Automated Data Preprocessing for Machine Learning Based Analyses

Akshay Paranjape
IconPro GmbH
Aachen, Germany
akshay.paranjape@iconpro.com

Praneeth Katta
IconPro GmbH
Aachen, Germany
praneeth.katta@iconpro.com

Markus Ohlenforst
IconPro GmbH
Aachen, Germany
markus.ohlenforst@iconpro.com

***Abstract*—Data preprocessing is crucial for Machine Learning (ML) analysis, as the quality of data can highly influence the model performance. In recent years, we have witnessed numerous literature works for performance enhancement, such as AutoML libraries for tabular datasets, however, the field of data preprocessing has not seen major advancement. AutoML libraries and baseline models like RandomForest are known for their easy-to-use implementation with data-cleaning and categorical encoding as the only required steps. In this paper, we investigate some advanced preprocessing steps such as feature engineering, feature selection, target discretization, and sampling for analyses on tabular datasets. Furthermore, we propose an automated pipeline for these advanced preprocessing steps, which are validated using RandomForest, as well as AutoML libraries. The proposed preprocessing pipeline can also be used for any ML-based algorithms and can be bundled into a Python package. The pipeline also includes a novel sampling method - "Bin-Based sampling" which can be used for general purpose data sampling. The validity of these preprocessing methods has been assessed on OpenML datasets using appropriate metrics such as Kullback-Leibler (KL)-divergence, accuracy-score, and r2-score. Experimental results show significant performance improvement when modeling with baseline models such as RandomForest and marginal improvements when modeling with AutoML libraries.**

***Index Terms*—*AutoML; Preprocessing; Feature Engineering; Feature Generation; Feature selection; Sampling*.**

I. Introduction

Data preprocessing is a crucial step in Machine Learning (ML) as the quality of data can have a significant influence on its performance. Preprocessing is performed to prepare the compatible dataset for analysis as well as to improve the performance of the ML model. Preprocessing steps can be roughly categorized into two types: model compatible preprocessing (Type 1) and quality enhancement preprocessing (Type 2). A common example of model compatible preprocessing step is the encoding of string values to either Label Encoded values or One-Hot Encoded values based on the model requirements. Preprocessing steps like data cleaning and missing value imputation fall into the category of Type 1, while other generic preprocessing steps like standardization and normalization, and cyclic transformation fall into Type 2 category. The focus of this paper is Type 2 preprocessing for supervised learning of tabular datasets.

In recent years, the ML field for tabular datasets has been heavily researched for performance enhancement of ML-based models, especially automated Machine Learning (AutoML) [6]- [8]. However only a few papers [1] [2] have investigated advanced Type 2 preprocessing steps (mainly feature generation). The validation of these preprocessing steps together with AutoML libraries has not been studied yet. We have considered three relevant AutoML libraries, namely AutoGluon [8], AutoSklearn [6], and H2O [7]. The different preprocessing steps supported by these AutoML libraries are summarized in Table I. It can be inferred from Table I that advanced preprocessing steps like feature engineering and feature selection have not been implemented in these AutoML libraries. In this paper, we first investigate some advanced Type 2 preprocessing steps and later propose an automated preprocessing pipeline based on our research. A validation study of the proposed pipeline is conducted on both the baseline model and AutoML libraries.

The automated preprocessing pipeline is designed with the main objective of automatically generating new features from existing input features to improve the performance metric. If the dataset size is large, feature engineering can take a longer computational time. Therefore, required research in the sampling field is evident, for which we propose the Bin-Based sampling method as an alternative to Random Sampling. Before proceeding with feature engineering, unnecessary, irrelevant, and highly insignificant features are removed since these features have neutral or significantly low information gain for the target variable. These three techniques, viz. feature engineering, feature selection, and sampling are the main aspects of this paper. Therefore, in Section II, related work for these three techniques is briefly presented. Further, in Section III, methodology is presented. In Section IV, experiments and the results obtained are tabulated. We conclude the work in Section V.

II. Related Work

A. Feature Engineering

Feature engineering is the process of generating new features with the help of domain knowledge. The construction of novel features for the enhancement of predictive learning is time-intensive and often requires field expertise. With the appropriate addition of features, predictive models can show significant performance improvement. Cognitio by Khurana et al. [1] demonstrated a novel method for automated feature engineering in supervised learning. Cognito performs row-wise transforms over instances for all valid features, each

 ISBN: 978-1-61208-976-8

TABLE I
PREPROCESSING STEPS INCLUDED IN DIFFERENT AUTOML SOLUTIONS

Name	AutoSklearn	AutoKeras	TPOT	AutoGluon	H2O
Balancing	yes	no	no	yes	yes
Categorical encoding	yes	yes	yes	yes	yes
Imputation	yes	yes	no	yes	yes
Standardization/Normalization	yes	yes	no	yes	yes
Others	Densifier, PCA, minority coalescence, select percentile	Data augmentation	Feature selector	Introduce "unknown category"	None

producing a new column or columns. The number of possible transformations is an unbounded space considering various combination of features. These function transforms could be unary, binary, or multiple transforms [1]. As the number of transforms can increase exponentially based on the number of input columns, the pruning step is included by Cognito for feature selection to ensure a manageable size of the dataset.

Katz et al. introduced a framework ExploreKit [2] for automated feature generation. Katz et al. demonstrate new feature findings by using unary operators such as inverse, addition, multiplication, division, etc., as well as higher-order operators. The huge number of features generated in ExploreKit are pruned and validated using a Ranking Model. A two-step approach is proposed by ExploreKit where the generated features in the first step are ranked based on meta-features in the second step.

Galhotra et al. [3] focus on an automated method to utilize domain structured knowledge to perform feature addition. They further developed a tool KAFE (Knowledge Aided Feature Engineering) to attain knowledge about similar analyses from 25 million tables available on the internet. Hoag et al. presented a neural network approach to generate new features from relational databases [4]. They use a set of Recurrent Neural Networks (RNNs) that takes as input a sequence of vectors and outputs a vector (with new generated features). The Data Science Machine [5] developed a deep feature engineering algorithm for relational databases and cannot be generalized to tabular datasets.

B. Feature selection

In contrast to feature engineering where new features are generated from the existing ones, feature selection implies selecting useful features from the available set of input features, i.e., a subset of features. Tereno et al. [10] consider various search strategies for feature selection namely heuristic and probabilistic. They illustrate the importance of removing unnecessary features based on class separability measures. The elimination of non-relevant features or features with negligible importance can significantly reduce computation time and resources [10]. Aliferis et al. [11] introduced an algorithmic framework to learn local casual structure for target structure that is later used to select features. The most popular approach for feature selection includes correlation, Bayesian error rate, information gain, entropy measures, etc., [12]. Elssied et al. [13] demonstrate the use of a one-way Analysis of Variance (ANOVA) F-test for feature selection in the context of email spam classification.

C. Sampling

ML algorithms should be trained on a complete dataset because a higher amount of data can improve performance. But a sample set can help get a quick overview of data quality as well as determine its characteristics. The popular approach of sampling is Random sampling with or without replacement [15]. The literature for sampling dates back to 1980 when Cochran [14] first introduced the concept of stratified sampling. Stratified sampling divides samples into homogeneous subgroups and later the data is randomly sampled from these subgroups. Rojas et al. [15] in their survey concluded that the majority of data scientists use random sampling, stratified sampling, or sampling by hand. Section III-B elaborates on the stratified sampling technique in detail.

III. METHODOLOGY

In this paper, we present an automated preprocessing pipeline that includes advanced preprocessing methods which are generally not available in AutoML libraries. The aim of this research is to develop an automated pipeline that can improve the performance of predictive modeling on tabular datasets. The proposed pipeline can be used with any ML algorithms as well as for AutoML libraries. The novelty aspects of this paper are as mentioned below:

- Hybrid Feature Engineering (HFE) method
- Generalized automated preprocessing pipeline
- Sampling technique

The proposed preprocessing pipeline is validated by analyzing its implementation on OpenML datasets [18]. This section consists of four preprocessing steps each representing an element of the proposed pipeline, namely feature selection, sampling, target discretization, and feature engineering. These steps are described in detail below with their pseudo algorithms.

A. Feature selection

Feature selection implies selecting important features from the list of input features or in other words eliminating less significant features. As mentioned in Section II, a popular approach for feature selection is the correlation coefficient. In this paper, we present a mixture of variance and correlation analysis for feature selection. Inspired by [10], feature selection has been categorized into three parts:

- removal of redundant features

 ISBN: 978-1-61208-976-8

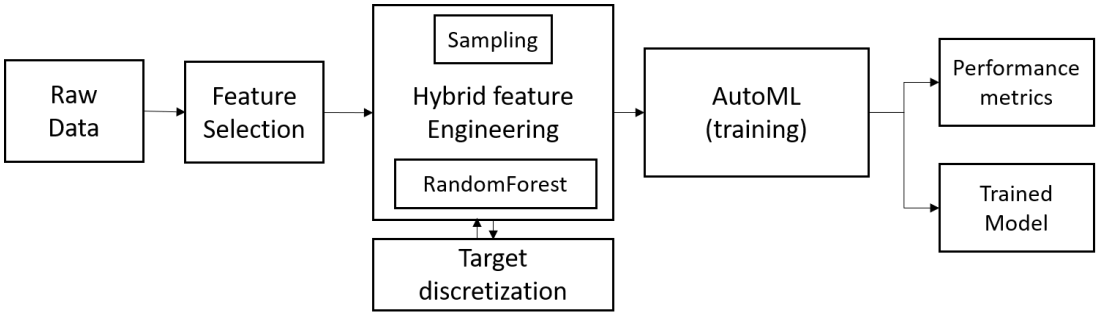

Fig. 1. Block diagram for preprocessing pipeline together with the integration of AutoML module

- removal of highly correlated features
- elimination of insignificant features with one-way ANOVA F-test for classification and correlation coefficient for regression

1) Redundant features: For each categorical input feature, the number of categories is measured. Two extreme cases are eliminated based on the number of unique values:

- Constant feature: single category features, eg., Machine No., Nationality, etc.,
- Feature with the number of categories equal to the number of samples, eg., Name, Email Id, etc.,

In both cases, the information gain is zero and hence the features are removed.

2) Correlation Threshold: In this step, a one-dimensional correlation among the input features is calculated. Pearson's correlation [19] coefficient as shown in (1) is calculated for all input features. Features are merged based on their correlation coefficient. Features with a correlation coefficient close to 1 would provide similar information for modeling and can thus be removed to save the computational power. Equation 1 shows the correlation between two features denoted as x and y.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

where, x_i, y_i are the i^{th} elements for features x and y.

3) Analysis of Variance: After the elimination of insignificant features from steps 1 and 2, the rest of the features are analyzed based on a one-way ANOVA test for a classification task. Each feature is divided into subgroups corresponding to its target value. The one-way analysis of variance is carried out with these subgroups to find the F-value, as described below.

$$\begin{aligned} SSR &= \sum_{i=i}^{k} n_i(\bar{X}_o - \bar{X}_i)^2 \\ SSE &= \sum_{i=i}^{k}\sum_{j}^{n_i}(\bar{X}_i - X_{ij})^2 \\ SST &= SSR + SSE \\ df_r &= k - 1 \\ df_e &= \sum_{i=1}^{k} n_i - k \\ MSR &= \frac{SSR}{df_r} \\ MSE &= \frac{SSE}{df_e} \\ F\_value &= \frac{MSR}{MSE} \end{aligned} \tag{2}$$

where k is the number of groups, n_i is the number of samples in group i, $\bar{X}_o$ is the global mean of all samples of the features and $\bar{X}_i$ is the mean of samples in subgroup i, df_r is regression degree of freedom and df_e is error degree of freedom. MSR is regression mean square, MSE is error mean square, SSR represents the regression sum of squares, SSE is the error sum of squares and SST is the total sum of squares.

For the regression datasets, F-value is calculated via a univariate linear regression test. Once the F-values are available for all features, they are ranked based on $p\text{-}values$ as suggested by Charles Poole [9]. Experimentally, it is found that a relative comparison of $p\text{-}values$ is a better evaluation of the significance of the feature. The features are arranged in decreasing order of $\text{-}log(p\text{-}values)$ and the maximum drop of $\text{-}log(p\text{-}value)$ over the features is computed by calculating their second-order gradients. The maximum drop in the $\text{-}log(p\text{-}value)$ is considered a threshold for that particular dataset. Features with $\text{-}log(p\text{-}value)$ less than this threshold are considered insignificant. In [9], a $p\text{-}value$ of 0.05 is considered an appropriate threshold value for selecting the insignificant features. Experimentally, it is found that a few datasets have many features with $p\text{-}values > 0.05$. To overcome this situation where many features could be removed because of a static threshold, the second-order gradient method

 ISBN: 978-1-61208-976-8

is used. By using a second-order gradient, the maximum drop of the $-log(p\text{-}value)$ can be detected. The advantage of this method is the threshold for the $p\text{-}value$ and it is set dynamically. The least important features are identified with the threshold as mentioned below:

$$threshold = \arg\max \left\{ \nabla \left(- \log \left(p\_values_{sorted} \right) \right) \right\} \quad (3)$$
$$features = features_{(p\_value < threshold)}$$

Based on the above three methods, the most important features are selected for later processing.

B. Sampling

The quality of a sampling method can be accessed based on the divergence of a sampled dataset with the original distribution. As mentioned in Section II, the most widely used sampling techniques are random sampling, stratified sampling, and hand-pick sampling [15]. Stratified sampling is the basis for our proposed approach, Bin-Based sampling.

1) Stratified Sampling: Stratified sampling [14] by Cochran is a well-known sampling technique that closely resembles the original distribution statistics. In stratified sampling, a population is divided into sub-populations *strata* followed by random sampling from these *strata*. The proportionate allocation of sampling steps for the creation of strata is summarized in Algorithm 1. These sub-populations are formed based on the nested grouping of columns. One of the disadvantages of stratified sampling is its non-realistic runtime which makes it difficult for real-time applications. Therefore, we propose a new sampling technique, Bin-Based sampling. Two major advantages of bin-based sampling are:

- faster than stratified sampling
- preserves original distribution better than random sampling

2) Bin-based sampling: The motivation behind bin-based sampling is to reduce the time complexity while maintaining the original population statistics. To achieve this, input features are divided into different bins based on their distribution. After the binning process, random samples are drawn from each bin for every feature. A union set of sample collection from every feature is now the bin-based-sampled population. Figure 2 shows the histogram of a feature before and after sampling. We can see that the distribution is preserved with Bin-Based sampling and it simplifies the probability of choosing a sample by lowering the number of possibilities. The joint conditional probabilities in stratified sampling are simplified into the conditional probabilities of each feature, as mentioned below.

$$P_{random}(s) = \frac{1}{N}$$
$$P_{bin-based}(s) = \frac{P(s|b_i)P(b_i)}{P(b_i|s)} \quad (4)$$
$$P_{bin-based}(s) = P(s|b_i) = \frac{1}{size(b_i)}$$

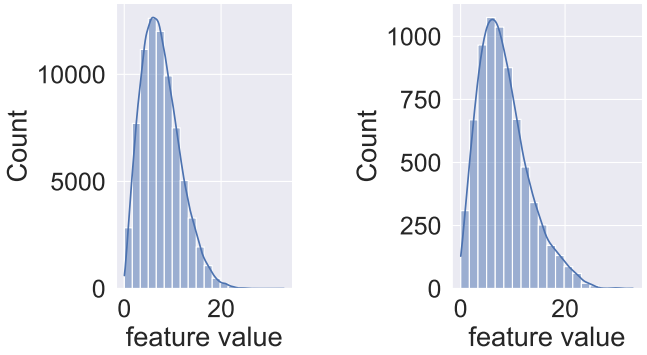

Fig. 2. Bin-based Sampling: Comparison of distribution for a feature before and after Bin-based sampling. (left: original distribution; right: sampled distribution)

where N denotes the number of samples, s is the sample point and b_i denotes the i^{th} bin. The pseudo-algorithm for Bin-Based sampling is provided in Algorithm 2.

Algorithm 1: Stratified sampling

```
Function Subsets(Data):
    for category/bin in feature_i from Data do
        if num_features in Data > 1 then
            Strata ← Subsets(Data.drop(feature_i))
        else
            Strata ← category/bin
    return Strata
Strata ← Subsets(Data)
for Stratum in Strata do
    stratumSamples ← RandomSample(stratum)
StratifiedSample ← Concat(stratumSamples)
Result: StratifiedSample
```

Algorithm 2: Binbased Sampling

```
for Feature_i in Features do
    if Feature_i is Categorical then
        for Category in Categories_Feature_i do
            Sample ← RandomSample(Category)
        end
        featureSample ← Concat(Sample)
    else
        Bins ← Discretize(Feature_i)
        for Bin in Bins do
            Sample ← RandomSample(Bin)
        end
        featureSample ← Concat(Sample)
    end
end
BinbasedSample ← Concat(featureSample)
Result: Binbased Sample
```

3) Sampling size: An optimal sampling size should ensure that the information loss is minimum. Cochran [14] has stated

 ISBN: 978-1-61208-976-8

an optimal size for sampled population based on the size of the population.

$$n_o = \frac{Z^2 p(1-p)}{e^2} \tag{5}$$

where e is desired level of precision (i.e., margin or error), p is the estimated proportion of the population, and Z is the z-score distribution value, defaulting to 0.475 for 95% distribution. For Bin-Based sampling, a sample size with a z-score distribution value of 0.475 is chosen, as suggested by Cochran [14].

C. Target Discretization

With the help of target discretization, a numerical output feature can be converted into categorical values, thereby transforming a regression task into a classification task. Based on the baseline regression model, a regression task will be converted to a classification task if the regression r2-score metric value is significantly low or unacceptable. The prediction of categorical values has less degree of freedom than the prediction of numerical values. Taking advantage of this fact, a classification analysis with AutoML might give a reasonable classification accuracy rather than concluding the datasets as not suitable for analysis. Here, each data point in the continuous domain is converted into a discrete class domain. Different types of target discretization methods can be considered based on domain expertise. As an automated solution, we have considered the discretization of the target variable based on its z-score values.

D. Feature Engineering

Feature engineering is the process of generating new features or transforming features from the existing set of features using domain knowledge. Feature engineering is performed to leverage the performance of the ML model. Since domain knowledge is not always available, we suggest an automated way to feature engineering - Hybrid Feature engineering, inspired by Cognito [1] and ExploreKit [2]. They are briefly described below.

1) Cognito: As mentioned in Section II, Cognito generates new features by performing unary and binary operations on the input features. As the number of input features and transform operators increases, the number of newly generated features increases exponentially in the order of $O(f.d^{k+1})$ where f is the number of features, d is the number of combinations and k is the number of transforms. For a feature D and transforms τ_1 and τ_2, $\tau_1(D)$, $\tau_2(D)$, $\tau_1\tau_2(D)$ are generated. These generated features have to be pruned to reduce the complexity of the problem. Feature selection is done on the generated features using information gain as a proxy measure of accuracy.

2) ExploreKit: ExploreKit [2] takes a similar approach towards feature engineering. Together with the unary and binary operators, ExploreKit considers higher-order operators. Among all the generated features, each feature is added to the dataset, and the rank of the feature is determined with the Ranking Model. The Ranking Model ranks the newly generated features based on either accuracy score (classification) or r2-score (regression). Low-rank features are removed and higher rank features are added to the original dataset so that more information can be extracted.

3) Hybrid Feature Engineering: Inspired by Cognito and ExploreKit, we propose a new Hybrid Feature engineering approach for feature engineering, which can be implemented in real-time with a modified ranking algorithm and additional usage of the Bin-Based sampling method. In Hybrid feature engineering, new features are generated with a single transformation on a feature or features at a time. This transformation are either a unary or a binary operator. For a feature D and unary transforms τ, $\tau(D)$ feature is generated. For features D_1, D_2 and binary transform τ, $\tau(D_1, D_2)$ feature is generated. These features are subjected to feature selection as described in Section III-A instead of using the Ranking Model directly. After the elimination of features with feature selection, most of the insignificant features are eliminated and we are left with a relatively less amount of features. These features are then ranked with the Ranking Model. High-ranked features are selected and added to the original dataset.

Consider $F = f_1, f_2, ..., f_n$ is a set of features, $T = \tau_1, \tau_2, ..., \tau_n$ is a set of transform functions and $F' = FXT$ denotes the set of new generated features. With the help of the Feature Selection technique, the most significant features, $I \subset FXT$ can be selected from the generated features. After Feature Selection, the set I is fine-tuned with Ranking Model R. (c.f. Algorithm 3, Algorithm 4)

$$I \leftarrow R(FXT) \tag{6}$$

Algorithm 3: Hybrid Feature Engineering

```
for Operator in Unary/Binary Operators do
    for Feature in Numerical-Features do
        NewFeatures ← Operator(Feature)
    end
    allNewFeatures_operator ←
     FeatureSelection(NewFeatures)
end
allNewFeatures = RankingModel(allNewFeatures)
Result: Generated Features
```

Algorithm 4: Ranking Model

```
thresholdf = baseModel(Dataset)
for i = 0 to allNewFeatures do
    Dataset.append(allNewFeatures[i])
    RankedFeatures = []
    featureScore = baseModel(Dataset)
    if featureScore ≤ thresholdf then
        continue
    else
        RankedFeatures.append(allNewFeatures[i])
    end
    return RankedFeatures
end
Result: Ranked features
```

IV. Experiments and Results

This section describes various experiments that are conducted together and their validation results. The goal of this validation study is to ensure that the proposed methods have a positive influence on the datasets. Validation is done on OpenML datasets [18]. RandomForest model is used as a baseline model. The proposed auto-preprocessing pipeline is also benchmarked against top-performing AutoML libraries [17], namely AutoSklearn, Autogluon, and H2O. The results are consistently compared with (w) and without (w/o) the auto-preprocessing libraries. Figure 1 illustrates the flow of the preprocessing pipeline.

A. Experimental Setup

For an individual experiment, a RandomForest model with 30 estimators was chosen for both classification and regression tasks. K-fold cross-validation was used to benchmark the results. All experiments are conducted on a Linux Virtual Machine with 16 GB of RAM and 4 cores. Experiments are conducted without using `dask` multiprocessing for AutoML libraries.

B. Datasets

OpenML datasets were used for benchmarking the results [6]. A comparative study shows that Hybrid Feature Engineering (HFE) performs better than modeling without HFE for around 35% of cases, no change in performance was observed for the rest of the datasets. This trend can be observed for both classification and regression tasks. It has to be noted that an increase in performance can be expected only if we have new features after the pruning method as explained in Section III. Only 35% of the OpenML datasets reported new features after the pruning step. Results of these datasets are provided in Tables II - VIII.

C. Feature selection

A comparative study was conducted to compare the performance with and without feature selection. It can be seen that for all datasets performance remains the same after the elimination of less significant features, as described in Section III-A. Reduction in the training time is not very significant for the baseline model, as the model has only 30 estimators and the size of the datasets is comparatively less. For MNIST, a total of 64 features out of 784 features were eliminated maintaining the same accuracy. Feature selection is the first step in the preprocessing pipeline. The results of the analysis are summarized in Tables II and III.

D. Bin-Based sampling

Bin-based sampling is used to reduce computation time for the baseline model used for feature engineering. Stratified sampling is known to sample a good representation from a population but at the cost of computation time with a time complexity O(n^2) where *n* is the number of input features. Random Sampling, on the other hand, has the complexity of O(1) and Bin-Based sampling, as explained in Section III-B O($\frac{n}{3}$). The sampling technique is validated using Kullback-Leibler (KL) divergence (7) considering the original distribution as the reference distribution [16].

$$D_{KL}(P\|Q) = \int_{-\inf}^{\inf} P(x) log\left(\frac{P(x)}{Q(x)}\right) dx \qquad (7)$$

where $P(x)$ is the original distribution and $Q(x)$ is the sampled distribution. We can observe that KL-divergence for Bin-Based sampling is comparatively much lower than random sampling. Stratified sampling has the least KL-divergence, but 10 times the computation time as can be inferred from Table IV. These experiments are conducted on a Linux VM with 256 GB of RAM. Experiments also revealed that Bin-Based sampling failed to perform well for the datasets of smaller sizes. The reason for this is that performing a binning operation and extracting random samples from each bin might cause a loss of information. Therefore, for such datasets of small size, sampling is not useful.

E. Hybrid Feature Engineering

Significant improvement in performance is achieved for a few OpenML datasets. These are summarized in Tables V and VI. A performance improvement for 35% of datasets is observed when testing over 50+ OpenML datasets. All tests are performed on a cross-validation split. It should be noted that performance improvement is achieved only if new significant features are generated after the pruning step. The performance with HFE is higher or similar, in no case did we encounter a decrease in performance.

F. Overall Pipeline

An auto preprocessing pipeline, as shown in Figure 1, is used together with AutoML libraries mainly AutoGluon, AutoSklearn, and H2O for benchmarking. Significant improvements compared to the baseline model are not achieved with AutoML libraries for all datasets shown in Tables V and VI. The results with AutoML libraries are shown in Tables VII and VIII. The stopping criteria for training of AutoML-libraries are set with the runtime limit. The benchmarking for AutoML libraries is done on a single core. The runtime across all AutoML libraries is set to 10 minutes and the results are 4-fold cross-validated. Overall, the combination of auto-preprocessing and AutoML libraries performed better or was similar to "only AutoML libraries". As mentioned in Table I, AutoML libraries do not consider feature engineering in their preprocessing step.

V. Conclusion

A significant amount of recent work in the field of automated-Machine Learning is being done, but the same has not been the case for data preprocessing. This paper reviews and suggests some advanced preprocessing steps that can either be used individually or combined as a pipeline. Although performance improvements cannot be ensured for all datasets, datasets that have inter-feature dependency can be observed to perform better. For example, length and width

TABLE II
VALIDATION OF FEATURE SELECTION TECHNIQUE FOR CLASSIFICATION TASK

OpenML dataset	Accuracy w/o feature selection	Accuracy with feature selection	Number of features removed	Difference in accuracy
11	0.607	0.607	3	0
54	0.753	0.753	0	0
188	0.579	0.579	0	0
333	0.908	0.908	3	0
335	0.977	0.977	2	0
470	0.661	0.661	4	0
1459	0.588	0.588	0	0
1461	0.692	0.692	2	0
23381	0.560	0.560	5	0
amazon-employee-access	0.943	0.943	3	0
australian	0.857	0.857	2	0
bank-marketing	0.692	0.692	2	0
credit-g	0.761	0.761	2	0
sylvine	0.941	0.941	7	0

TABLE III
VALIDATION OF FEATURE SELECTION TECHNIQUE FOR REGRESSION TASK

OpenML dataset	r2-score w/o feature selection	r2-score with feature selection	Number of features removed	Difference in r2-score
537	0.484	0.484	0	0
495	0.616	0.616	5	0
344	0.999	0.999	2	0
215	0.948	0.948	1	0
189	0.579	0.579	0	0
507	0.391	0.390	0	0

TABLE IV
SAMPLING COMPARISON ON OPENML DATASETS CALCULATED OVER 100 TRIALS

OpenML dataset	Mean of KL-divergence Bin-Based sampling	Mean of KL-divergence Stratified sampling	Mean of KL-divergence Random sampling	Time (in sec) Bin-Based sampling	Time (in sec) Stratified sampling
183	0.017	0.173	0.057	0.359	5.230
223	0.067	0.079	0	0.273	7.600
287	0.076	0.356	0.027	0.399	4.807
307	0.0	0.097	0.006	0.214	7.572
528	0.0	0.0215	0.0	0.054	0.489
537	0.190	0.886	1.160	2.052	133.939
550	0.0	0.011	0.004	0.302	0.738
Amazon-employee-access	0.019	0.460	0.753	0.466	2.112
Blood-transfusion	0.065	0.002	0.001	0.062	0.069
Phoneme	0.0	0.168	0.143	0.580	1.383

of a workpiece can be combined to form a new feature "area of the workpiece", which can have a significant impact on the ML-based model. The proposed method does it without domain knowledge in an automated manner. This paper also introduces a new sampling method that can be used for general application as well as for ML-based modeling. We used the Bin-Based sampling method during the Feature Engineering step to generate new features and select them using a Ranking Model. Usage of sampled data for Feature Engineering has significantly reduced the preprocessing time. It can be concluded that a significant performance improvement of around 4-7% is observed for the analysis conducted with the baseline model on OpenML datasets. For the same set of datasets, a marginal improvement was observed for analysis with the AutoML libraries. The proposed pipeline is currently not parallelized. Parallelization can significantly reduce the time for feature engineering and this we would like to focus on in our future work.

REFERENCES

[1] U. Khurana, D. Turaga, H. Samulowitz and S. Parthasrathy, "Cognito: Automated Feature Engineering for Supervised Learning." *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 2016: pp. 1304-1307.

[2] G. Katz, E. C. R. Shin and D. Song, "ExploreKit: Automatic Feature Generation and Selection," *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 979-984, doi: 10.1109/ICDM.2016.0123.

[3] S. Galhotra, U. Khurana, O. Hassanzadeh, K. Srinivas, H. Samulowitz and M. Qi, "Automated Feature Enhancement for Predictive Modeling using External Knowledge," *2019 International Conference on Data Mining Workshops (ICDMW)*, 2019, pp. 1094-1097, doi: 10.1109/ICDMW.2019.00161.

[4] H. T. Lam, T. N. Minh, M. Sinn, B. Buesser, and M. Wistuba, "Neural Feature Learning From Relational Database." *arXiv: Artificial Intelligence*, 2018.

TABLE V
HYBRID FEATURE ENGINEERING FOR CLASSIFICATION DATASETS WITH BASELINE MODEL

OpenML datasets	Number of features	Number of classes	Accuracy before	Accuracy after	Percentage Gain	New features
188	14	5	0.466	**0.506**	8.386	2
1461	7	2	0.692	**0.718**	4.748	2
1459	7	10	0.588	**0.635**	7.952	1
54	18	4	0.753	**0.759**	0.786	2

TABLE VI
HYBRID FEATURE ENGINEERING FOR REGRESSION DATASETS WITH BASELINE MODEL

OpenML datasets	Number of features	r2-score before	r2-score after	Percentage Gain	New features
189	8	0.579	**0.615**	6.227	1
507	6	0.390	**0.411**	5.361	1
537	8	0.484	**0.494**	2.000	1
495	13	0.616	**0.632**	2.700	2

TABLE VII
OVERALL PREPROCESSING PIPELINE PERFORMANCE COMPARISON WITH AUTOML LIBRARIES (CLASSIFICATION - ACCURACY)

OpenML datasets	AutoGluon		AutoSklearn		H2O		RandomForest	
	w/o	w	w/o	w	w/o	w	w/o	w
188	0.728	0.726	0.674	0.696	0.717	0.739	0.466	0.506
1461	0.914	0.914	0.906	0.906	0.907	0.907	0.692	0.718
1459	0.815	0.82	0.919	0.919	0.922	0.927	0.588	0.635
54	0.858	0.857	0.839	0.839	0.707	0.708	0.753	0.759

TABLE VIII
OVERALL PREPROCESSING PIPELINE PERFORMANCE COMPARISON WITH AUTOML LIBRARIES (REGRESSION - R2-SCORE)

OpenML datasets	AutoGluon		AutoSklearn		H2O		RandomForest	
	w/o	w	w/o	w	w/o	w	w/o	w
189	0.913	0.913	0.902	0.903	0.913	0.918	0.579	0.615
507	0.731	0.741	0.753	0.753	0.762	0.761	0.390	0.411
537	0.815	0.821	0.862	0.865	0.861	0.869	0.484	0.494
495	0.496	0.495	0.494	0.494	0.441	0.442	0.616	0.632

[5] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors." *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015: pp. 1-10.

[6] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum and F. Hutter, "Efficient and Robust Automated Machine Learning." *NIPS 2015*.

[7] E. LeDell and S. Poirier, "H2O AutoML: Scalable Automatic Machine Learning". *7th ICML Workshop on Automated Machine Learning (AutoML)*, July 2020.

[8] N. Erickson et al., "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data." *ArXiv abs/2003.06505*, 2020.

[9] C. Poole, "Low P-values or narrow confidence intervals: which are more durable?" *Epidemiology* 12, 2001: pp. 291-294.

[10] T. Terano, H Liu and L. P. Arbee, "Chen Knowledge Discovery and Data Mining." *4th Pacific-Asia Conference, PAKDD 2000*, Kyoto Japan, 2000.

[11] C. F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X. D. Koutsoukos, "Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification Part I: Algorithms and Empirical Evaluation." *J. Mach. Learn. Res. 11*, 2010, pp. 171–234.

[12] C. Jie, L. Jiawei, W. Shulin and Y. Sheng, "Feature selection in machine learning: A new perspective." *Neurocomputing* 300, 2018: pp. 70-79.

[13] N. O. F. Elssied, O. Ibrahim and A. H. Osman, "A Novel Feature Selection Based on One-Way ANOVA F-Test for E-Mail Spam Classification." *Research Journal of Applied Sciences, Engineering and Technology 7*, 2014: pp. 625-638.

[14] Cochran William, Sampling Techniques, 3rd edition, *John Wiley and Sons*, 1978.

[15] J. A. R. Rojas, M. B. Kery, S Rosenthal, and A. Dey, "Sampling techniques to improve big data exploration." 2017 *IEEE 7th Symposium on Large Data Analysis and Visualization (LDAV)*, 2017: pp. 26-35.

[16] J. M. James, "Kullback-Leibler Divergence". *International Encyclopedia of Statistical Science*, 2011.

[17] P. Gijsbers, E. LeDell, J. K. Thomas, S. Poirier, B. Bischl, and J. Vanschoren, "An Open Source AutoML Benchmark." *ArXiv abs/1907.00909*, 2019.

[18] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: networked science in machine learning." *SIGKDD Explorations 15(2)*, 2013, pp. 49-60.

[19] Kirch Wilhelm, "Pearson's Correlation Coefficient." *Encyclopedia of Public Health*, 2008. Springer, Dordrecht. https://doi.org/10.1007/978-1-4020-5614-7_2569 .

 ISBN: 978-1-61208-976-8