Supplementary material for the manuscript *"Classification Models for Resource-Constrained Hardware and Sensors"*

Lucas Tsutsui da Silva, Vinicius M. A. Souza, and Gustavo E. A. P. A. Batista

V Evaluation of EmbML classifiers

This section presents detailed results for subsections V-B and V-C of the original manuscript.

B Classification Time

We compare the average time that each model spent to classify an instance. Figure 1 and Figure 2 respectively show the executions of the EmbML codes derived from WEKA and scikit-learn models. In these figures, each column of graphs describes a microcontroller, and each row denotes a dataset. Then, for a given combination of dataset and microcontroller, these graphs illustrate the time performance of different classifiers using the supported representations for real numbers. The missing points are those that could not be determined because their classifier code was larger than the microcontroller's memory.

C Memory Usage

For memory consumption, we separately compare data (SRAM) and program (flash) memories among the supported classifiers in Figure 3 (SRAM for WEKA models), Figure 4 (flash for WEKA models), Figure 5 (SRAM for scikit-learn models), and Figure 6 (flash for scikit-learn models). The graphs presented in these figures are combined in rows and columns, as explained previously.

VI Evaluation of code modifications

This section presents detailed results for subsections VI-A and VI-B of the original manuscript.

A Approximations for Sigmoid Function in MLP

In Figure 7 and Figure 8, we exhibit the average classification time comparison of the WEKA and scikit-learn MLP models, respectively, using the provided options for the sigmoid activation function. They do not contain the column corresponding to the ATmega328/P microcontroller, because there was not an MLP version able to execute in it.

B If-Then-Else Statements and Iterative Decision Trees

We compare the average classification time results in Figure 9 and Figure 10 for WEKA and scikit-learn decision tree models, respectively.

VII Comparison with related tools

This section presents detailed results for the section VII of the original manuscript. The chosen models, their identifiers and tools that support them are listed below.

- ***J48* WEKA** (referred as **C1**) is supported by EmbML and weka-porter. For this experiment, we used the if-then-else version of EmbML.
- ***SVC* (polynomial kernel) scikit-learn** (referred as **C2**) is supported by EmbML, m2cgen, and sklearn-porter.
- ***SVC* (RBF kernel) scikit-learn** (referred as **C3**) is supported by EmbML, m2cgen, and sklearn-porter.
- ***LinearSVC* scikit-learn** (referred as **C4**) is supported by EmbML, m2cgen, and sklearn-porter.
- ***DecisionTreeClassifier* scikit-learn** (referred as **C5**) is supported by EmbML, emlearn, m2cgen, and sklearn-porter. In this experiment, we selected the if-then-else version of EmbML.
- ***MLPClassifier* (ReLU) scikit-learn** (referred as **C6**) is supported by EmbML and emlearn.
- ***LogisticRegression* scikit-learn** (referred as **C7**) is supported by EmbML and m2cgen.

For each classifier, we used the different datasets and microcontrollers to compare average classification time, shown in Figure 11, and memory consumption (SRAM + flash), shown in Figure 12, between classifiers from EmbML and related tools.

In these figures, each graph contains the results corresponding to a specific dataset. The graphs incorporate only the time or memory values associated with a high accuracy rate to prevent including poor solutions – *e.g.*, the FXP16 versions of EmbML classifiers are faster but usually achieve lower accuracy rate than their corresponding FLT version. Therefore, for each combination of microcontroller, dataset, and classifier, we used the following steps to decide the values to plot:

1. unite the outcomes of the given combination produced by classifiers from EmbML and other tools;
2. determine the mean accuracy value of these results;
3. eliminate the results that achieve an accuracy lower than the mean value;
4. sort the remaining data in ascending order of classification time (or memory usage);
5. choose, at most, the three first results from the sorted list.

For instance, considering the mean classification time comparison for the *DecisionTreeClassifier* model, created using D2, and running on the AT91SAM3X8E microcontroller, Table 1 presents the values estimated using the test set during the experiments. Since the average accuracy rate is 80.54%, we exclude the EmbML/FXP16 version. Then,

we sort the remaining results in ascending order of mean classification time and select the first three, which are: EmbML/FLT, EmbML/FXP32, and emlearn. Finally, the time results of these versions are the chosen points included in the D2 graph. This process repeats for each dataset, microcontroller, and classifier.

Table 1: An example of mean classification time results for the *DecisionTreeClassifier* model, D2 dataset, and AT91SAM3X8E microcontroller.

Classifier version	**Accuracy rate**	**Mean classification time**
EmbML/FLT	84.02%	23.47 μs
EmbML/FXP32	84.28%	3.15 μs
EmbML/FXP16	63.06%	2.82 μs
emlearn	83.93%	20.85 μs
m2cgen	83.93%	27.13 μs
sklearn-porter	84.02%	36.93 μs

In the graphs, the large markers represent the lowest values for a specific combination of dataset, microcontroller, and classifier. Thus, for the example shown in Table 1, the EmbML/FXP32 point is larger than the EmbML/FLT and emlearn points in the D2 graph of Figure 11. Similarly, in many other cases, the EmbML classifiers are those that reach the best results for both classification time and memory consumption. In fact, Table 2 shows a comparison using these two metrics to evaluate, for each dataset, the number of cases that EmbML classifiers accomplish the best result and the total number of cases – *i.e.*, combinations of microcontroller and classifier – that at least one classifier code was able to execute. Therefore, the proposed method to compare the time and memory performances indicates that the EmbML classifiers were able to produce the best mean classification time in at least 70.97% of the cases, and the smallest memory consumption in at least 77.14%. These results reveal that EmbML classifiers frequently perform better than the other solutions, which is evidence that EmbML is an advantageous alternative.

Table 2: Overall time and memory comparison of classifiers from EmbML and related tools.

Dataset	**Cases which EmbML classifiers achieve the lowest time results**	**Cases which EmbML classifiers achieve the smallest memory results**	**Total number of cases**
D1	25 (71.43%)	27 (77.14%)	35
D2	27 (75.00%)	30 (83.33%)	36
D3	27 (77.14%)	30 (85.71%)	35
D4	22 (70.97%)	27 (87.10%)	31
D5	28 (77.78%)	35 (97.22%)	36
D6	23 (85.19%)	21 (77.78%)	27

Figure 1: Mean classification time for WEKA classifiers.

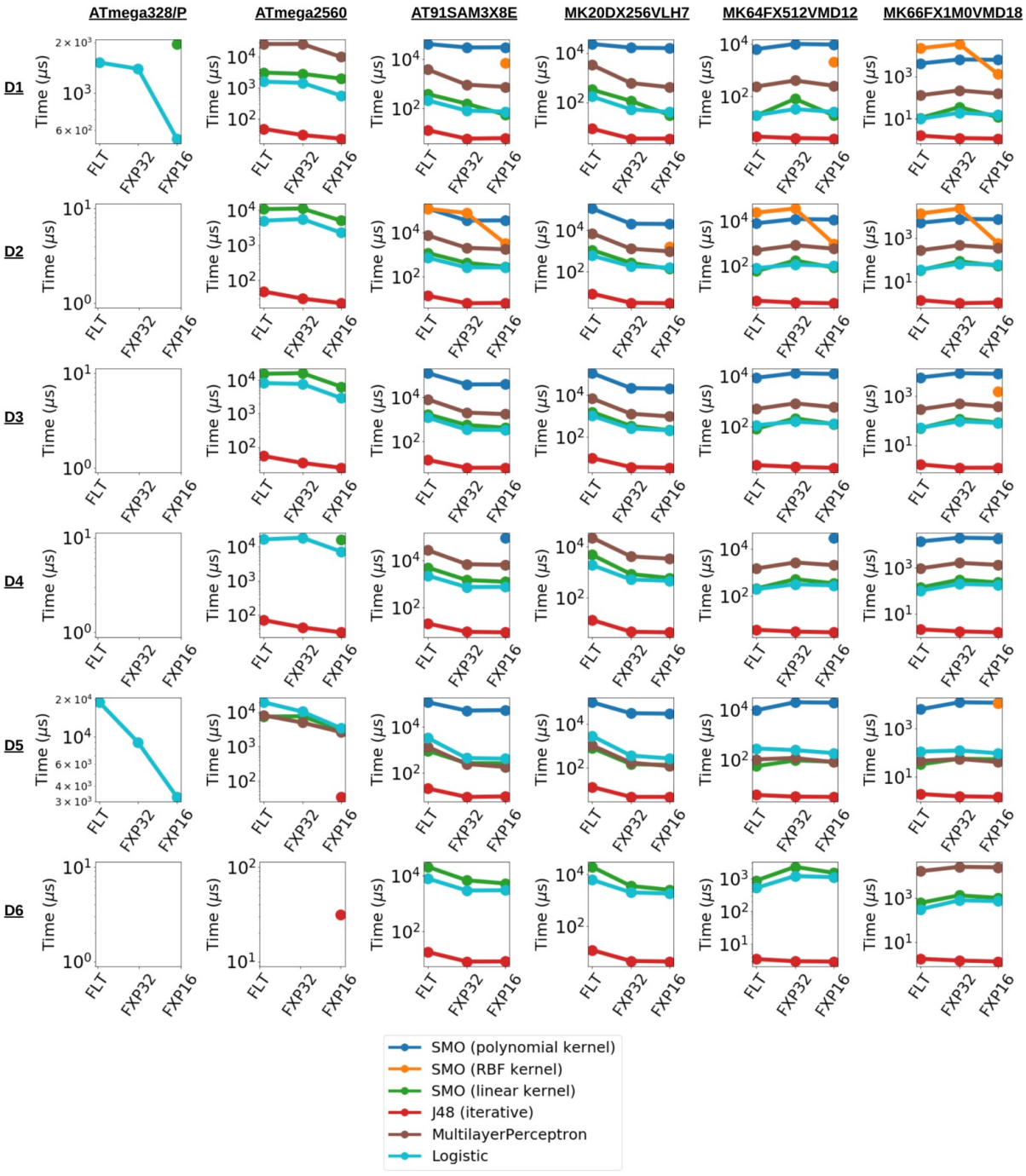

Figure 2: Mean classification time for scikit-learn classifiers.

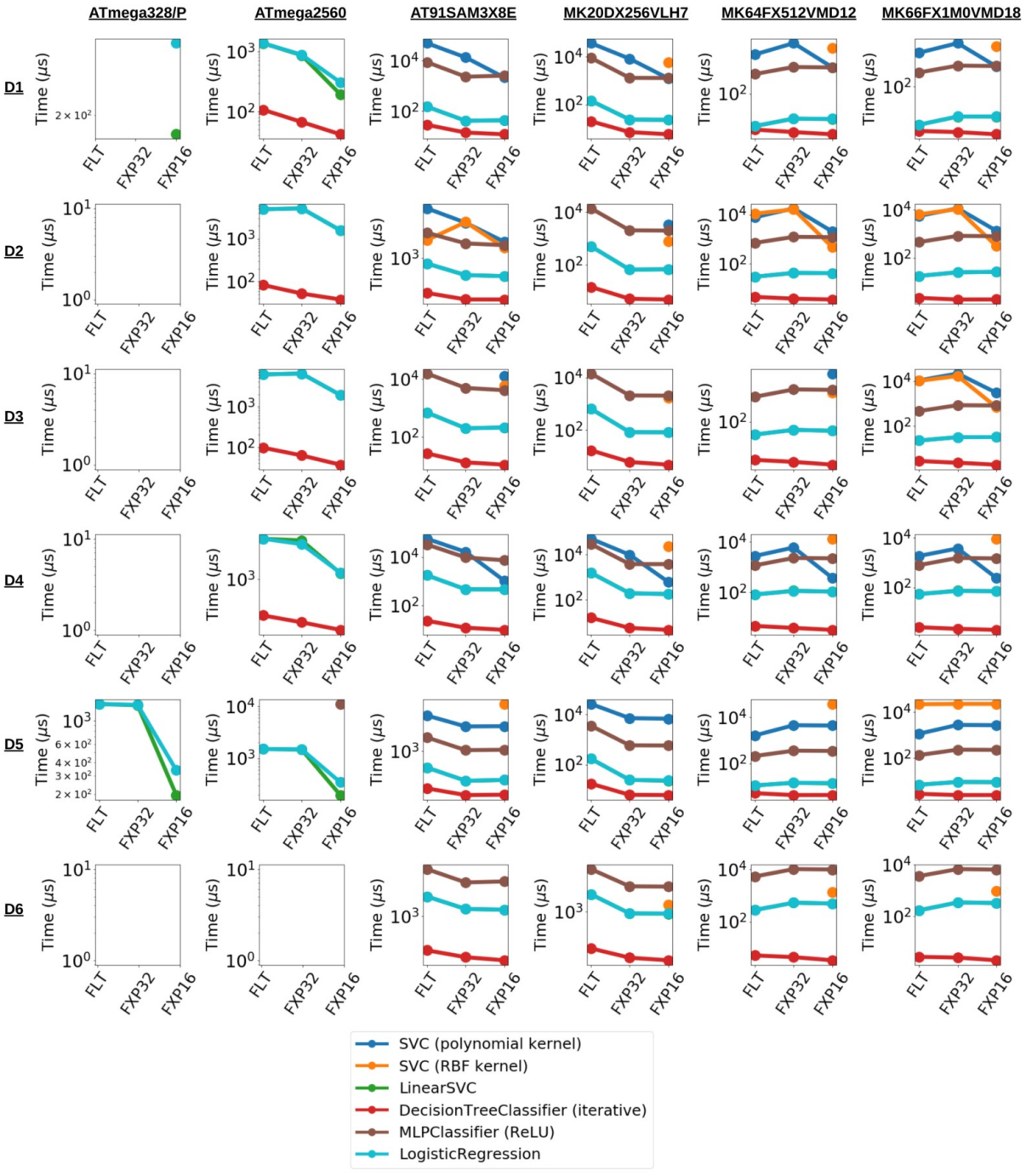

Figure 3: SRAM memory consumption for WEKA classifiers.

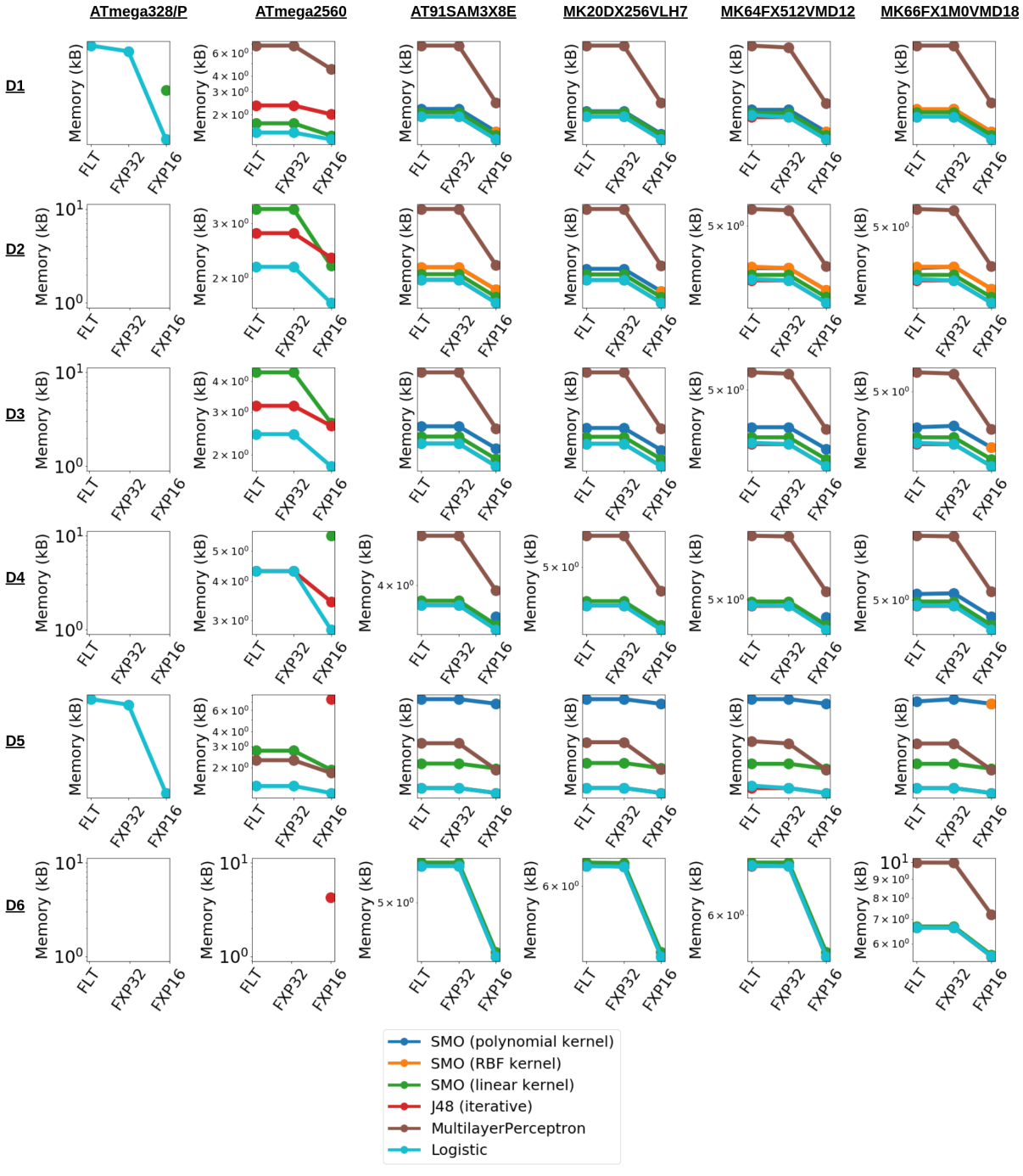

Figure 4: Flash memory consumption for WEKA classifiers.

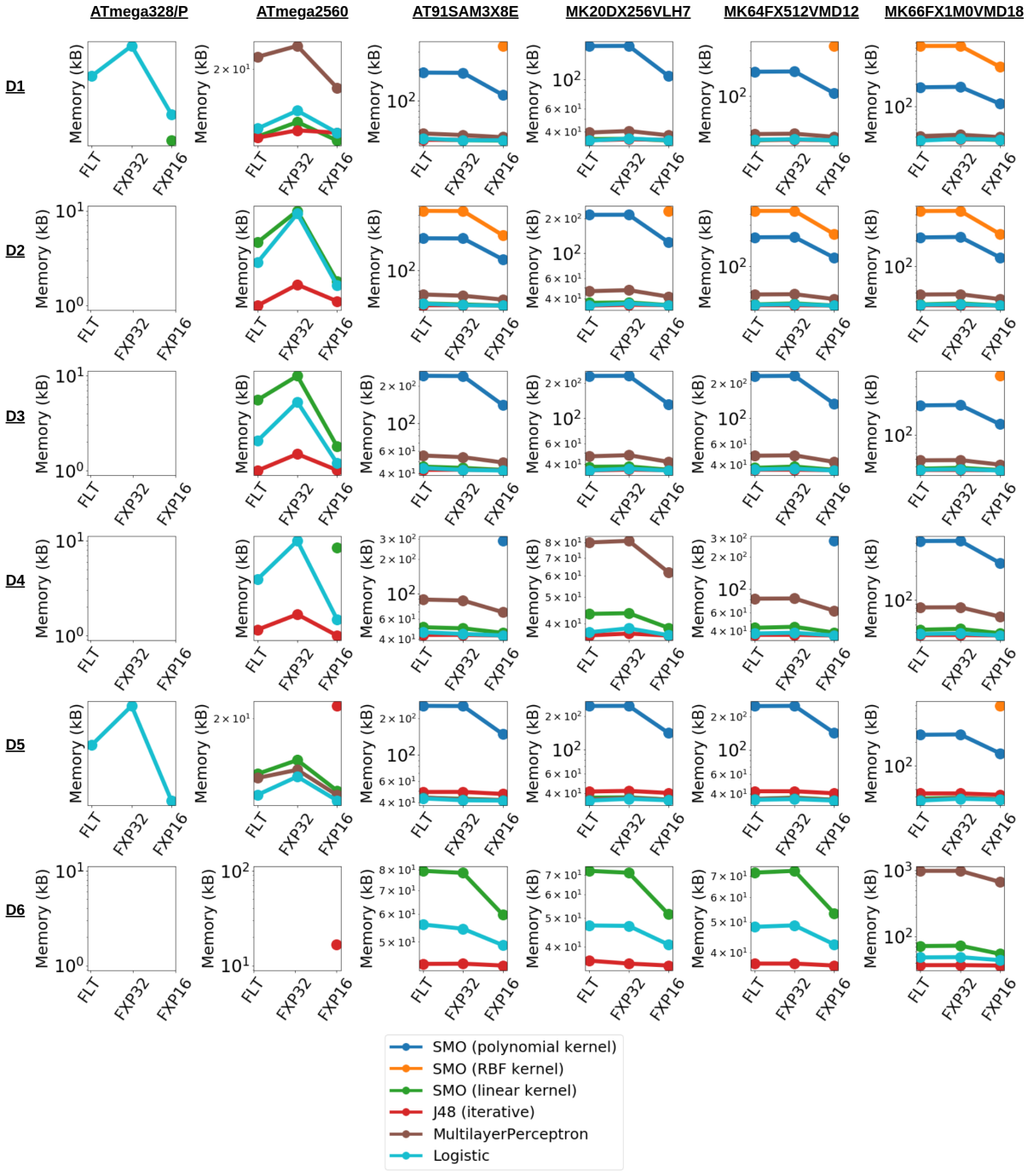

Figure 5: SRAM memory consumption for scikit-learn classifiers.

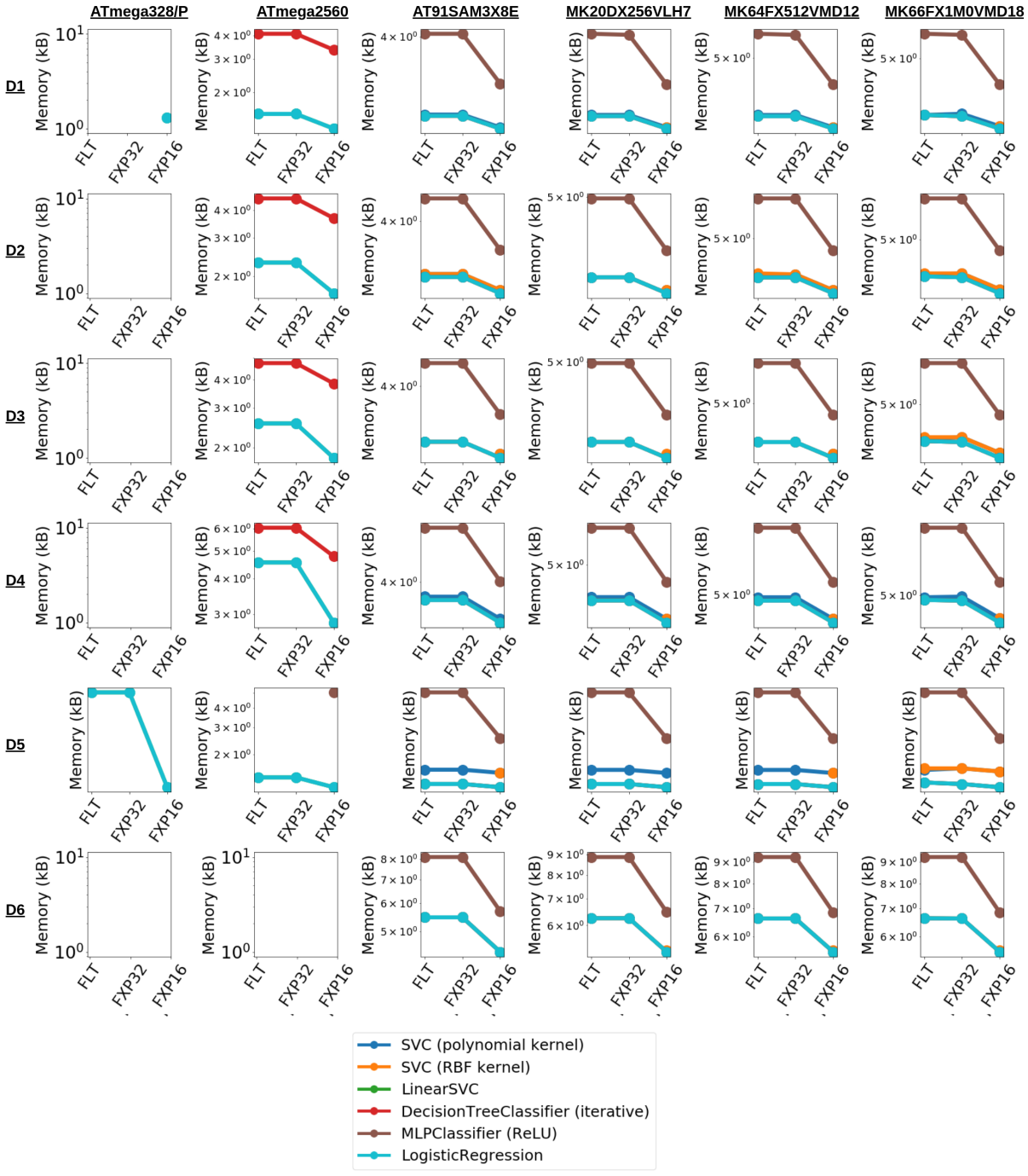

Figure 6: Flash memory consumption for scikit-learn classifiers.

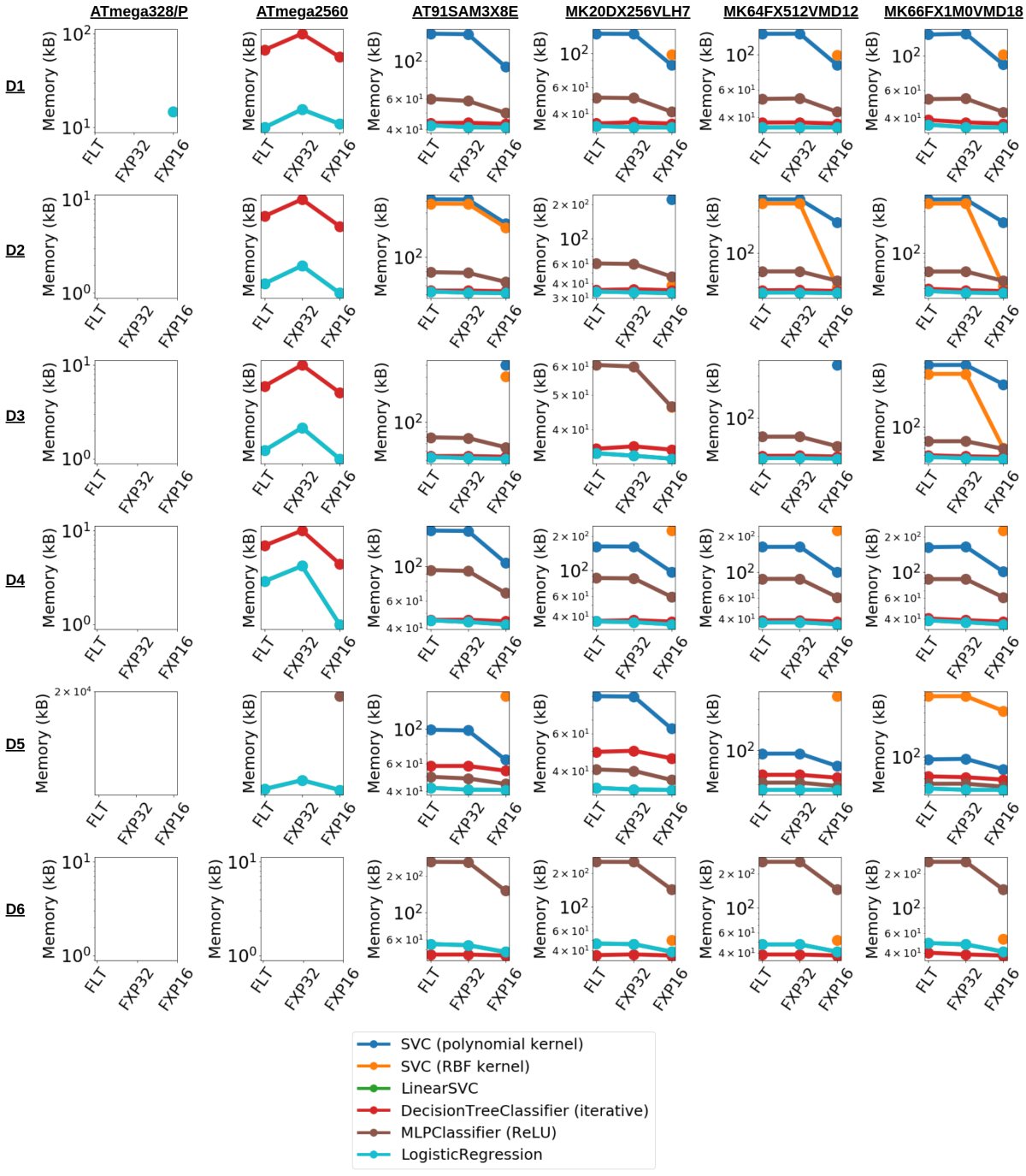

Figure 7: Mean classification time for the *MultilayerPerceptron* WEKA classifiers.

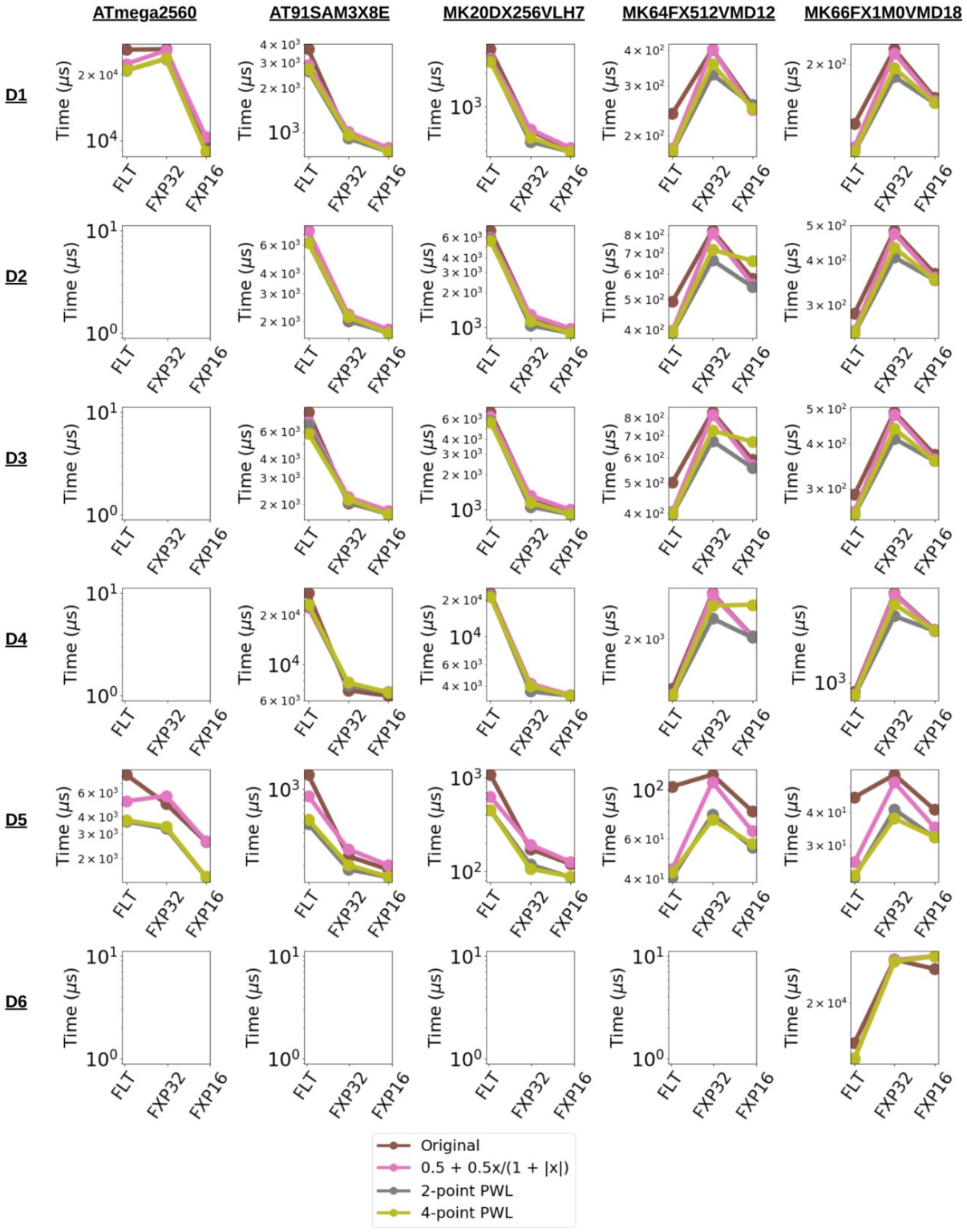

Figure 8: Mean classification time for the *MLPClassifier* scikit-learn classifiers.

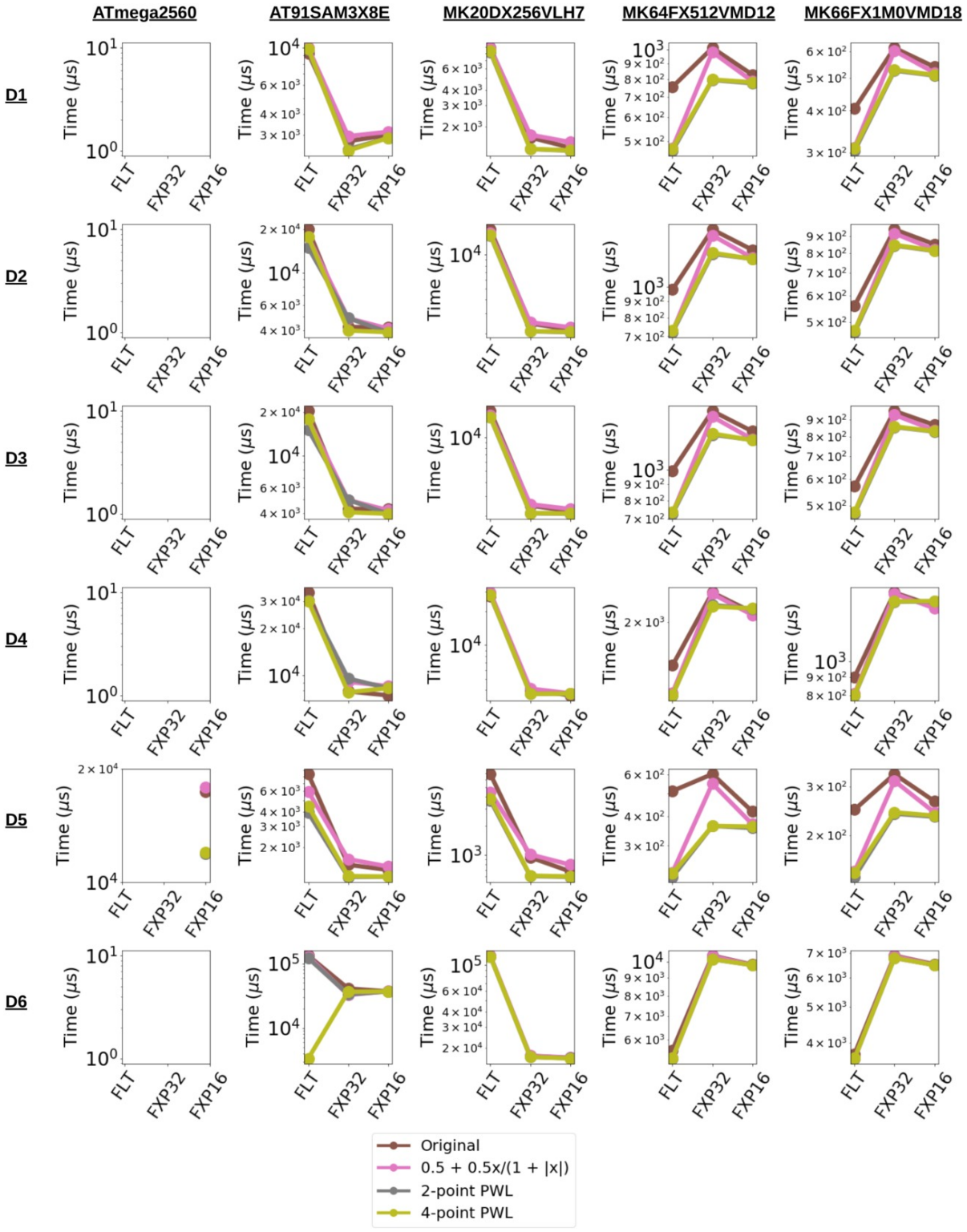

Figure 9: Mean classification time for the *J48* WEKA classifiers.

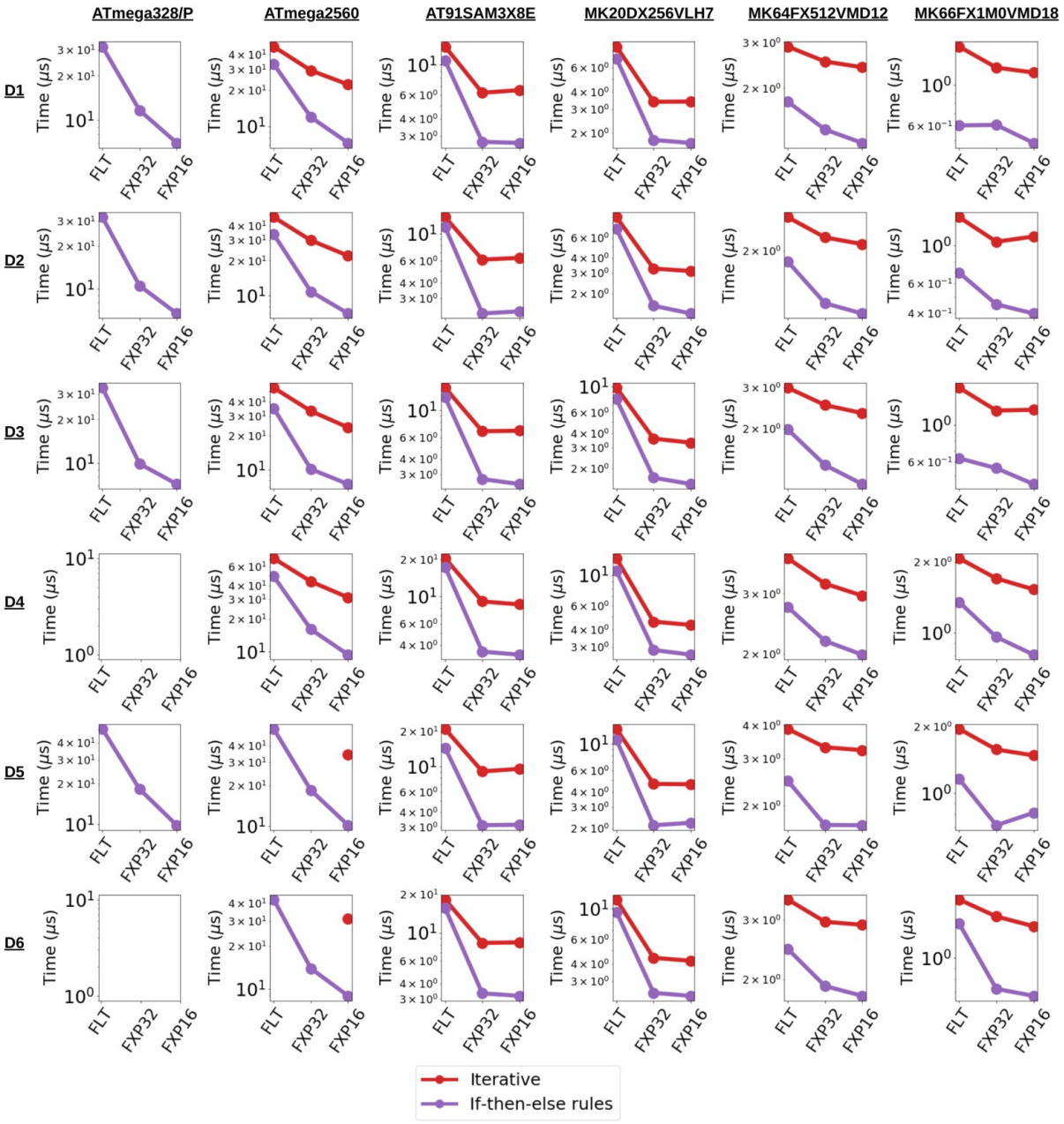

Figure 10: Mean classification time for the *DecisionTreeClassifier* scikit-learn classifiers.

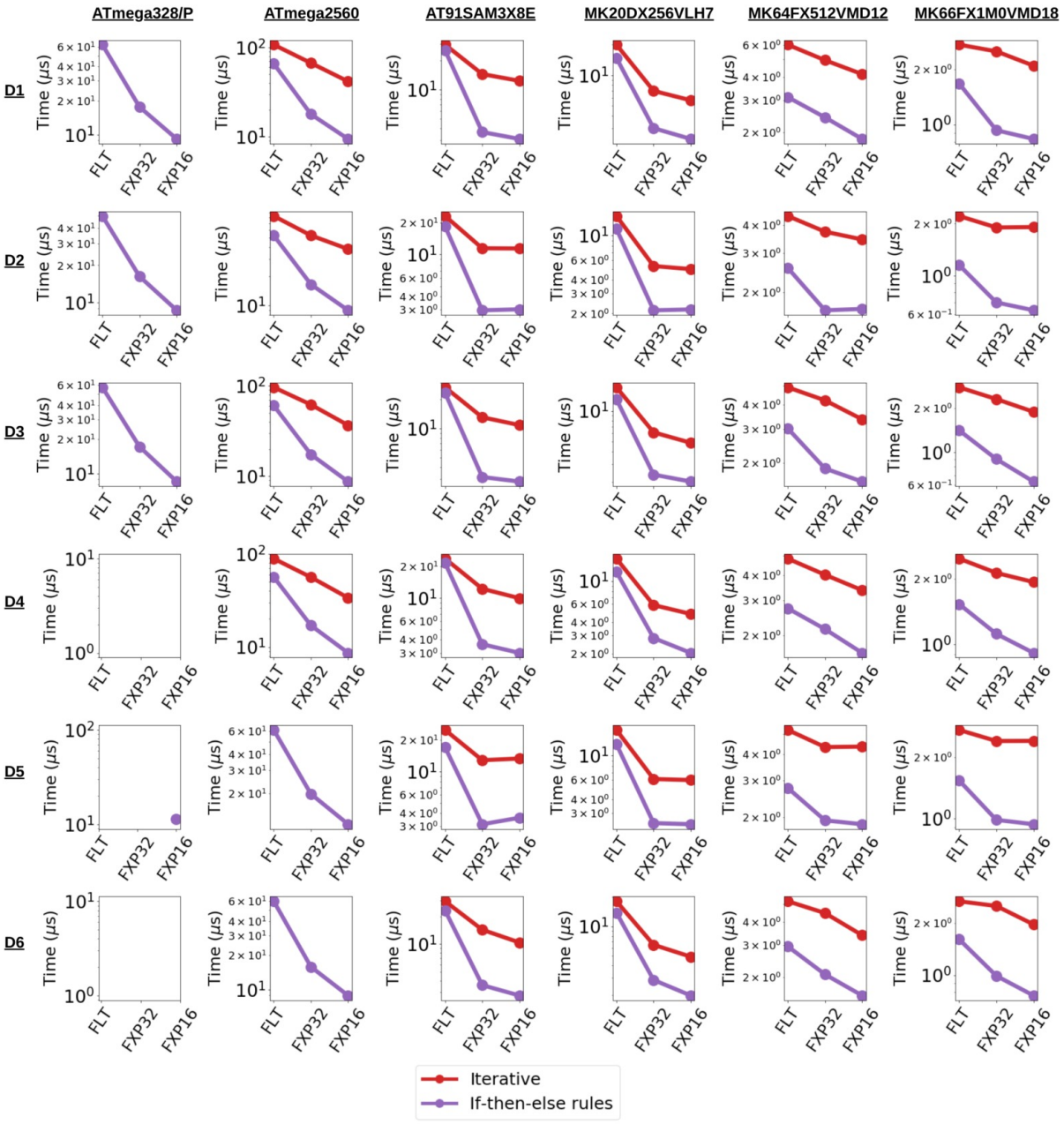

Figure 11: Mean classification time comparison between classifiers from EmbML and related tools.

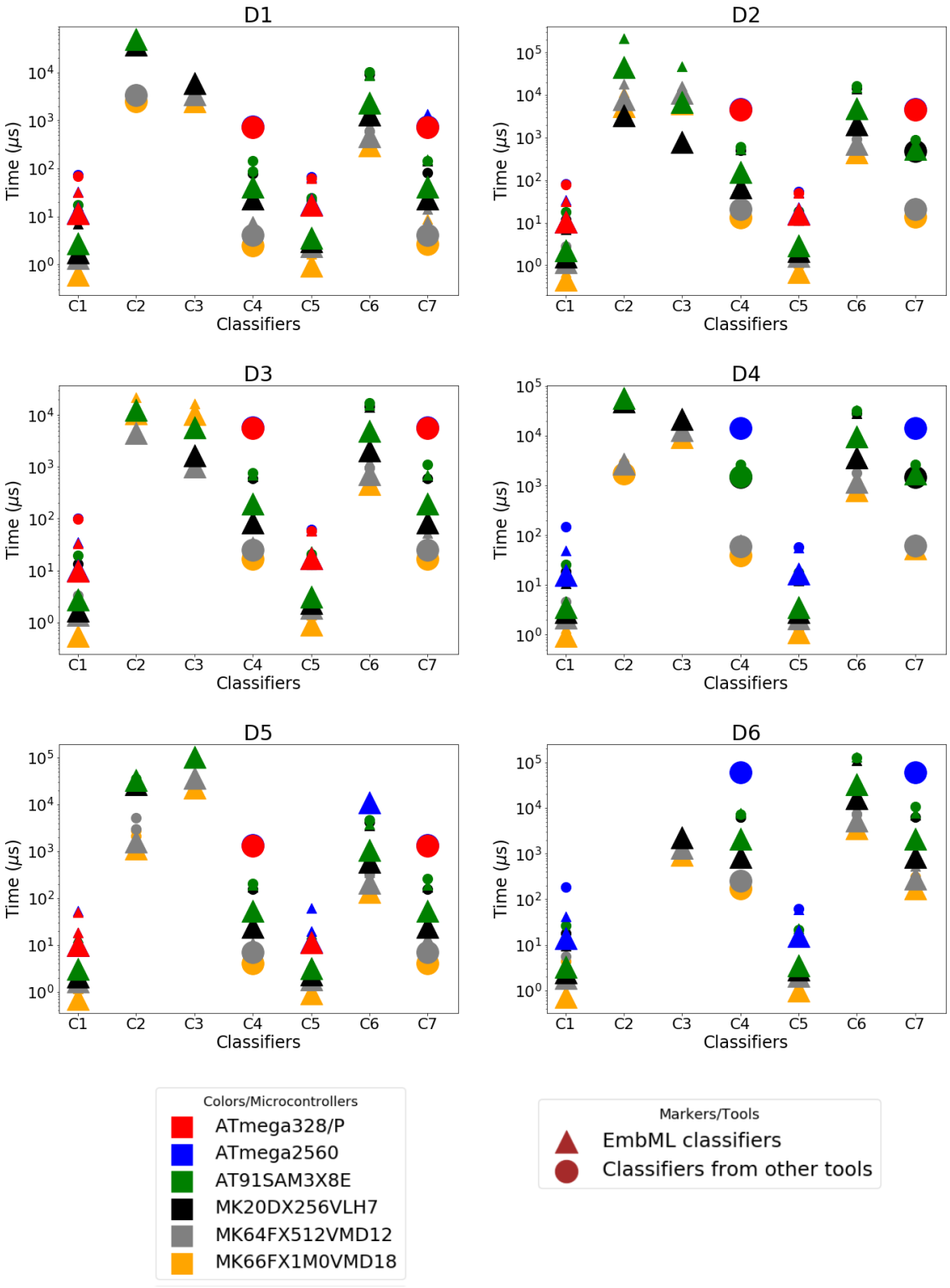

Figure 12: Memory usage comparison between classifiers from EmbML and related tools.