Applied Machine Learning - Basic

Prof. Daniele Bonacorsi

Lecture 6

Data Science and Computation PhD + Master in Bioinformatics

University of Bologna

Communications

2 bonus virtual lectures! incl. notebooks!

Focus on:

- **Matplotlib**
  - https://unibo.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=65556357-e7c9-42aa-aa31-ae6c0162120a#
- **Seaborn**
  - https://unibo.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=f4198b86-9440-4568-8583-ae6c01529aec#

Recap

(…)

Advices for applying ML

- Evaluating a learning model → Evaluate an hypothesis
- Model selection and TVT sets
- Bias vs variance diagnosis
- Regularisation and bias vs variance
- Learning curves

NEXT:

Performance metrics and skewed classes

1110001110100100110011010010100111010100100101
1100101010010101010101101001010101011110001110
0100110011010010100111010100100101011100101010
0011100Advices00for01applying00ML1010101011101
0100101010011010100110010010101110101010010101
0100101010101111000111010010011001101001010011
1011011010111001010100101010101101000101010111
1000111010010011001101001010011101010010010101
1011011001100 0Bias10and10Variance0100100100101
1110110101110010101001010101010110100010101011
0001110101101 0What01to01do11next1011101010010
0101110011101011001101011100101010010101010110
1000101010111100011100101010001101111000100110
1010010100111010101011100010010110010101010101
1101000111010101001110101110110101101011101011
1000000010101011000011100001011000110101101110

We've talked about **how to evaluate learning algos**, talked about **model selection**, talked a lot about **bias and variance**.

So how does this help us figure out what are potentially fruitful things to try to do to improve the performance of a learning algorithm?

Let's go back to our original motivating example and then go straight to the result.

Debugging a learning algo

Suppose you have implemented regularised linear regression to predict housing prices. However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

We discussed:

- getting more training examples
- try smaller sets of features
- try getting additional features
- try adding polynomial features (squares, cross products, etc)
- try increasing/decreasing lambda

Let's revisit them in view of what we now know a bit more.

What to try next

- getting more training examples —> **CHECK YOUR VARIANCE**
  - good to fix high-variance problems (i.e. your CV error is quite bigger than training error)
  - but if you have high bias, this will not help, so do <u>not</u> waste time in collecting more examples..
- try smaller sets of features
  - good to fix high-variance problems
  - but if you have high bias, this will not help, so do <u>not</u> waste time in carefully selecting features…
- try getting additional features
  - good to fix high-bias problems (h too simple, so add features to make h more able to fit the training set)
- try adding polynomial features (squares, cross products, etc)
  - another way to fix high-bias problems
- try increasing lambda
  - good to fix high-variance problems
- try decreasing lambda
  - good to fix high-bias problems

1110001110100100110011010010100111010100100010
1011100101010010101010101101001010101011111000
1110100100110011010010100111010100100101011110
01110Advices00for01applying00ML1010101011101010
1010010101001101010011001001010111010101001001
0101010110100101010101011110001110100100110011
0
1001010011101011011010111001010100101010101
1010001010101111000111010010011001101001001
1101010010010101110010101001001110101101101
1011110111001010100101010101101000101010111
00001110Error01metrics01for11skewed10classes1
0010011101011001101011100101010010101010101
1000101010111100011100101010001101111000100
0011010010100111010101011100010010110010101
0101011101000111010101001110101110110101010
0101101011111000011100001011000110101101110

No deep preliminary discussion on error analysis and the importance of having error metrics, but <u>use of intuition</u> that it is of **great importance to have a single real number evaluation metric for a learning algorithm**.

In most cases, **accuracy** is just fine. But there is one and common important case, where it is particularly tricky to come up with an appropriate evaluation metric. That is the case of so-called **skewed classes**.

Consider the problem of cancer classification, where we have features of medical patients and we want to decide whether or not they have cancer.

Example : cancer/not-cancer classification

→ train a logistic regression model $h_\theta(x)$ ⇒ a classifier

$$y = \{0, 1\}$$

benign malignant

→ test your classifier : e.g. find a 1% error on test set

⇒ impressive ! 99% correct diagnosis !

Example : malignant / benign classification

→ train a logistic regression model $h_\theta(x)$ ⇒ a classifier

$y = \{0, 1\}$

benign malignant

→ test your classifier : e.g. find a 1% error on test set

⇒ impressive ! 99% correct diagnosis !

→ suppose you find out that only 0.5% of patients have cancer
(eg you assign $y=0$ to everyone and you are wrong in 0.5% of the cases)

⇒ no longer impressive :(

if you IGNORE features x, you do even better !
("non learning algo")

When a non-learning algo does a good job, better than a simple learning algo, we are in peculiar case:

always "y=0" or
always "y=1"

positive examples
<<
negative examples

=> "skewed classes"

When a non-learning algo does a good job, better than a simple learning algo, we are in peculiar case:

always "y=0" or
always "y=1"

positive examples << negative examples ⇒ "skewed classes"

This poses a tricky question:

if I go from a learning algo at 99.1% accuracy (0.9% error) to one at 99.7% accuracy (0.3% error), how can I be sure the latter is an improvement, or a random mod that e.g. predicts =0 more often and I am in skewed classes scenario?

With skewed classes, accuracy ↗ is not always an indicator of improvements in the classifier

You need a different error metric → PRECISION / RECALL

Precision / Recall

actual → / predicted ↓	1	0
1		
0		

Precision / Recall

predicted ↓ \ actual →	1	0
1	TRUE POSITIVE	
0		TRUE NEGATIVE

Precision / Recall

predicted ↓ \ actual →	1	0
1	TRUE POSITIVE	FALSE POSITIVE
0	FALSE NEGATIVE	TRUE NEGATIVE

Precision / Recall

predicted ↓ \ actual →	1	0
1	TRUE POSITIVE	FALSE POSITIVE
0	FALSE NEGATIVE	TRUE NEGATIVE

NOTE :

- TRUE vs FALSE
  → position in the matrix
  ⇒ depends on "predicted" and "actual"

- POSITIVE vs NEGATIVE
  → depends on "predicted" only

Precision / Recall

predicted ↓ \ actual →	1	0
1	TP	FP
0	FN	TN

PRECISION

"Among all patients predicted to have cancer, how many actually have it?"

→ how "precise" have you been?

$$\Rightarrow \frac{\#\ TP}{\underbrace{\#\ \text{predicted } P}_{TP+FP}} = \frac{TP}{TP+FP}$$

better if this is high!

Precision / Recall

predicted ↓ \ actual →	1	0
1	TP	FP
0	FN	TN

PRECISION

"Among all patients predicted to have cancer, how many actually have it?"

$$\stackrel{\text{def}}{=} \frac{TP}{\underbrace{TP+FP}_{\text{predicted P}}} \quad \text{(should be as high as possible!)}$$

RECALL

"Among all patients that actually have cancer, how many did we predict to have it?"

→ "how many of the TP were "recalled" (found)?"

$$\Rightarrow \frac{\cancel{\#}\ TP}{\cancel{\#}\ \text{actual } P} = \frac{TP}{TP+FN}$$

better if this is high!

Precision / Recall

predicted ↓ \ actual →	1	0
1	TP	FP
0	FN	TN

PRECISION

"Among all patients predicted to have cancer, how many actually have it?"

$$\stackrel{\text{def}}{=} \frac{TP}{\underbrace{TP+FP}_{\text{predicted P}}} \quad \text{(should be as high as possible!)}$$

RECALL

"Among all patients that actually have cancer, how many did we predict to have it?"

$$\stackrel{\text{def}}{=} \frac{TP}{\underbrace{TP+FN}_{\text{actual P}}} \quad \text{(should be as high as possible!)}$$

Precision / Recall

$$\text{PRECISION} \stackrel{\text{def}}{=} \frac{TP}{TP+FP}$$

Intuitively: the **precision** is the ability of the classifier not to label as positive a sample that is negative.

$$\text{RECALL} \stackrel{\text{def}}{=} \frac{TP}{TP+FN}$$

Intuitively: the **recall** is the ability of the classifier to find all the positive samples.

predicted \ actual	1	0
1	TP	FP
0	FN	TN

Example :

classifier that predicts $y=0$ always :

$\Rightarrow$ $TP = 0$ $\Rightarrow$ PRECISION $= 0$

RECALL $= 0$

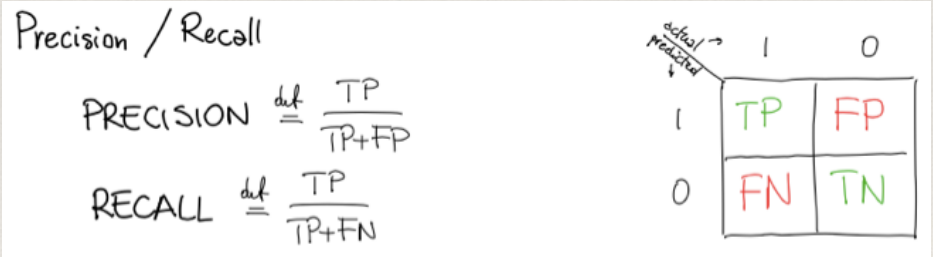
C.Y.U.

Quiz

Your algorithm has a performance on the test set given by this matrix.
What is the algo's **precision**?

	1	0
1	80	20
0	80	100

- 0.5
- 0.8
- 0.08
- 0.1



C.Y.U.

Quiz

Your algorithm has a performance on the test set given by this matrix.
What is the algo's **precision**?

	1	0
1	80	20
0	80	100

- 0.5
- **0.8**
- 0.08
- 0.1

Precision / Recall

$$\text{PRECISION} \stackrel{def}{=} \frac{TP}{TP+FP}$$

$$\text{RECALL} \stackrel{def}{=} \frac{TP}{TP+FN}$$

predicted ↓ / actual →	1	0
1	TP	FP
0	FN	TN

C.Y.U.

Quiz

Same as before. What is the algo's **recall**?

- 0.5
- 0.8
- 0.08
- 0.1

	1	0
1	80	20
0	80	100

Precision / Recall

$$\text{PRECISION} \stackrel{def}{=} \frac{TP}{TP+FP}$$

$$\text{RECALL} \stackrel{def}{=} \frac{TP}{TP+FN}$$

actual → / predicted ↓	1	0
1	TP	FP
0	FN	TN

C.Y.U.

Quiz

Same as before. What is the algo's **recall**?

- **0.5**
- 0.8
- 0.08
- 0.1

	1	0
1	80	20
0	80	100

Precision / Recall

$$\text{PRECISION} \stackrel{def}{=} \frac{TP}{TP+FP}$$

$$\text{RECALL} \stackrel{def}{=} \frac{TP}{TP+FN}$$

predicted ↓ / actual →	1	0
1	TP	FP
0	FN	TN

When we define **precision** and **recall**, usually we use the convention that y is equal to 1 in the presence of the *more rare class*.

- So if we are trying to detect rare conditions such as cancer, hopefully that's a rare condition, precision and recall are defined setting y = 1, rather than y = 0: the presence of the rare class is the driver.

Summary

As we were not satisfied about “accuracy” only, as it could be cheated easily by a non-learning algo, we introduced **precision** and **recall**.

Now, more generally, even for settings where we have very skewed classes, it is not possible for an algorithm to"cheat" and somehow get a very high precision and a very high recall by doing some simple thing like predicting y=0 (or y=1) all the time.

We now are more confident to state if a classifier is actually a good classifier: **precision/recall are more useful evaluation metrics to tell if one algorithm may be doing well in real applications**.

1110001110100100110011010010100111010100100101
0111001010100101010101101001010101011110001 1
1010010011001101001010011101010010010101110010
1110Advices00for01applying00ML1010101011101010
1001010100110101001100100101011101010100101010
1010110100101010101111000111010010011001101001
0100111010110110101110010101001010101010110100
0101010111100011101001001100110100101001110101
0010010101110010101001001110101101101011100101
0100101010101010110100010101011110001110010100 01
1110111100Error01metrics01for11skewed10classes1
0010011101011001101011100101010010101010101101
0001010010Tradeoff01precison11vs10recall1000110
1101010111000100101100101010101010111010001110
1010100111010111011010101010110000111000010110
1111101101110100011001110101000011010110101110

We talked about precision and recall as evaluation metrics for classification problems with skewed constants.

For many applications, we'll want to somehow control the **trade-off between precision and recall**.

Cancer example $\rightarrow$ train logistic regression

$$\text{PRECISION} \stackrel{\text{def}}{=} \frac{TP}{\underbrace{TP+FP}_{\text{predicted P}}}$$

$$\text{RECALL} \stackrel{\text{def}}{=} \frac{TP}{\underbrace{TP+FN}_{\text{actual P}}}$$

$$0 \leq h_\theta(x) \leq 1$$

$$\text{predict} \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 \\ 0 & \text{if } h_\theta(x) < 0.5 \end{cases}$$

this classifier will give me some values for

Cancer example $\rightarrow$ Train logistic regression

$$\text{PRECISION} \stackrel{def}{=} \frac{TP}{\underbrace{TP+FP}_{\text{predicted P}}}$$

$$\text{RECALL} \stackrel{def}{=} \frac{TP}{\underbrace{TP+FN}_{\text{actual P}}}$$

$$0 \leq h_\theta(x) \leq 1$$

$$\text{predict} \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 \\ 0 & \text{if } h_\theta(x) < 0.5 \end{cases}$$

this classifier will give me some values for

Suppose we want to predict $y=1$ only if we are VERY confident

e.g.

$$\begin{cases} 1 & \text{if } h_\theta(x) \geq \cancel{0.5} \quad 0.7 \\ 0 & \text{if } h_\theta(x) < \cancel{0.5} \quad 0.7 \end{cases}$$

Cancer example → Train logistic regression

$$\text{PRECISION} \stackrel{\text{def}}{=} \frac{TP}{\underbrace{TP+FP}_{\text{predicted P}}}$$

$$\text{RECALL} \stackrel{\text{def}}{=} \frac{TP}{\underbrace{TP+FN}_{\text{actual P}}}$$

$$0 \leq h_\theta(x) \leq 1$$

$$\text{predict} \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 \\ 0 & \text{if } h_\theta(x) < 0.5 \end{cases}$$

this classifier will give me some values for

Suppose we want to predict $y=1$ only if we are VERY confident

e.g.

$$\begin{cases} 1 & \text{if } h_\theta(x) \geq \cancel{0.5} \quad 0.7 \\ 0 & \text{if } h_\theta(x) < \cancel{0.5} \quad 0.7 \end{cases}$$

more "precise"!

$\Rightarrow$ higher precision

I do not know TP (numerator), but for sure predicted P ($=TP+FP$) ↘

$\Rightarrow$ denominator ↘ $\Rightarrow$ **precision** ↗

Cancer example → train logistic regression

$$\text{PRECISION} \overset{def}{=} \frac{TP}{\underbrace{TP+FP}_{\text{predicted P}}}$$

$$\text{RECALL} \overset{def}{=} \frac{TP}{\underbrace{TP+FN}_{\text{actual P}}}$$

$$0 \leq h_\theta(x) \leq 1$$

$$\text{predict} \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 \\ 0 & \text{if } h_\theta(x) < 0.5 \end{cases}$$

this classifier will give me some values for

Suppose we want to predict $y=1$ only if we are VERY confident

e.g.

$$\begin{cases} 1 & \text{if } h_\theta(x) \geq \cancel{0.5} \quad 0.7 \\ 0 & \text{if } h_\theta(x) < \cancel{0.5} \quad 0.7 \end{cases}$$

more "precise"!

I "recall" (find) less TP w.r.t. before

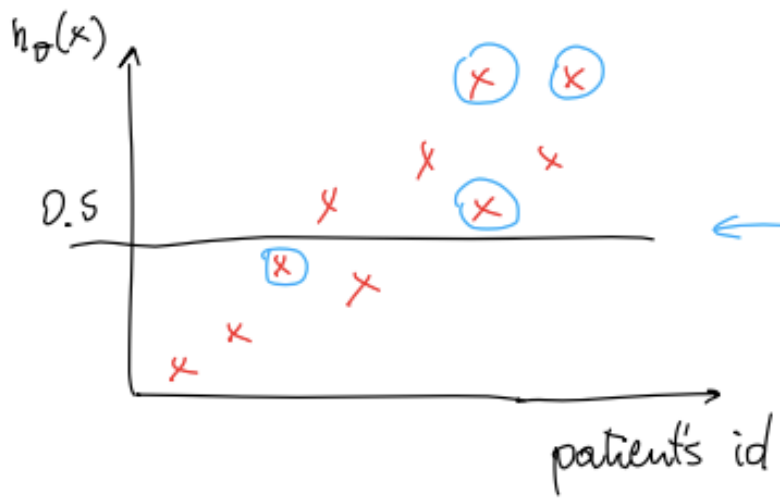
$\Rightarrow$ higher precision, but lower recall

I do not know TP (numerator), but for sure predicted P ($=TP+FP$) ↘
$\Rightarrow$ denominator ↘ $\Rightarrow$ **precision** ↗

Now predicting $y=1$ on a smaller # patients
I increased the purity of the sample:
TP (numerator) ↘, plus FN at the denominator ↗ $\Rightarrow$ **recall** ↘

Let's check with an example

Example :



× value of $h_\theta(x)$ for a patient

⊗ same, but this has cancer

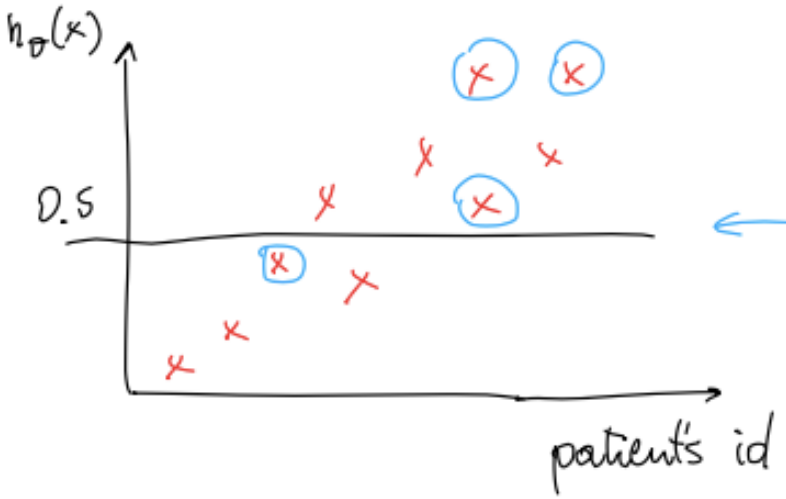
threshold at 0.5

actual / predicted	1	0
1	TP	FP
0	FN	TN

Example :

× value of $h_\theta(x)$ for a patient

⊗ same, but this has cancer



threshold at 0.5

actual → / predicted ↓	1	0
1	TP	FP
0	FN	TN

$$\text{PRECISION} \triangleq \frac{TP}{\underbrace{TP+FP}_{\text{predicted P}}}$$

$$\text{RECALL} \triangleq \frac{TP}{\underbrace{TP+FN}_{\text{actual P}}}$$

0.5 ⇒

actual → / predicted ↓	1	0
1	3	3
0	1	3

⇒

$$\text{Precision} = \frac{3}{3+3} = 0.5$$

only 50% of selected have cancer

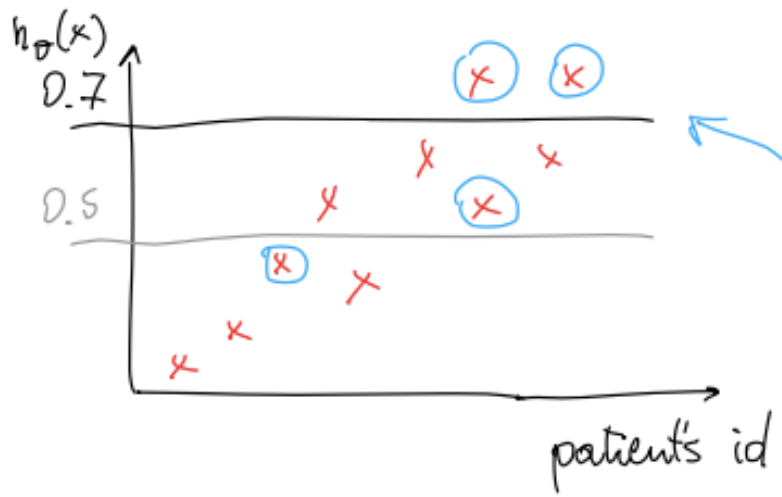
$$\text{Recall} = \frac{3}{3+1} = 0.75$$

I selected 3/4 of them

Example :

× value of $h_\theta(x)$ for a patient

⊗ same, but this has cancer



threshold at 0.7

actual → / predicted ↓	1	0
1	TP	FP
0	FN	TN

$$\text{PRECISION} \triangleq \frac{TP}{\underbrace{TP+FP}_{\text{predicted P}}}$$

$$\text{RECALL} \triangleq \frac{TP}{\underbrace{TP+FN}_{\text{actual P}}}$$

0.7 ⇒

actual → / predicted ↓	1	0
1	2	0
0	2	6

⇒

$$\text{Precision} = \frac{2}{2+0} = 1$$ perfect!

$$\text{Recall} = \frac{2}{2+2} = 0.5$$ I selected 50% of them

Example summary :

	prec	recall
0.5 →	0.5	0.75
0.7 →	1.0	0.5

⇓ prec ↗

⇓ recall ↘

Cancer example → Train logistic regression

$$\text{PRECISION} \stackrel{\text{def}}{=} \frac{TP}{\underbrace{TP+FP}_{\text{predicted P}}}$$

$$\text{RECALL} \stackrel{\text{def}}{=} \frac{TP}{\underbrace{TP+FN}_{\text{actual P}}}$$

$$0 \leq h_\theta(x) \leq 1$$

$$\text{predict} \begin{cases} 1 & \text{if } h_\theta(x) \geq \cancel{0.5} \quad \cancel{0.7} \quad 0.9 \\ 0 & \text{if } h_\theta(x) < \cancel{0.5} \quad \cancel{0.7} \quad 0.9 \end{cases}$$

We can push this further ↗ ⇒ predict $y=1$ ONLY IF we are 90% certain it is cancer

⇐ a large fraction of those patients will turn out to have cancer

(again) ⇒ even higher precision and even lower recall

Suppose instead you have a different goal: you want to avoid missing too many cases of cancer $\longrightarrow$ i.e. avoid FN

[NOTE: as a real-life example, it is compelling to fight FN, i.e. sick but no diagnosis!
In other words, when in doubt, trigger $y=1$ so they make exams and figure out...]

Suppose instead you have a different goal: you want to avoid missing too many cases of cancer $\longrightarrow$ i.e. avoid FN

[NOTE: as a real-life example, it is compelling to fight FN, i.e. sick but no diagnosis! In other words, when in doubt, trigger $y=1$ so they make exams and figure out...]

In this case, you need a lower threshold instead:

$$0 \leq h_\theta(x) \leq 1$$

$$\text{predict} \begin{cases} 1 & \text{if } h_\theta(x) \geq \cancel{0.5} \;\; 0.3 \\ 0 & \text{if } h_\theta(x) < \cancel{0.5} \;\; 0.3 \end{cases}$$

Suppose instead you have a different goal: you want to avoid missing too many cases of cancer $\longrightarrow$ i.e. avoid FN

[NOTE: as a real-life example, it is compelling to fight FN, i.e. sick but no diagnosis! In other words, when in doubt, trigger $y=1$ so they make exams and figure out...]

In this case, you need a lower threshold instead:

$$0 \leq h_\theta(x) \leq 1$$

$$\text{predict} \begin{cases} 1 & \text{if } h_\theta(x) \geq \cancel{0.5} \; 0.3 \\ 0 & \text{if } h_\theta(x) < \cancel{0.5} \; 0.3 \end{cases}$$

$\Rightarrow$ lower precision and higher recall

a larger fraction of patient we are saying to have cancer will eventually turn out not to have it

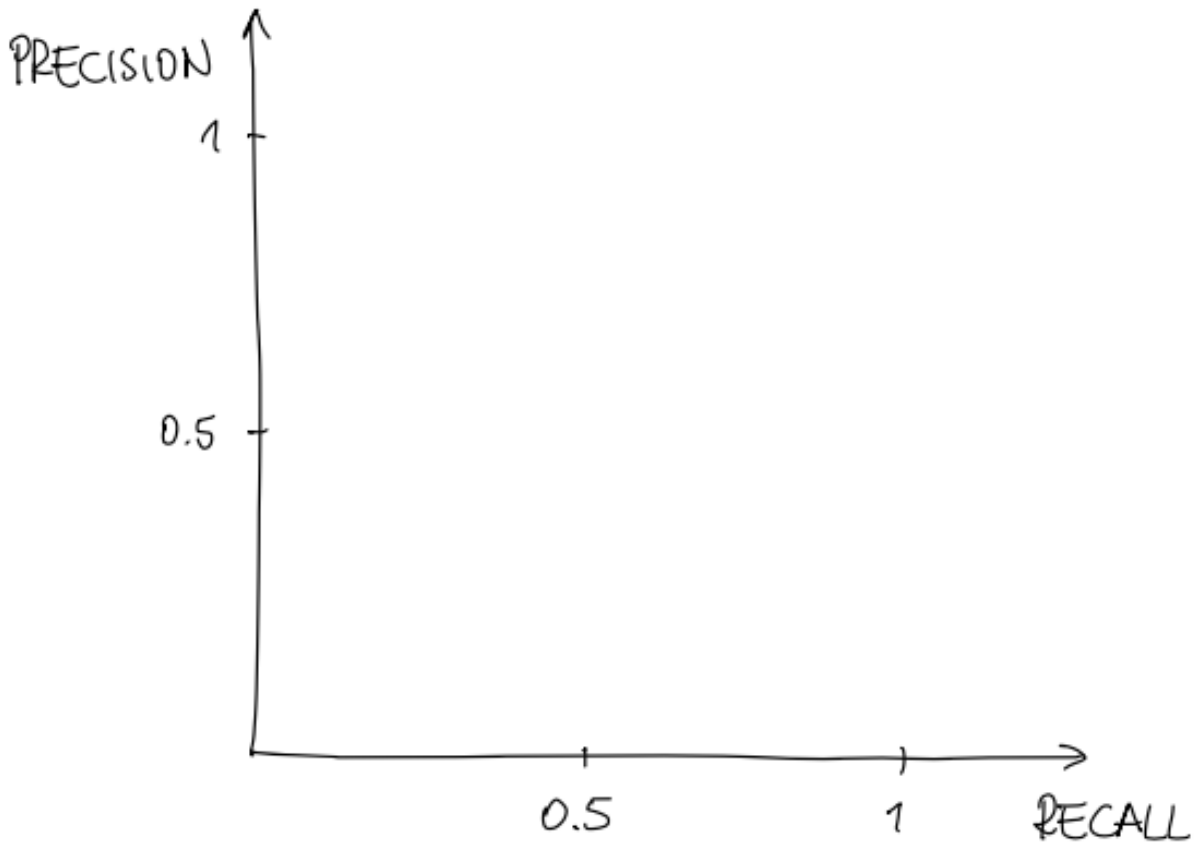
we are going to correctly flagging a larger fraction of patients that actually do have cancer

If this threshold can be different from 0.5 - and there seems to be a motivation - is there a way to choose this threshold automatically?

Or - more generally - if we leave this open, how do we compare different models having different precision/recall values?

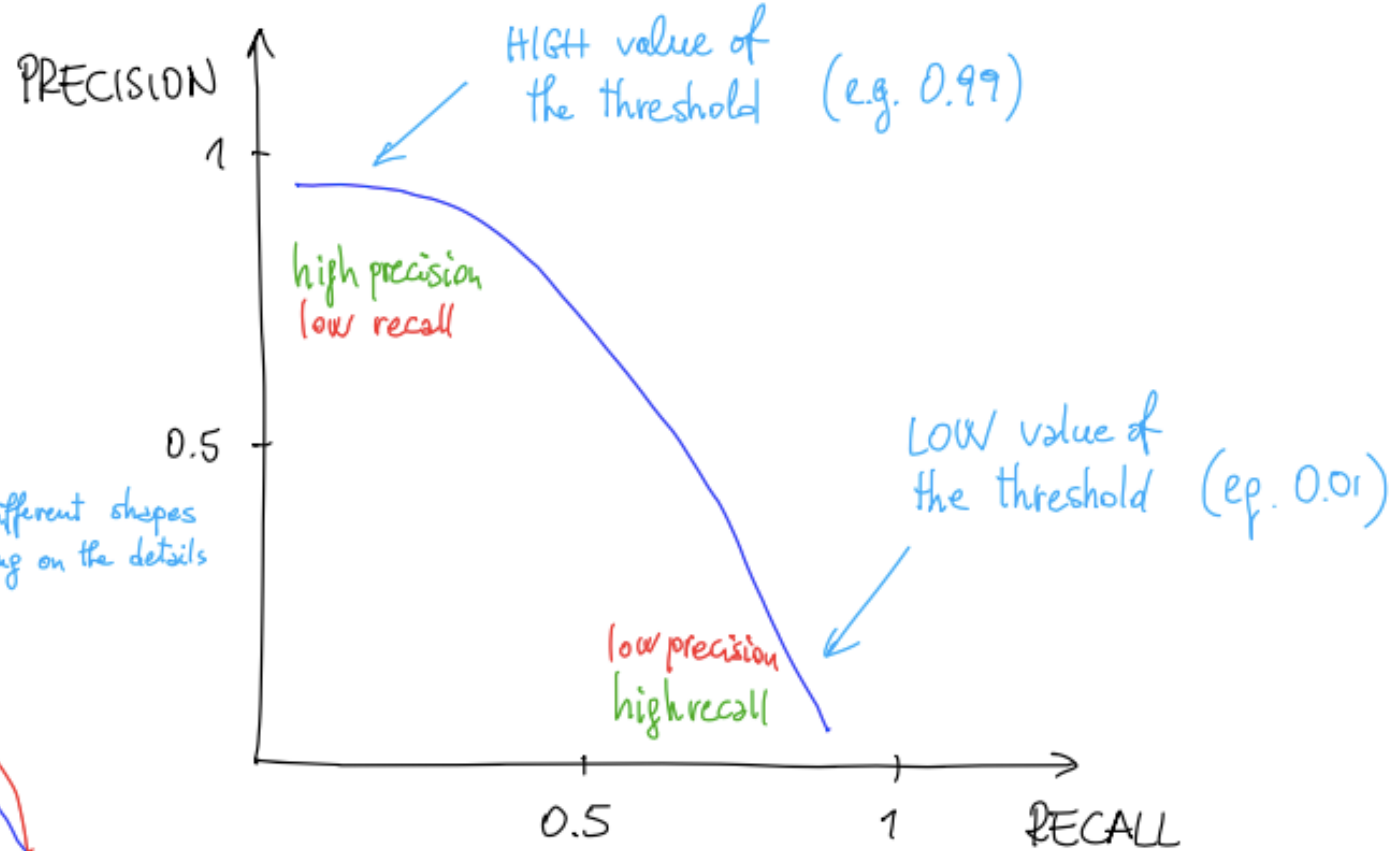
More generally : predict $y=1$ if $h_\theta(x) \geq$ "threshold"

For most classifiers there is a <u>trade-off</u> between precision and recall, and as you vary the threshold you can plot some curve

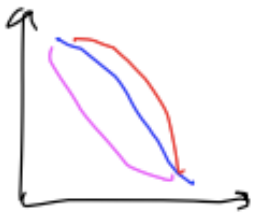


More generally : predict $y=1$ if $h_\theta(x) \geq$ "threshold"

For most classifiers there is a <u>trade-off</u> between precision and recall, and as you vary the threshold you can plot some curve



NOTE : many different shapes possible, depending on the details of the classifiers.



Suppose you have either a) 3 different algos
b) same algo but with 3 different thresholds

	PRECISION (P)	RECALL (R)
algo 1	0.5	0.4
algo 2	0.7	0.1
algo 3	0.02	1.0

would it be easier to just have 1 number, then? (i.e. accuracy)

How do you decide which is best ?
(BTW, we now have 2 raw nbs and not just 1 as metric...)

You can try : average $= \frac{P+R}{2}$

Suppose you have either a) 3 different algos
b) same algo but with 3 different thresholds

	PRECISION (P)	RECALL (R)	average
algo 1	0.5	0.4	0.45
algo 2	0.7	0.1	0.4
algo 3	0.02	1.0	0.51

How do you decide which is best?

You can try : average = $\cancel{\frac{P+R}{2}}$

→ not so good. Best is algo 3, an extreme, that predicts $y=1$ all the time... Not a great way to select the best classifier.

Suppose you have either a) 3 different algos
b) same algo but with 3 different thresholds

	PRECISION (P)	RECALL (R)	F1 score	
algo 1	0.5	0.4	0.444	best
algo 2	0.7	0.1	0.175	
algo 3	0.02	1.0	0.0392	worst

Introduce F1 score (F score), a way to combine P and R:

$$F_1 \text{ score} \overset{\text{def}}{=} 2\,\frac{P \cdot R}{P+R}$$

(just one way!)
→ hystorical...

if one is 0 ⇒ $F_1 = 0$ (~same if small)
Both need to be largish for a good F1

$P=0$ or $R=0 \Rightarrow F_1 = 0$
$P=1$ and $R=1 \Rightarrow F_1 = 1$

Suppose you have either a) 3 different algos
b) same algo but with 3 different thresholds

	PRECISION (P)	RECALL (R)	F1 score	
algo 1	0.5	0.4	0.444	best
algo 2	0.7	0.1	0.175	
algo 3	0.02	1.0	0.0392	worst

Introduce F1 score (F score), a way to combine P and R:

$$F_1 \text{ score} \overset{\text{def}}{=} 2\,\frac{P \cdot R}{P+R}$$

Intuitively: the **F1** can be interpreted as a weighted harmonic mean of precision and recall (best at 1 and worst at 0)

(just one way!) → historical...

if one is 0 ⇒ $F_1 = 0$ (~same if small)
Both need to be largish for a good F1

$P=0$ or $R=0$ ⇒ $F_1 = 0$
$P=1$ and $R=1$ ⇒ $F_1 = 1$

Summary

We discussed the notion of trading off between **precision** and **recall**, and how we can vary the threshold that we use to decide whether to predict y=1 or y=0. By varying the threshold, you can control such trade off

- e.g. "we need to be at least 70% confident or 90% confident in order to predict y=1"

We discussed about the **F1** scorer, a single real number evaluation metric whose definition is based on precision and recall.

If your goal is to automatically set that threshold to decide what's really y=1 vs y=0, one reasonable way to do so would be:

- to try a range of different values of thresholds
- to evaluate these different thresholds on your cv set
- to pick whatever value of threshold gives you the highest F1 score

This is a reasonable way to automatically choose the threshold for your classifier.

1110001110100100110011010010100111010100100100
1011100101010010101010101101001010101011111000
1110100100110011010010100111010100100101010011
011110Advices00for01applying00ML1010101011101010
1010010101001101010011001001010111010101001001
0101010110100101010101011110001110100100110011
0
1001010011101011011010111001010100101010101
1010001010101111000111010010011001101001010011
1101010010010101110010101001001110101101101011
1100101010010101010101101000101010111100011111
10110110Error01metrics01for11skewed10classes1
0010011101011001101011100101010010101010101101
10001110ROC01and11AUC0011010011101010010001101
1101010111000100101100101010101010111010001111
0101010011101011101101010101010110000111000010
0110110101010101010101000111000110101101011110

Re-visit with your memory the “metrics” discussion we had

- accuracy enough or not, more metrics, which ones, etc..

We add few details, in a few simple slides.

ROC curve

Are you comfortable with performance parameters whose values change according to the classification threshold?

A **ROC curve** (**Receiver Operating Characteristic curve**) is a graph showing the performance of a classification model <u>at all classification thresholds</u>.

It is a curve in the TPR-FPR space:

$$\text{True Positive Rate } (\mathbf{TPR}) = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False Positive Rate } (\mathbf{FPR}) = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Reminder:

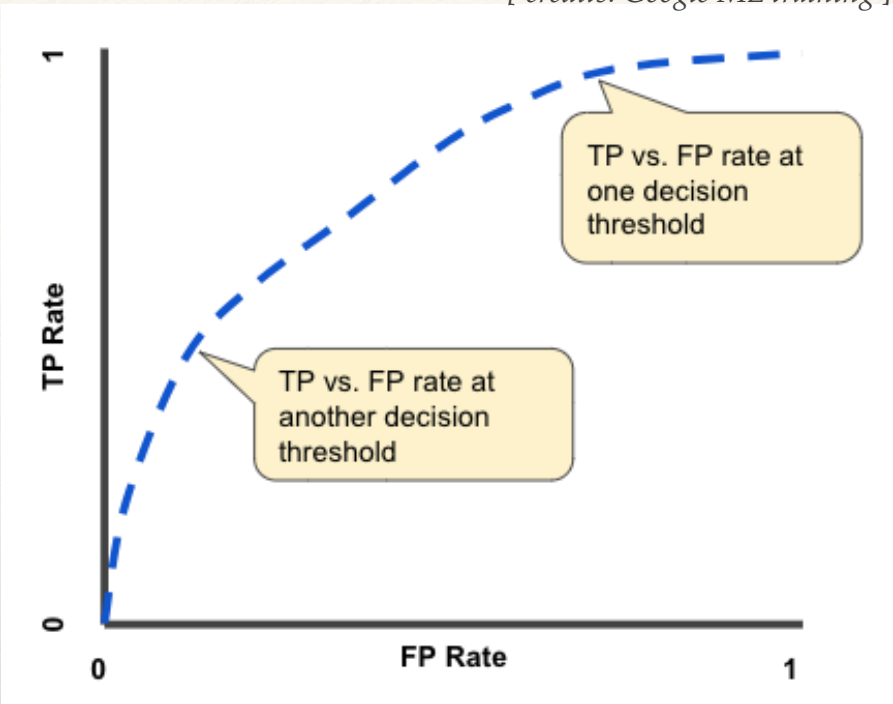
$$\text{PRECISION} \stackrel{\text{def}}{=} \frac{\text{TP}}{\underbrace{\text{TP}+\text{FP}}_{\text{predicted P}}}$$

$$\text{RECALL} \stackrel{\text{def}}{=} \frac{\text{TP}}{\underbrace{\text{TP}+\text{FN}}_{\text{actual P}}}$$

An ROC curve plots TPR vs. FPR at different classification thresholds.

- every point on the ROC curve corresponds to just one value of the classification threshold
- e.g. lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

[credits: Google ML training]



To compute the points in an ROC curve, we could evaluate a logistic regression model many times with different classification thresholds

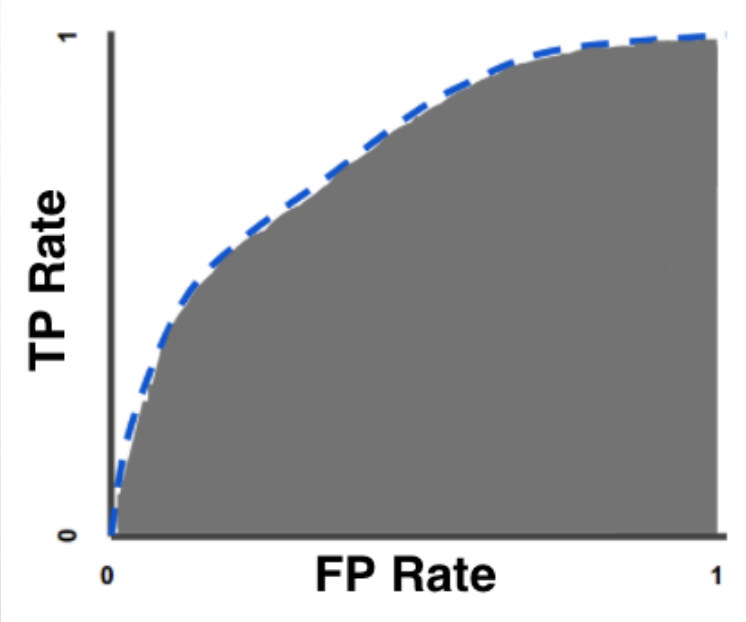
- but this would be inefficient..

Fortunately, there's an efficient, sorting-based algorithm that can provide this information for us, called **AUC**.

AUC: Area Under the ROC Curve

AUC ("**Area under the ROC Curve**") measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1)

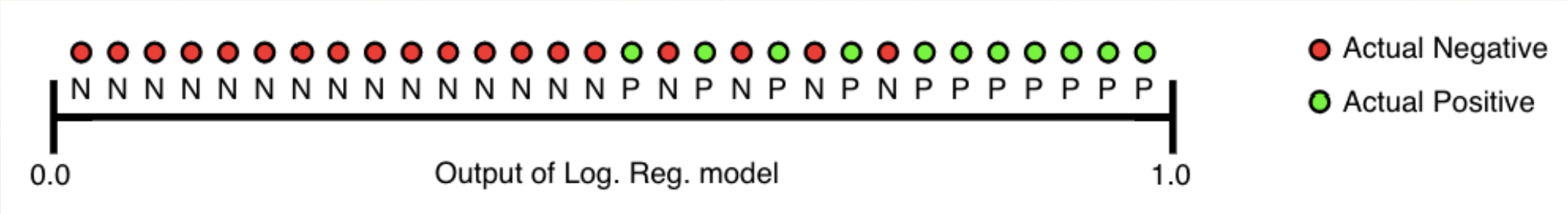
AUC provides an aggregate measure of performance across all possible classification thresholds.



AUC ranges in value from 0 to 1

- a model whose predictions are **100% correct** has **AUC = 1.0**
- a model whose predictions are **100% wrong** has **AUC = 0.0**

AUC: Area Under the ROC Curve



One way of interpreting AUC is as *the probability that the model ranks a random positive example "higher" than a random negative example*

- E.g. take the example above, which are arranged from left to right in ascending order of logistic regression predictions: AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example example

Why AUC?

AUC is desirable for the following two reasons:

- AUC is **classification-threshold-invariant**. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.
- AUC is **scale-invariant**. It measures how well predictions are ranked, rather than their absolute values.

However, both these reasons come with caveats, which may limit the usefulness of AUC in certain use cases:

- **Classification-threshold invariance is not always desirable**. In cases where there are wide disparities in the cost of false negatives vs. false positives, it may be critical to minimise one type of classification error
  - e.g. when doing email spam detection, you likely want to prioritise minimising FP (even if that results in a significant increase of FN). AUC isn't a useful metric for this type of optimisation.
- **Scale invariance is not always desirable**. For example, sometimes we really do need well-calibrated probability outputs, and AUC won't tell us about that.

Applied Machine Learning - Basic

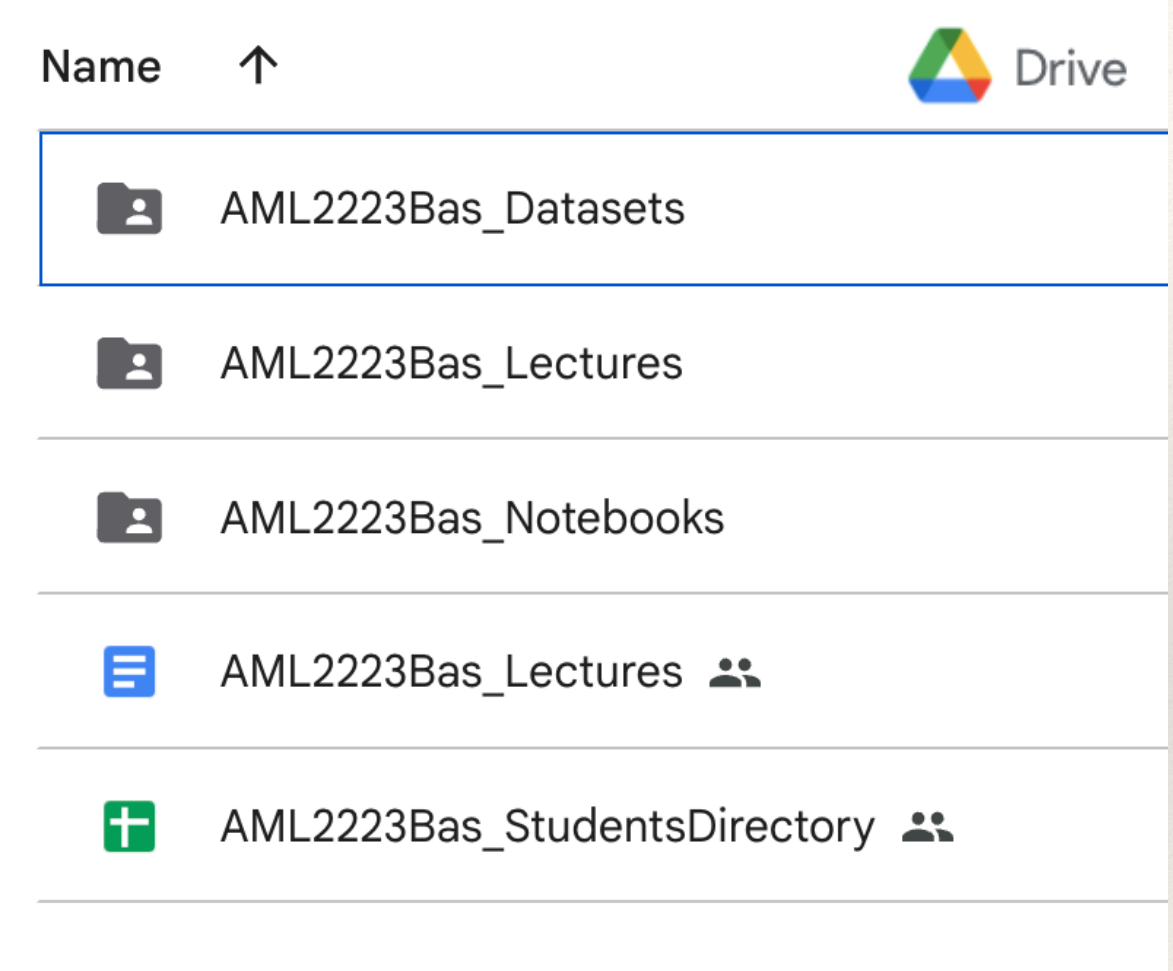
Prof. Daniele Bonacorsi

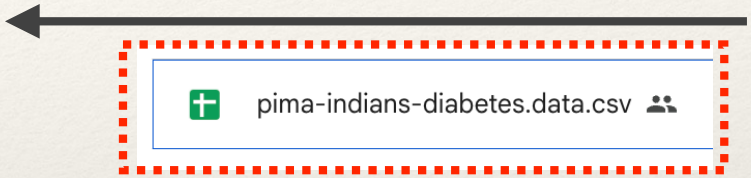
Hands-on:

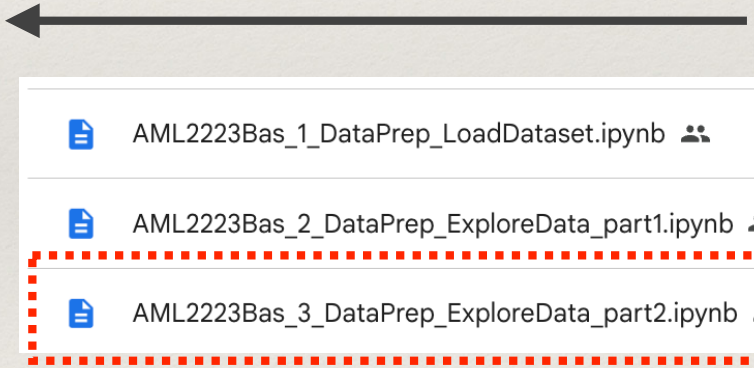
3 - Data prep - explore data (part 2)

Data Science and Computation PhD + Master in Bioinformatics
University of Bologna

Look into these:







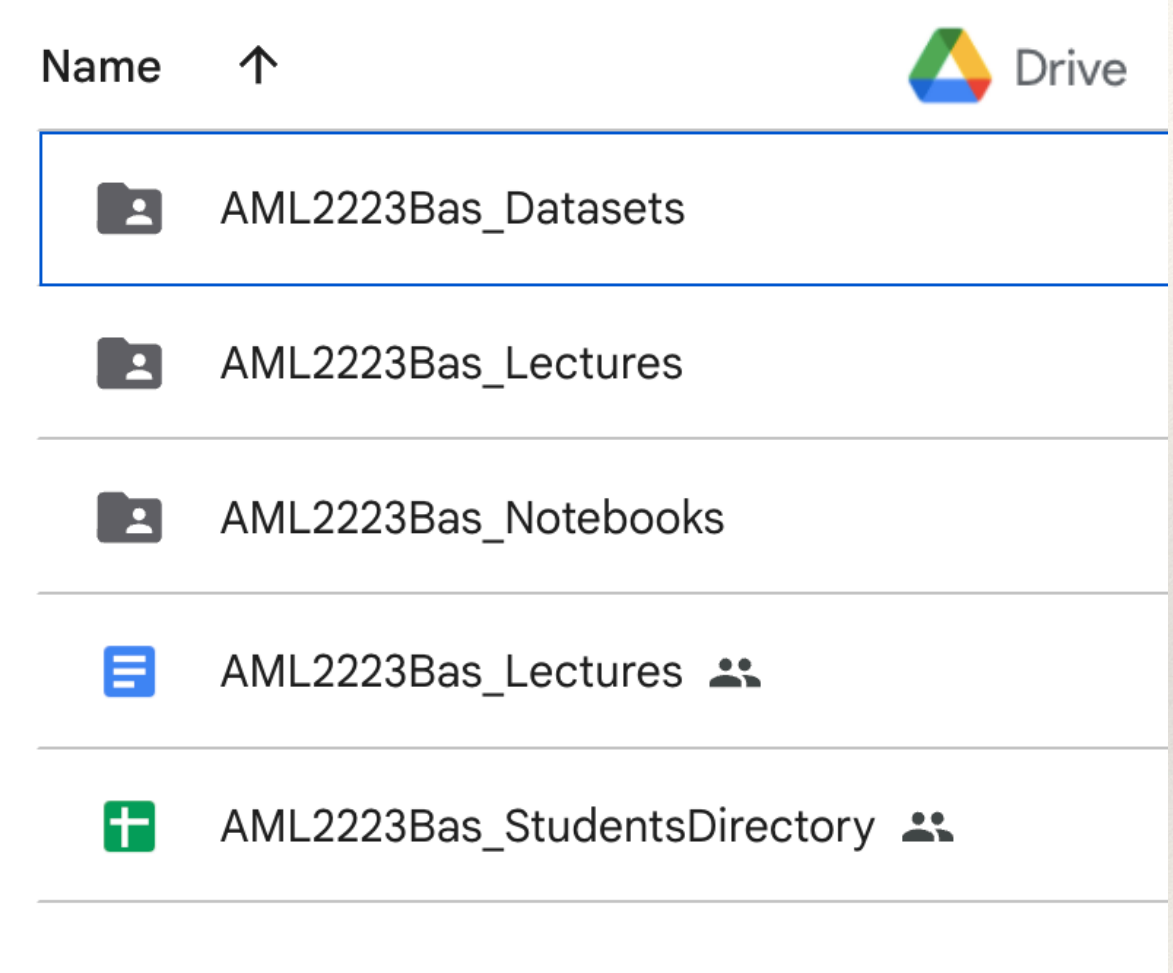
Applied Machine Learning - Basic

Prof. Daniele Bonacorsi

Hands-on:

4 - Data prep - Prepare data

Data Science and Computation PhD + Master in Bioinformatics
University of Bologna



Notebook nb.4

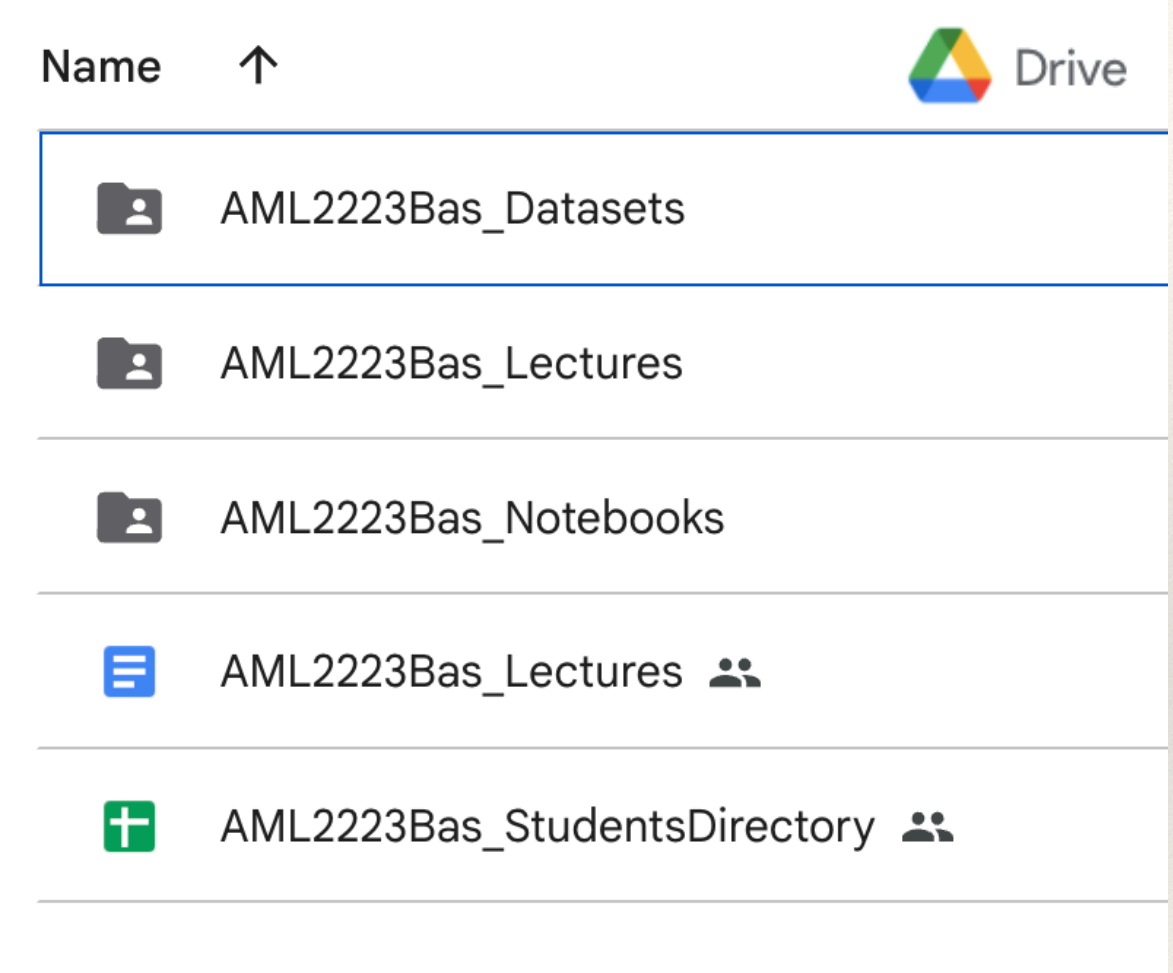
Applied Machine Learning - Basic

Prof. Daniele Bonacorsi

Hands-on:

5 - Data prep - Feature selection

Data Science and Computation PhD + Master in Bioinformatics
University of Bologna



Notebook nb.5