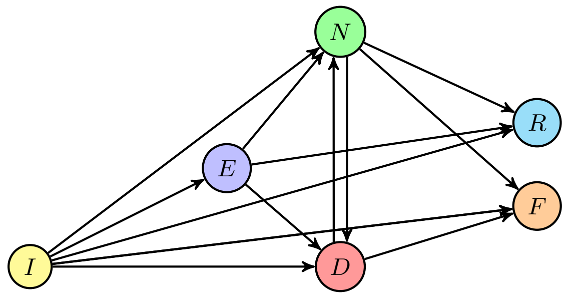

Figure 5: Precedence graph generated from the adjacency matrix in Table 4.

Table 5: Effective pipeline prototypes.

ID	Pipeline prototype
1	$I \rightarrow E \rightarrow N \rightarrow R \rightarrow F$
2	$I \rightarrow E \rightarrow N \rightarrow F \rightarrow R$
3	$I \rightarrow E \rightarrow R \rightarrow D \rightarrow F$
4	$I \rightarrow E \rightarrow D \rightarrow R \rightarrow F$
5	$I \rightarrow E \rightarrow D \rightarrow F \rightarrow R$

For a better reading, in Figure 5, we visualize Table 4 in form of a graph, where nodes represent the kinds of transformations and the directed edges represent a precedence order between them. Out of the graph, we generate the final pipeline prototypes by taking all the maximum length variations (ordered arrangements without repetition) of the nodes, respecting the precedence rules (i.e., not contradicting the direction of existing edges). The result is the set of five pipeline prototypes shown in Table 5. This set consisting of *compatible*, *meaningful* and *promising* pairs of transformations is in fact the final recommendation of our method. To demonstrate the effectiveness of our pipelines, in the next section, we evaluate their impact on the overall results of machine learning tasks.

4 EVALUATION

The aim of our experimental study is three-fold:

(1) Check whether there exists a universal pipeline prototype that works best for every classification problem considered (i.e., dataset and ML algorithm) (Section 4.1).
(2) Assess and compare the performance of the effective pipelines constructed using our method against the set of exhaustively generated pipeline prototypes (Section 4.2).
(3) Assess and compare the impact of dedicating a portion of the optimization time to the effective pipelines constructed using our method, with the impact of using the whole optimization time for the hyper-parameters of the ML algorithm (Section 4.3).

The experiments were performed on an Intel Core i7 machine with 12 cores, running at 3.20 GHz with 64 GB of main memory. As a platform for running the SMBO optimization algorithm we use HyperOpt. Furthermore, the datasets used in the experiments are the ones from the OpenML-CC18 suite. Finally, the classification algorithms considered are *NB*, *KNN*, and *RF*. All the experiments for a single algorithm, on average took approximately two weeks[6].

Table 6: Exhaustive set of pipeline prototypes generated using the compatible precedence graph of Table 1a.

ID	Pipeline prototype	ID	Pipeline prototype
1	$I \rightarrow E \rightarrow N \rightarrow D \rightarrow F \rightarrow R$	13	$I \rightarrow E \rightarrow F \rightarrow N \rightarrow D \rightarrow R$
2	$I \rightarrow E \rightarrow N \rightarrow D \rightarrow R \rightarrow F$	14	$I \rightarrow E \rightarrow F \rightarrow N \rightarrow R \rightarrow D$
3	$I \rightarrow E \rightarrow N \rightarrow F \rightarrow D \rightarrow R$	15	$I \rightarrow E \rightarrow F \rightarrow D \rightarrow N \rightarrow R$
4	$I \rightarrow E \rightarrow N \rightarrow F \rightarrow R \rightarrow D$	16	$I \rightarrow E \rightarrow F \rightarrow D \rightarrow R \rightarrow N$
5	$I \rightarrow E \rightarrow N \rightarrow R \rightarrow D \rightarrow F$	17	$I \rightarrow E \rightarrow F \rightarrow R \rightarrow N \rightarrow D$
6	$I \rightarrow E \rightarrow N \rightarrow R \rightarrow F \rightarrow D$	18	$I \rightarrow E \rightarrow F \rightarrow R \rightarrow D \rightarrow N$
7	$I \rightarrow E \rightarrow D \rightarrow N \rightarrow F \rightarrow R$	19	$I \rightarrow E \rightarrow R \rightarrow N \rightarrow D \rightarrow F$
8	$I \rightarrow E \rightarrow D \rightarrow N \rightarrow R \rightarrow F$	20	$I \rightarrow E \rightarrow R \rightarrow N \rightarrow F \rightarrow D$
9	$I \rightarrow E \rightarrow D \rightarrow F \rightarrow N \rightarrow R$	21	$I \rightarrow E \rightarrow R \rightarrow D \rightarrow N \rightarrow F$
10	$I \rightarrow E \rightarrow D \rightarrow F \rightarrow R \rightarrow N$	23	$I \rightarrow E \rightarrow R \rightarrow D \rightarrow F \rightarrow N$
11	$I \rightarrow E \rightarrow D \rightarrow R \rightarrow N \rightarrow F$	23	$I \rightarrow E \rightarrow R \rightarrow F \rightarrow N \rightarrow D$
12	$I \rightarrow E \rightarrow D \rightarrow R \rightarrow F \rightarrow N$	24	$I \rightarrow E \rightarrow R \rightarrow F \rightarrow D \rightarrow N$

4.1 Universal pipeline prototype

The goal of this experiment is to demonstrate the difficulty of blindly finding the right pipeline prototype (i.e., without considering any meaningful or promising precedence). In Table 6, we list the exhaustive set of pipeline prototypes generated considering the compatible precedence graph in Table 1a (i.e., 24 compatible permutations). In a real scenario, this number is too high for splitting the time budget in order to optimize them. Yet, for the sake of this experiment, we exhaustively optimize all the prototypes, for each dataset. Thus, for each pipeline prototype and for each dataset, the SMBO algorithm is configured to assign a 200 seconds time budget to the phase of instantiating and optimizing the pipeline prototype, and another 200 seconds to the phase of optimizing the hyper-parameters of the ML algorithm.

The results obtained are shown in Figure 6. The enumerated prototypes are listed in the ordinate axis and each stacked bar represents the percentage of cases for which that prototype achieved the best performance across different ML algorithms (the contribution of each algorithm is represented with a different color). In an ideal scenario, for a pipeline to be considered *universal*, it should perform best in all or at least most of the cases, which is clearly not happening. Observe that, even the best performing pipeline is only the best in 11% of the cases, which is obviously far from being *universal*. Hence all (or at least several) pipelines need to be evaluated together, in order to obtain optimal results.

4.2 Exhaustive versus effective prototypes

Given that there is no single universal pipeline, one can opt for feeding all the possible prototypes (see Table 6) to the optimization algorithm and get the optimal results. As before, we assign a budget of 200 seconds for the optimization of each prototype, hence 80 minutes in total for all the set of 24 *exhaustive prototypes* in order to find the optimal pipeline for every dataset. On the other hand, we take only the five *effective prototypes* resulting from the application of our method and assign just 40 seconds time budget for the optimization of each one of them, hence 200 seconds in total. With the aim of comparing the two, and thus roughly understanding how close we are to the optimal case, we dedicated the same

[6]The source code and the datasets for reproducing the experiments can be found in https://github.com/josephgiovanelli/effective_preprocessing_pipeline_evaluation