ARTIFICIAL INTELLIGENCE

LEI/3, LMA/3, MBE/1

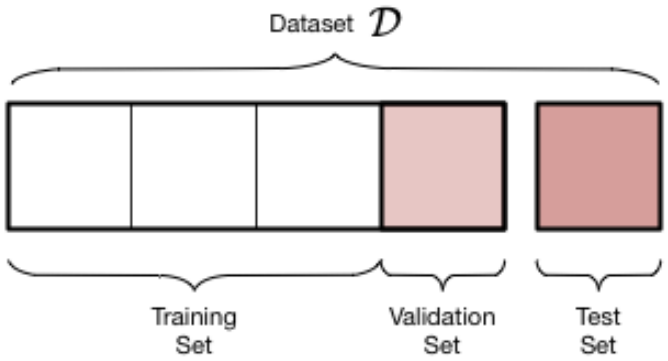
University of Beira Interior, Department of Informatics

Hugo Pedro Proença

hugomcp@di.ubi.pt, **2022/23**

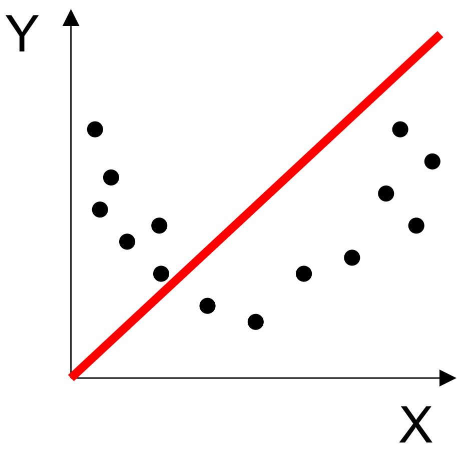
Machine Learning: Experimental Setup

- The design of the experimental procedure to learn/evaluate machine learning models is sensitive
  - Badly designed experiments lead to erroneously optimistic/pessimistic estimates of the system performance
- One of the golden rules in machine learning is that the data should be split in three disjoint subsets:
  - **Learning (Training) set**: this is the set of instances used to fit the parameters of the hypothesis (model).
    - In case of supervised learning, it consists of pairs of a input vectors and the corresponding ground truth, also known as the target or label.
  - **Validation set**. It provides an unbiased evaluation of a model performance during the learning process, while tuning the model **hyper-parameters** (e.g., acceptance/rejection threshold)
  - **Test set**. It is used to provide an unbiased evaluation of a final model.

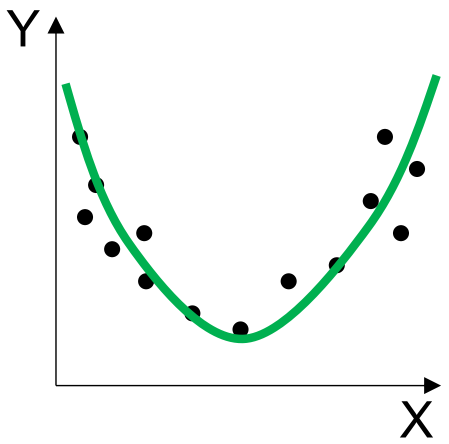


Machine Learning: Overfitting

- Overfitting it is one of the most classical problems in Machine Learning problems.
- It occurs when the our model fits *“**too well”*** **the learning data**, but is **fails to generalize to new data**, i.e., the data where we actually want to use the model
- This is particularly probable when the model has a large number of parameters
  - In such case, the model has too many degrees-of-freedom
  - Nowadays, the breakthrough models based in deep-learning frameworks have a huge number of parameters
  - VGG-16 network, proposed in 2014, has 138,000,000 parameters!



Underfitted



OK

Overfitted

Machine Learning: Overfitting/Underfitting

- The Occam's razzor is a principle from philosophy that states that:
  - *»Entia non sunt multiplicanda praeter necessitatem»*
- This can be translated to:
  - *"More things should not be used than are necessary"*
- Which in practical terms states that simple models should (in case of **comparable effectiveness**) be preferred over more complex ones.

Wiliam of Ockham

- In linear and logistic regression, this is equivalent to force the inferred parameters of our model to be small.
- This is done by adding a term to the cost function we want to minimize:
  - It is called the "**regularization term**" (and λ the regularization weight)
  - Consider that $\theta = \{\theta_0, \theta_1, \ldots, \theta_D\}$

$$J(\boldsymbol{\theta}) = \frac{1}{2N}\sum_{i=1}^{N}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{i=1}^{D}\theta_i^2$$

Machine Learning: Overfitting/Underfitting

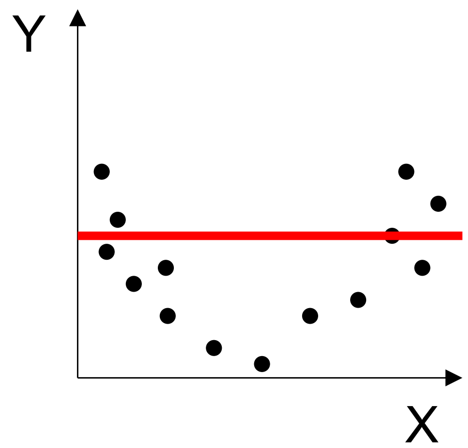
- Consider the following model:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

- Suppose that we set λ too large. What happens?
- Minimizing the J() function, it will force that $\theta_1 \ldots \theta_4$ will be approximately 0

$$J(\boldsymbol{\theta}) = \frac{1}{2N} \sum_{i=1}^{N} \left(h_{\theta}(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{i=1}^{D} \theta_i^{\,2}$$

- Hence, the inferred model will be given by:



Poor Fitting!!
(Underfitted)

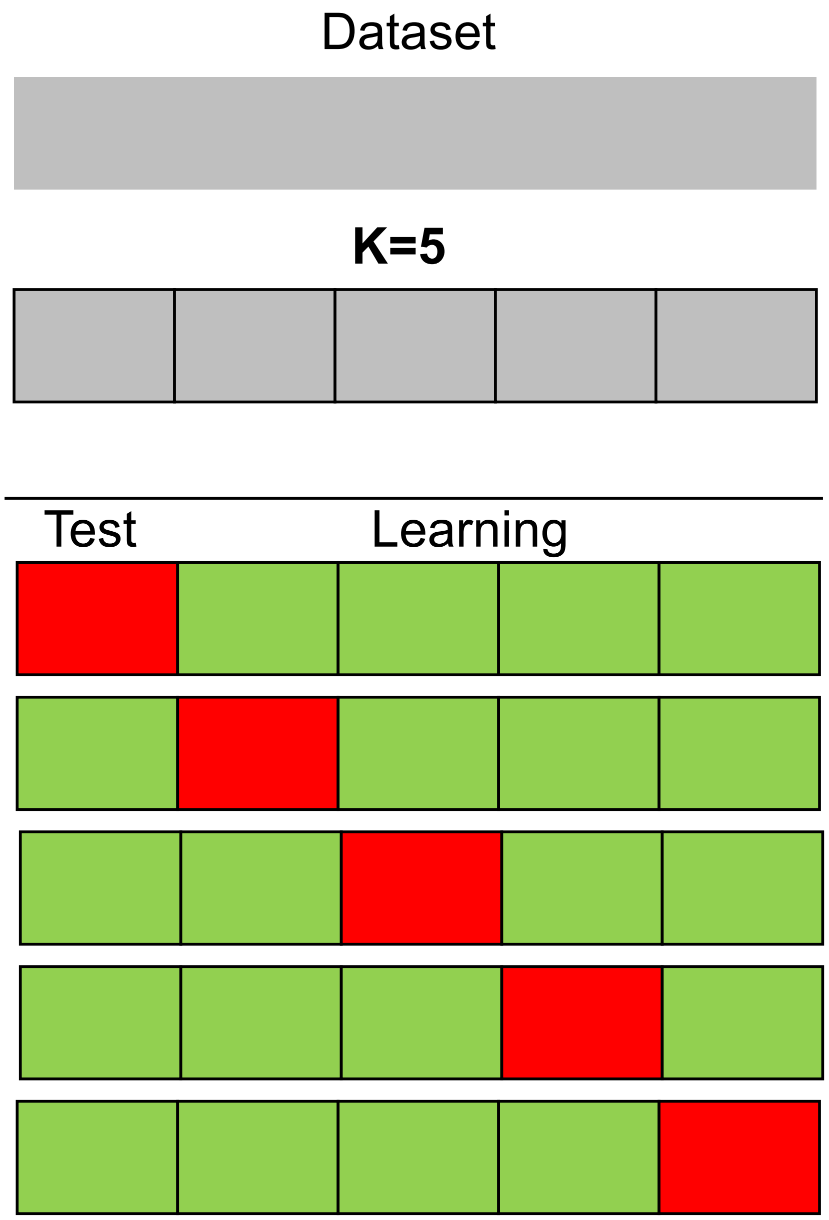
Machine Learning: Overfitting/Underfitting

- In practice terms, this adds one extra-parameter λ to our problem.
  - This parameter is not part of the model, but instead, it is used during the learning process
  - These are called “**hyper-parameters**”
- We saw that:
  - Too large values will lead to underfitted models
  - Too small values will lead to overfitted models
- Typically, the choice of λ can be made according to the performance in the validation set.
- To adapt the linear and logistic regression learning processes, in order to obtain regularized models, one just have to consider that:

$$\frac{\int}{\int \theta_i} \lambda \sum_{i=1}^{D} {\theta_i}^2 = 2\lambda\theta_i$$

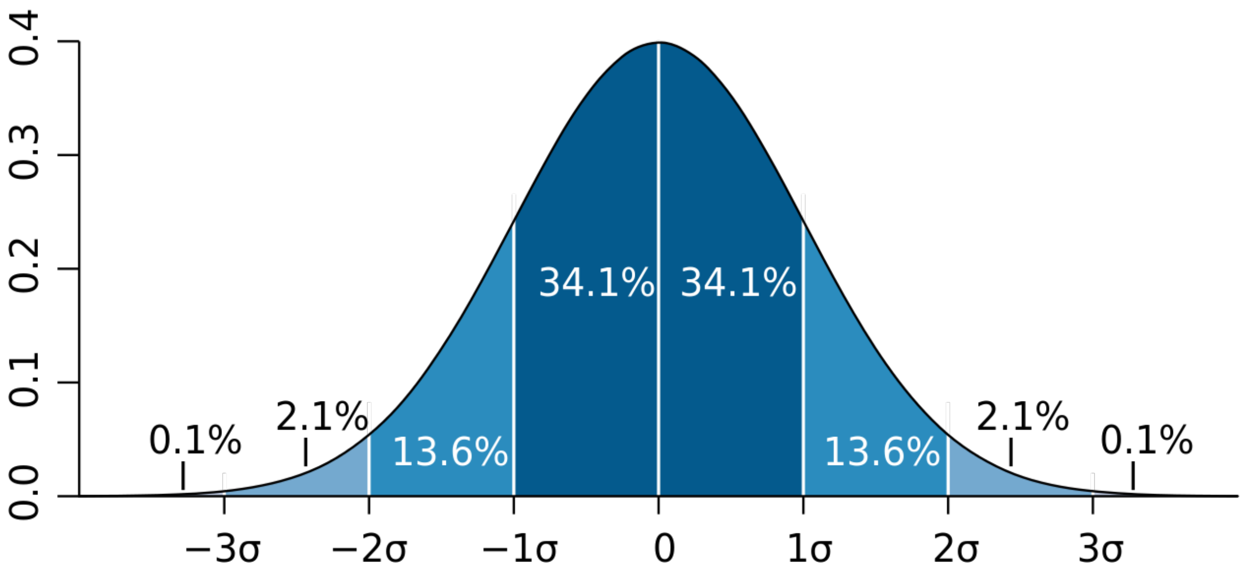
K-Fold Cross Validation

- It is a statistical method used to estimate the performance of machine learning hypotheses (models).
- It is one of the most commonly used, being easy to understand and to implement, with estimates generally having comparable bias than other more sophisticated methods (e.g., bootstrapping)
- It is a **resampling** technique.
  - The value for "K" is defined at the beginning
  - The available data is randomly split at K samples (groups)
  - The model is fitted "K" times, each time using 1 group as **test set** and the remaining (k-1) groups as **learning data**
  - Performance is obtained for the test set
  - The final performance is given by the mean value of the "K" performance values.



K-Fold Cross Validation

- Also, typically results are given in a (“mean” ∓ “standard deviation”) performance values
  - E.g.: “0.70 ∓0.02” means that it is expected that the model performs well 70% of the times, with “typical” variations of more or less 2%
- It has roots in the “***law of big numbers***” and in the “***theorem of the central limit***”
- Considering that repeated observed performance values will approach their “true mean” and that they follow a Gaussian distribution, one can conclude that about 68.2% of the times, the model performance will lie in the “mean ∓ standard deviation” interval.



Bootstrapping

- It is closely related to K-fold cross validation and follows the same idea:
  - Generates multiple subsets, by sampling from a single, original dataset.
  - Each of the “*new*” sets can be used to estimate performance.
  - Since there are multiple sets (and therefore multiple estimates), one can also obtain the mean, standard deviation or a confidence interval for the estimate.
- The key difference is that bootstrapping **resamples the data *with replacement***.
  - Given a dataset containing N points, bootstrap picks a data point uniformly at random, adds it to the bootstrapped set, *puts that data point back into the dataset*, and repeats.
  - Why put the data point back?
  - In a real setting, data would come from the “real distribution of the data”.
  - But all we have is a dataset (i.e., a sample), we don’t have the real distribution of the data. Out set is supposed to represent the underlying distribution, i.e., **it is an *empirical* distribution of data**.
  - The rule is to simulate sub-sets by drawing from the empirical distribution.
  - Hence, the data point must be put back, because otherwise the empirical distribution would change after each draw.

Confusion Matrix

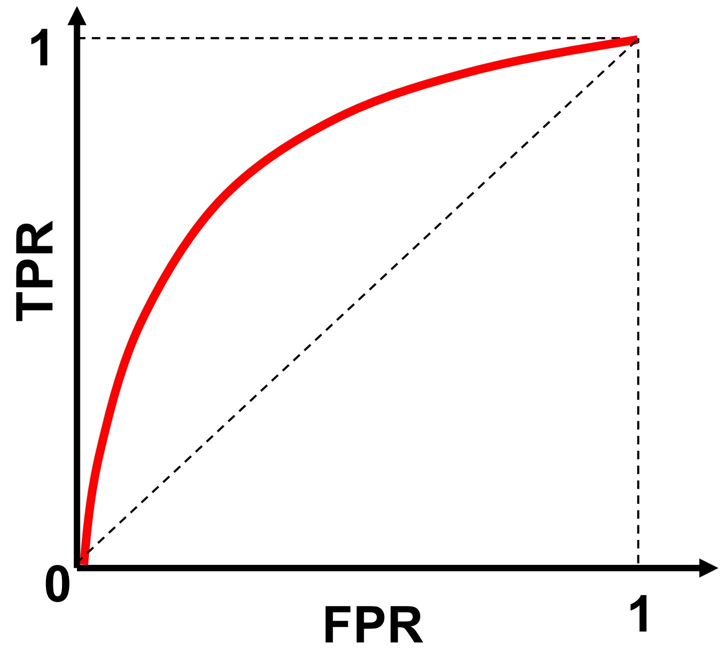
- Also known as an error matrix, this table summarizes the model performance, providing more information that the simple “accuracy” value.
- For a binary classification problem, it is a table with two rows and two columns, reporting the number of *false positives*, *false negatives*, *true positives*, and *true negatives*.
  - Each row corresponds to one predicted outcome (class)
  - Each column corresponds to one actual (ground-truth) class

		Prediction outcome		
		positive	negative	
Actual value	positive	TP	FN	$TP + FN$
	negative	FP	TN	$FP + TN$
		$TP + FP$	$FN + TN$	

- The model **accuracy** is given by: $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision**: $\frac{TP}{TP+FP}$ (*when it predicts “yes”, how likely it is correct?*)
- **Recall**: $\frac{TP}{TP+FN}$ (*what is the proportion of “yes” that are actually detected?*)

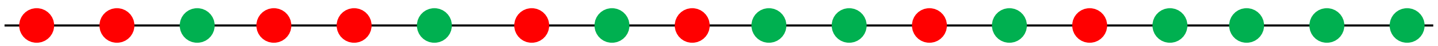
ROC: Receiver Operating Characteristic

- A **R**eceiver **O**perating **C**haracteristic curve (**ROC)**, is a graphical plot that illustrates the performance of a binary classifier system, with respect to changes in its discrimination threshold.
- This curve shows the relationship between two measures:
  - True Positive Rate
  - False Positive Rate
- The **T**rue **P**ositive **R**ate (**TPR**) is also known as recall and is given by:
  - $TPR = \frac{TP}{TP+FN}$
- The **F**alse **P**ositive **R**ate (**FPR**) (1 – specificity) is given by:
  - $FPR = \frac{FP}{FP+TN}$
- This plot gives the TPR vs. FPR at different acceptance thresholds.
  - Low thresholds classify more items as positive, which increases both the TPR and FPR
  - High thresholds classify less items as positive, which decreases both the TPR and FPR

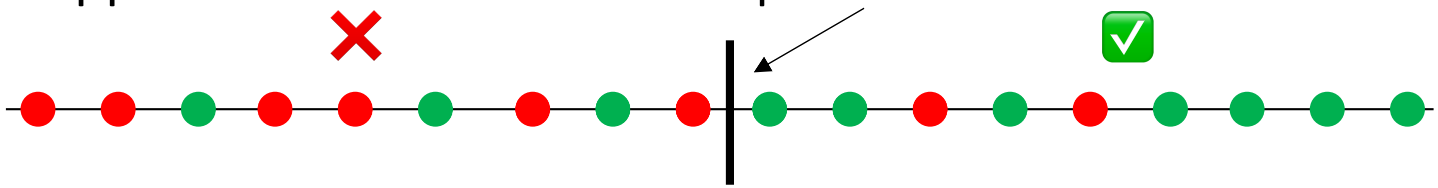


ROC: Receiver Operating Characteristic

- To obtain the data for a ROC curve, we start by sorting the output scores, obtained for the evaluation set:
    - Consider that red dots correspond to class “0” (the *negative* class), and green dots to class “1” (the positive class)



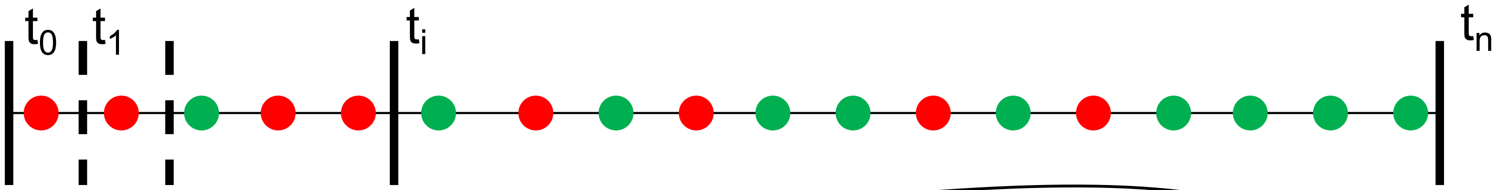
- What happens when we set the acceptance threshold at?



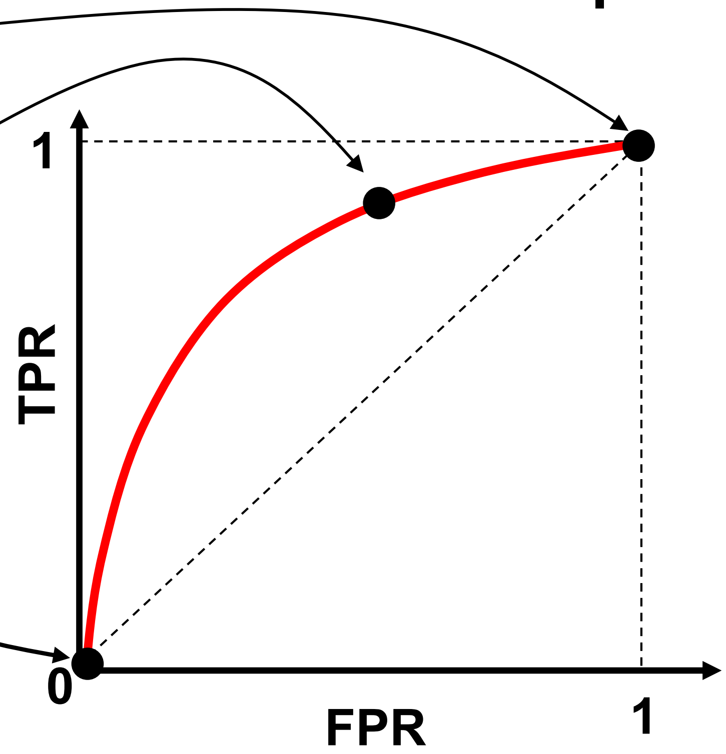
- 6 (out of 8) negative samples are correctly rejected. TNR=6/8
- 2 (out of 8) negative samples are erroneously considered as positive. FPR = 2/8
- 7 (out of 10) positive samples are correctly accepted. TPR = 7/10
- 3 (out of 10) positive samples are erroneously considered as negative. FNR = 3/10

ROC: Receiver Operating Characteristic

- Next, we obtain the TPR/FPR values for all possible acceptance thresholds:

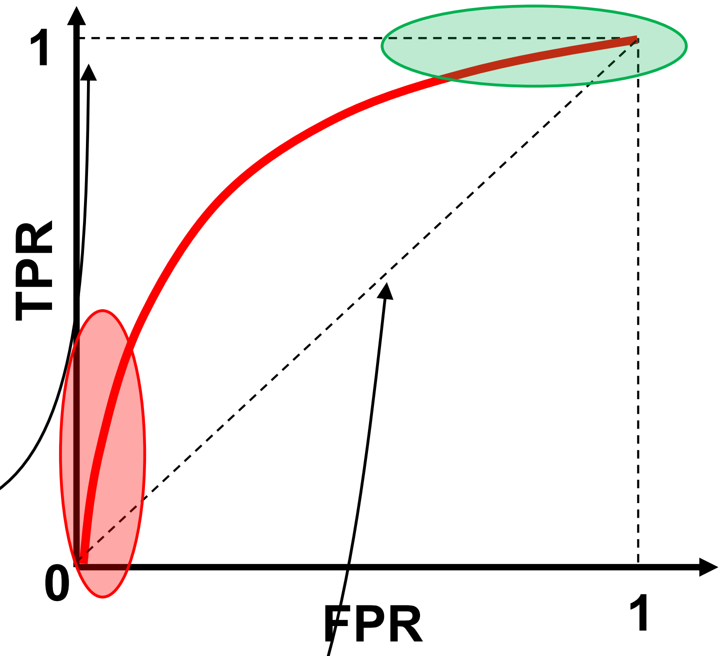


- At t_0, we have TPR=1, FPR=1
- At t_1, …
- At t_i, we have TPR=0.9, FPR=0.5
- …
- At t_n, we have TPR=0, FPR=0



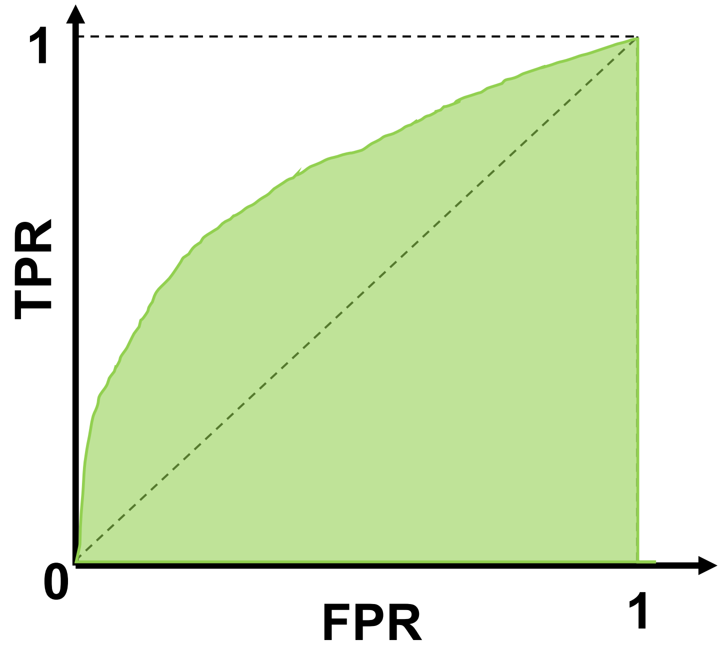
ROC: Receiver Operating Characteristic

- The ROC curve reports all the possible performance parameterizations of our model:
  - Either tuned for security or convenience
- When comparing two models, the best one would have the ROC curve above the other most times
- The optimal performance will correspond to the (0,1) point in the plot
- The $x_i=y_i$ line corresponds to the worst possible model, with performance equal of a random number generator.



AUC: Area Under Curve

- The ROC curve shows all possible parameterization, and it is given as a plot
- To obtain a numeric value that sumarizes the effectiveness of a model, it is typically used the **A**rea **U**nder **C**urve metric.
- It is given by:
  - $\int_0^1 f(x)dx$
- with $f(x)$ corresponding to the ROC curve values.
- AUC = 1 is the **perfect system** that obtains optimal performance with all possible acceptance thresholds
- AUC = 0.5 is the “random number” generator (**worst possible system**)



Machine Learning: Exercise

- Consider the "banknote.csv" dataset, available at the course web page, and taken from the "*UCI: Machine Learning Repository*", at the University of California
- Suppose that we are interested in developing a machine learning model, able to distinguish between genuine and forged bank notes.
- To do that, experts told us that we would have to measure four features in each note:
  1. variance of the wavelet transformed image
  2. skewness of wavelet transformed image
  3. curtosis of wavelet transformed image
  4. entropy of the image
- The fifth column gives us the class information, i.e., whether the note is genuine (1) or a fake (0)

- Start from the "logistic_regression.py" script, and implement the regularized version of logistic regression
  - See how different values of λ lead to different models
- Implement the "K-fold" cross validation and bootstrapping performance evaluation strategies
  - Report the corresponding "mean $\mp$ standard deviation", and AUC values.