xAI

eXplainable Artificial Intelligence

What xAI is (“opening the black box”)

- Deep Neural Networks functions as “black boxes”
    - Intrinsically non-interpretable, “raw”, features (pixels, voxels, wavelets, *etc*.)
    - Highly non-linear (“hierarchical representations”)
- *xAI is a broad field of research in AI concerning development of methods to increase trust and understanding of ML model’s predictions*
    - xAI tries to make the “box” more transparent and understandable
    - “Why did you think that’s a cat?”
- The “shades-of-grey-box” problem
    - Trade-off between interpretability and representation power of ML models
    - Simple models (*e.g.*, linear models, decision trees) interpretable, yet admittedly not very powerful
    - Complex models (like deep neural networks) powerful, yet difficult “to understand”

What is an explanation?

- It depends on whom you ask
  - Developers (“is it working?”), expert users (*e.g.*, doctors; “which elements influenced the decision?”), end users (*e.g.*, patients; “what does it mean?”)
  - xAI mainly skewed towards developers
- What’s a good explanation?
  - (Most likely) not a simple rule
  - Visual, conceptual (text), exemplars?
  - Local vs global
  - Different methods can (and do) disagree
  - There are no clear measures to quantify the goodness of an explanation
  - (Is segmentation an explanation?)
- Is everything explainable?
  - Decision boundaries in 4+ dimensions
  - Blaise Pascal and the *esprit de finesse*
  - Split-brain patients and flashing words

Why xAI is important

- Helps ML developers understand system outputs simply and quickly
- Can suggest solutions and spot anomalies to investigate
- Makes possible to understand why a mistake was made
  - Or was not made (correct answer for the “wrong” reason)
  - Dog in the snow = wolf
- And train the system to stop it from happening again
- Helps to meet the “right to explanation” required by the GDPR
- Helps in avoiding discrimination and biases in decisions

Evaluating xAI

- The “human side”
  - Understandability: Can humans easily grasp the explanations provided by the xAI method?
  - Actionability: Do the explanations help users make informed decisions or take appropriate actions?
  - Trustworthiness: Do users trust the explanations and the underlying AI model?
- The “machine side”
  - Fidelity: How accurately do the explanations reflect the true workings of the AI model?
    - “Decision gradient” aligns with explanation? Counterfactuals do change the decision?
  - Sensitivity: Are explanations stable and consistent across similar inputs?
    - Perturbation analysis
  - Completeness: Do explanations cover all relevant factors?
    - Comparing explanations with domain knowledge or alternative xAI methods
- The “interface side”
  - Do humans change their attitude toward/their way of using the AI system?
  - Is it this change appropriate/useful/sound?
- It’s all quite difficult to quantify!

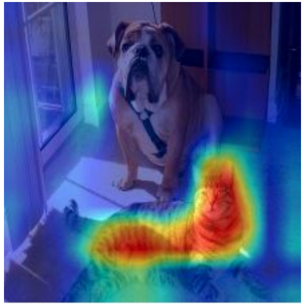
Taxonomy of xAI methods

- Importance methods
  - What part of the input data influenced the decision (and how much)?
  - Region in an image, features, “representations”
- Exemplar methods
  - What training points impacted most the decision? - “Proponents” and “opponents”
  - Counterfactual explanations: “minimal” input change to make the machine change its mind
- Distillation methods
  - Train an interpretable (“white box”) model to reproduce the output of a complex model
  - “Structural” distillation (*e.g.*, NN into a random forest), “complexity” distillation (large NN into smaller NN)
- Intrinsic methods (“Interpretable AI”)
  - Models that provide, alongside the main output, an explanation
  - Optimizing both model performance and a certain quality of the explanations produced
  - Prototypes, templates, concepts

Importance examples

- CAM (Class Activation Maps), Grad-CAM, Integrated Gradients, Smooth Gradients
  - Aim to identify which regions in the image were relevant to a specific class
  - By looking at the magnitude of the gradients (class C_k, representation R_i) flowing through the network layers
  - Useful to measure how much each pixel/region activate the class predicted by the network
- Occlusion sensitivity
  - “Perturbing” (masking) the input
  - And looking at the effects on classification
  - Saliency = output variation

$$\frac{\partial C_k}{\partial R_i}$$



predicted class: cat

class dog

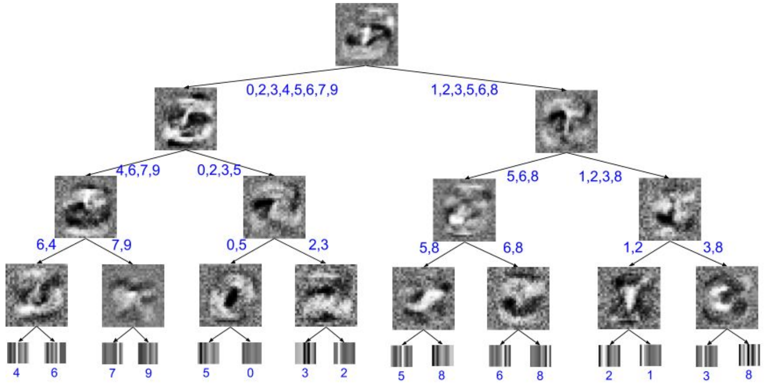
Exemplar example: Gradient tracing/TrackIN

- Data influence: How much has this training point influenced the prediction for this test point?
- Retrain the model without the training point and test
- This is prohibitively expensive!
- Resorting to approximations

$$\sum_{e} \nabla_{\mathbf{w}_e} \mathrm{loss}(x_{\mathrm{train}}) \cdot \nabla_{\mathbf{w}_e} \mathrm{loss}(x_{\mathrm{test}})$$

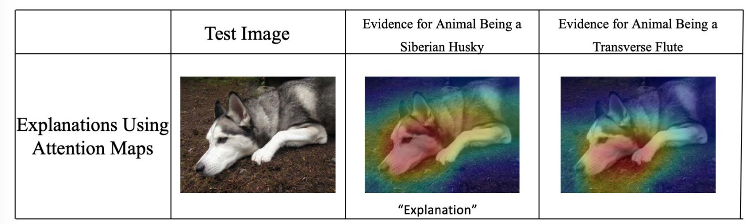
Distillation example

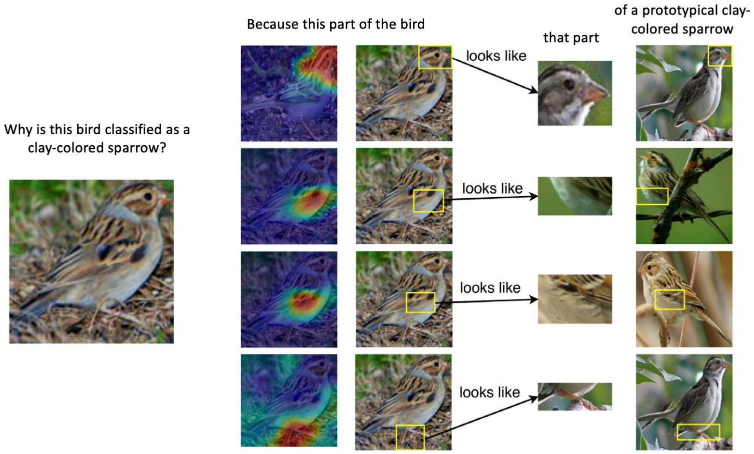
- DNN into a “soft decision tree”
- Each node is a “learned representation”
- Explanation = most probable path
- Why distillation?
  - Simply, the DNN is more powerful
  - And you can build better decision trees by using “**soft label**” generated by the DNN



Intrinsic example

- Heat-maps look good, don't they?
  - Explanation seems more like "attention", not related to the specific class
  - They can be quite "unsensitive" (distillation can help with that)
- "This looks like that" — prototypes
  - The network dissects the image by finding prototypical parts
  - And combines evidence from the prototypes to make a final classification
  - Explanation mimicking (in part) the way an ornithologist would reason





C.Chen et al., "This Looks Like That: Deep Learning for Interpretable Image Recognition", NeurIPS 2019

Extreme gradient boosting (XGBoost)

Why XGBoost?

The XG Boosting algorithm uses advanced regularization techniques to suppress weights, prevent overfitting, and enhance its performance in real-world scenarios. On top of this, the implementation allows the algorithm to cache data and utilize multiple CPU cores for speedy processing. The enhanced performance and speed have made ***XGBoost one of the most popular machine-learning algorithms in recent years****.*

What is XGBoost?

XGBoost gives a prediction function f, by regressing a set of parameters θ on a dataset of predictors $\mathbf{X}$ related to the dependent variable of interest y.

Function f is non-linear and admits general interactions between predictors.

$$y = f(\mathbf{X}|\theta)$$

Ensemble ML

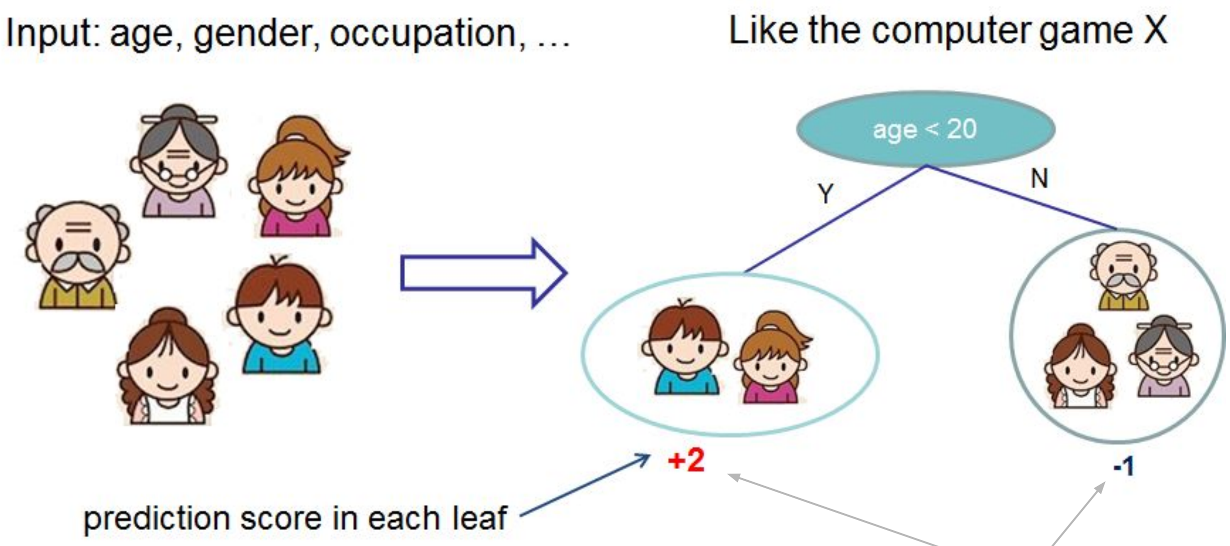
- “Weak” learners: simple models (simple = does not overfit)
- Ensembles of weak learners can be very strong
- Bagging (parallel)
  - Each weak learner “looks” only at part of the features
  - We average (or count the “votes”) the answers of the weak learners
  - Each weak learner is trained independently
- Boosting (sequential)
  - The first weak learner approximates $\hat{y}^{(1)} = f^{(1)}(x) \approx y$
  - The second one learns $\hat{y}^{(2)} = y - \hat{y}^{(1)}$
  - The *t*-th learner learns $\hat{y}^{(t)} = y - \hat{y}^{(t-1)}$
  - “Gradient boosting” is a generalisation

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

$$
\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} \omega(f_i) \\
&= \sum_{i=1}^{n} [2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2] + \omega(f_t) + \text{c} \\
&= 2\sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \omega(f_t) + \text{c}
\end{aligned}
$$

g stands for gradient
h stands for… second derivative (Hessian)

Trees



$$f_t(x) = w_{q(x)}$$

Deep trees overfit
Shallow trees are weak learners

Learning the leaf weights

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

Number of leafs

Regularisation

$$\begin{aligned}
\text{obj}^{(t)} &= 2\sum_{i=1}^{n}[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\
&= 2\sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T \\
&= 2\sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T
\end{aligned}$$

Best solution

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

$$\text{obj}^* = -\frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

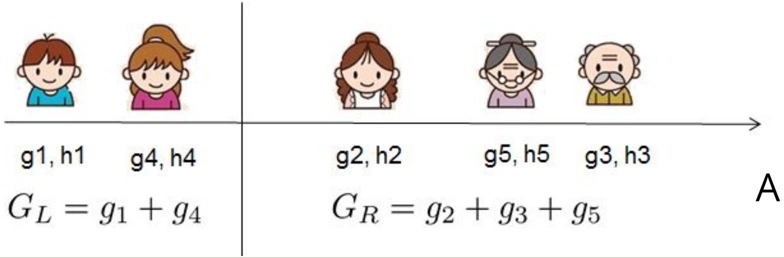
Learning the tree structure

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

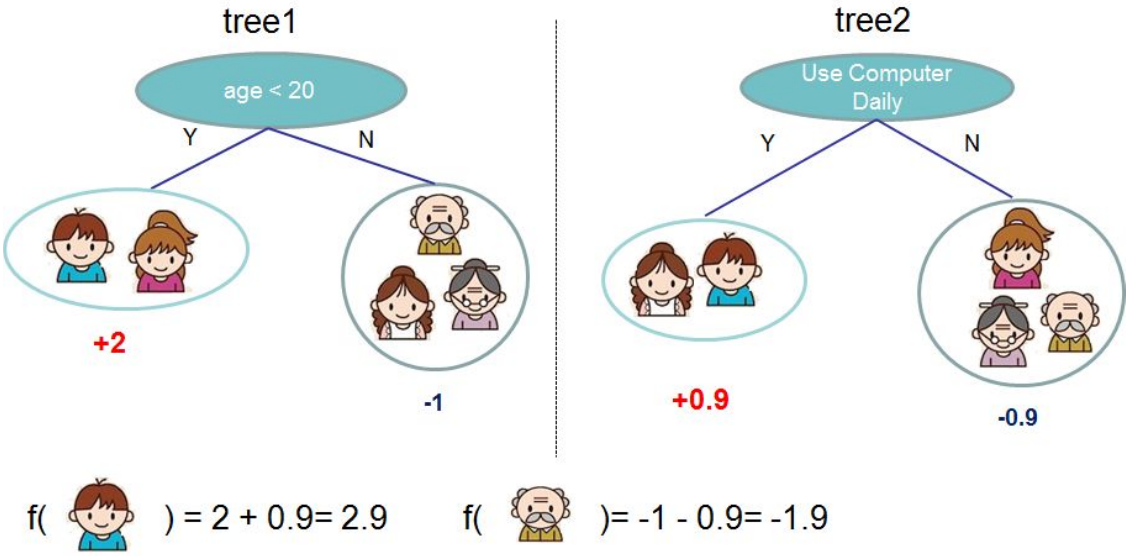
$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}-\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right]-\gamma$$

If the gain is smaller than **γ**, we would do better not to add the new branch (this is usually called "**<u>pruning</u>**")



A left to right scan is sufficient to find the best split

Forests



Bagging:
Random Forest

Boosting:
Boosted trees

What about “extreme” in XGBoost?

*“XGBoost is an **optimized distributed** gradient boosting library designed to be **highly efficient, flexible, and portable**. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve many data science problems in a **fast and accurate** way. The same code runs on major distributed environment and **can solve problems beyond billions of examples**”*

There are other gradient boosting libraries, such as LightGBM and CatBoost

SHAP

What is SHAP?

Here SHAP is just an algorithm that, given the function *f* returned by XGBoost and given one individual i (one set of values for the predictors $\mathbf{X}_i$ - for example: age 37, BMI 28.4, systolic blood pressure 135), estimates the contribution of each predictor to the specific prediction y_i.

In simple words, SHAP gives a principled estimation of, for example, **how much the prediction y_i would change if the age of individual i were not observed**

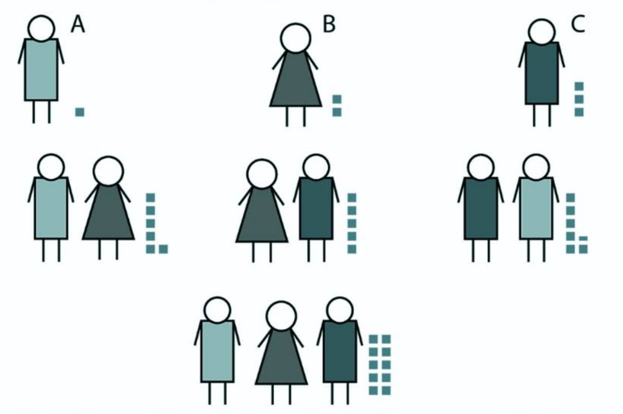
$$\Delta y_i = \mathrm{SHAP}(\mathrm{age}_i)$$

$$\mathrm{SHAP}(\mathrm{age}_i) = f(\mathbf{X}_i|\theta) - f(\mathbf{X}_i \setminus \mathrm{age}_i|\theta)$$

SHAP is an Importance Method

Shapley value

- A game theory concept introduced by Lloyd Stowell Shapley (Nobel Prize in economic sciences)
- **The Shapley value gives a "fair" estimate of the contribution of each player participating in a collaborative work**
- There is a game, there are N players; for each possible team T of k (≤ N) players there is a payoff V(T) (that measures the "value of the team")
- Knowing V(T) for all the possible teams, how can we measure the "value of each single player"?



$$\begin{aligned}\varphi_i(v) &= \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(n-|S|-1)!}{n!}(v(S \cup \{i\}) - v(S)) \\ &= \sum_{S \subseteq N \setminus \{i\}} \binom{n}{1, |S|, n-|S|-1}^{-1} (v(S \cup \{i\}) - v(S))\end{aligned}$$

SHAP

- In SHAP the players are the predictors
- You have a predictive function/model (XGBoost, in our case) $f(\mathbf{X}|\theta)$ and a data point $\mathbf{X}_i$ you want to analyse
- The value V(T) of a team T (a subset of the predictors) is the prediction given by f when you (conditionally) average over the predictors NOT in T
- Very hard to compute exactly. SHAP makes some simplifying assumptions
- These simplifications make SHAP very efficient to compute for Decision Trees (the atomic element of XGBoost) → **TreeSHAP**
- There exist many instantiations of SHAP, for different machine learning algorithms

Individualists, synergies, and conflicts

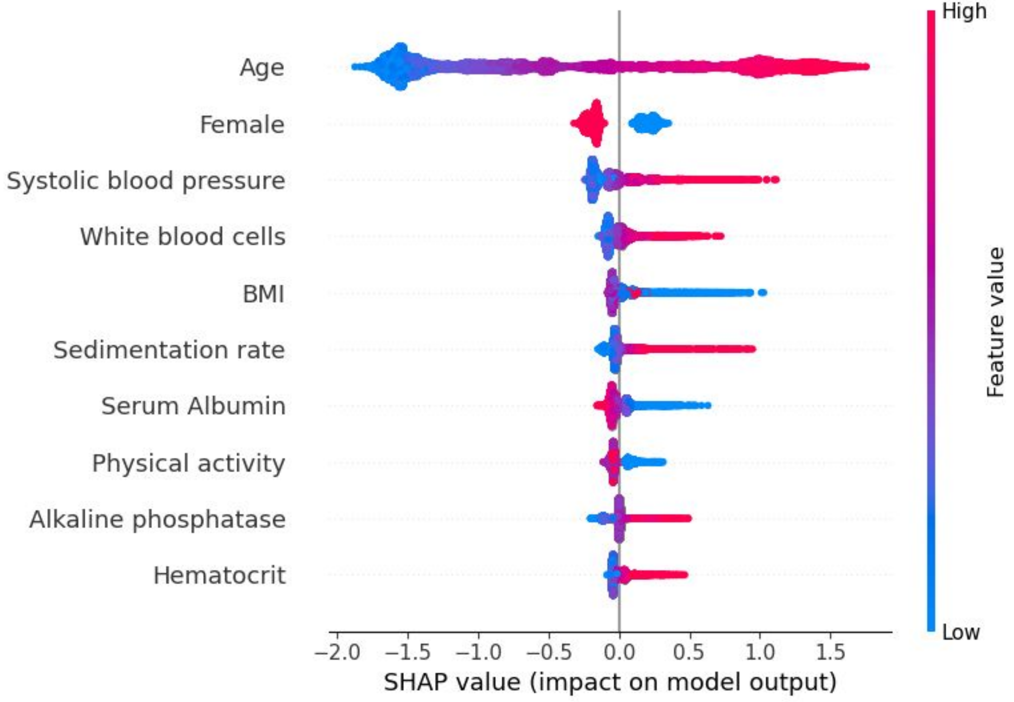
- An individualistic player C is one that gives the same "boost" v_C to every team (**v_{AC} = 0** for all players A, with A != C)
- Synergy: player A adds v_A; player B adds v_B; $V(A + B) > v_A + v_B$
  - A and B boost each other game (**v_{AB} > 0**)
- Conflict: player A adds v_A; player B adds v_B; $V(A + B) < v_A + v_B$
  - A and B compete for the same "role" (**v_{AB} < 0**)
- **Main effect** and **interaction values**

$$V(T) = \sum_{A \in T} v_A + \frac{1}{2} \sum_{(A,B) \in T} v_{AB}$$

$$\boxed{\mathrm{SHAP}_A = v_A + \frac{1}{2} \sum_B v_{AB}}$$

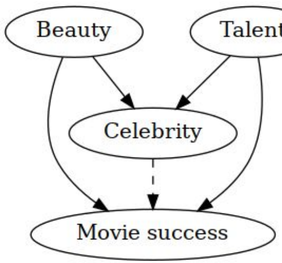
A quick example on mortality

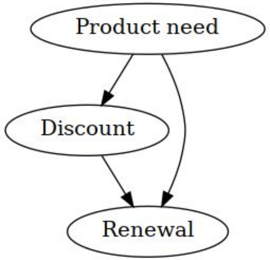
- Age is by far the most significant predictor
- Sex (second place) acts largely as a constant factor
- There is, quite consistently, a monotonic relationship between feature value (*e.g.*, Age) and the effect on prediction (SHAP)

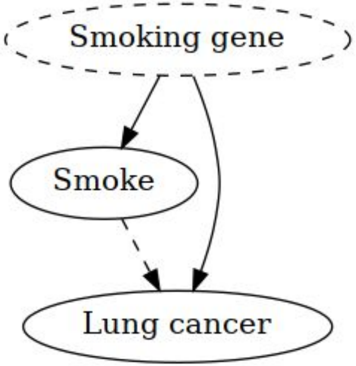


Explanation is not causation

- AI doesn’t try to build a “physical model” of the data
- AI just “wants to predict”
- AI can discover, and take advantage of, highly complex and non-linear “correlations” among predictors, and between predictors and response
    - A good collider can be better than the real causes (and more parsimonious)
    - Direct and mediated effects entangled in explanations
- xAI just unearths these “correlations on steroids”
    - Don’t forget that <u>you are explaining your model, not reality</u>
- And, of course, unobserved confounders are the hardest problem with or without AI







Yet...

- It is difficult to dismiss the predictive power of AI
  - Prediction is important *per se* (Where and when can I intervene most effectively?)
  - And there are anyhow many factors that we cannot control, but are determinant
- And the patterns of “influence” uncovered by SHAP are just too intriguing
- At the very least, AI + xAI represents today one of the most promising instruments for scientific *exploration*
- As for *explanation*:
  - Strongly nonlinear causal effects (*e.g.*, U-shaped, one predictor “gating” the effects of another) will be mostly under-detected and under-measured by traditional methods
  - xAI can make the most out your prior knowledge
  - There are attempts to make AI (and xAI) more “causality-aware”
  - **Double Machine Learning** tries to extract casual knowledge from observational data
- But we leave that for a second course...

Ci vediamo **giovedì** prossimo (27 giugno) in Aula Nitti-Bovet (qui di fronte), via del Castro Laurenziano, 10!