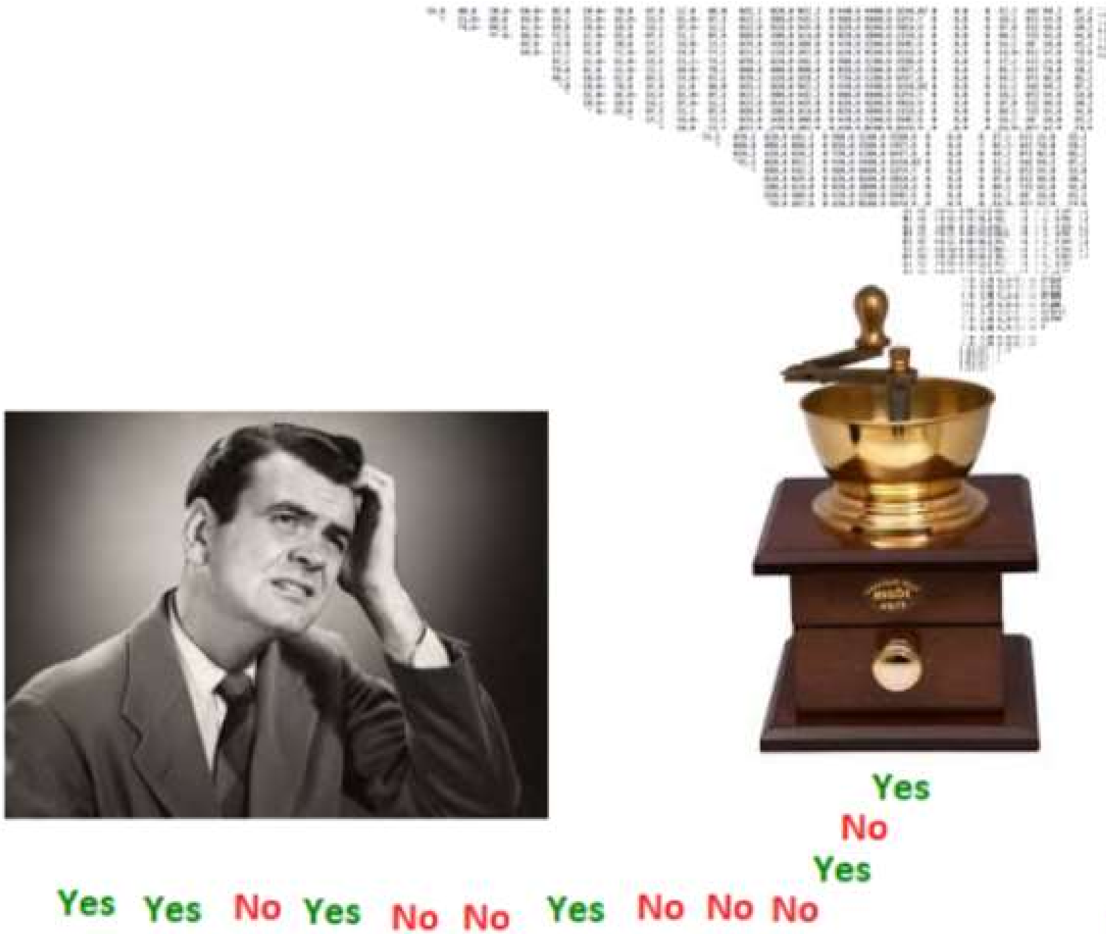
Inspired solutions

YUXI GLOBAL

www.yuxiglobal.com

TCXP - A Scalable Algorithm for Explaining Individual Tree-based Classifier Predictions

mateo.restrepo@yuxiglobal.com - Head of Data Analytics at Yuxi Global, Medellín



Inspired solutions
YUXI
GLOBAL

Outline

- Context and terminology
- Explanations??? Please explain yourself!
- What is TCXP?
- TCXP vs. LIME
- Demo on real data

Inspired solutions
YUXI
GLOBAL

Context and terminology

A **(hard) binary classifier** is just a (computable) function C that takes a vector of covariates (features) and outputs a a result in $\{0, 1\}$

$$\begin{aligned} C: \quad \mathbb{R}^f \to & \quad \{-1, +1\} \\ \mathbf{x} \mapsto & \quad y \end{aligned}$$

A **(soft) binary classifier** is :

$$\begin{aligned} p : \mathbb{R}^f & \to [0, 1] \\ \mathbf{x} & \mapsto p^+(\mathbf{x}) \end{aligned}$$

$p^+(\mathbf{x})$ is interpreted as the (estimated) probability that $\mathbf{x}$ belongs to the *positive class*.

Inspired solutions
YUXI
GLOBAL

Context and terminology

A **(hard) binary classifier** is just a (computable) function C that takes a vector of covariates (features) and outputs a a result in $\{0, 1\}$

$$
\begin{aligned}
C: \quad \mathbb{R}^f \to & \quad \{-1, +1\} \\
\mathbf{x} \mapsto & \quad y
\end{aligned}
$$

A **(soft) binary classifier** is :

$$
\begin{aligned}
p : \mathbb{R}^f & \to [0, 1] \\
\mathbf{x} & \mapsto p^+(\mathbf{x})
\end{aligned}
$$

$p^+(\mathbf{x})$ is interpreted as the (estimated) probability that $\mathbf{x}$ belongs to the *positive class.*

(supervised) Machine learning is the ~~science~~ *art* of automatically constructing an optimal C (or p^+) from many examples $\{(\mathbf{x}^{(i)}, y^{(i)}) \ : \ i = 1, \ldots, N\}$.

Examples from industry

- **Credit risk:** $\mathbf{x}$ = information about a customer and a credit product $p^+(\mathbf{x})$ is the probability that she will default.
- **Customer churn:** $\mathbf{x}$ = information about a customer's behavior $p^+(\mathbf{x})$ is the probability that he will stop being my client.
- **Online-advertisement:** $\mathbf{x}$ = information about an online ad and a person that is looking at it $p^+(\mathbf{x})$ the probability that person will click on it

Explanations (or lack thereof) in the context of ML models

- Explanation for prediction:
  - Answer the question: **Why** did the model predict $\hat{y}^{(i)}$ on input $\mathbf{x}^{(i)}$?
  - What was each *feature*'s contribution to the prediction?

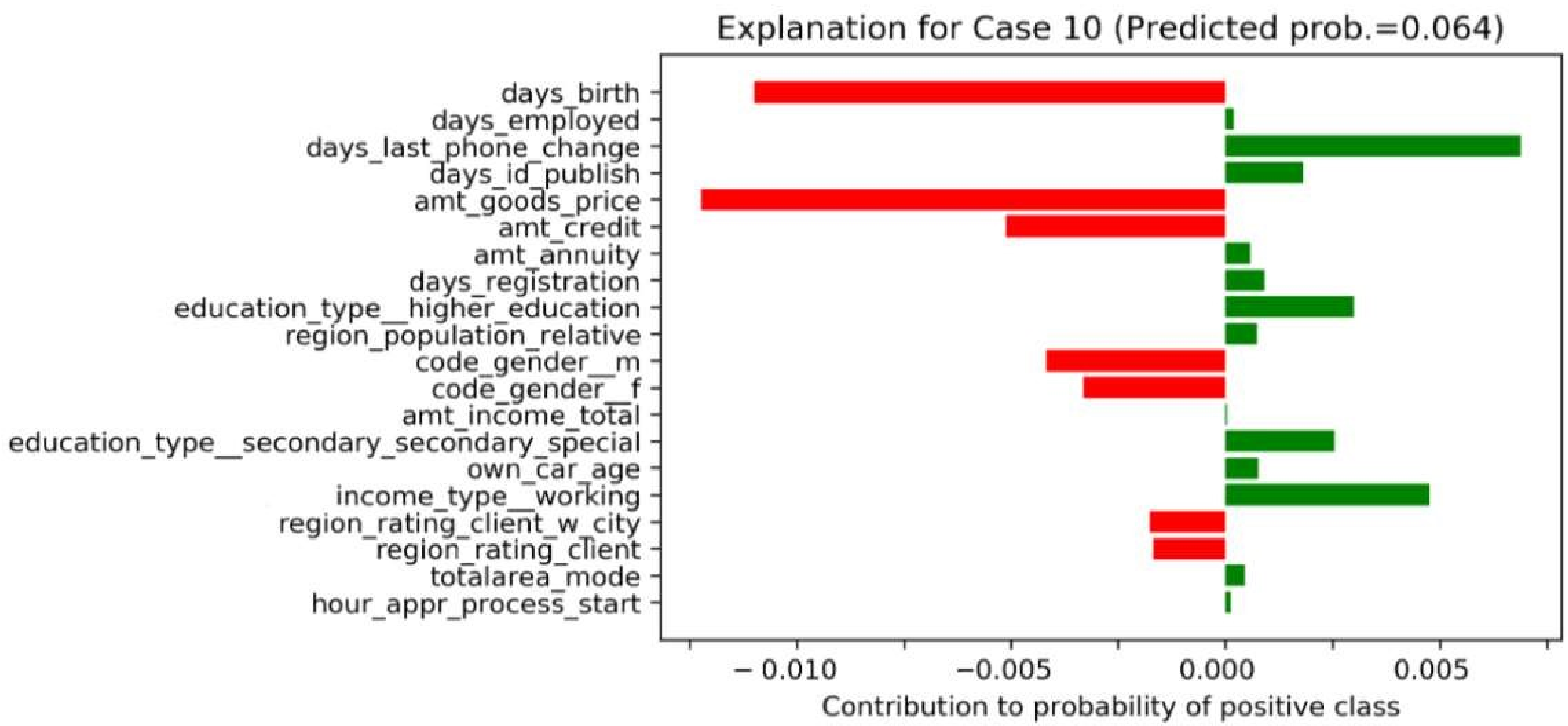
Explanations (or lack thereof) in the context of ML models

- Explanation for prediction:
  - Answer the question: **Why** did the model predict $\hat{y}^{(i)}$ on input $\mathbf{x}^{(i)}$?
  - What was each *feature*'s contribution to the prediction?

- In industries (such as banking, insurance):
  - Sales staff sometimes ask about individual predictions...
  - Predictive analytics promises **actionable insights**:
    - Individual prediction $\rightarrow$ individual action

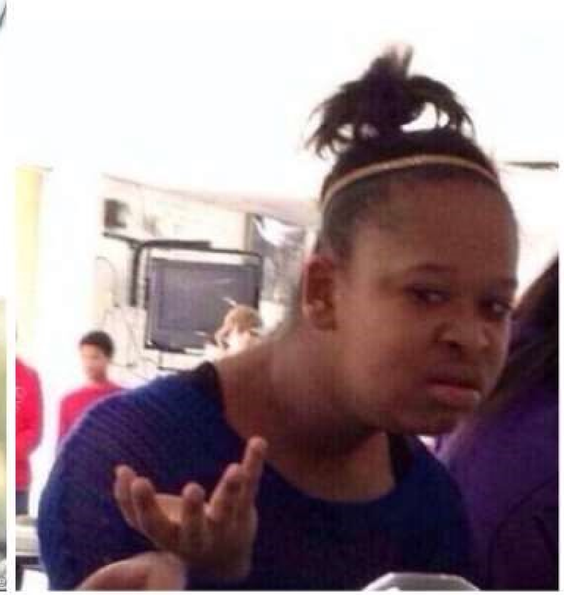
Inspired solutions YUXI GLOBAL

- **One way to implement**: Quantify each *feature*'s contribution to the prediction?
- Something like this:



Explaining advanced ML algorithms to sales staff

Explaining advanced ML algorithms to sales staff



Inspired solutions
YUXI
GLOBAL

Besides... You need to produce explanations BY LAW!

- *Algorithmic Fairness Provisions* of the **General Data Protection regulation (GDPR)**:
- *"The Right to Explanation of Automated Decision mandates that the data subject has a right to get an explanation about decisions made by algorithms and a right to opt-out of some algorithmic decisions altogether if they are not satisfied with it."*
- Article: Deep Learning going illegal in Europe

Inspired solutions
YUXI GLOBAL

A dichotomy

- Predictions by *simple algoritms* (e.g for **Logistic regression**) are "easy" to explain.

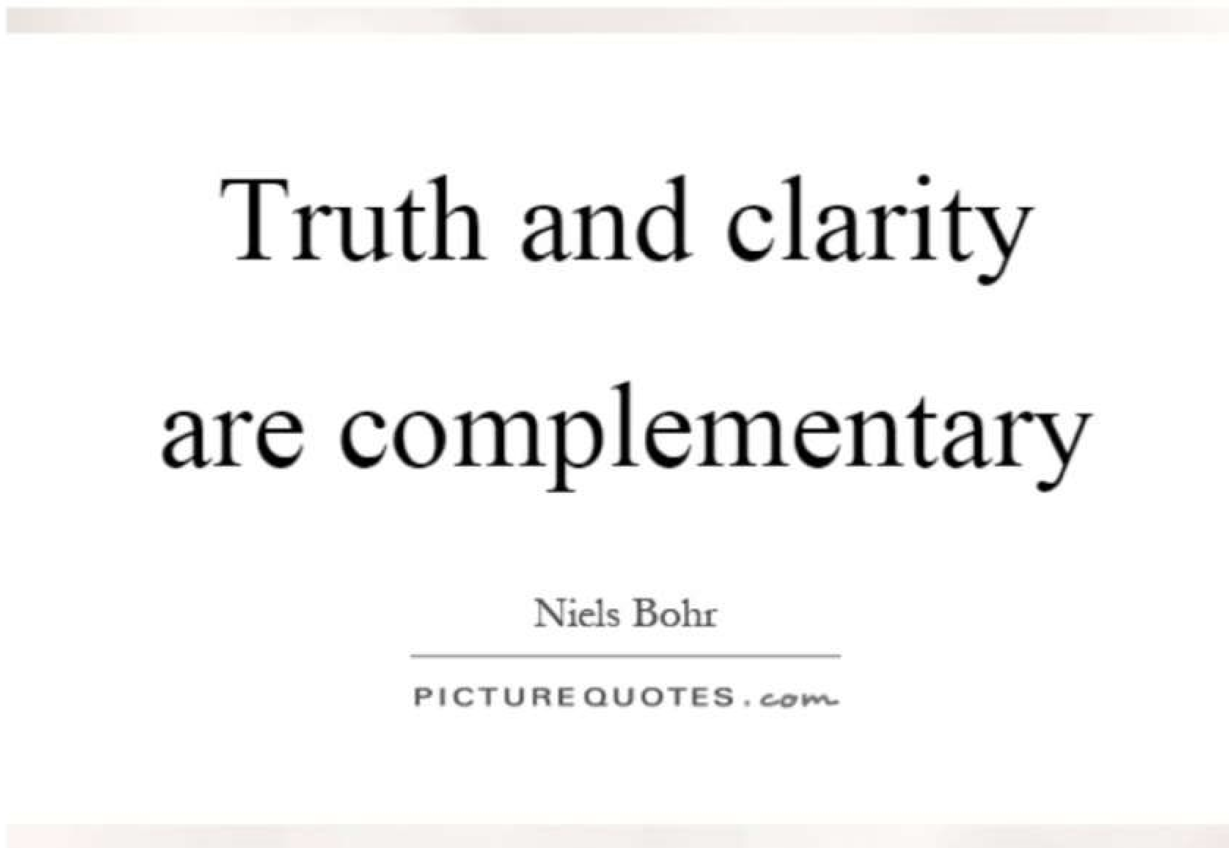
- Predictions by *advanced algorithms*, e.g. random forests, neural networks, XGBoost are *hard* to explain
  - black-box nature and high internal complexity of these models

Inspired solutions
YUXI
GLOBAL

A dichotomy

- Predictions by *simple algoritms* (e.g for **Logistic regression**) are "easy" to explain.

- Predictions by *advanced algorithms*, e.g. random forests, neural networks, XGBoost are *hard* to explain
  - black-box nature and high internal complexity of these models



What is TCXP?

- An algo to generate **interpretable explanations** for *individual* tree-based classifier predictions.
    - **Simple** and **scalable**

What is TCXP?

- An algo to generate **interpretable explanations** for *individual* tree-based classifier predictions.
  - **Simple** and **scalable**

- **Definition:** An **explanation** fo an individual prediction $p^+(\mathbf{x}(i))$:

$$(p_0(i), \Delta p_1(i), \ldots, \Delta p_f(i)) \quad \text{such that}$$

$$p_0(i) + \sum_{j=1}^{f} \Delta p_j(i) = p^+(\mathbf{x}^{(i)})$$

  $\Delta p_j(i)$ is interpreted as the *contribution* to the prediction coming from the j-th feature, $x_j(i)$.

What is TCXP?

- An algo to generate **interpretable explanations** for *individual* tree-based classifier predictions.
  - **Simple** and **scalable**

- **Definition:** An **explanation** fo an individual prediction $p^+(\mathbf{x}(i))$:

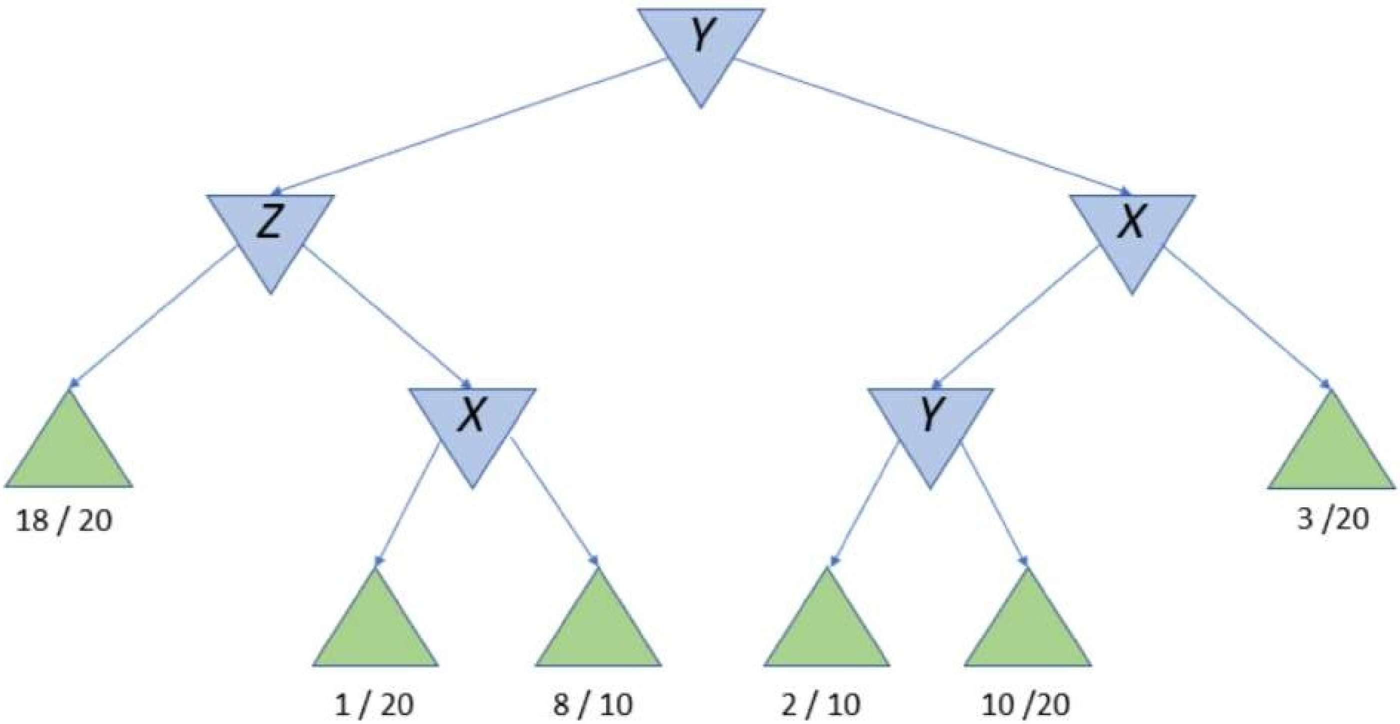
$$(p_0(i), \Delta p_1(i), \ldots, \Delta p_f(i)) \quad \text{such that}$$

$$p_0(i) + \sum_{j=1}^{f} \Delta p_j(i) = p^+(\mathbf{x}^{(i)})$$

  $\Delta p_j(i)$ is interpreted as the *contribution* to the prediction coming from the j-th feature, $x_j(i)$.

- **How**?
  - **Basic idea:** carry out *careful accounting* of probability contributions of each variable.

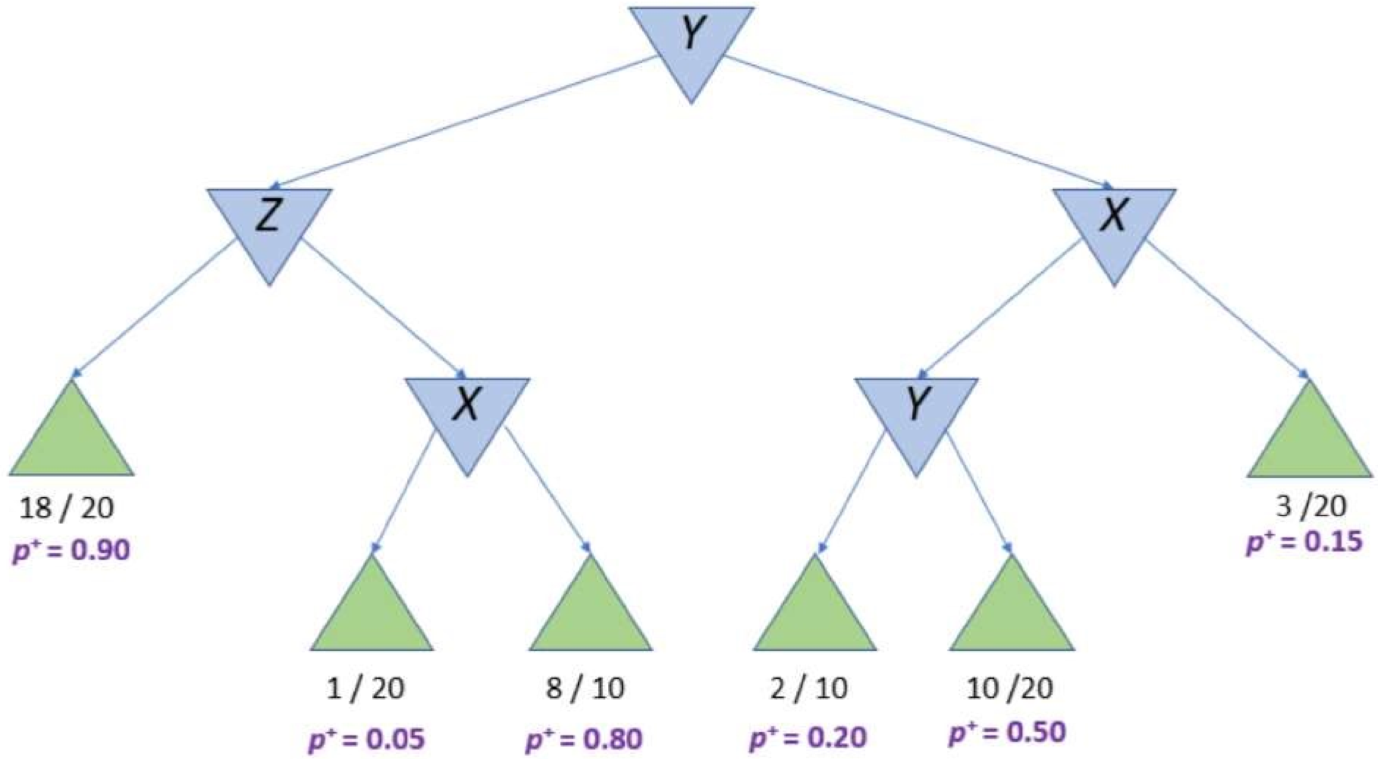
Binary classification through a tree: leaf counts



A classification trees has **internal decision** nodes, each using a single variable, and final (non-decision) **leaves**

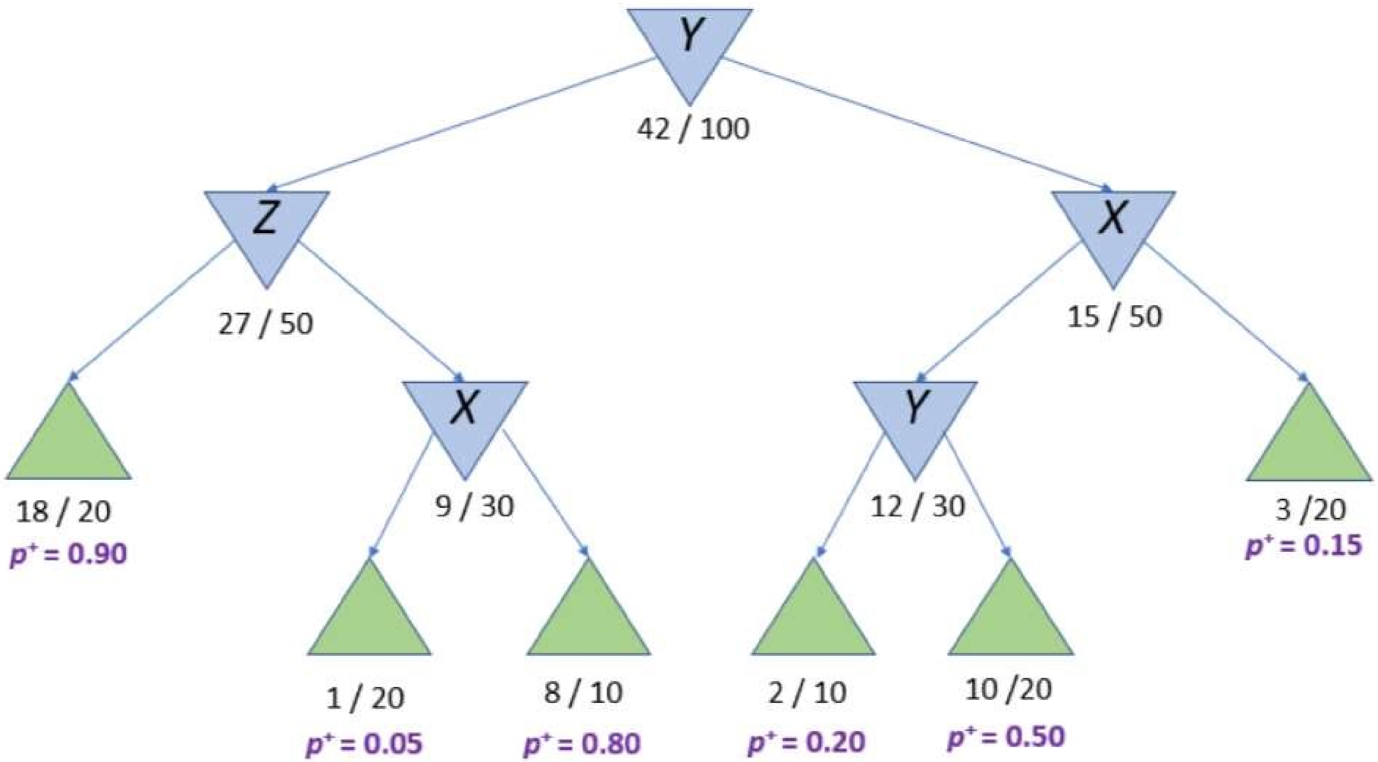
For each leaf node k we record count of positive class over total count: $(n_k^+,\ n_k^+ + n_k^-)$

Binary classification through a tree: probability estimates



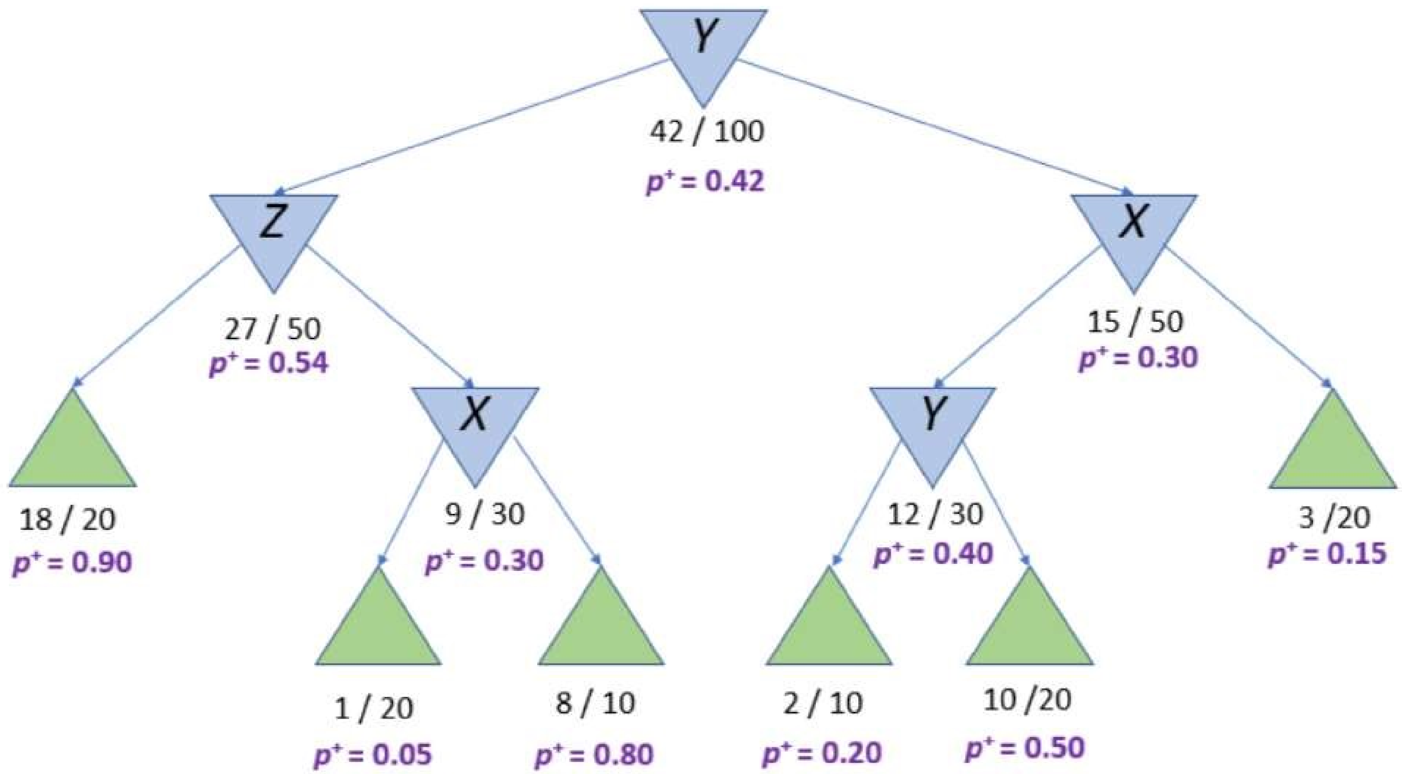
For each leaf $k, p_k^+ = n_k^+/(n_k^+ + n_k^-)$

Explanation generation: all node counts



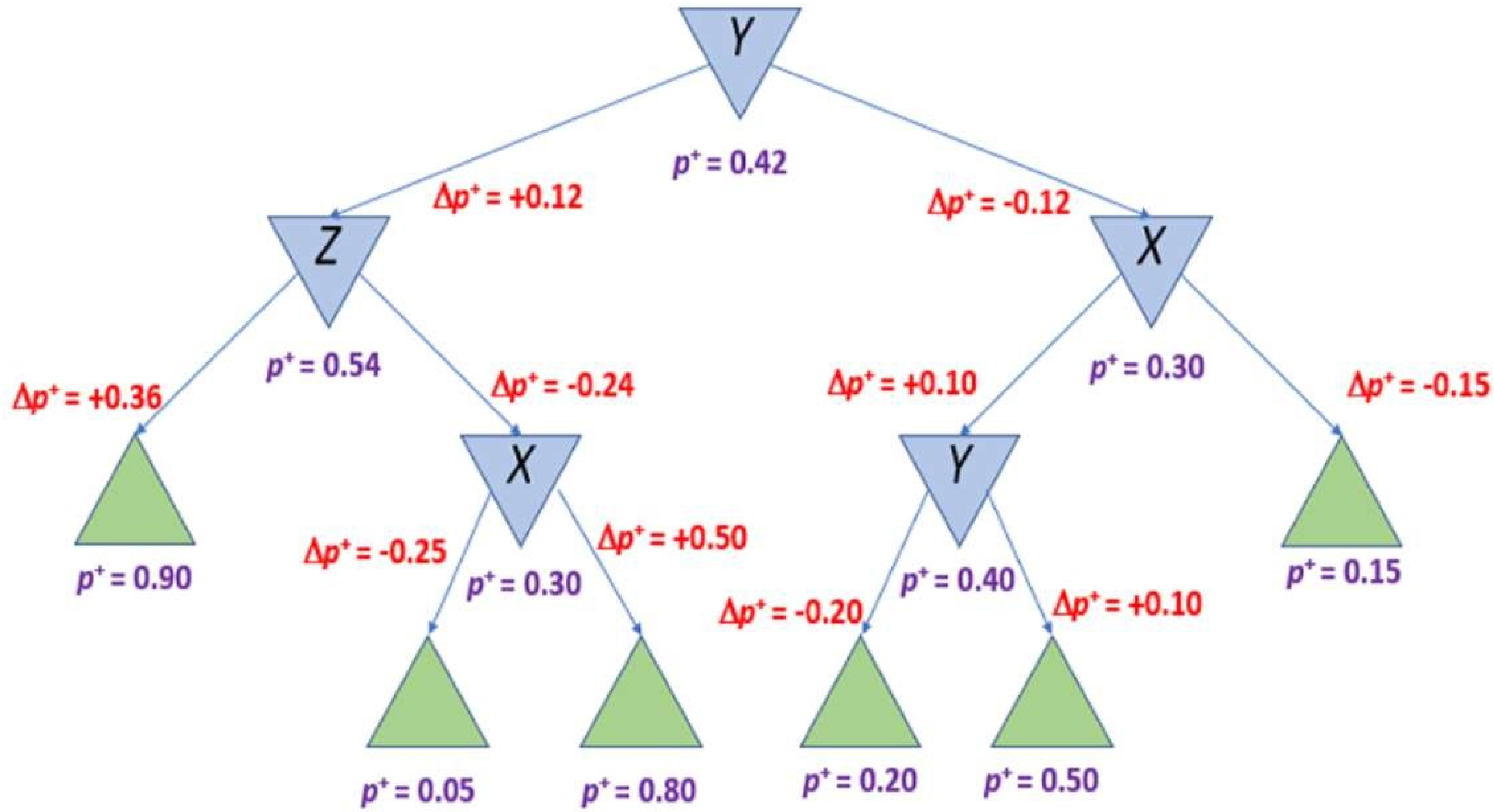
For each internal node compute $(n_k^+,\, (n_k^+ + n_k^-))$

Explanation generation: all node probs



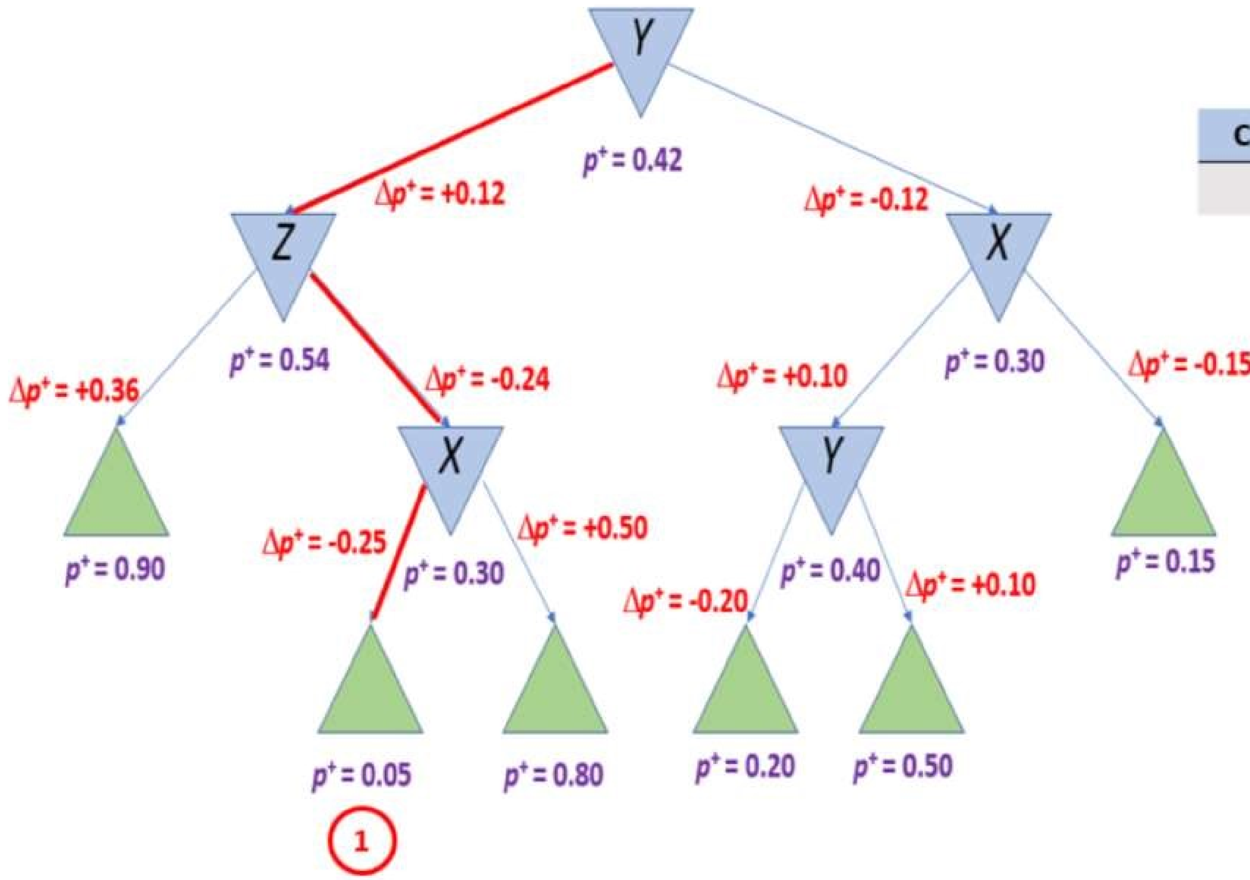
For each internal node node $k, p_k^+ = n_k^+ / (n_k^+ + n_k^-)$

Explanation generation: deltas on all edges



If k is the parent of l, compute $\Delta p^{+}_{(k,l)} := p^{+}_{l} - p^{+}_{k}$

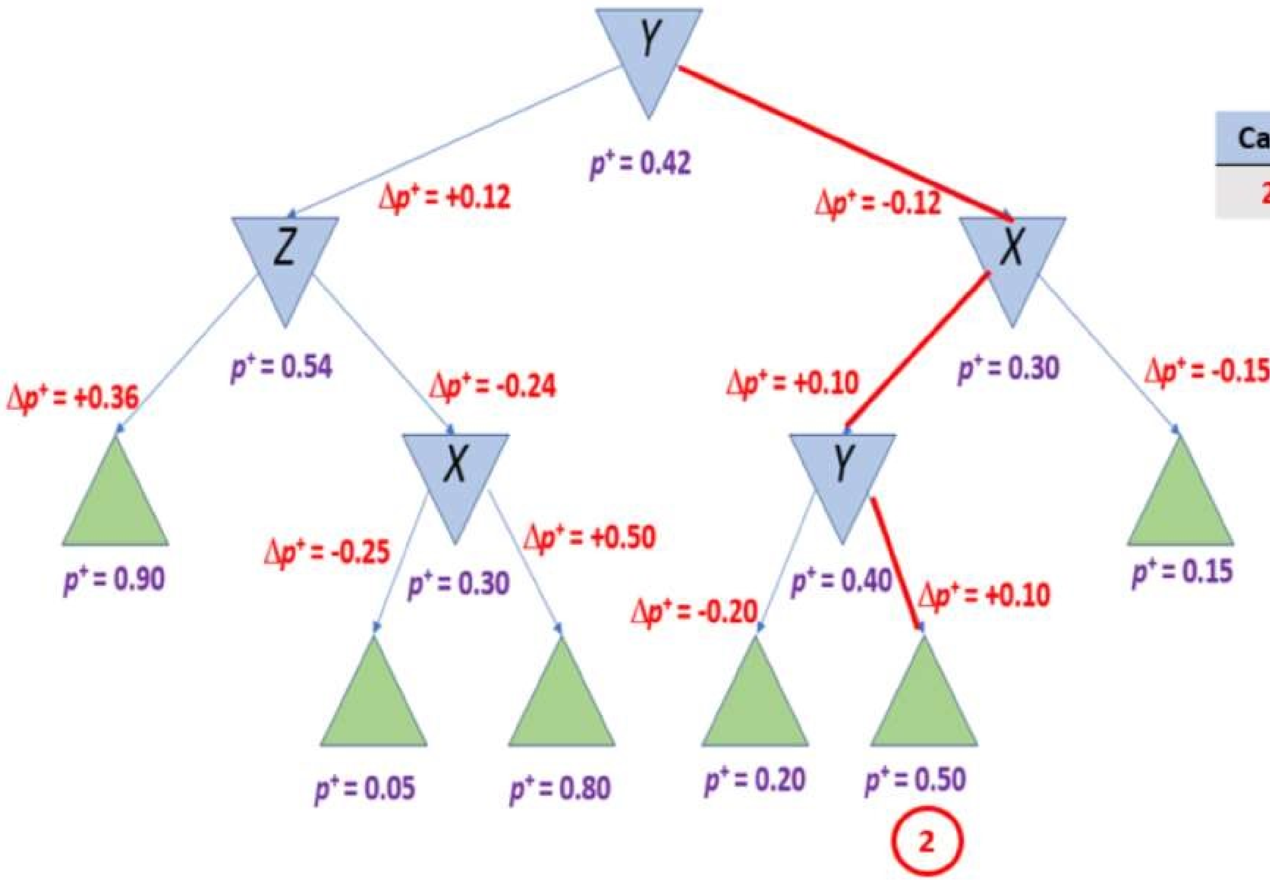
Explanation generation: assigning deltas to variables - Case 1



Case	X	Y	Z
1	-0.25	+0.12	-0.24

First delta is attributable to Y, second to Z, third to X

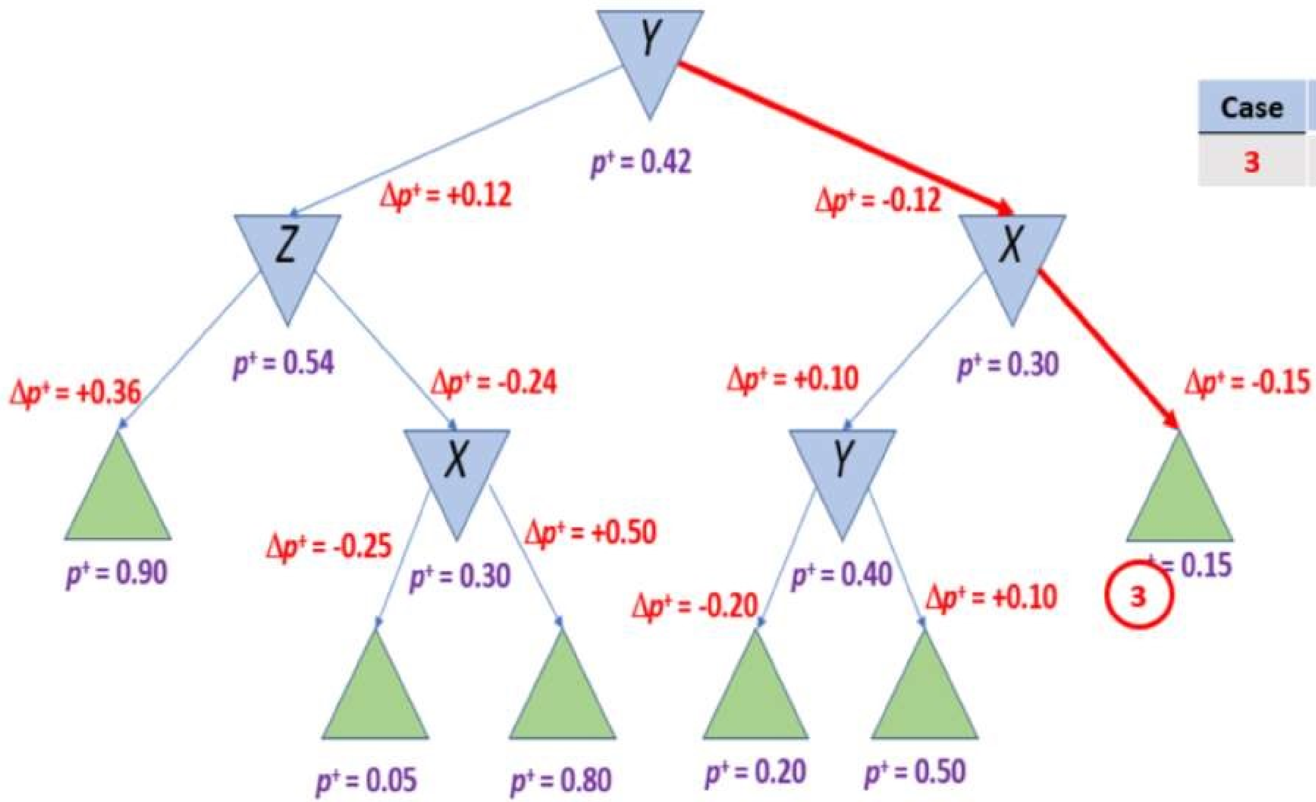
Explanation generation: assigning deltas to variables - Case 2



Case	p0	X	Y	Z
2	0.42	+0.10	-0.02	0.00

First delta is attributable to Y, second to X, third to Y again.

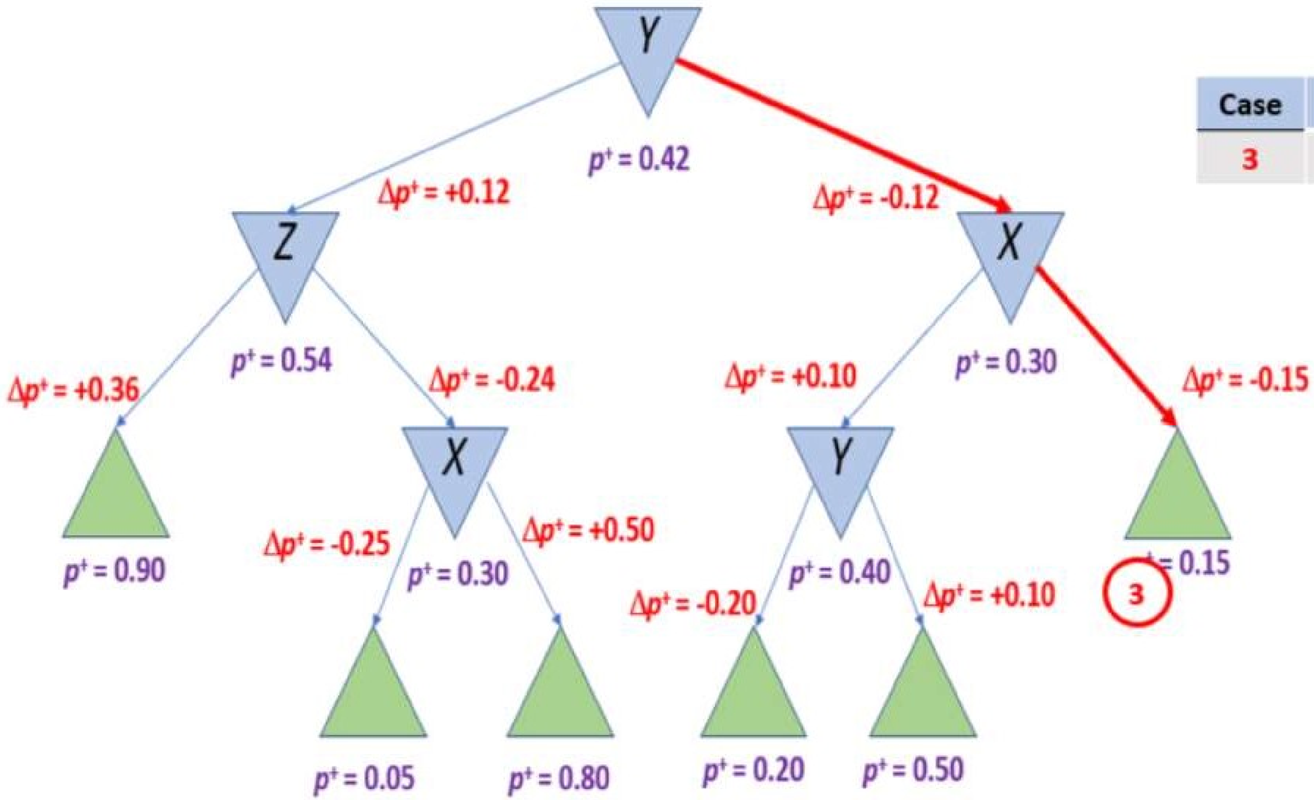
Explanation generation: assigning deltas to variables - Case 3



Case	p0	X	Y	Z
3	0.42	-0.15	-0.12	0.00

First delta is attributable to Y, second to X, none to Z

Explanation generation: assigning deltas to variables - Case 3



Case	p0	X	Y	Z
3	0.42	-0.15	-0.12	0.00

First delta is attributable to Y, second to X, none to Z

Another approach for explanation generation: LIME

LIME: **L**ocal **I**nterpreatable **M**odel-agnostic **E**xplanations

Basic Idea: For each $\mathbf{x}(i)$:

- Generate (100s of) random samples of a neighborhood around $\mathbf{x}(i)$.
- Compute prediction using model M for each sample.
- Fit linear ML model to predictions.
- Cast coefficients of linear model as variable importances.

Another approach for explanation generation: LIME

LIME: **L**ocal **I**nterpreatable **M**odel-agnostic **E**xplanations

Basic Idea: For each $\mathbf{x}(i)$:

- Generate (100s of) random samples of a neighborhood around $\mathbf{x}(i)$.
- Compute prediction using model M for each sample.
- Fit linear ML model to predictions.
- Cast coefficients of linear model as variable importances.

Main drawback:

- It is **very slow** to compute an explanation for a single datapoint!
    - $\rightarrow$ This doesn't scale!

Comparison between LIME and TCXP

Feature	LIME	TCXP
Model Agnostic	☺	☹
Easy to visualize	☺	☺
Scalable	☹	☺
Handles local non-linearities	☹	☺
Python Implementation	☺	☺
Spark Implementation	☹	☺

Extras

- Visit Yuxi Global's site: www.yuxiglobal.com
- Add me on LinkedIn: https://www.linkedin.com/in/mateorestrepo/
- Get the codez: https://github.com/YuxiGlobal/data-analytics
- More about LIME:
  - https://arxiv.org/abs/1602.04938
  - https://github.com/marcotcr/lime
- O'Reilly: An in introduction to machine-learning interpreatability

Inspired solutions

YUXI GLOBAL

¡Gracias!