All Relevant Feature Selection

Racapping Our Path So Far

We started with a somewhat clear goal

- Given data containing candidate correlates and a discrete target
- ...We aimed at identifying the most relevant correlates

We applied a baseline approach (Lasso) to:

- Obtain a surrogate for our data-generation process
- Analyze the impact of each candidate correlate (feature)
- Identify the most relevant correlates

Our baseline turned out to be largely insufficient, so we:

- Trained a non-linear model to obtain a more reliable surrogate
- Learn to assess importance via a permutation-based method
- Learned to explain individual examples via SHAP

*We still have a couple of major open problems...

Open Problems

There's a mistmatch between local and global explanations

- We are using SHAP to assess local feature effects
- ...And permutation importance for global feature effects

As a side effect, there may be inconsistences in our analysis

We still don't know how to identify the most relevant features

- Like in the Lasso appproach we could think of using a threshold
- ...But we still don't know how such threshold should be calibrated

It's time that we fix both of them

Global Feature Analysis via SHAP

SHAP explanations can be aggregated to get global importance scores

By default, this is done by averaring absolute SHAP values:

$$\bar{\phi}_j(x) = \frac{1}{n} \sum_{i=1}^{m} |\phi_j(x_i)|$$

- Other aggregation functions can also be used (e.g. max)

By using aggregated SHAP scores

...We ensure that our local and global analysis have a similar semantic

- Permutation Importance are a perfectly viable approach
- ...And sometimes may be more appropriate than SHAP

However, when doing a rigorous analysis consistency is important

Global Feature Analysis via SHAP

The SHAP library provide convenience functions to plot aggregated values

Here's how to plot mean (absolute) SHAP values:

```
In [ ]: shap.plots.bar(shap_values, max_display=7)
```

Global Feature Analysis via SHAP

The SHAP library provide convenience functions to aggregated values

Here's how to display the maximum (absolute) SHAP values:

```
In [ ]: shap.plots.bar(shap_values.max(0), max_display=7)
```

Semantics for Feature Selection

A viable approach for feature selection consists in solving:

$$\operatorname*{argmin}_{S \subseteq \mathcal{X}} \left\{ |S| : \hat{y} = \hat{f}_S(x_S), L\left(y, \hat{y}\right) \leq \theta \right\}$$

Where $\boldsymbol{x}, \boldsymbol{y}$ denote all the training data. Intuitively:

- We search for the smallest subset of features $\boldsymbol{S}$
- ...Such that a model $\hat{f}_S$ trained over only over them
- ...Still has an acceptable (cross-validation) accuracy

Heuristics (e.g. greedy search) can be used to improve scalability

Semantics for Feature Selection

A viable approach for feature selection consists in solving:

$$\mathop{\mathrm{argmin}}_{S \subseteq \mathcal{X}} \left\{ |S| : \hat{y} = \hat{f}_S(x_S), L\left(y, \hat{y}\right) \leq \theta \right\}$$

Where $\boldsymbol{x}, \boldsymbol{y}$ denote all the training data. Intuitively:

- We search for the smallest subset of features S
- ...Such that a model $\hat{f}_S$ trained over only over them
- ...Still has an acceptable (cross-validation) accuracy

Heuristics (e.g. greedy search) can be used to improve scalability

This optimization-driven approach

- ...Can be customized by adjusting the constraint and cost function
- ...Can reduce data storage and location costs on the deployed model

Semantics for Feature Selection

If we care just about cost and accuracy, the optimization approach is perfect

But it is not suitable for our current case study...
Can you tell why?

Semantics for Feature Selection

If we care just about cost and accuracy, the optimization approach is perfect

But it is not suitable for our current case study...
Can you tell why?

For a number of reasons:

- We care about finding all the relevant features, not a minimal set
- How should the accuracy threshold be calibrated?
- What about the noise induced by retraining?

Semantics for Feature Selection

If we care just about cost and accuracy, the optimization approach is perfect

But it is not suitable for our current case study...
Can you tell why?

For a number of reasons:

- We care about finding all the relevant features, not a minimal set
- How should the accuracy threshold be calibrated?
- What about the noise induced by retraining?

If we wish to use ML for data analysis, we need another approach

...In particular, we will rely on statistical hypothesis testing (HT)

Statistical Hypothesis Testing

HT is one of the main mathematical workhorses of scientific research

HT builds evidence for a hypothesis by refuting a competing one:

- We start from a hypothesis H and some data x
- We formulate a competing null hypothesis H_0
- We define a test statistic $T(X)$, monotonically related to H
- We define the theoretical probability of $T(X)$ under H_0, i.e. $P(T(X) \mid H_0)$
- We compute the its empirical value for our data $t = P(T(x) \mid H_0)$
- We compute the probability that $T(X)$ is as extreme as t under H_0, i.e.:

$$p = P(T(X) \geq t \mid H_0)$$

- If $p \leq 1 - \alpha$ for some confidence α, we reject the null hypothesis

This is probably very confusing...

Let's make an example for our case

Hypothesis, Data, and Null-Hypothesis

First, we need to define our hypothesis and data

We care about identifying correlates, so a possible choice might be:

- $H \equiv "r(X, Y) \geq r^*"$, for some correlation measure r
- $\text{data} \equiv \text{"x, y"}$, i.e. our sample

Hypothesis, Data, and Null-Hypothesis

First, we need to define our hypothesis and data

We care about identifying correlates, so a possible choice might be:

- $H \equiv "r(X, Y) \geq r^*"$, for some correlation measure r
- $\text{data} \equiv \text{"x, y"}$, i.e. our sample

Now we need a competing null hypothesis

A good choice might be $H_0 \equiv \text{"the observed result is due to chance"}$

- In most cases, the null hypothesis assumes what we observe is due to chance
- If we manage to reject it, we can claim that H is more likely true
- The tricky part is choosing a H_0 for which we can compute probabilities
- ...Without introducing unnecessary assumptions

Now we need some "test" related to H and H_0

..And it must be something for which we can compute probabilities

How do we do that?

Test Statistic and Theoretical Probability

Let's consider the event "$r(X, Y) < r^*$"

Since it has a binary outcome, it will follow a Bernoulli distribution

- If we assume that the correlation is due to chance...
- ...Then the associated probability should be $^1/_2$

Let's pretend we make repeated experiments

The number of observe events $T(X, Y)$ will follow a binomial distribution

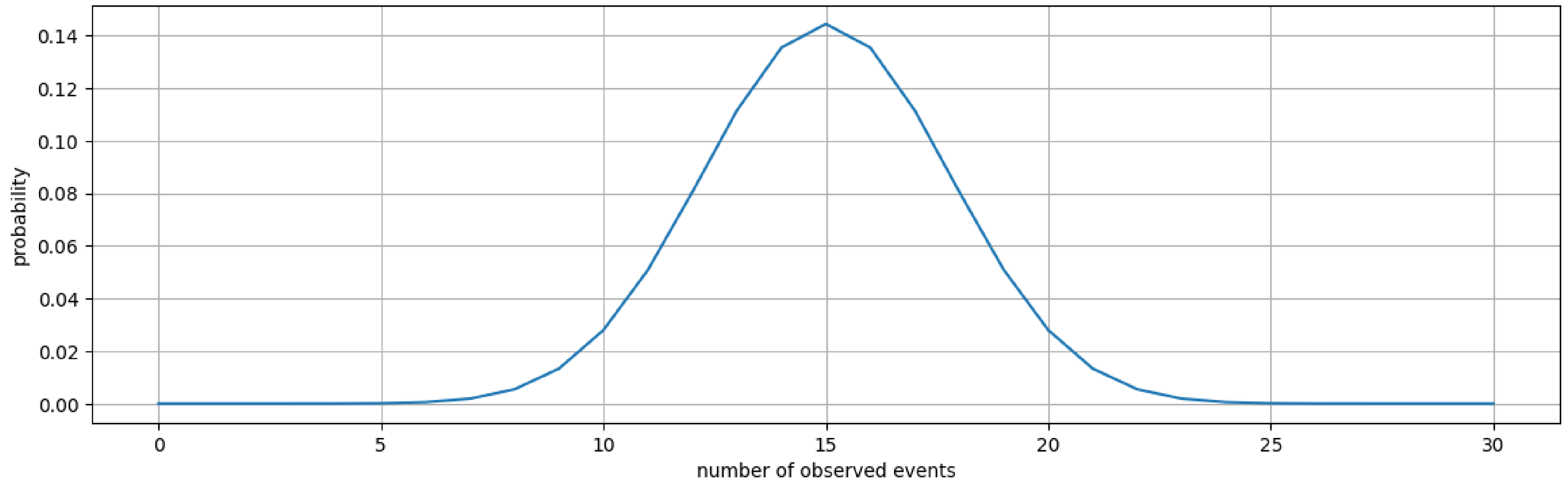
- Given the number of experiments n
- ...The probability of $T(X, Y)$ should be $P(T \mid H_0) = B(n, {}^1/_2)$

T **is our test statistics,** $B(n, {}^1/_2)$ **its theoretical distribution under** H_0

Theoretical Probability Computation

We can easily compute and plot the distribution

```
In [4]: util.binomial_plot(n=30, p=0.5, figsize=figsize)
```



- This tells us how likely we are to observe a certain number of events
- ...Assuming that the null hypothesis is true

What about the empirical probability?

We need to simulate lack of correlation

...Without additional assumptions

Empirical Probability Computation

We can use a Monte-Carlo approach

The trick is once again relying on permutations

- If we shuffle the values of one variable (say the values x of X)
- ...We can get a correlation with Y only by chance
- ...But we otherwise preserve the distribution of the sample

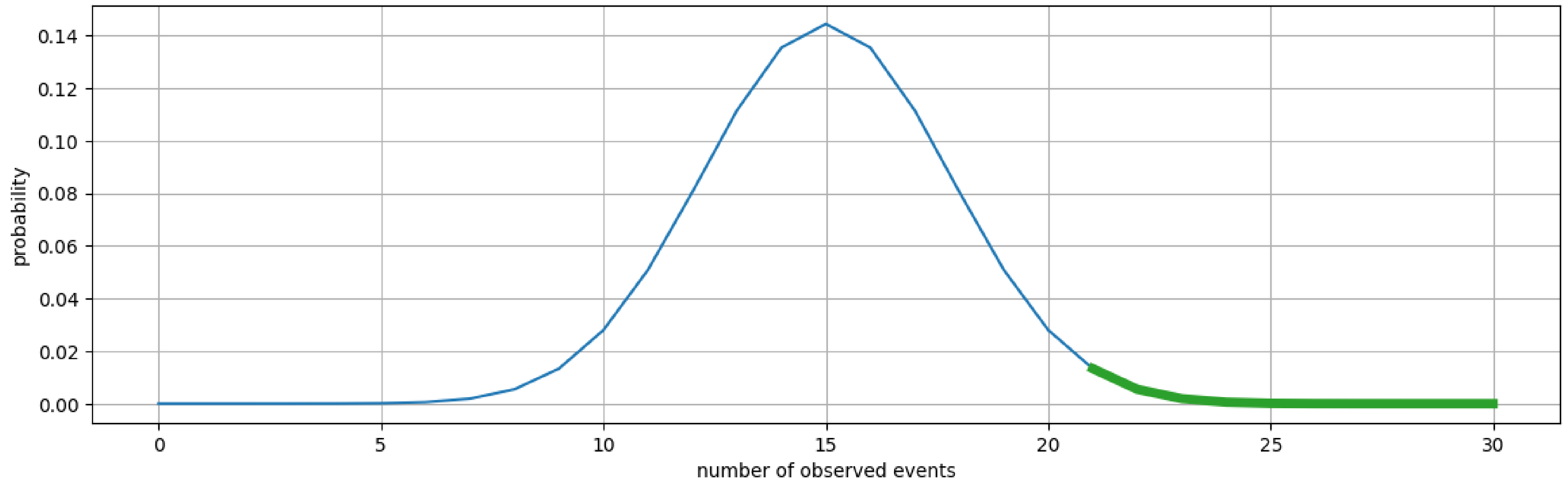
We can mitigate sampling noise via repeated experiments

- Then, we take our empirically observed number of events $t = T(x, y)$
- ...And we match it against the theoretical probability
- We care about the probability that $T(X, Y) \geq t$
- ...Since any value larger than t would still support the null-hypothesis

p-Value and the Statistical Test

Basically, there is a "target interval" in the distribution

```
In [5]: util.binomial_plot(n=30, p=0.5, r_alpha=0.01, figsize=figsize)
```

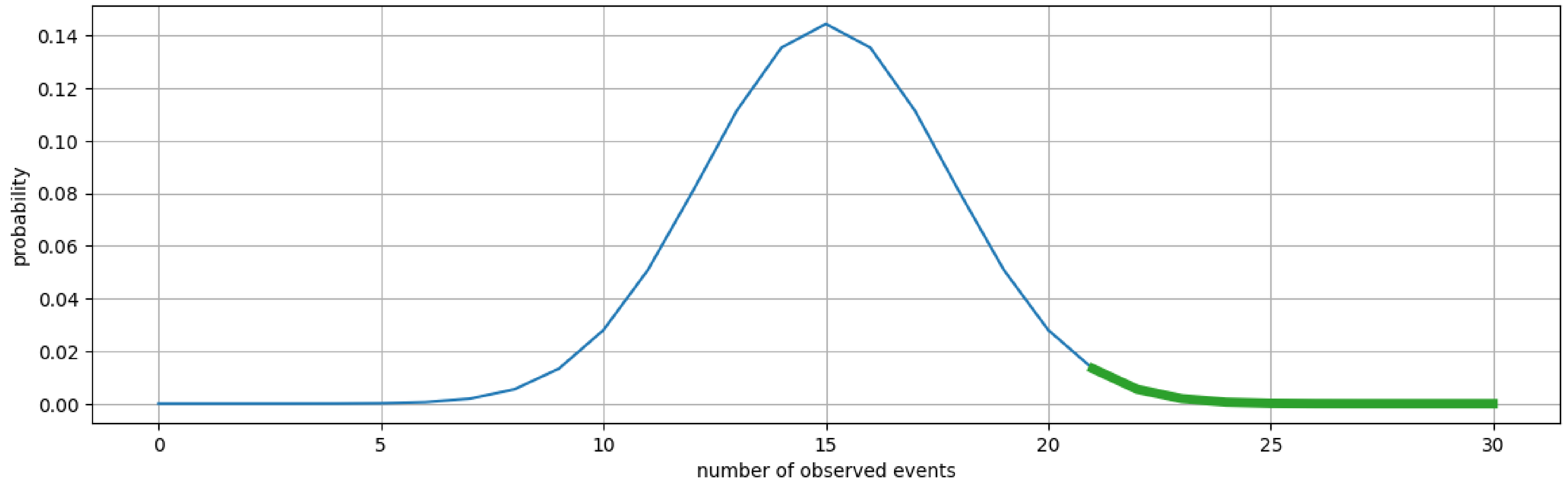


- For any value in the interval, we have $P(T(X,Y) \geq t \mid H_0) \leq 1 - \alpha$
- ...Where α is our desired confidence level

p-Value and the Statistical Test

Basically, there is a "target interval" in the distribution

```
In [6]: util.binomial_plot(n=30, p=0.5, r_alpha=0.01, figsize=figsize)
```

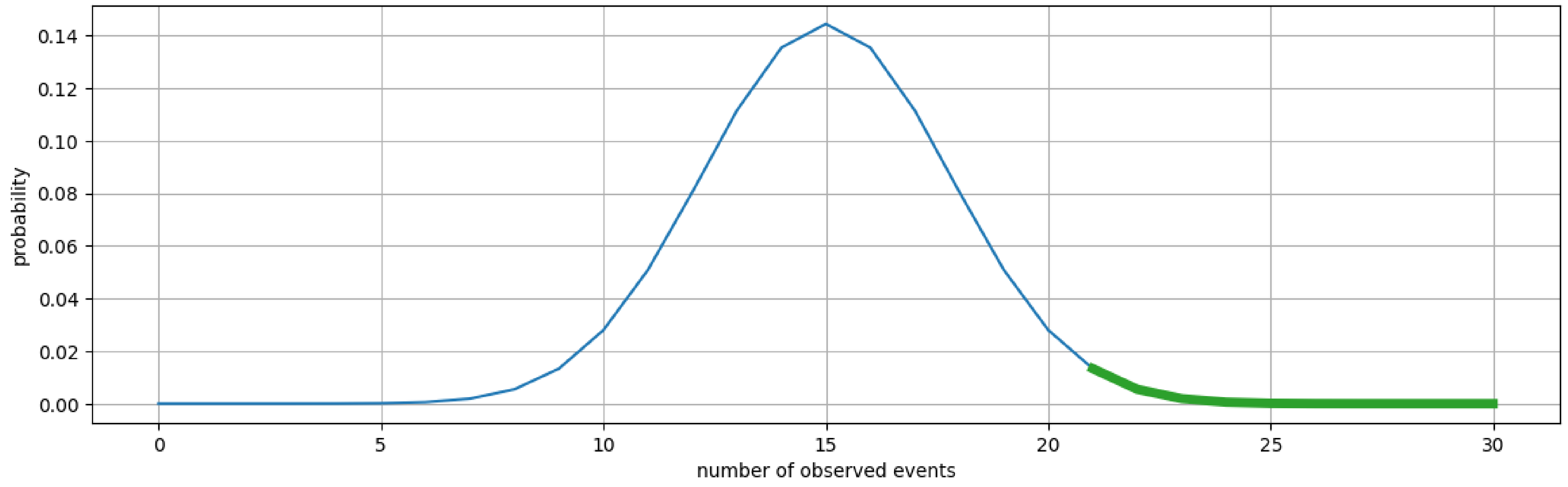


- We still need a threshold (i.e. α) to define the interval
- ...But it's a probability, so it's easier to define (usually $\alpha = 0.01$ or $\alpha = 0.05$)

p-Value and the Statistical Test

Basically, there is a "target interval" in the distribution

```
In [7]: util.binomial_plot(n=30, p=0.5, r_alpha=0.01, figsize=figsize)
```



- In practice it's more common to compute the p-value $P(T(X,Y) \geq t \mid H_0)$
- ...Which can then be immediately compared with $1 - \alpha$

Back to the Procedure Description

The procedure should be clearer now

Let's recap the steps:

- We start from a hypothesis H and some data x
- We formulate a competing null hypothesis H_0
- We define a test statistic $T(X)$, monotonically related to H
- We define the theoretical probability of $T(X)$ under H_0, i.e. $P(T(X) \mid H_0)$
- We compute the its empirical value for our data $t = P(T(x) \mid H_0)$
- We compute the probability that $T(X)$ is as extreme as t under H_0, i.e.:

$$p = P(T(X) \geq t \mid H_0)$$

- If $p \leq 1 - \alpha$ for some confidence α, we reject the null hypothesis

Testing a Hypothesis and Its Negation

In our case, the method works also for testing lack of correlation

- Our hypothesis becomes $\neg H \equiv$ "$r(X, Y) < r^*$"
- The null hypothesis is the same as before
- The test statistics is just $n - T(X, Y)$, for the same T as before

Then we can proceed as in the previous case

Since we are relying on the same test statistics

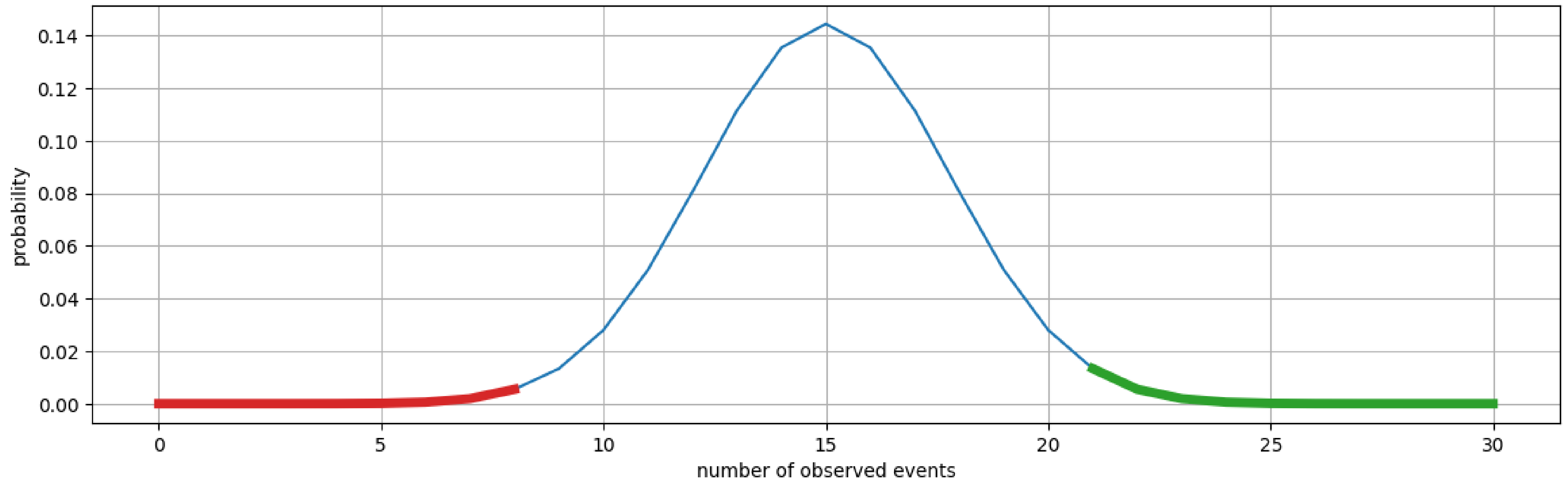
...We can use the same set of experiments to test both hypotheses

- Intuitively, in both cases we look at the number of events $T(xy)$
- If $T(x, y)$ is sufficiently high, it's likely that the H holds
- ...If $T(x, y)$ is suficiently low, it's likely that $\neg H$ holds

Testing a Hypothesis and Its Negation

In other words, we will end up having two target intervals

```
In [8]: util.binomial_plot(n=30, p=0.5, l_alpha=0.01, r_alpha=0.01, l_color='tab:red', r_color='tab:
```



- If $T(x, y)$ lands in the green region, we support H (e.g. correlation)
- If $T(x, y)$ lands in the red region, we support $\neg H$ (e.g. lack of correlation)
- If $T(x, y)$ lands in the center region, we support no claim

Boruta

The approach we have just seen is the backbone of the Boruta algorithm

- The Boruta algorithm is a SotA feature selection method
- ...That relies in statistical HT to determine relevant features

Like in our analysis, the method relies on surrogate models

...And in particular on tree ensembles (the name refers to a Slavic forest spirit)

- As a consequence, the algorithm can deal with non-linear correlations
- ...And accounts for interactions between multiple features

Boruta is an all-relevant feature selector

- This makes it particularly well suited for scientific analyses
- ...But it can be used to reduce data collection costs or improve generalization

Tested Hypothesis in Boruta

Boruta relies on a measure of feature importance

- The original algorithm and the BorutaPy package use permutation importance
- The more recent BorutaShap package relies on average SHAP values

In both cases, importance is computed w.r.t. a reference dataset

The hypothesis H being tested is more general than ours and consists in:

"Feature j is important among those in the dataset, according to the chosen metric"

- This is much more general than the one we considered
- ...And it require as slighly more sophisticated test statistics

Test Statistics in Boruta

The main idea is stil to rely on permuted features

Let (x, y) be our original dataset

- First, we augmented it by introducing permuted versions $\tilde{\mathcal{X}}$ of all features
- These are called shadow features by the algorithm. Let their values be $\tilde{x}$

Then, we train a predictive model on $(x, \tilde{x}, y)$

Let $\phi_j(x, , \tilde{x}, y)$ be the importance of feature j, on the augmented dataset

- If the feature is important, its ϕ_j should beat the shadow features
- Therefore, we can consider the event

$$\phi_j((x, \tilde{x}), y) > \max_{j \in \tilde{\mathcal{X}}} \phi_j((x, \tilde{x}), y)$$

Test Statistics in Boruta

Then the testing statistics T is similar to the one we used:

- The algorithms performs multiple experiments (retraining the model)
- ...And counts the number of times the event is satisfied ("hits")

The thereotical distribution for T under H_0 is mostly a binomial

- The algorithm needs to apply some statistical corrections
- ...Since we are testing multiple features together (we have a $\max$)

Boruta tests both the positive and negative hypothesis

Therefore, at the end of the process:

- Some features will be confirmed important
- Some features will be confirmed unimportant
- Some features will be remain tentative

Using Boruta in Practice

We'll use Boruta through the BorutaShap package

```
In [59]: bfs = BorutaShap(importance_measure='shap', classification=True)
         bfs.fit(X=X, y=y, n_trials=100, sample=False, train_or_test='test', normalize=True, verbose=
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

```
4 attributes confirmed important: ['u1', 'u12', 'u10', 'u13']
11 attributes confirmed unimportant: ['u9', 'u6', 'u3', 'u0', 'u2', 'u14', 'u11', 'u7', 'u
4', 'u5', 'u8']
0 tentative attributes remains: []
```

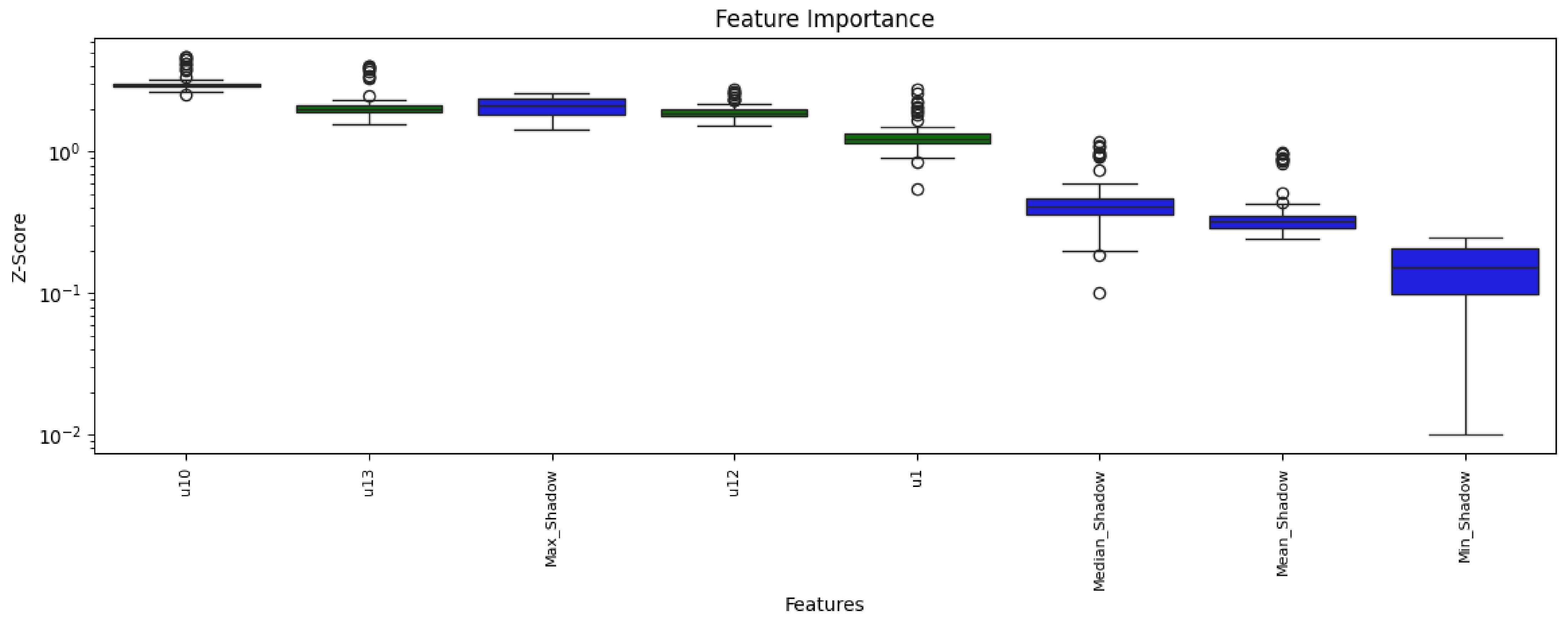
- We can choose to use either the testing or training importance
- The algorithm also determines the best number of estimators
- The algorithm allow the use of a clever sampling procedure
- ...To reduce the number of averaged SHAP values (and therefore the run-time)

Warning: as of Nov 2023, the PyPI version of BorutaShap is not compatible with the most recent scikit-learn release (this lecture uses the GitHub version)

Using Boruta in Practice

We can plot the the ϕ_j distribution for the confirmed important features

```
In [58]: bfs.plot(which_features='accepted', figsize=figsize)
```

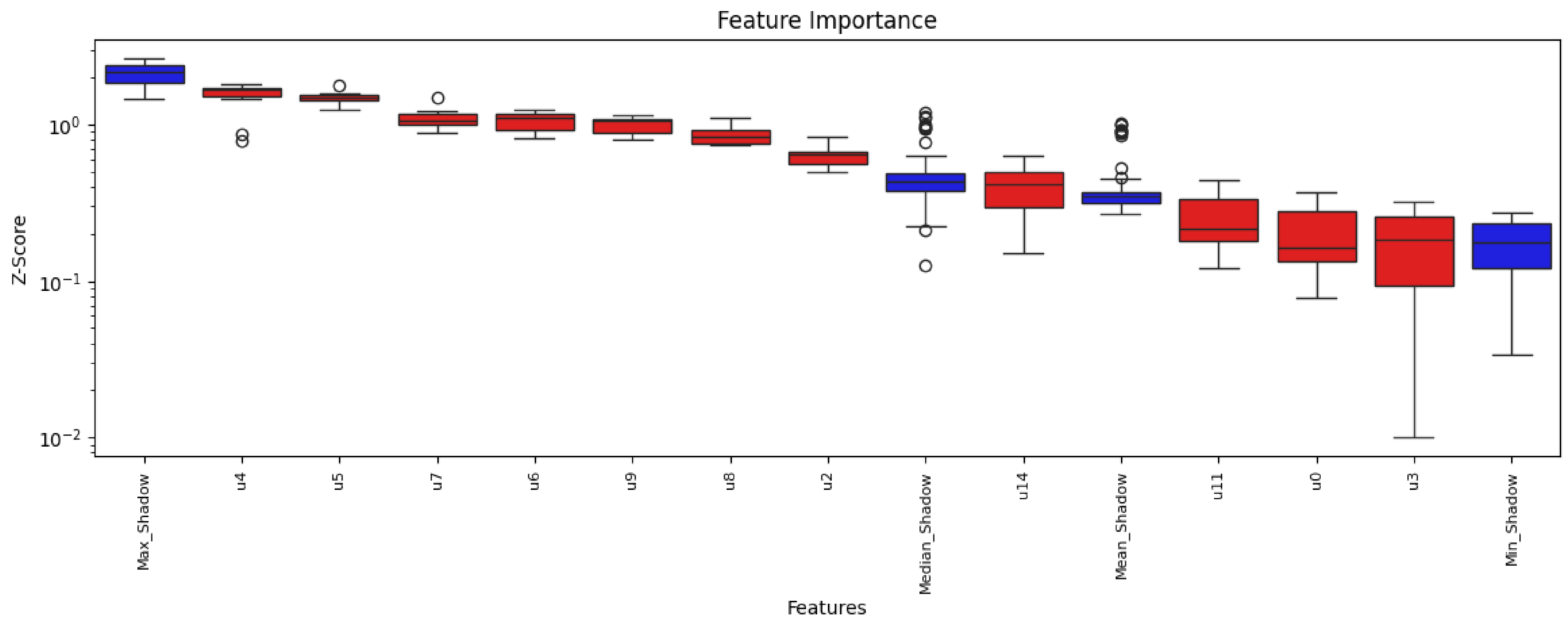


- Distribution data for the shadow features is shown for comparison

Using Boruta in Practice

We can to the same for the confirmed unimportant features

```
In [60]: bfs.plot(which_features='rejected', figsize=figsize)
```



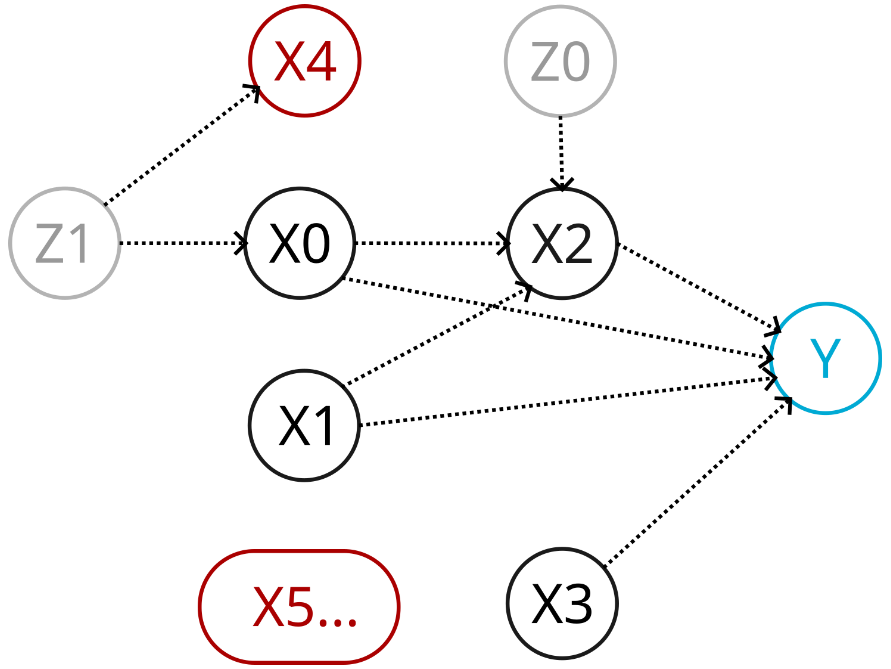
- Rejected features are shown in red

Ok, but... Did it work?

In our controlled setting, we can inspect the ground truth process

Checking the Ground Truth

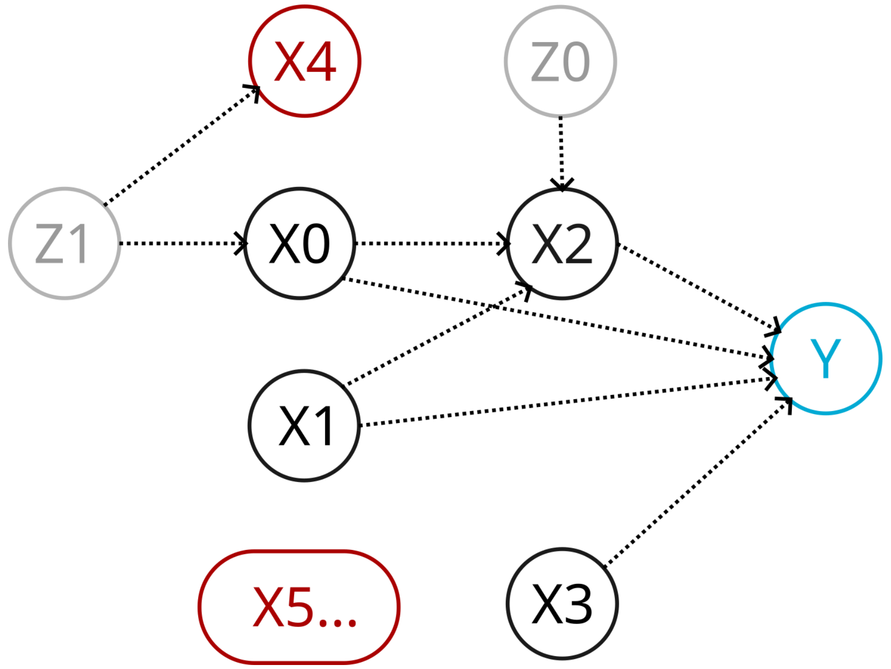
The ground-truth process is described by this causal graph:



- The Y variable (in **blue**) is the target
- The variables in **black** are those that are relevant
- The variables in **gray** are not observable, i.e. latent
- The variables in **red** are irrelevant

Checking the Ground Truth

The process was engineered to contain several classical cases

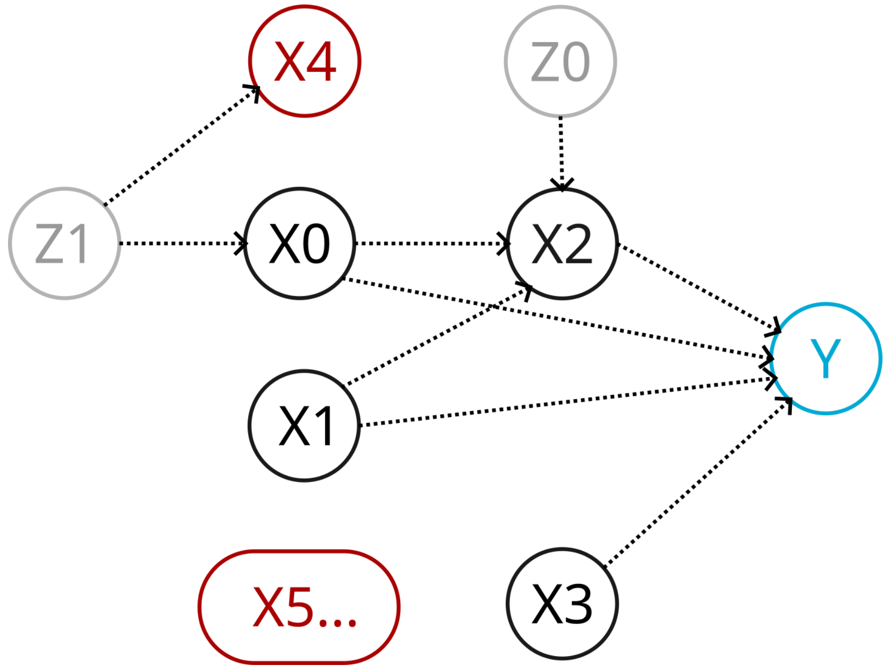


X_2 is a mediator beween X_0, X_1 and Y

- The variable partially hides the effect of X_0 and X_1
- If it does that completely, even Boruta cannot mark X_0 and X_1 as important
- Depending on the use case, this might be an issue

Checking the Ground Truth

The process was engineered to contain several classical cases

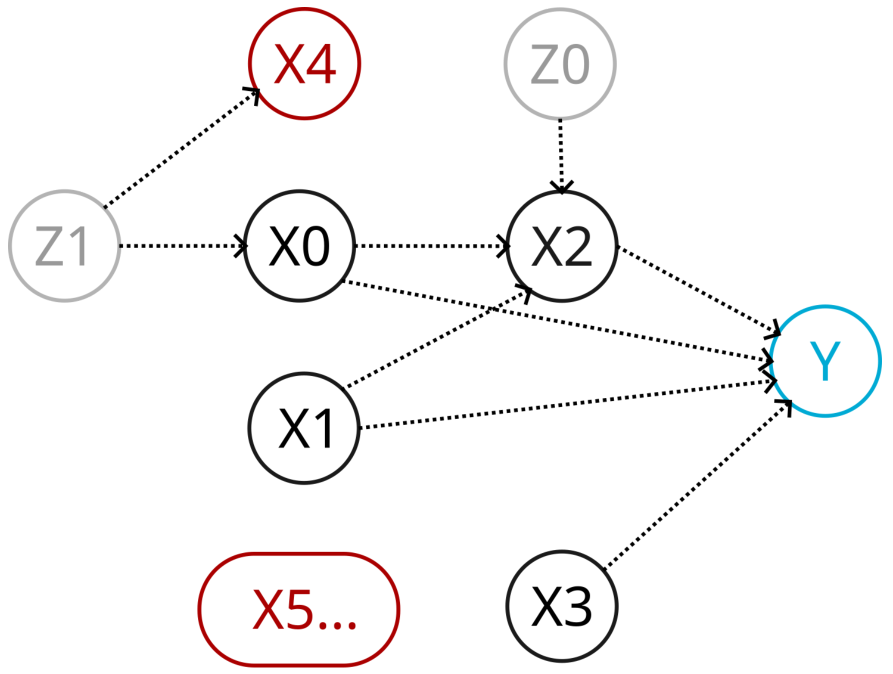


X_2 is also a complete mediator for Z_0

- ...But in this case it is a good thing!
- Z_0 is not observed, but we can account for that at least indirectly

Checking the Ground Truth

The process was engineered to contain several classical cases

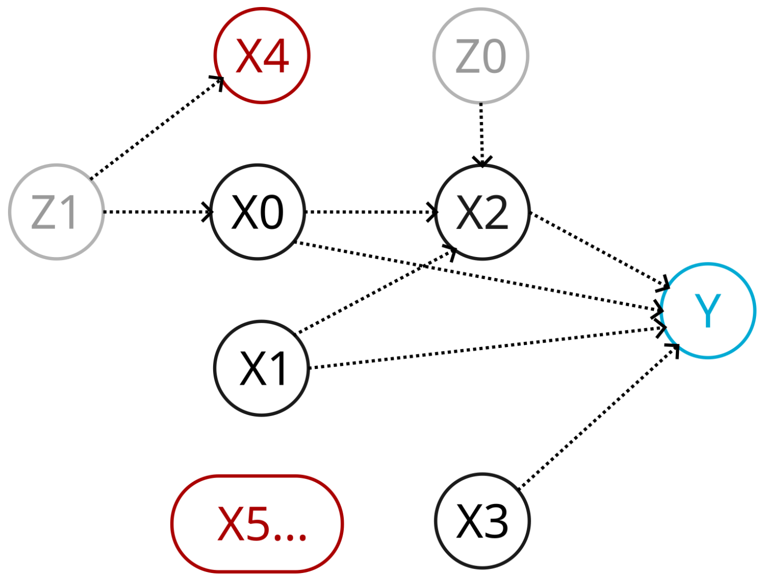


Z_1 is a confounder and causes a correlation between X_1 and X_0

- It is totally mediated by X_1, which is a good thing
- ...But it also causes a correlation between X_0 and X_4
- This might trick a model into considering X_4 as important

Checking the Ground Truth

Now let's check how accurate our importance estimate is:



```
In [65]: print(f'The accepted feature are {bfs.accepted}')
         print(f'...Which correspond to {[name_map[f] for f in bfs.accepted]}')

         The accepted feature are ['u1', 'u12', 'u10', 'u13']
         ...Which correspond to ['X1', 'X2', 'X3', 'X0']
```

If everything went as a planned, we should have found all relevant variables!

A Few Final Remarks

ML models are not just for prediction!

- They can be used for generation, anomaly detection, decision support
- ...And also as tools for a scientific analysis!

Explainability is an important topic in AI

- It is one of the main approaches to make an AI model transparent
- This critical when AI systems need to interact with human users
- ...And for some domains it is also required by existing regulations

Beware of correlated features

- Strongly correlated features (e.g mediated-mediator) may mislead algorithms
- Dealing with those is still a partially open problem!