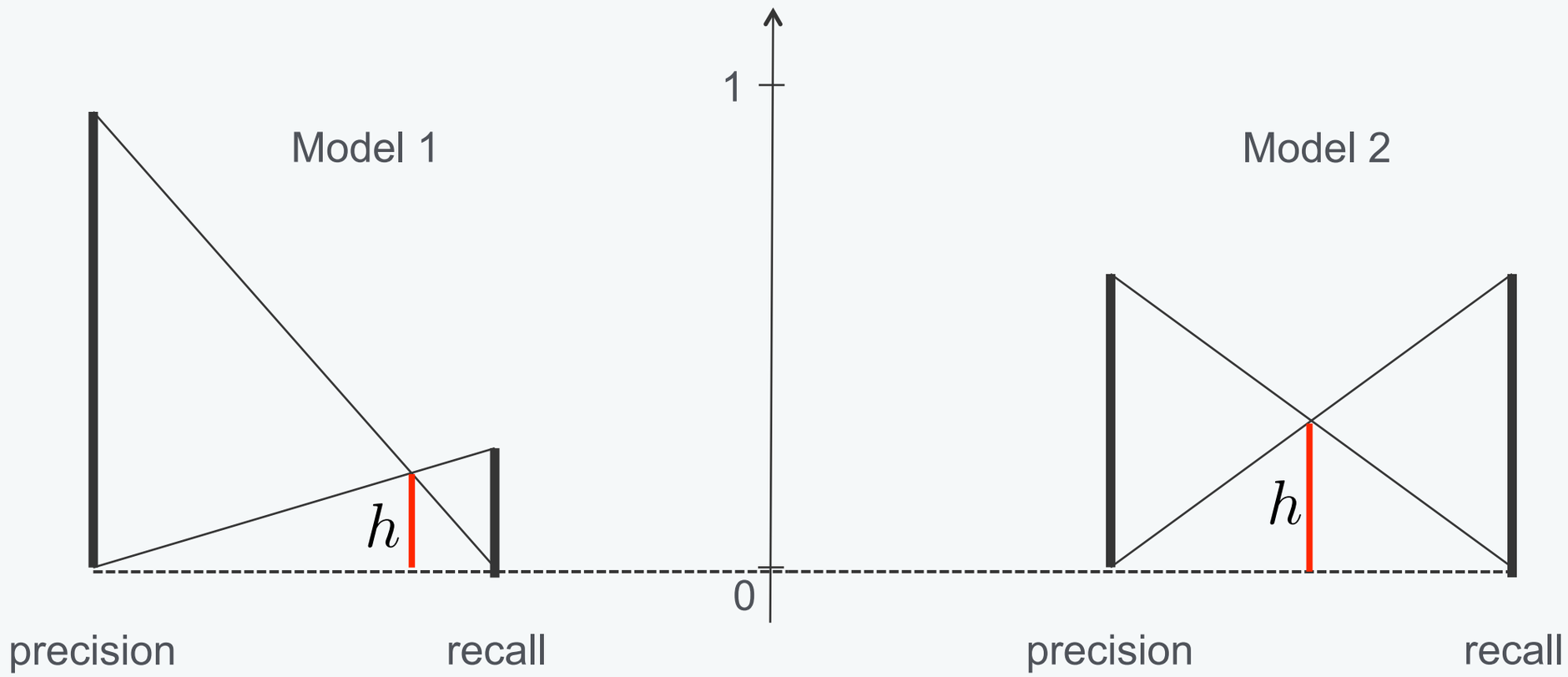
Classification Metrics for Binary Classification – Part 2 – :

- The **F1 Score** is just the **harmonic mean** of recall and precision.

$$\mathbf{F1} = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- As shown in the figure below, the F1 score punishes the extreme values: Model 2 is better than Model 1.

- The F1 score should be used for **Imbalanced datasets** instead of the accuracy score.



h is half the harmonic mean of recall and precision (i.e, the F1 score)

An example of an Imbalanced dataset – Part 1 –

- The objective is to create a predictor for the final exam of a class full of exceptional pupils. The notation system consists in 4 different labels : A (the best) – B (less good) – C (bad) – D (the worst).

- As we are dealing with gifted students, the distribution of the actual results is the following:

A	B	C	D
200 students	10 students	10 students	10 students

- We want to compare two models:
  - The first model is a very optimistic model. It predicts a lot of A labels.
  - The second model has a more spread out distribution over the different labels.

- The first model obviously beats the second one in term of accuracy (since it predicts a lot of A labels and the dataset contains 83% of A labels).

- But we know that the accuracy score is not a good evaluation metric for this kind of problems.

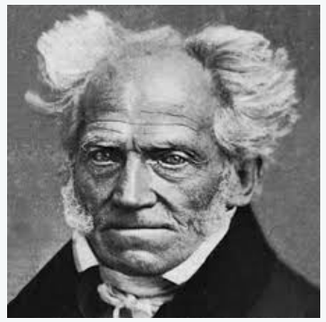
- Thus, we will compare the two models based on their confusion matrix and F1 score.

An example of an Imbalanced dataset – Part 2 –

Confusion Matrix for each Model

First Model
(Optimistic one)

Second Model
(Pessimistic one)



Predictions

Actual Values	A	B	C	D
A	198	2	0	0
B	7	1	0	2
C	0	8	1	1
D	2	3	4	1

$$\textbf{Accuracy Score} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = 0.87$$

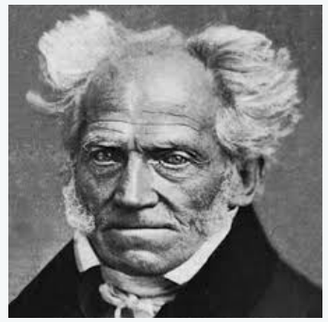
Predictions

Actual Values	A	B	C	D
A	100	80	10	10
B	0	9	0	1
C	0	1	8	1
D	0	1	0	9

$$\textbf{Accuracy Score} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = 0.54$$

An example of an Imbalanced dataset – Part 3 –

Precision for each Model



$$\textbf{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Predictions

Actual Values	A	B	C	D
A	198	2	0	0
B	7	1	0	2
C	0	8	1	1
D	2	3	4	1

TP : 198 FP : 9	TP : 1 FP : 13	TP : 1 FP : 4	TP : 1 FP : 3

Predictions

Actual Values	A	B	C	D
A	100	80	10	10
B	0	9	0	1
C	0	1	8	1
D	0	1	0	9

TP : 100 FP : 0	TP : 9 FP : 82	TP : 8 FP : 10	TP : 9 FP : 12

$$\textbf{Average Precision} = \frac{\text{Precision A} + \text{Precision B} + \text{Precision C} + \text{Precision D}}{4}$$

$$\textbf{Average Precision(First Model)} = 0.36$$

$$\textbf{Average Precision(Second Model)} = 0.49$$

An example of an Imbalanced dataset – Part 4 –

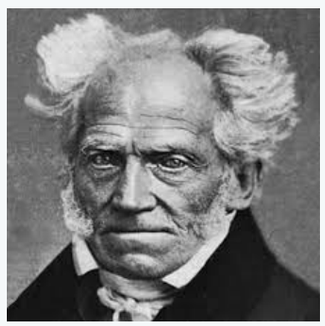
Recall for each Model

Predictions

Actual Values	A	B	C	D	
A	198	2	0	0	TP = 198 / FN = 2
B	7	1	0	2	TP = 1 / FN = 9
C	0	8	1	1	TP = 1 / FN = 9
D	2	3	4	1	TP = 1 / FN = 9

Average Recall (First Model) = 0.32

$$\textbf{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$



Predictions

Actual Values	A	B	C	D	
A	100	80	10	10	TP = 100 / FN = 100
B	0	9	0	1	TP = 9 / FN = 1
C	0	1	8	1	TP = 8 / FN = 2
D	0	1	0	9	TP = 9 / FN = 1

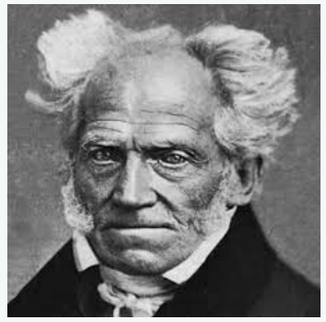
Average Recall (Second Model) = 0.77

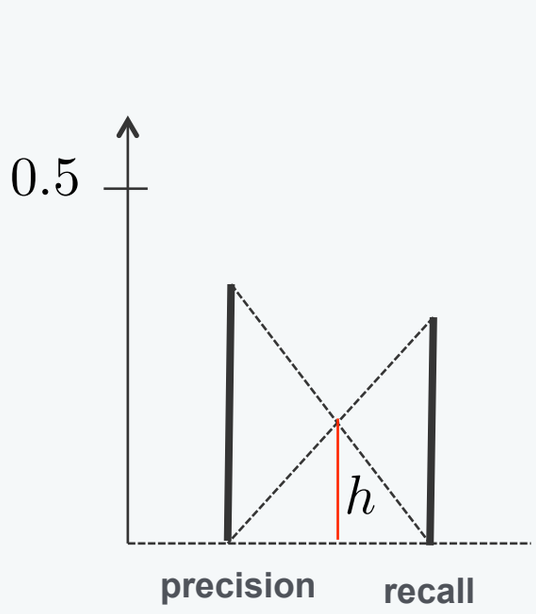
An example of an Imbalanced dataset – Part 5 –

First Model

Comparing the two Models

Second Model

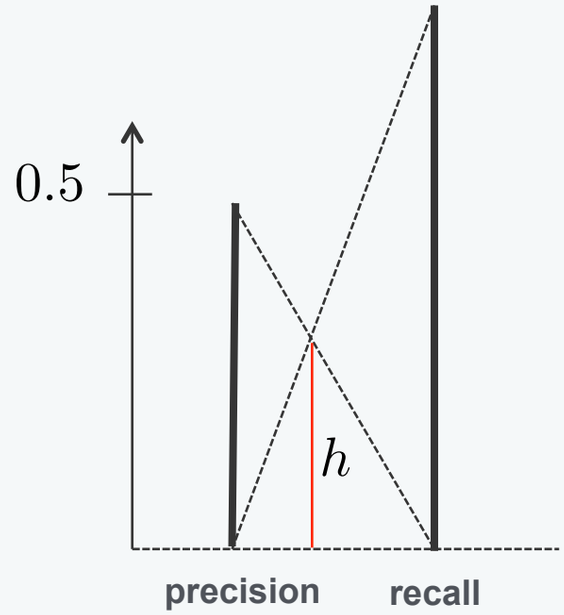




Reminder

$$\mathbf{F1} = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$h = \frac{F1}{2}$$



0.87

0.33

👎

Accuracy Score

F1 Score

0.54

👍

0.60

By calculating the F1 score, we realize that the second Model (the pessimistic one) has a better performance than the first one (the Optimistic one) on this imbalanced dataset.

The Receiver Operator Characteristic (ROC) graph

- During the prediction phase (after the training process), most of the classifiers output a more precise information than just the class label for each new data point x^* . They also output the probability p that this data point belongs to the positive class and, a fortiori, the probability of belonging to the negative class (which is $1 - p$)

Test set features

`Trained classifier

Predictions

True Targets

$$\begin{pmatrix} 80 & 20 & 30 \\ 03 & 04 & 12 \\ \vdots & \vdots & \vdots \\ 18 & 09 & 20 \end{pmatrix}$$

$$\begin{pmatrix} 0.99 \\ 0.23 \\ \vdots \\ 0.47 \end{pmatrix} \qquad \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

- How do we go from continuous predictions to discrete ones ?

- We will explain this on the example of Logistic Regression.

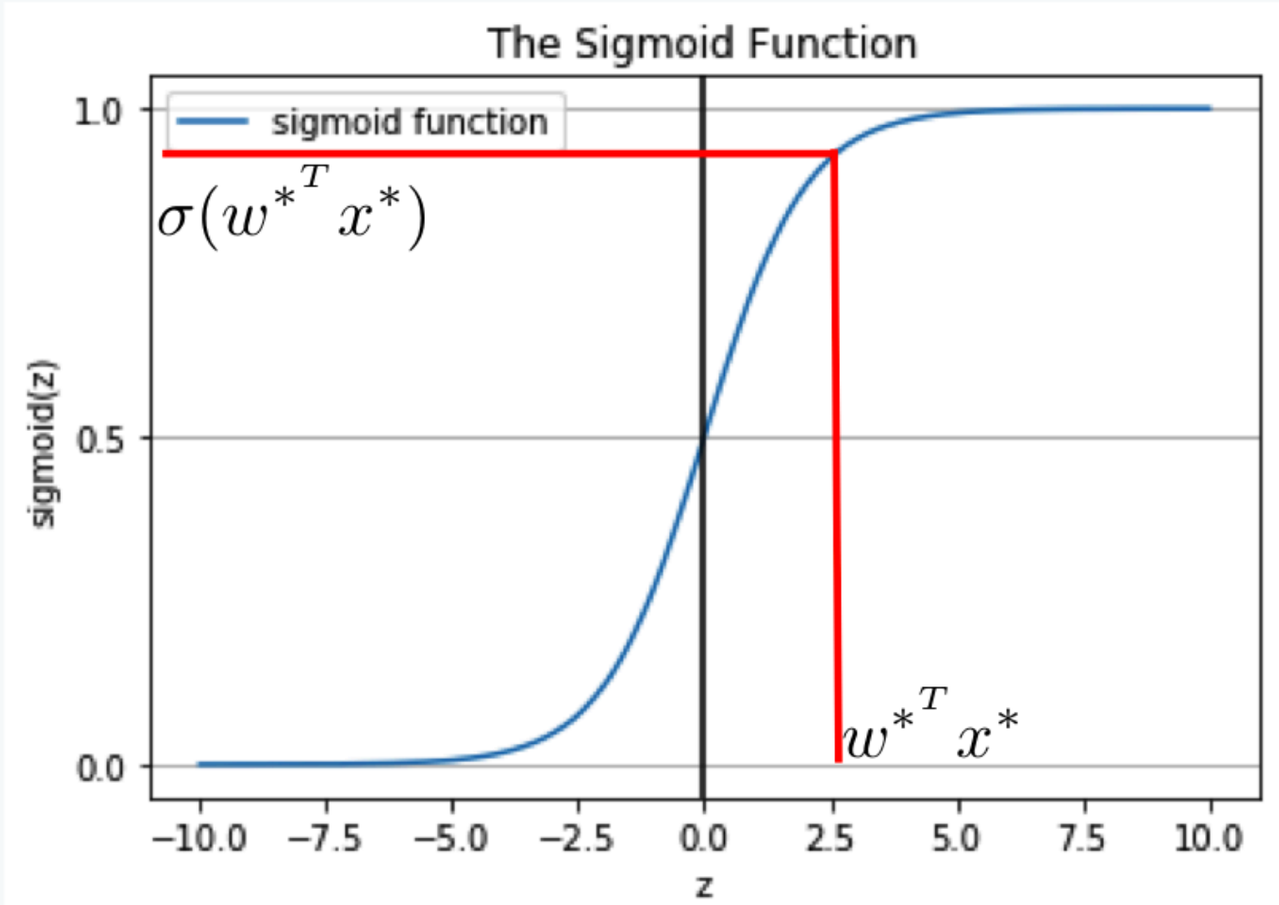
The Receiver Operator Characteristic (ROC) graph

- Logistic regression predicts the conditional distribution of the label $Y^* \in \{0, 1\}$ given the feature vector $x^* \in \mathbb{R}^D$ as follows:

$$P(Y^* = 1 \mid X^* = x^*) = \sigma({w^*}^T x^*)$$

Determined by training the model on the training set

where σ refers to the **sigmoid function** $\sigma : z \mapsto \frac{1}{1 + e^{-z}}$

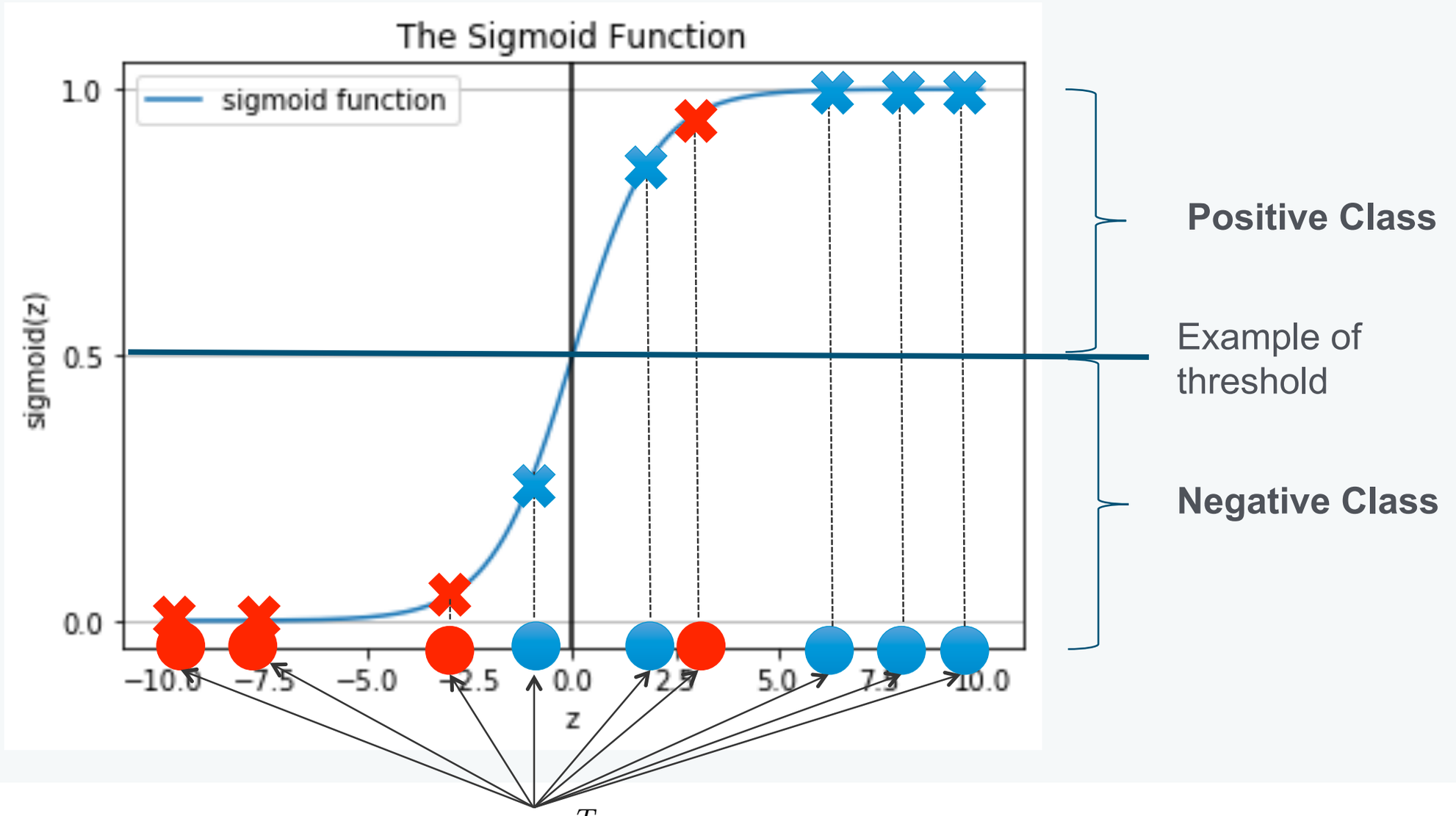


The Receiver Operator Characteristic (ROC) graph

- Let's consider 9 pairs $(x_i, y_i)_{i \in \{1,\dots,9\}}$ in our test set. (The blue point are associated to the positive class and the red ones are associated to the negative class).

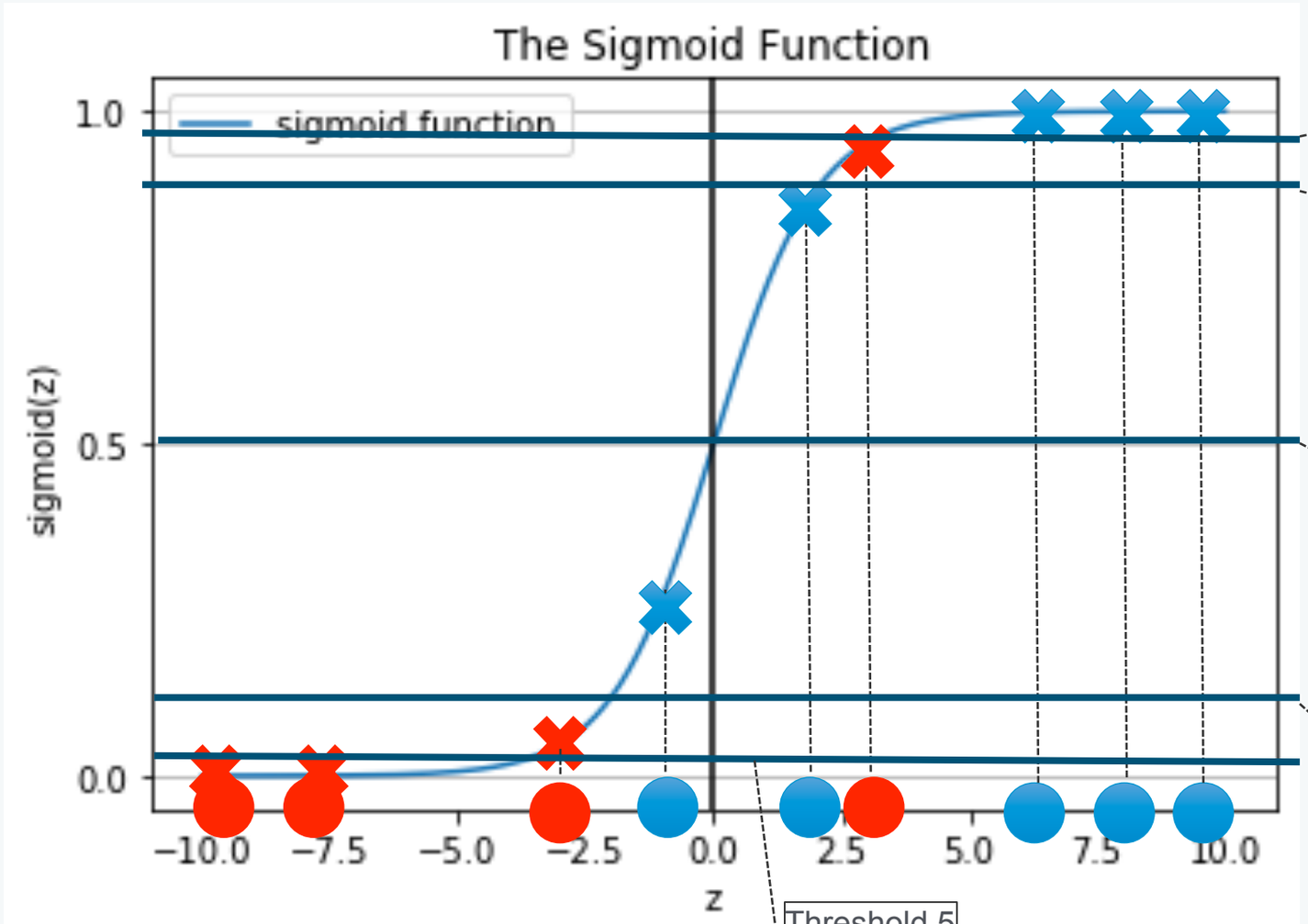
- As the model has been trained, we can generate the predictions $\forall i \in \{1, \dots, 9\} \quad p_i = \sigma({w^*}^T x_i)$

- To turn the predictions to class labels, we need to define a threshold above which we consider the points as belonging to the positive class.



$$z_i = {w^*}^T x_i$$

The Receiver Operator Characteristic (ROC) graph



Threshold 1

		Predicted 0	Predicted 1
Actual	0	4	0
	1	2	3

Threshold 2

		Predicted 0	Predicted 1
Actual	0	3	1
	1	2	3

Threshold 3

		Predicted 0	Predicted 1
Actual	0	3	1
	1	1	4

Threshold 4

		Predicted 0	Predicted 1
Actual	0	3	1
	1	0	5

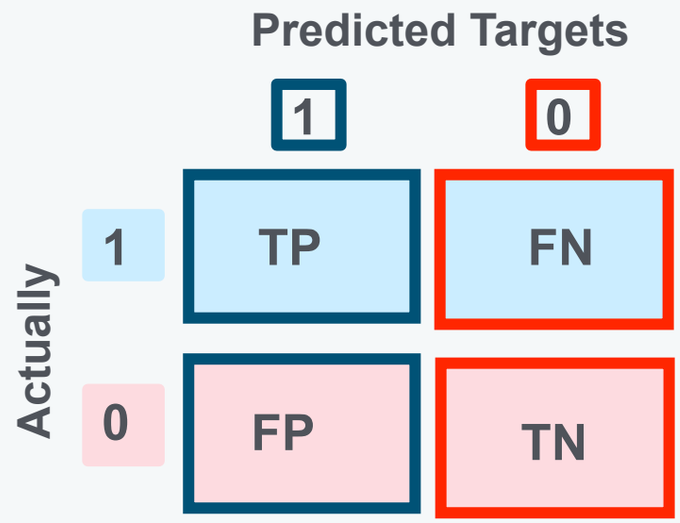
Threshold 5

		Predicted 0	Predicted 1
Actual	0	2	2
	1	0	5

The Receiver Operator Characteristic (ROC) graph

- In order to plot the ROC graph, we first need to define the True Positive Rate and the True negative Rate:

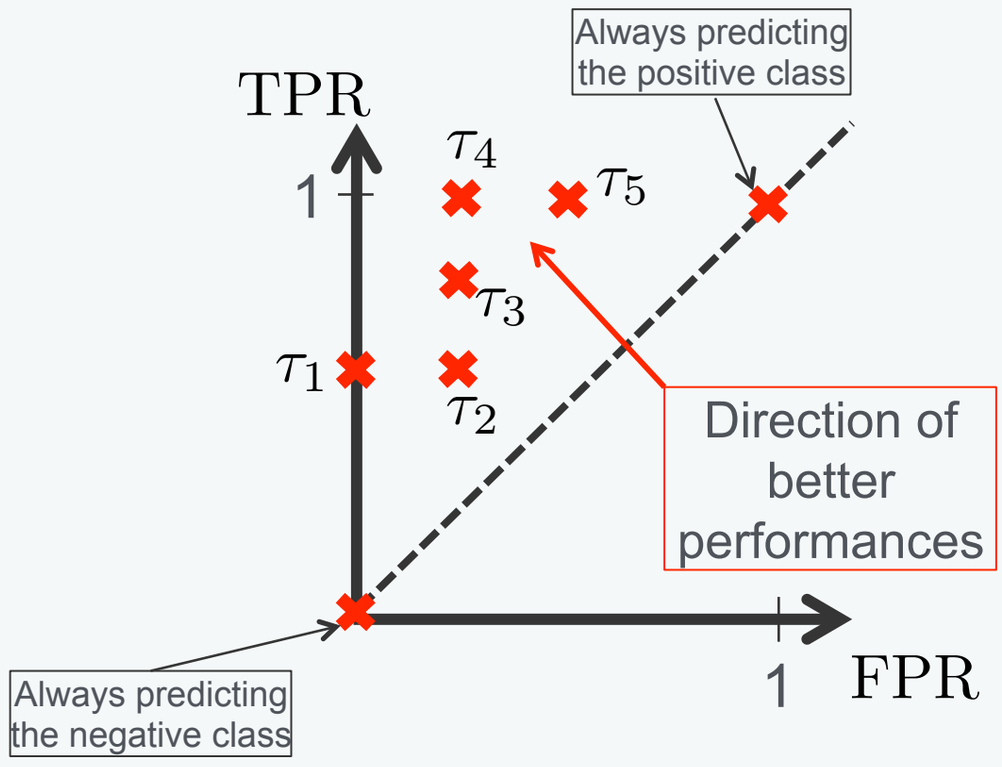
$$\text{True Positive Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$



$$\text{False Positive Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

- We can then plot the True Positive Rate (TPR) vs the False Positive Rate (FPR), over all the thresholds.

- The different thresholds are denoted $\tau_1, \ldots, \tau_5$

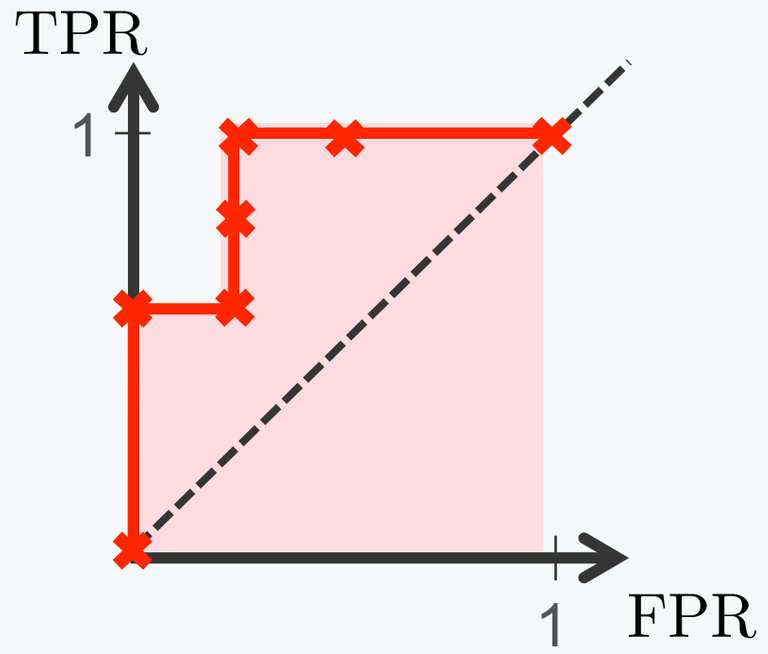


	TPR	FPR
τ_1	0.6	0
τ_2	0.6	0.25
τ_3	0.8	0.25
τ_4	1	0.25
τ_5	1	0.5

The Area Under the Curve (AUC)

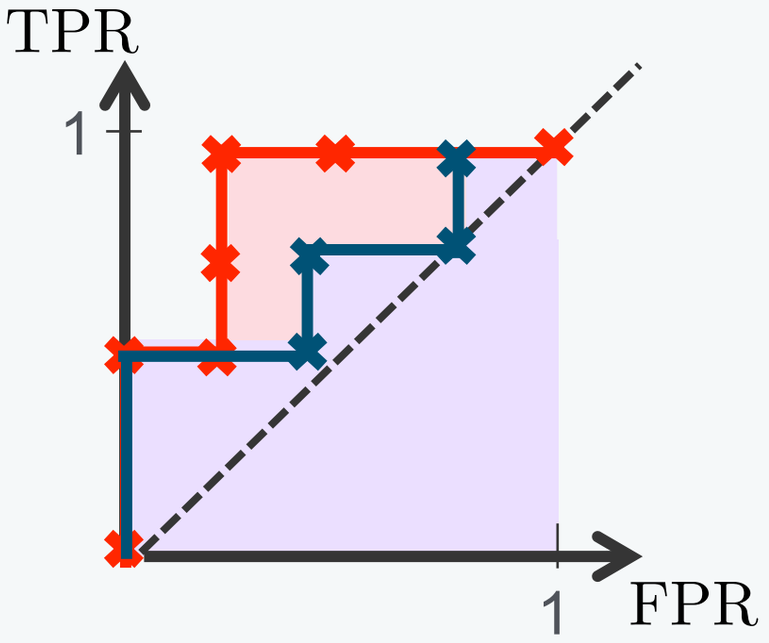
- The area under the ROC curve (called AUC) evaluates the model at all possible cut-off points.

- The AUC, by summarizing the ROC curve in one measure, gives better insights about how well the classifier is able to separate the positive and negative classes.



- If the blue ROC curve represents an other model (Random Forest for instance), we can see from the figure below that the red ROC curve (representing Logistic Regression) is better.

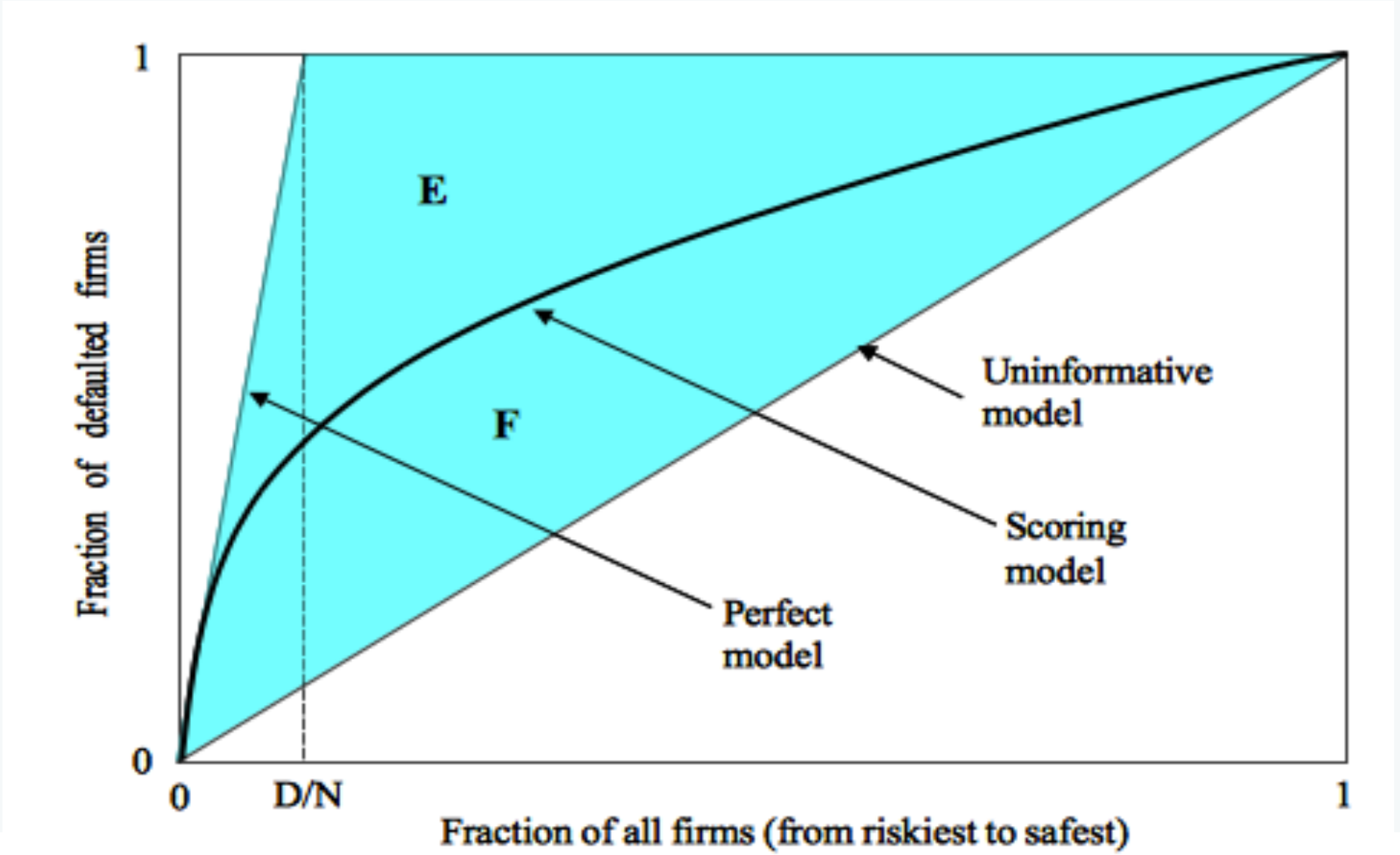
- The higher the AUC, the better the model.



The Gini coefficient (Gini)

Let us consider the case of a default model, where we aim at identifying defaulting firms from features.

- So far we have considered the AUC. A subset of it is useful to compute the Gini coefficient.
- Gini = (Area F)/(Area E). Gini = 2*ROC - 1
- The upper bound is 100%.
- Models with better ranking produce higher Gini coefficients
- NOTE: these Gini coefficients, like ROC are dataset specific and cannot be compared across datasets



Going beyond simple Logit models

- Logistic regression predicts the conditional distribution of the label $Y^* \in \{0, 1\}$ given the feature vector $x^* \in \mathbb{R}^D$ as follows:

$$P(Y^* = 1 \mid X^* = x^*) = \sigma({w^*}^T x^*)$$

Note that instead of looking for simple weights w^*, it is possible to think of an unknown function we wish to uncover, but given the size of our dataset, we limit ourselves to the second order of its Taylor expansion.

While still considering a logistic transformation, we have to estimate the parameters of a quadratic function, hence the denomination of “Quadratic Logit”

$$\beta_0 + \sum_{i=1}^{K} \beta_i x_i \rightarrow \beta_0 + \sum_{i=1}^{K} \beta_i x_i + \sum_{i=1}^{K} \sum_{j=1}^{K} \gamma_{ij} x_i x_j$$

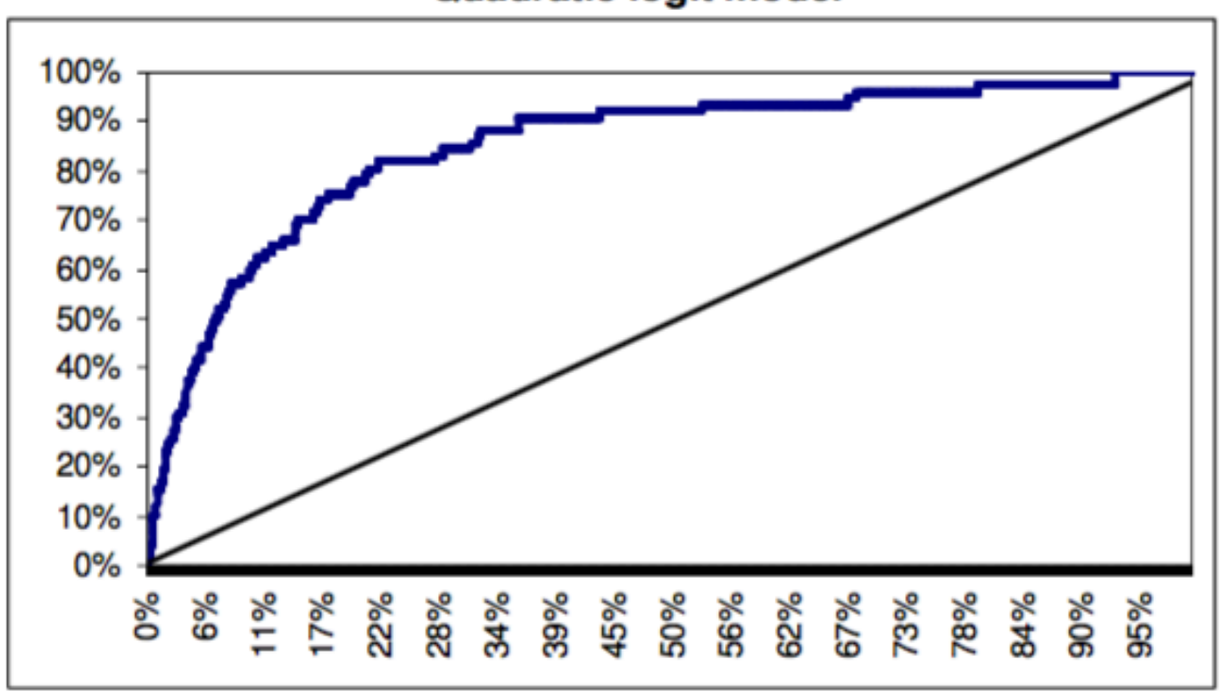
We could alternatively approximate this unknown function using Gaussian kernels with spread centres

In this case, we talk about a “Kernel Logit”.

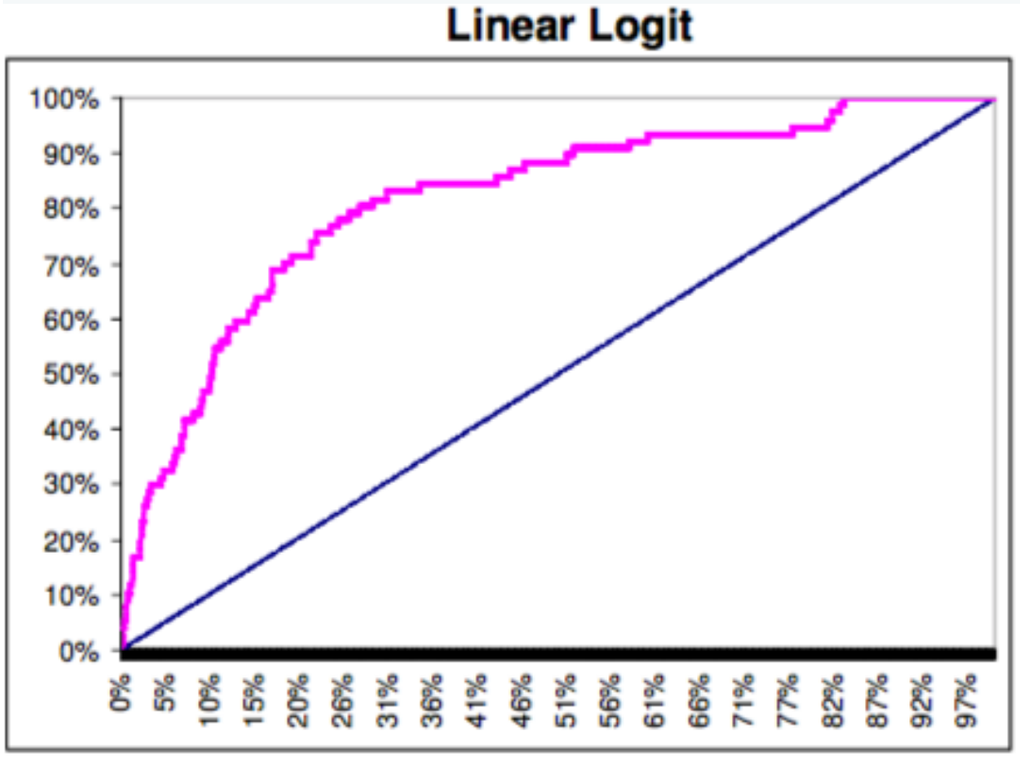
$$\beta_0 + \sum_{i=1}^{K} \beta_i x_i \rightarrow \beta_0 + \sum_{i=1}^{K} \sum_{j=1}^{J} \delta_{ij} \exp\left[-\left(x_i - a_j\right)^2 / \sigma^2\right]$$

Comparing their Gini coefficients on an identical dataset

The Quadratic Logit model looks better, but…



Gini coefficient: **68.2%**



Gini coefficient: **62.7%**

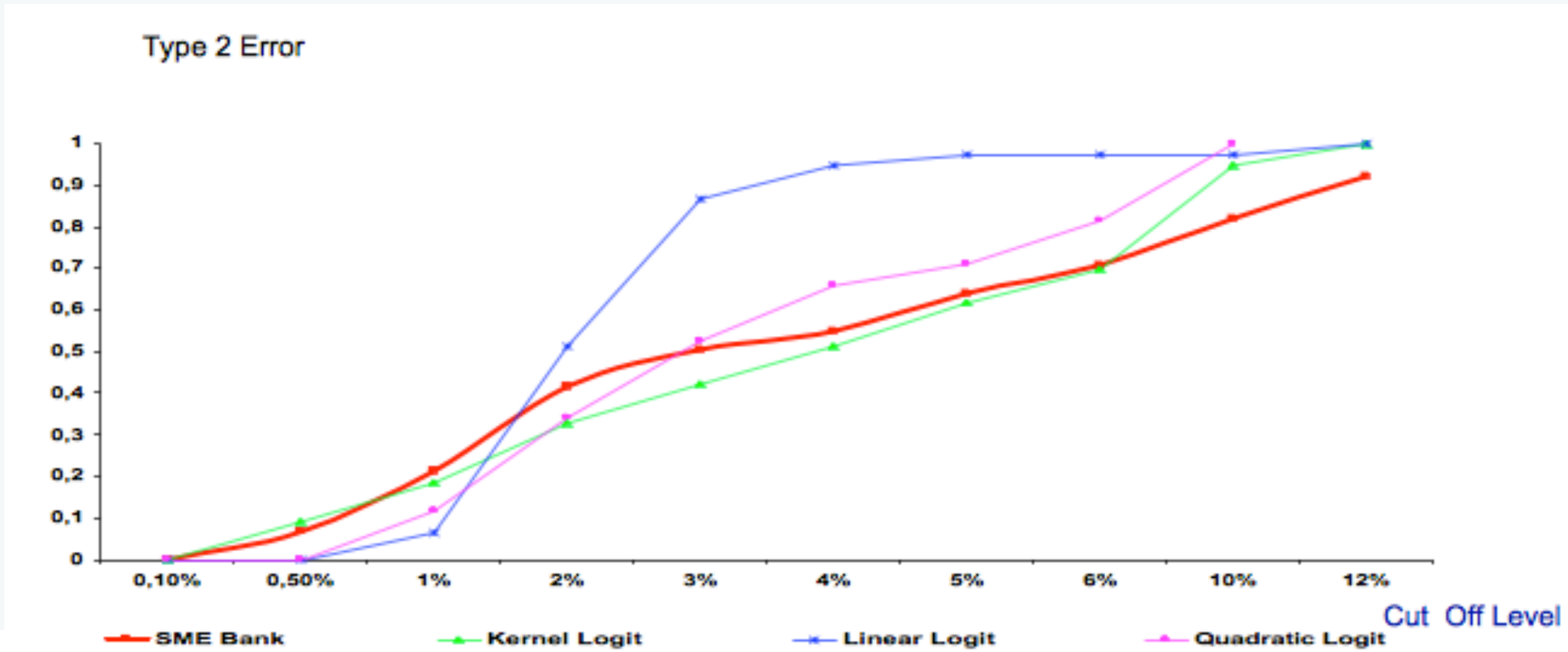
Empirically choosing between models: the lending bank case

We have here 4 competing classification models. A bank has built its own model (SME model) and wishes to compare it with various logit models. They all look to identify defaulting firms considering the same sample and the same features. We would like to chose at the same time an optimal model and a relevant cut-off to identify defaulting firms.

In this case a firm labelled 1 is a firm in default

Type 2 error (false negative) = the firm is in default (1) but is predicted as a non defaulter (0)

Caveat: missing a “true defaulter” is much more important from a P&L perspective than assuming a firm will default, where as it will not. The choice of the cut-off point will depend on the risk aversion of the lending bank! It is likely that the logit model will be preferred, although it is not the best.



Bearing in mind the implicit assumptions made to find the ‘best model ’

When we look to estimate and fit a model using a traditional Maximum Likelihood approach, looking to optimize:

$$\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(p(x_i)\right)+\left(1-y_i\right)\log\left(1-p(x_i)\right)\right)$$

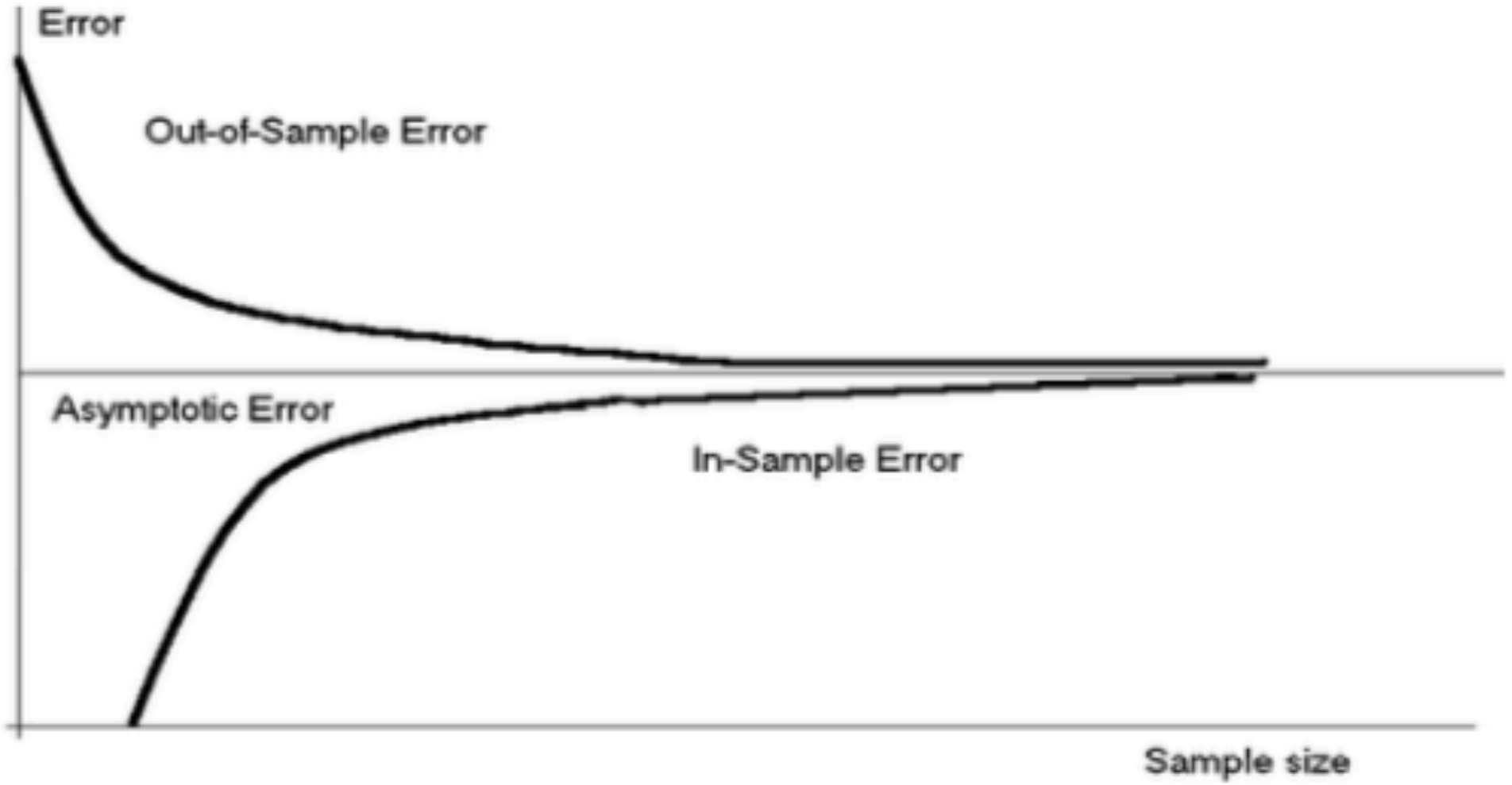
We in fact make two assumptions:

1. The utility function of the user of the model is logarithmic. It is not a bad assumption, but far from a perfect one
2. Success and errors carry the same weight, which is not always the case, as seen on the previous slide.

The consequence is that different fitted models optimised through an MLE process may not lead to an optimal decision process for people having specific preferences.

A model may be preferable to another due to the size of the dataset available

The speed of convergence will typically vary with the kind of scoring model used. Defining the appropriateness of a scoring model like a Logit rather than any other ones such as a Probit or a K-nearest Neighbor, will depend on the highest obtained speed of convergence, given the fixed amount of data available.



Generalizing to Multiclass Classification

- The generalization of to the **multiclass** setting (i.e classifying into **one of K** categories with K greater than 2) can easily be achieved by the One Over All (OOA) approach,

- The OOA approach consists in turning the multiclass classification problem into K binary classification ones as follows:
  - For each class k among the K possible classes, we can create K new datasets by keeping the same input data X and turning the target data Y into a binary one : positive for the class k and negative for all the other classes.
  - You can then train K binary classifiers $(\mathcal{C}_k)_{k\in\{1,\dots K\}}$, where each $\mathcal{C}_k$ is associated to the class k
  - At prediction time, for each new sample, the prediction is given by selecting randomly one of the classes k for which the classifier $\mathcal{C}_k$ predicted a positive output.
- Let's take an example with $K = 3$:

Training

$$Y = \begin{pmatrix}1\\2\\3\\1\\3\end{pmatrix}$$

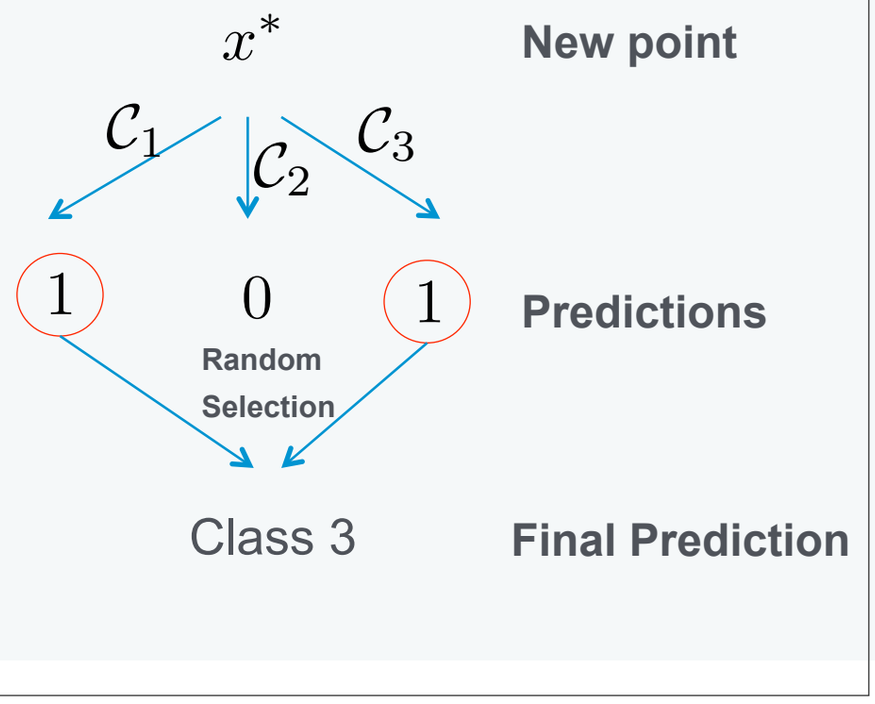
Original Training targets

$\mathcal{C}_1$ → $Y = \begin{pmatrix}1\\0\\0\\1\\0\end{pmatrix}$ Labels for $\mathcal{C}_1$

$\mathcal{C}_2$ → $Y = \begin{pmatrix}0\\1\\0\\0\\0\end{pmatrix}$ Labels for $\mathcal{C}_2$

$\mathcal{C}_3$ → $Y = \begin{pmatrix}0\\0\\1\\0\\1\end{pmatrix}$ Labels for $\mathcal{C}_3$

Prediction



Take away

In the end, in the case of a supervised learning approach, fitting a model always boils down to optimising an objective function.

There are many ways to define an objective function and it will always depend on the error measure used.

Below, we list some of them in a non exclusive manner. It is important to realise that there is not a unique relevant measure.

- ***Quality of fit / loss measures***: there are in fact many different loss functions such as mean square error (MSE), mean absolute error (MAE), R^2, Kullback-Leibler divergence (KL) , Signal to Noise Ratio (SNR)

  ⇒ They assume an implicit utility function (for instance the difference between the MSE and the MAE is related to the relative importance of small v.s. large losses)

  ⇒ Note that some measures are related to a group of observations irrespective of their underlying distribution (MSE, MAE, R^2, SNR) while other measures are related to distributions (KL)

- ***Classification measures***: ROC/AUC, Accuracy, Precision/Recall, Hit Ratio

  ⇒ These measures depend on the dataset used and are not comparable across datasets

  ⇒ Some of these measures enable the modeller to use differentiated rewards / penalties for successes & failures with differentiated cost functions. This again equivalent to assuming a utility function

Go to the following link and take Quiz 1 :

https://mlfbg.github.io/MachineLearningInFinance/

Programming Session