The Problem
Given a set of objects $O$, and a utility function $U : 2^O \rightarrow \mathbb{R}$ defined for all $X \subseteq O$, find an optimal subset of $O$ - $\text{Opt}$, such that:

$$\text{Opt} = \arg\max_{X \subseteq O} U(X)$$

The Idea
- Iterative solution.
- Start with an empty subset.
- In each iteration add the most contributing element.
- End once adding an element reduces the selected subset’s utility.

The maximal utility value of a subset which contains the given element and all previously selected elements, formally:

$$\text{Cont}(o, \text{selected}) \max_{s \in \text{unselected}\{o\}} U(\text{selected} \cup \{o\} \cup s)$$

Or using a more practical approach:

$$\max_{s \in \text{KSubsets}} U(\text{selected} \cup \{o\} \cup s)$$

Where $\text{KSubsets}$ is a sample of $K$ subsets of $\text{unselected}\{o\}$.

The Algorithm

```
SubsetSelect(O, U):
    selected = \emptyset
    unselected = O
    repeat while unselected ≠ \emptyset:
        Cont(0, selected, unselected) = \{\}
        if U(selected U (maxim)) > U(selected):
            selected ← selected U (maxim)
        else:
            return selected
        return selected
    Cont(0, o, selected, unselected):
        subsets = KSubsets(0, unselected, 0, selected U \{o\})
        return max_{s \in \text{subsets}} U(\text{selected} U \{o\})
```

Experiment #1
Algorithm performance using different implementations of KSubsets & values of K:

- NonrandomImprovingKS gives the best performance; value of K is insignificant for algorithm performance.
- XOR5 dataset is hard to handle.

Experiment #2
Using results of multiple runs to generate better results, using the assumption that mostly selected features are “worth” selecting:

- Using the results of multiple runs of the SubsetSelect algorithm, we can generate a small series of subsets, some of which are expected to have a high utility value.
- A useful method to deal with inherently difficult datasets, such as XOR5.

Accuracy Function Calculation
1. Reduce dataset to the relevant features.
2. Create K folds using the reduced dataset. Each fold represents a (train, test) pair.
3. For each fold - learn a classification tree using ID3 from the train set, and compute classification accuracy using the test set.
4. Return average classification accuracy from stage 3.

Implementations