Low-code AutoML-augmented Data Pipeline – A Review and Experiments

Low-code AutoML-augmented Data Pipeline – A Review and Experiments

Ulla Gain* and Virpi Hotti

School of Computing, University of Eastern Finland, Finland

*Email: ulla.gain@uef.fi; virpi.hotti@uef.fi

Abstract. There is a lack of knowledge concerning the low-code autoML (automated machine learning) frameworks that can be used to enrich data for several purposes concerning either data engineering or software engineering. In this paper, 34 autoML frameworks have been reviewed based on the latest commits and augmentation properties of their GitHub content. The PyCaret framework was the result of the review due to requirements concerning adaptability by Google Colaboratory (Colab) and the BI (business intelligence) tool. Finally, the low-code autoML-augmented data pipeline from raw data to dashboards and low-code apps has been drawn based on the experiments concerned classifications of the "Census Income" dataset. The constructed pipeline preferred the same data to be a ground for different reports, dashboards, and applications. However, the constructed low-code autoML-augmented data pipeline contains changeable building blocks such as libraries and visualisations.

1. Introduction

Common data models and common data lakes are examples to promote both low-code application development and AI-BI-insights generation. However, both individuals and organisations build their data models and data storages, or at least they make experiments concerning data pipelines from raw data into insights. At the same time, there is pressure to predict and even prevent the behaviour of things such as customers.

There are several machine learning (ML) frameworks that can be adapted to build and deploy state-of-the-art machine learning models for predictions and detections as well as consume the models for unseen data. However, there is a lack of knowledge concerning the low-code autoML frameworks that can be used to generate models with minimum setting up parameters as well as to automate data processing workflows (a.k.a., pipelines). There are some open-source Python packages for pipeline development. However, some autoML frameworks orchestrate the entire pipeline from data preparations into adaptable models. Those frameworks automate, for example, missing value imputations, categorical data transformations, and hyperparameter tunings.

The review is based on free and open-source software (FOSS) ML frameworks [1]. First, we research whether there are the autoML wrappers of the state-of-the-art methods to do either supervised or unsupervised learning on tabular data. Our research questions are as follows: Are selected ML frameworks committed during the last three years? Whether they are, then we research possibilities to use the framework to label classes, clusters and outliers as well as pre-calculate continuous values for tabular data. Finally, we will figure out is the autoML framework based on Python because we have deployment requirements concerning Google Colaboratory and the Microsoft Power BI tool. Google Collaboratory can be used to upstreaming data, and several models can be generated and evaluated

without feature engineering, such as transforming categorical values into numerical ones. Moreover, the best ML model shall be runnable in the Python script of the BI tool such as Microsoft Power BI. The main reason for that is to provide insights-driven data pipelines where data is ingested, unified, and mastered, as well as analysed and enriched to provide a ground for reports, dashboards, and applications.

2. Review Results

Twenty-nine frameworks seem to be under construction based on the latest commits and augmentation properties of their GitHub content (Table 1).

Table 1. The autoML frameworks and the latest commits (15.10.2020).

Framework	GitHub	2018	2019	2020
Acme	https://github.com/deepmind/acme			x
AdaNet	https://github.com/tensorflow/adanet			x
Analytics Zoo	https://github.com/intel-analytics/analytics-zoo			x
auto_ml	https://github.com/ClimbsRocks/auto_ml		x	
Blocks	https://github.com/mila-udem/blocks		x	
Detectron2	https://github.com/facebookresearch/detectron2			x
Dopamine	https://github.com/google/dopamine			x
Fastai	https://github.com/fastai/fastai/			x
Featuretools	https://github.com/Featuretools/featuretools			x
FlyingSquid	https://github.com/HazyResearch/flyingsquid			x
Karate Club	https://github.com/benedekrozemberczki/karatecluB			x
Keras	https://github.com/keras-team/keras		x	
learn2learn	https://github.com/learnables/learn2learn/			x
Lore	https://github.com/instacart/lore			x
Mljar	https://github.com/mljar/mljar-supervised			x
MLsquare	https://github.com/mlsquare/mlsquare			x
NeuralStructuredLearning	https://git hub.com/tensorflow/neural-structured-learning			x
NNI	https://github.com/Microsoft/nni			x
NuPIC	https://github.com/benedekrozemberczki/karatecluB		x	
Plato	https://github.com /uber-research/plato-research-dialogue-system			x
Polyaxon	https://github.com/polyaxon/polyaxon			x
PyCaret	https://github.com/pycaret/pycaret			x
Pyro	https://github.com/uber/pyro			x
Pythia	https://github.com/facebookresearch/pythia			x
PyTorch	https://github.com/pytorch/pytorch			x
ReAgent	https://github.com/facebookresearch/ReAgent			x
RLCard	https://github.com/datamllab/rlcard			x
Scikit-learn	https://github.com/scikit-learn/scikit-learn			x
Streamlit	https://github.com/streamlit/streamlit			x
TF Encrypted	https://github.com/tf-encrypted/tf-encrypted			x
Theano	https://github.com/Theano/Theano			x
Thinc	https://github.com/explosion/thinc			x
Turi & TuriCreate	https://github.com/apple/turicreate			x
XAI	https://github.com/EthicalML/xai		x	

There are only two frameworks (Table 2) that can be used to label classes, clusters and outliers as well as pre-calculate continuous values for tabular data. Observe that other augmentation purposes are not predefined due to vague descriptions of the frameworks. However, the PyCaret framework is only one low-code machine learning library that can be used by Google Colaboratory and Microsoft Power BI [2].

Table 2. Appraised autoML frameworks and their augmentations (outliers, clusters, classes, and pre-calculations) for tabular data and other purposes.

Framework	Outliers	Clusters	Classes	Pre-calculations	Other purposes
Acme					reinforcement learning
AdaNet			x		
Analytics Zoo				time-series	computer vision, NLP, recommendation
Detectron2					object detection
Dopamine					reinforcement learning
Fastai			x		image classification, image segmentation, text-based sentiments, recommendation
Featuretools					automate feature engineering
FlyingSquid					labelling
Karate Club					unsupervised learning on graph-structured data
learn2learn					meta-learning
Lore					standardise ML techniques across multiple libraries
Mljar			x	regression	machine-learning pipelines
MLsquare			x		recommendation
MMF (fka Pythia)					vision and language modelling
NeuralStructuredLearning					image classification
NNI					manages AutoML experiments
NuPIC					unsupervised learning on graph-structured data
Plato					conversational AI agents
Polyaxon					container-native engine for running machine learning pipelines
PyCaret	x	x	x	regression, time series	association mining, NLP, machine learning pipelines
Pyro					deep probabilistic modelling
PyTorch					provide tensor routines
ReAgent					an end-to-end platform for applied reinforcement learning (RL) developed
RLCard					toolkit for reinforcement learning in card games
Scikit-learn	x	x	x	regression, time-series	pre-processing, model selection, dimensionality reduction
Streamlit					to create apps for machine learning projects
TF Encrypted					enable training and prediction over encrypted data
Theano					define, optimise, and evaluate mathematical expressions involving multi-dimensional arrays
Thinc					composing models
Turi & TuriCreate					recommendations, object detection, image classification, image similarity

3. Experiment-based Low-code AutoML-augmented Data Pipeline

The "Census Income" dataset (https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data) columns are "Age", "Workclass", "fnlwgt", "Education", "Education-Num", "Marital-Status", "Occupation", "Relationship", "Race", "Sex", "Capital-Gain", "Capital-Loss", "Hours-per-week", "Country", and "Over-50K". The PyCaret framework contains 18 classification models that are used to construct models when we figure out features that can be used to predict whether the yearly incomes are more or less than 50K (i.e., target='Over-50K'). The PyCaret framework highlights preferred models based on several metrics (Figure 1). Further, the PyCaret framework illustrates the most important features. The classified labels can be used, for example, as slicers in a report (Figure 2) to get a deeper understanding concerning meaningful fields.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	CatBoost Classifier	0.8735	0.9292	0.6577	0.7823	0.7143	0.6339	0.6380	12.2419
1	Light Gradient Boosting Machine	0.8718	0.9275	0.6610	0.7740	0.7127	0.6309	0.6344	0.3644
2	Gradient Boosting Classifier	0.8650	0.9196	0.5919	0.7957	0.6784	0.5954	0.6061	5.5714
3	Extreme Gradient Boosting	0.8624	0.9182	0.5795	0.7938	0.6694	0.5853	0.5972	4.2616
4	Ada Boost Classifier	0.8602	0.9149	0.6172	0.7577	0.6800	0.5918	0.5971	1.7603
5	Random Forest Classifier	0.8492	0.8812	0.5872	0.7338	0.6520	0.5573	0.5631	0.3189
6	Linear Discriminant Analysis	0.8408	0.8938	0.5670	0.7136	0.6312	0.5315	0.5376	0.6360
7	Ridge Classifier	0.8400	0.0000	0.5107	0.7452	0.6054	0.5095	0.5242	0.0775
8	Extra Trees Classifier	0.8333	0.8803	0.6058	0.6707	0.6363	0.5286	0.5299	2.9349
9	Decision Tree Classifier	0.8116	0.7485	0.6269	0.6052	0.6157	0.4909	0.4912	0.3048
10	Logistic Regression	0.7992	0.6063	0.2822	0.7100	0.3968	0.3050	0.3544	0.4646
11	Naive Bayes	0.7944	0.8065	0.3019	0.6602	0.4138	0.3097	0.3448	0.0427
12	K Neighbors Classifier	0.7704	0.6628	0.3046	0.5415	0.3897	0.2618	0.2783	0.4402
13	SVM - Linear Kernel	0.7325	0.0000	0.2614	0.7276	0.2906	0.1900	0.2639	1.7342
14	Quadratic Discriminant Analysis	0.2628	0.5149	0.9940	0.2457	0.3939	0.0126	0.0616	0.3607

```
<catboost.core.CatBoostClassifier at 0x7f8b13a65828>
```

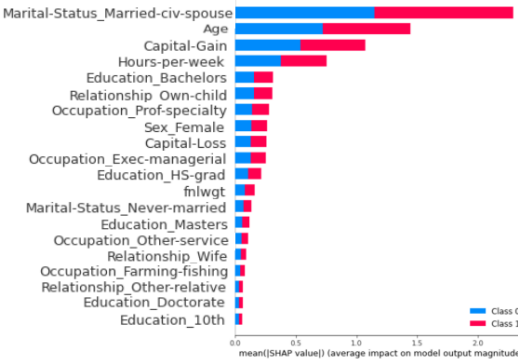

Figure 1. Classifiers and comparable metrics, the yellow coloured of which refer the best ones and the stacked bar illustrate meaningful features. Observe that the underscore ("_") delimit separates the feature name from the feature value when the feature type is categorical.

Figure 2. The classified labels as slicers for the meaningful fields.

We made experiments where Google Collaboratory (Colab) is used as a sandbox for the experiments of the PyCaret framework. Microsoft Power Platform that contains Power BI and Power Apps is used to provide reports, dashboards, and low-code applications. In our experiment, two Python scripts have been run in the Microsoft Power BI query editor – one to build the classifier and another to classify the dataset [3]. The autoML-augmented data pipeline (Figure 3) from raw data to dashboards and low-code apps has been drawn based on the experiments such as classifications of the "Census Income" dataset.

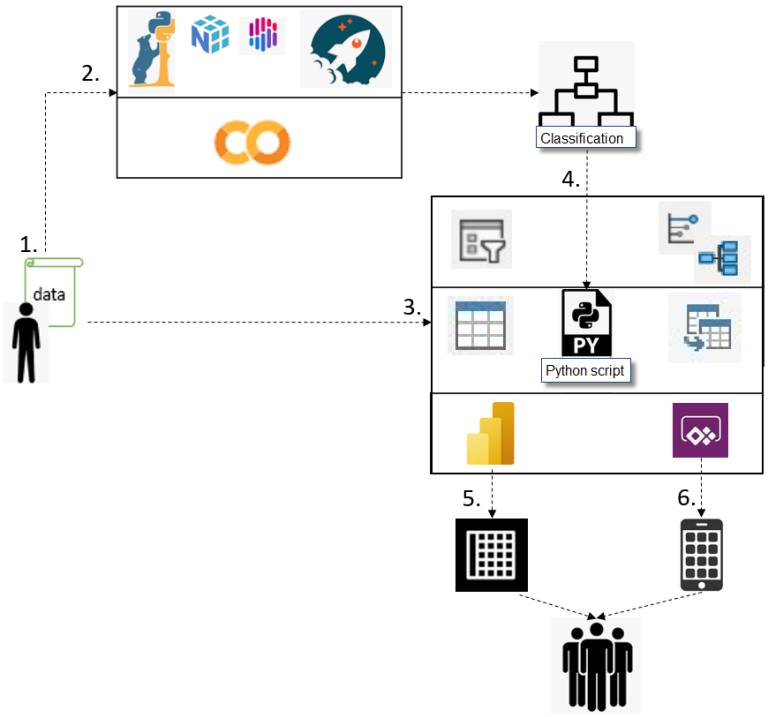

Figure 3. Example of the low-code autoML-augmented data pipeline.

The meanings of the numbered pipeline items are the following:

1) *Curated or unmanifestable data*. When new technologies such as low-code autoML wrappers are evaluated, then the known datasets have been used because they are curated datasets the insights of which have been verified. However, data are usually unmanifestable, and we will figure out both attributes and instances as well as what data will tell us.
2) *Explorable or learnable data*. CO stands for Google Collaboratory (Colab), and it is easy to ingest data to explore or build models from it. Nowadays, plenty of notebooks are used to make experiments concerning new technologies such as the low-code autoML wrappers. Moreover, the Colab-liked environments facilitate setups concerning the development environments because there are pre-installed packages and new installations are easy to make. The icons in our example (Figure 3) are the followings: a 'tree climbing bear' stands for the Pandas library the functions of which serve both data framing and exploratory analysis, a cube with N stands for Numpy, colour bars stand for SHAP (SHapley Additive exPlanations), and a rocket stands for the PyCaret library the functions of which serve both modelling and pipelining. Two other icons, a cube with N for Numpy and colour bars for SHAP (SHapley Additive exPlanations), are examples of the libraries that are used by PyCaret.
3) *Schema-relatable or stand-alone data*. When data seem to be valid for further processing, then some transformations might be made to prepare data either to follow the selected schema (e.g., Microsoft Common Data Model) or to get insights from data without the predefined scheme.
4) *Model files or best model identifiers (id)*. Power BI (a yellow column image) offers possibilities to run Python scripts at the table level. It is possible to run model files that are produced by other tools. However, we have some interoperability problems when we tried to run the model files, and Power BI is not flexible to make experiments and comparisons between the models. Therefore, the best model identifier (id) is used to create a model and enrich the tabular data.
5) *Dashboards and reports*. Power BI (a yellow column image) offers several visualisation possibilities such as quick insights and visual-based analyses as well as even AI visuals the names of which are "Key influencers" and "Decomposition tree". A funnel image stands for a

slicer within the AI visuals to give insights from the data. Further, we formed tiles from the dashboards for the low-code applications.

6) *Low-code applications*. Power Apps (diamonds inside the shape) contains standards entities such as Account that are deployable. We merged data tables into one Excel file in our experiments, and then we create a canvas app from the file. Further, we used tiles of the dashboards in low-code applications and the low-code applications as parts of the reports and dashboards.

The power of the low-level autoML framework is mainly in predefined parameters concerning model setup. The PyCaret framework (i.e., the Python package) requires two mandatory parameters (data and target column) for setup models and the rest of the setup parameters are either ML task (e.g., classification or regression) specific or common ones. We did not report the effects of the changes concerning the setup parameters. However, our guideline of the setup parameters is based on the following two groups where the verb in the list identifier serves as memory support to perceive the main function of the parameter:

- Feature collection
  - *Reduce*. pca = False, pca_method = 'linear' | 'kernel' | 'incremental', pca_components = None
  - *Bin*. bin_numeric_features = None
  - *Group*. group_features = None, group_names = None
  - *Ignore*. ignore_features = None
  - *Permutate*. feature_selection = False, feature_selection_method = = 'classic' | 'boruta', feature_selection_threshold = 0.8
  - *Drop*. remove_multicollinearity = False, multicollinearity_threshold = 0.9
  - *Combine*. feature_interaction = False, interaction_threshold = 0.01, feature_ratio = False
  - *Relate*. polynomial_features = False, polynomial_degree = 2, polynomial_threshold = 0.1; trigonometry_features = False
  - *Detect*. remove_outliers = False, outliers_threshold = 0.05
  - *Cluster*. create_clusters = False, cluster_iter = 20
- Feature values
  - *Impute*. categorical_imputation = 'constant', numeric_imputation = 'mean'
  - *Type*. categorical_features = None, numeric_features = None, date_features = None, ordinal_features = None
  - *Encode*. high_cardinality_features = None, high_cardinality_method = 'frequency' | 'clustering'
  - *Unwant*. combine_rare_levels = False, rare_level_threshold = 0.10
  - *Rescale*. normalize = False, normalize_method = 'zscore' | 'minmax' | 'maxabs' | 'robust'
  - *Reshape*. transformation = False, transformation_method = 'yeo-johnson' | 'quantile'
  - *Retarget*. transform_target = False, transform_target_method = 'box-cox' | 'yeo-johnson'
  - *Replace*. ignore_low_variance = False

4. Conclusions

There are several ML tasks, and they can be grouped in several ways [1,4]. When we compared the autoML frameworks with the repository containing a curated list of ML libraries [4], we realised that PyCaret is categorised to handle "Model Training Orchestration". However, some libraries (a.k.a., frameworks) such as TPOT and ktrain include autoML frameworks, but we cannot use them in the Power BI pipelines.

Business use cases within outcome-centric descriptions of the low-code machine learning libraries (or wrappers such as PyCaret) are essential to increase the awareness of ML-based augmentations such as outliers, clusters, and classes. Lack of understanding of the algorithms and setup parameters are pitfalls when we adapt the functionalities of the wrappers. However, the autoML insights at least raise questions and the awareness of the autoML possibilities, especially when business users can use them in BI tools without pressure concerning details of multiple algorithms.

There are ten related studies of 14 hits that have been found from Google Scholar within the search phrase autoML+low-code. However, these studies do not overlap with ours. There was one low-code library for augmented machine learning called ktrain [5] that have been used to classify texts and images as well as to build an end-to-end QA system. However, some discussion concerning low-code development practices has been highlighted in the context of the sentiment analysis [6]. Low-code cases can be perceived as part of the AI context. Therefore, the lesson learned from AI functionality in enterprise contexts has been presented [7] as well as challenges concerning automated workflows that conduct embedded ML in Business Process Management Software (BPMS) [8]. In general, autoML and low-code platforms are the implementations of AI [9], or AI is used to empower something such as to assess and manage critical issues of performance and stability of the applications [10]. There are several skill requirements concerning AI tasks [11] as well as open-source tools and commercial ones (e.g., Amazon machine learning) [12]. In general, open-source technologies [13] and fairness in ML [14] are meaningful in low-code autoML development.

Nowadays, the same data is preferred to be a ground for different reports, dashboards, and applications. Data engineering and software engineering disciplines are quite near each other. The low-code autoML frameworks (or wrappers) that are usable in the BI tools give a possibility to augment understanding concerning, for example, classes for tabular data. In general, the low-code autoML frameworks are cognitive supportive when they manifest insights from datasets.

References

[1] Mardjan M 2020 *Free and Open Machine Learning Release 1.0.1* Available at https://readthedocs.org/projects/freeandopenmachinelearning/downloads/pdf/latest/
[2] Moez A 2020 *Machine Learning in Power BI using Pycaret. Towards data science.* Available at https://towardsdatascience.com/machine-learning-in-power-bi-using-pycaret-34307f09394a
[3] PyCaret 2020 Available at https://pycaret.readthedocs.io/en/latest/
[4] The Institute for Ethical AI & Machine Learning 2020 Available at https://github.com/EthicalML/awesome-production-machine-learning
[5] Maiya A S 2020 ktrain: A low-code library for augmented machine learning Preprint arXiv:2004.10703
[6] Carvalho A and Harris L 2020 Off-the-shelf technologies for sentiment analysis of social media data: Two empirical studies *AMCIS 2020 Proceedings* pp 1–10
[7] Casati F, Govindarajan K, Jayaraman B, Thakur A, Palapudi S, Karakusoglu F, Chatterjee D 1999 Operating Enterprise AI as a Service *International Conference on Service-Oriented Computing* pp 331–344
[8] Thakur A, Palapudi S, Karakusoglu F, Chatterjee D 2019 Operating Enterprise AI as a Service *Service-Oriented Computing: 17th International Conference, ICSOC* 11895
[9] Taulli T 2019 Implementation of AI *In:Artificial Intelligence Basics* (Apress Berkeley CA) pp 143–159 https://doi.org/10.1007/978-1-4842-5028-0_8
[10] Taulli T 2020 RPA Vendors *The Robotic Process Automation Handbook* pp 217–258
[11] Dibia V, Cox A, Weisz J 2018 Designing for democratisation: introducing novices to artificial intelligence via maker kits Preprint arXiv:1805.10723
[12] Sakhnyuk P A and Sakhnyuk T I 2020 Intellectual Technologies in Digital Transformation. *IOP Conference Series: Materials Science and Engineering* **873** 1
[13] Atwal H 2020 DataOps Technology In:*Practical DataOps* (Apress, Berkeley, CA) pp 215–247 https://doi.org/10.1007/978-1-4842-5104-1_9
[14] Caton S and Haas C 2020 Fairness in machine learning: a survey. *Preprint arXiv:2010.04053*