In [1]:

```
%run Latex_macros.ipynb
%run beautify_plots.py
```

In [2]:

```
# My standard magic !  You will see this in almost all my notebooks.

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# Reload all modules imported with %aimport
%load_ext autoreload
%autoreload 1

%matplotlib inline
```

Feature Importance

Given the n features in $\mathbf{x}$, which are the "most important" ?

The multiple trees in a Random Forest offer several ways to answer this question.

Importance: Decrease in Impurity

Recall that the question that splits the examples corresponding to a node is chosen so as to maximize Information Gain.

One method of measuring the importance of $\mathbf{x}_j$ is the amount of impurity decrease it creates.

- For each feature x_j
  - find each node n in *any* tree in the forest with question (j, v) for *any* v
    - compute the information gain of the split on (j, v)
  - average the information gain across all such nodes

That is, how much does impurity decrease when $\mathbf{x}_j$ is used in a question.

- This is a biased method
    - Recall the universe of possible values of $\mathbf{x}_j$ is V_j
    - Larger $|V_j|$ means $\mathbf{x}_j$ is more likely to appear in a questions
        - e.g., when $\mathbf{x}_j$ is a continuous variable that has been made discrete
    - So $\mathbf{x}_j$ will appear in more questions

Importance: Permutation importance

Let's consider building one tree from bootstrapped sample S.

Create another sample S', derived from S by *permuting* the values of $\mathbf{x}_j$.

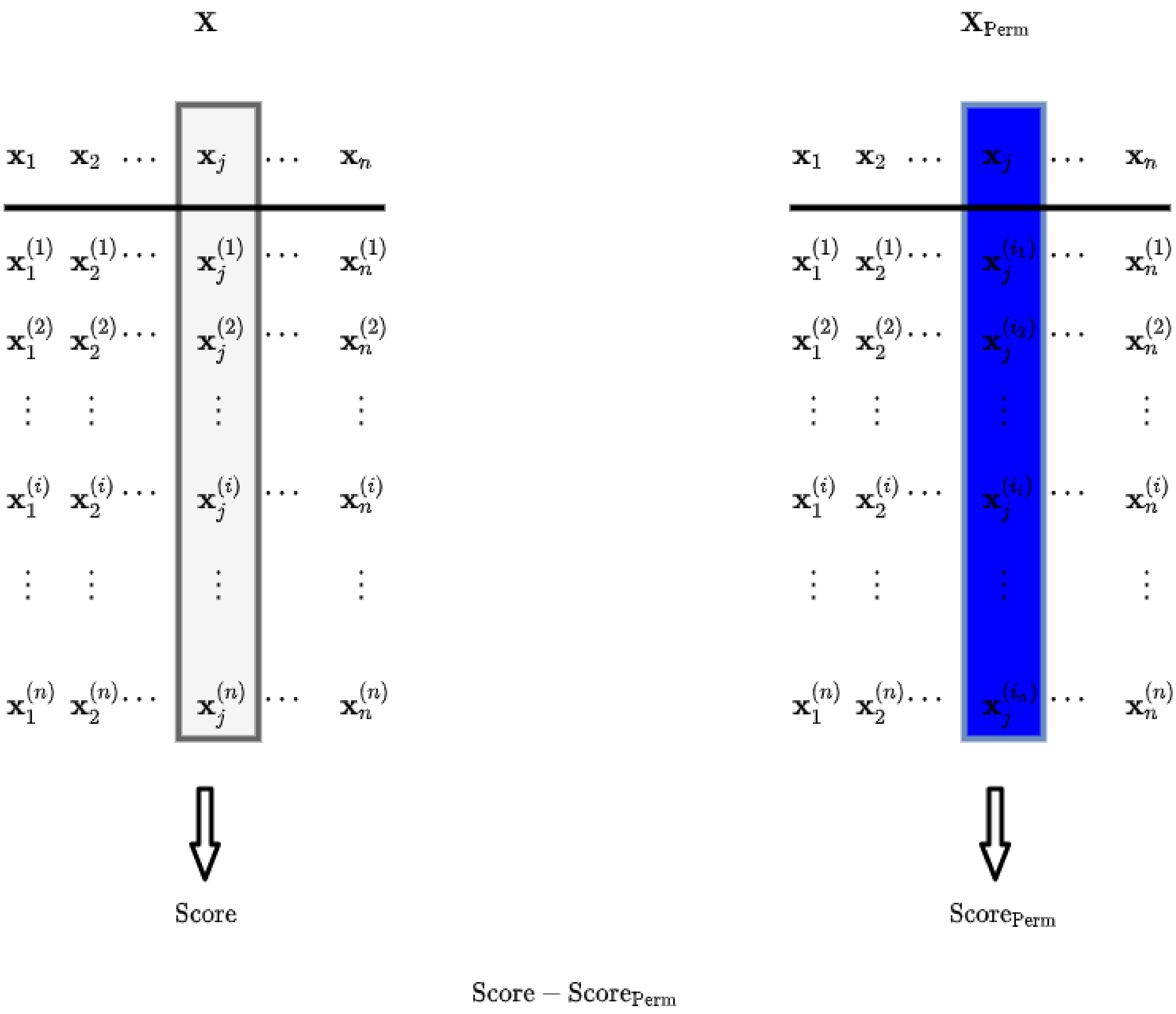
- maintains the unconditional distribution of $\mathbf{x}_j$
- breaks the correlation of $\mathbf{x}_j$ with the target and other features

We can now measure the importance of $\mathbf{x}_j$ as

- the change in out of bag accuracy of the tree built from S and S'.

That is, if $\mathbf{x}_j$ is unimportant, then permuting its values should have little effect on accuracy.

Permutation Importance, feature j



$\mathbf{X}$
$\mathbf{X}_{\mathrm{Perm}}$
$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_j \quad \cdots \quad \mathbf{x}_n$
$\mathbf{x}_j^{(i_1)} \quad \mathbf{x}_j^{(i_2)} \quad \mathbf{x}_j^{(i_i)} \quad \mathbf{x}_j^{(i_n)}$
Score
$\mathrm{Score}_{\mathrm{Perm}}$
$\mathrm{Score} - \mathrm{Score}_{\mathrm{Perm}}$

Permutation importance also has issues

- may be biased if $\mathbf{x}_j$ is strongly correlated with another feature $\mathbf{x}_{j'}$

In that case $\mathbf{x}_{j'}$ may compensate for the permuted $\mathbf{x}_j$, making $\mathbf{x}_j$ seem unimportant.

In [4]:
```
print("Done")
```

```
Done
```