Report on AUTO-ML experiments

Student Name: Provost Simon.
Professor Name: Alex A. Freitas.

Jul 2021

1 Introduction

The following experiment demonstrates how the Auto-ML (i.e.: Auto-Sklearn) framework handles classification of medical datasets across a variety of different dimensions in the field of Healthcare. The datasets and their caracteristics used for the experiments are available in the section 2. However, to be consistent each run was performed on a Google cloud compute engine virtual machine instance with the following characteristics:

Table 1: Google Cloud virtual machine instance characteristics

Param name	**Param value**
Machine type	e2-standard-2
CPU	2 vCPU
Virtual Machine Memory	8 Go
Image size	10 Go

2 Datasets characteristics

The datasets used in the experiments that follow are described in detail in the table 2. (1) The dataset name indicates the dataset's name; (2) The number of samples indicates the total number of samples available for this dataset; (3) The number of features indicates the number of features available without the class labels column; and (4) The number of classes indicates the number of distinct class labels associated with the appropriate dataset, which also indicates whether it is a binary/multi class classification problem. Finally, (5) the column class_label x indicates the proportion of samples for that class in the entire dataset.

Here are the datasets source:

- Thoracic Surgery dataset [1].
- Diabetic Retinopathy Debrecen dataset [2].
- Estimation of obesity levels based on eating habits and physical condition [3].
- Breast Cancer Coimbra Data Set [4].
- Heart failure clinical records Data Set [5].

Table 2: Datasets used for the experiments.

dataset name	number of samples	number of features	number of classes	class_label_1(%)	class_label_2(%)	class_label_3(%)	class_label_4(%)	class_label_5(%)	class_label_6(%)	class_label_7(%)
Breast-cancer-Coimbra	116	9	2	44.827586	55.172414					
diabetic-retinopathy-Debrecen	1151	19	2	46.915725	53.084275					
Heart-failure-clinical-records_dataset	299	12	2	67.892977	32.107023					
ObesityDataSet_raw_and_data_synthetic	2111	16	7	12.884889	13.595452	16.627191	14.069162	15.348176	13.737565	13.737565
Thoracic-Surgery-binary-survival	470	16	2	85.106383	14.893617					

3 Auto-Sklearn characteristics

The following table depicts the Auto-Sklearn's parameters used for the experiments (refer to the Auto-Sklearn API for more details):

Table 3: Auto-Sklearn parameters

Param name	Param value
time_left_for_this_task	3600
per_run_time_limit	360
n_jobs	4 (i.e.: maximum 4, if only 2 available 2 will be taken)
memory_limit	5000
seed	2/11/21/42/85
resampling_strategy	holdout
ensemble_size	50

4 Auto-Weka characteristics

The following table depicts the Auto-Weka's parameters used for the experiments (refer to the Auto-Weka API for more details):

Table 4: Auto-Weka parameters

Param name	**Param value**
-t <name of training file>	training_set of the fold (i)
-T <name of test file>	test_set of the fold (i)
-timeLimit <limit> (approximately in minutes)	60 (3600 seconds)
-seed <seed>	85
-memLimit <limit> (in MiB)	5000
-nBestConfigs <limit>	50

5 Phase 1 results of the experiment

5.1 Experiment details

The following experiment uses Auto-Sklearn to search for the best model to classify an entire dataset given.

For the results available in the table 5, Auto-Sklearn[6] (i.e.: An Auto-ML Framework) was applied to the entire dataset chosen. This relies on Auto-ML doing an internal partition of the data into training and validation during its run.

5.2 Table columns details

Each row shows a set of different metrics: (1) The dataset's name; (2) the classifier selected by the AutoML pipeline; (3 & 4) the search time limit in seconds as well as the duration of the algorithm; (5) the seed value, which ensures reproducibility of the model if the results are consistent across different seeds (for seeds chosen see section 3); (6) the accuracy score of this classifier on the test data previously determined by the initial data split at the start of the pipeline; (7 & 8) The recall& precision score which is the ability of the classifier to find all the positive samples; (9 & 10) The F1 score macro and micro are used which depicts for macro the following: compute te metric independently for each class and thus to compute an average that compares all classes equally, and for micro: aggregates the contributions of all classes to compute the average metric; Finally, (11) the AUROC (i.e: receiver operating characteristic curve) score, which is a performance metric for discrimination: it tells us about the model's ability to discriminate between cases - positive examples - and non-cases - negative examples - (i.e.: Higher is the score better is the classifier).

Table 5: Optimum classifiers chosen from the Auto-Sklearn pipeline on the datasets presented section 2. Hyperparameters are available in the appendix (see HP column). Note: HP stands for Hyper Parameters.

Dataset name	HP	Classifier	Search Time limit	Algorithm time run (s)	Seed	Accuracy	Precision	Recall	F1 score macro	F1 score micro	Auroc
breast-cancer	Breast-1	mlp	3600	2.482066	11	0.827586	0.825758	0.816176	0.819876	0.827586	0.816176
breast-cancer	Breast-2	qda	3600	3.506922	2	0.517241	0.558333	0.551471	0.512019	0.517241	0.551471
breast-cancer	Breast-3	mlp	3600	3.074595	21	0.793103	0.798077	0.795238	0.792857	0.793103	0.795238
breast-cancer	Breast-4	random_forest	3600	2.54832	42	0.793103	0.798077	0.795238	0.792857	0.793103	0.795238
breast-cancer	Breast-5	random_forest	3600	6.722548	85	0.655172	0.637019	0.658333	0.633838	0.655172	0.658333
Heart-failure	Heart-failure-1	libsvm_svc	3600	2.202007	11	0.92	0.919118	0.902669	0.91	0.92	0.902669
Heart-failure	Heart-failure-2	mlp	3600	4.372969	2	0.866667	0.831104	0.849206	0.839194	0.866667	0.849206
Heart-failure	Heart-failure-3	k_nearest_neighbors	3600	2.542999	21	0.826667	0.766667	0.733918	0.747475	0.826667	0.733918
Heart-failure	Heart-failure-4	random_forest	3600	4.143286	42	0.746667	0.738889	0.736437	0.737521	0.746667	0.736437
Heart-failure	Heart-failure-5	k_nearest_neighbors	3600	3.839199	85	0.8	0.716374	0.725152	0.720497	0.8	0.725152
Obesity	obesity-1	gradient_boosting	3600	22.408372	11	0.965909	0.966143	0.96609	0.965447	0.965909	0.999414
Obesity	obesity-2	gradient_boosting	3600	131.172882	2	0.982955	0.98177	0.98325	0.982323	0.982955	0.999707
Obesity	obesity-3	gradient_boosting	3600	7.056944	21	0.971591	0.969771	0.969468	0.969479	0.971591	0.998308
Obesity	obesity-4	gradient_boosting	3600	28.593236	42	0.960227	0.958619	0.960982	0.959452	0.960227	0.99933
Obesity	obesity-5	gradient_boosting	3600	9.831563	85	0.982955	0.981575	0.982844	0.982134	0.982955	0.999591
Thoracic-Surgery	Thoracic-Surgery-1	sgd	3600	1.728338	11	0.79661	0.401709	0.494737	0.443396	0.79661	0.494737
Thoracic-Surgery	Thoracic-Surgery-2	bernoulli_nb	3600	2.939708	2	0.847458	0.438596	0.480769	0.458716	0.847458	0.480769
Thoracic-Surgery	Thoracic-Surgery-3	bernoulli_nb	3600	2.434407	21	0.779661	0.562319	0.509247	0.473214	0.779661	0.509247
Thoracic-Surgery	Thoracic-Surgery-4	gradient_boosting	3600	3.708382	42	0.79661	0.571014	0.511213	0.481319	0.79661	0.511213
Thoracic-Surgery	Thoracic-Surgery-5	decision_tree	3600	3.101079	85	0.830508	0.588406	0.516215	0.498726	0.830508	0.516215
Diabetic-retinopathy	diabetic-retinopathy-1	mlp	3600	13.145924	11	0.704861	0.704915	0.704371	0.70443	0.704861	0.704371
Diabetic-retinopathy	diabetic-retinopathy-2	random_forest	3600	13.08779	2	0.715278	0.718083	0.718083	0.715278	0.715278	0.718083
Diabetic-retinopathy	diabetic-retinopathy-3	libsvm_svc	3600	3.151908	21	0.784722	0.785642	0.787194	0.784556	0.784722	0.787194
Diabetic-retinopathy	diabetic-retinopathy-4	mlp	3600	7.289334	42	0.760417	0.760417	0.762557	0.759928	0.760417	0.762557
Diabetic-retinopathy	diabetic-retinopathy-5	mlp	3600	4.642596	85	0.767361	0.77019	0.76804	0.767021	0.767361	0.76804

6 Phase 2 & 3 results of the experiments

6.1 Experiment details

The following experiments perform a 10-fold cross validation of the dataset using a hand-made python program and Auto-Sklearn/Weka.

For the result of the table below, Auto-Sklearn[6] (i.e.: An Auto-ML Framework), Auto-Weka (i.e., another Auto-ML Framework), as well as Scikit Learn (i.e.: A Machine learning framework) was applied to follow the experiment. Nonetheless, a Python program has been made (see pseudo-code section 6.1.1 and 6.1.2) in order to follow a 10-fold cross-validation process for each framework respectively.

Table 6: Auto-Sklearn versus Auto-Weka on the average of the 10 fold-cross validation process

	Auto-Sklearn Classifier (most chosen)	Auto-Weka Classifier (most chosen)	Auto-Sklearn Accuracy	Auto-Weka Accuracy	Auto-Sklearn Precision	Auto-Weka Precision	Auto-Sklearn Recall	Auto-Weka Recall	Auto-Sklearn AUROC	Auto-Weka AUROC
Diabetic-retinopathy-Debrecen	mlp	LWL	**0.750**	0.7131	**0.752**	0.72	**0.750**	0.71	0.750	**0.77**
Obesity Dataset raw and data syntheti	gradient_boosting	AdaBoostM1	**0.967**	0.9208	**0.965**	0.92	**0.965**	0.92	**0.998**	0.99
Thoracic Surgery Binary Survival	gradient_boosting	All different	0.72	**0.8532**	**0.51**	0.31	0.52	**0.85**	**0.52**	0.49
Breast cancer Coimbra	mlp	Bagging	**0.72**	0.6977	0.70	**0.72**	0.69	**0.70**	**0.69**	0.68
Heart failure clinical records	random_forest	All different	**0.86**	0.8405	0.83	**0.84**	0.82	**0.84**	0.82	0.82

NB: 'All Different' means that each fold presented a different algorithm chosen by the Auto-ML pipeline.
NBB: If you were looking to see the results for each fold and for each frame<ork or dataset, see the appendix.

6.1.1 Pseudo-code of the Python Program (Auto-Sklearn)

The program's steps for the Auto-Sklearn experiment are as follows: (1) The program splits the dataset randomly into ten-folds; (2) drops one fold for later use; (3) uses the remaining folds to feed the Auto-ML pipeline to find the best model using this training samples; (4) once Auto-Sklearn output the best fitted model, the fold dropped previously becomes useful because Scikit-learn will use it to determine the viability of the best fitted model on this "unseen" sample data; and (5) this process is repeated ten times, with each time adding another fold for the test_-set (i.e.: unseen data), and the remains one feeding Auto-Sklearn to generate another fitted best model on that data. Finally, as a result, the program would output ten models with ten results on unseen data for one dataset. Additonally, via the help of bash script, this entire process is then repeated for each dataset.

Algorithm 1 Three main steps of the 10-fold cross-validation process. *Note: See snippets belows for more details.*

```
1: k_folds_split(k_folds ← 10)
2: cross_validation_process(all_10_folds)
3: show_latex_cross_validation_process()
```

Algorithm 2 Function that split the dataset into 10 random folds.

```
1: function K_FOLDS_SPLIT(entireDataset, k_folds)
2:     data ← entireDataset.copy()
3:     data ← data.sample(frac ← 1)
4:     all_10_folds ← array_split(data, k_folds)
5: end function
```

Algorithm 3 Function that do the cross-validation process over the 10 random folds. *Note: The reSampling if condition determines whether a dataset needs to be re sampled due to inconsistency in its data. At the moment, the feature has been used only once and does not result in something better, the feature remains still available.*

```
1: function CROSS_VALIDATION_PROCESS(all_10_folds, reSampling ← False)
2:
3:     for idx, fold ∈ enumerate(all_10_folds) do
4:         training_set ← all_10_folds.copy()
5:         training_set.pop(idx)
6:         test_set ← fold
7:
8:         input, output ← __get_inputs_outputs_from_folds(training_set)
9:         if reSampling then
10:            input, output ← reSamplingSMOTE(training_set, input, output)
11:        end if
12:        setup(input, output)
13:        fit_predict()
14:
15:        save_model(path ← filepath)
16:        loaded_classifier ← load_model(path ← filepath)
17:        all_best_models.append(loaded_classifier)
18:
19:        crossValUnseenDataOutput ← __predict_unseen_data_cross_validation_pro-
   cess(test_set, loaded_classifier)
20:        ensemble_results.append(crossValUnseenDataOutput[0])
21:    end for
22: end function
```

6.1.2 Pseudo-code of the Python Program refactored (Auto-Weka)

About Auto-weka's experiment the python program showed section 6.1.1 was refactored. The steps of the new program are as follows: (1) The program randomly divides the dataset into ten folds; (2) [while looping over the tenfold each time:] save one fold as a ".arff" (i.e., WEKA supported file) file for later use (i.e., test set); (3) [while looping over the tenfold each time:] save the remaining folds (i.e., training set) as a ".arff" file; (4) After having saved all the test_set and training_set files, a bash script loop over the ten training set previously saved with a java command line that run an Auto-Weka analysis with it, and in addition perform a test over the best fitted model with a test set provided as parameter. Additionally, this entire process is then repeated for each dataset along the bash script.

Algorithm 4 Function that save the 10 random folds into training/test set for Auto-Weka usage.

```
1:  function CROSS_VALIDATION_PROCESS(all_10_folds, reSampling ← False)
2:
3:      for idx, fold ∈ enumerate(all_10_folds) do
4:          training_set ← all_10_folds.copy()
5:          training_set.pop(idx)
6:          test_set ← fold
7:
8:          input, output ← __get_inputs_outputs_from_folds(training_set)
9:          if reSampling then
10:             input, output ← reSamplingSMOTE(training_set, input, output)
11:         end if
12:         inputArffDf ← input.copy()
13:         output ← output.apply(str).astype("category")
14:         inputArffDf['class'] ← output
15:         trainingAttributes = self.save_set_arff(inputArffDf, "training_set_fold", str(idx), True)
16:         self.save_set_arff(test_set, "test_set_fold", str(idx), False, trainingAttributes)
17:     end for
18: end function
```

Algorithm 5 The Bash function that run auto-weka for each of the 10-fold (i.e, test_set and training_set) available following the python program execution.

```
1: function WEKACV
2:     for i ∈ 0..9 do
3:         FILE_TRAIN ← "./training_set_old_i.arff"
4:         FILE_TEST ← "./test_fold_i.arff"
5:         if FILE_TRAIN or FILE_TEST do not exist then
6:             continue
7:         end if
8:         java -cp autoweka.jar weka.classifiers.meta.AutoWEKAClassifier -t FILE_TRAIN -T
   FILE_TEST -timeLimit 60 -seed 85 -memLimit 5000 -nBestConfigs 50
9:     end for
10: end function
```

References

[1] M. Zikeba, J. M. Tomczak, M. Lubicz, and J. 'Swikatek, "Boosted svm for extracting rules from imbalanced data in application to prediction of the post-operative life expectancy in the lung cancer patients," *Applied Soft Computing*, 2013.

[2] B. Antal and A. Hajdu, "An ensemble-based system for automatic screening of diabetic retinopathy," *Knowledge-based systems*, vol. 60, pp. 20–27, 2014.

[3] F. M. Palechor and A. de la Hoz Manotas, "Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from colombia, peru and mexico," *Data in brief*, vol. 25, p. 104344, 2019.

[4] M. Patrício, J. Pereira, J. Crisóstomo, P. Matafome, M. Gomes, R. Seiça, and F. Caramelo, "Using resistin, glucose, age and bmi to predict the presence of breast cancer," *BMC cancer*, vol. 18, no. 1, pp. 1–8, 2018.

[5] D. Chicco and G. Jurman, "Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone," *BMC medical informatics and decision making*, vol. 20, no. 1, pp. 1–16, 2020.

[6] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning, 2015," *URL http://papers. nips. cc/paper/5872-efficient-and-robust-automated-machine-learning*.