IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 June 2022, accepted 19 July 2022, date of publication 16 August 2022, date of current version 14 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3198662

RESEARCH ARTICLE

Auto-Prep: Efficient and Automated Data Preprocessing Pipeline

MEHWISH BILAL[1], GHULAM ALI[1], MUHAMMAD WASEEM IQBAL[2], MUHAMMAD ANWAR[3], MUHAMMAD SHERAZ ARSHAD MALIK[4], AND RABIAH ABDUL KADIR[5]

[1]Department of Computer Science, University of Okara, Okara 56300, Pakistan
[2]Department of Software Engineering, Superior University, Lahore 54000, Pakistan
[3]Division of Science and Technology, Department of Information Sciences, University of Education, Lahore 54000, Pakistan
[4]Department of Information Technology, Government College University Faisalabad, Faisalabad 38000, Pakistan
[5]Institute of IR4.0, Universiti Kebangsaan Malaysia (UKM), Bangi, Selangor 43600, Malaysia

Corresponding author: Rabiah Abdul Kadir (rabiahivi@ukm.edu.my)

This work was supported by Universiti Kebangsaan Malaysia with Research Grant Scheme under Project GPK-4IR-2020-020.

ABSTRACT Data preprocessing is crucial in the Machine Learning pipeline because the models' learning ability directly affects the quality of data and the underlying information acquired from this stage. Nevertheless, surprisingly, there are many alternatives for each transformation task, which makes an inexperienced user overwhelmed. A simple Python-based Auto-preprocessing architecture for Automated Machine Learning is developed to offer automated, interactive, and data-driven support to help the users perform data preprocessing tasks efficiently. The suggested method provides valuable insights into a dataset and can handle standard data preprocessing tasks adeptly. Initially, it detects the data problem and presents it to the end-user using compelling visualizations. Then, it recommends the most effective data cleaning and preparation method to the user after evaluating the state-of-the-art candidate techniques. For evaluation, the proposed architecture is employed on ten different and diverse datasets for automatic data preprocessing before passing it to an ML algorithm. The results are then compared with the results generated by the same ML algorithm but implemented on manually preprocessed data. The results have shown that not only did this approach make the whole process uncomplicated and facile, but it was also able to improve the performance of the model significantly.

INDEX TERMS Automated machine learning, data preprocessing, feature engineering.

I. INTRODUCTION

Machine learning models have been applied in different areas in recent years and managed to solve many complicated tasks [1]. Remarkably, all the models were designed manually by specialists by the trial-and-error method, which implies that even a specialist needs considerable time and resources to develop a well-performing model [2]. As a result, the demand for machine learning experts in the market has increased. Nevertheless, to lower these exorbitant development costs, an innovative concept of automating the complete ML pipeline has emerged, i.e., automated machine learning (AutoML). So, the purpose of AutoML is to decrease the data scientists' demand and enable the various domain specialists to develop ML applications automatically without any prior ML knowledge [3]. The core processes of AutoML include data-preprocessing, feature-engineering, Model-generation, and Models-evaluation [4].

Indeed, the crucial stage of the ML pipeline is the data preprocessing phase. Data preprocessing is imperative to such an extent that mostly 50-80 percent of analysis time is spent on it [5]. The explanation behind this is that an aptly preprocessed dataset may generate better results [6]. One can employ the best learning model, but the model might perform imprecisely if the dataset is not correctly prepared (e.g., attain low accuracy).

Now, if the data preprocessing is so significant and should be performed by unskilled users, then, in that case, a method

The associate editor coordinating the review of this manuscript and approving it for publication was N. Ramesh Babu.

must be found that makes the preprocessing effortless, i.e., assisting the user in efficiently implementing this task. At present, no automated technique is mature enough to handle this task accurately and requests significant human intervention. Although present techniques permit users to implement different preprocessing algorithms, they do not consider which one will be more suitable for the given dataset to retrieve the maximum information, and consequently to achieve better accuracy. Hence, the preprocessing stage in current automated techniques usually is a simple transformation that does not aim to improve the models' performance, but their primary purpose is to convert the dataset into the algorithm's acceptable format [7].

Currently, for a Structured dataset, the detection of attributes data types is very challenging and relies on humans to manually specify them [3]. After identifying data types, the most common data-preprocessing phases comprise missing values, Categorial attributes encoding, features scaling, and features reduction. Against all the required preprocessing stages, many alternative methods are available.

In this research, we are concerned with the automation of all the above-mentioned tasks of data preprocessing. A simple Python-based approach — Auto-Prep — is developed to offer automated, interactive, and data-driven support to help the user perform data preprocessing tasks efficiently. With the file path of the dataset and target column name as the inputs, the approach can show valuable information about the given dataset, for example, the presence of duplicates, kinds of features present, and the data types of each present feature. The approach is also competent at dealing with common data problems. It first detects the data problems and presents them to the end-user using effective visualization techniques. Next, it selects the most suitable technique and recommend it to user to clean data efficiently. To make it clearer, we summarize the functionalities of the approach as follows:

1. Automatic detection of duplicate rows
2. Automatic detection of Features data types
3. Automatic Missing Data Imputation
4. Automatic Categorical Features Encoding
5. Automatic Feature Reduction
6. Automatic Feature Scaling

To evaluate this approach, we first apply it to ten different and diverse datasets and implement a machine learning algorithm on each. The results are then compared with the results generated by the same ML algorithm implemented on manually preprocessed data. The results have shown that not only this approach made the whole process uncomplicated and facile but was also able to improve the performance of the model.

II. LITERATURE WORK

A. DATA PREPROCESSING

Every day huge amount of data is created. Machine learning (ML) can learn and predict from these datasets to make them more valuable. However, a significant issue is that real life data is hardly ever clean [8], and poor quality of data can have a significant impact on the performance of learning algorithms [9]. As necessary and unavoidable as data preprocessing is, it is a monotonous and inconvenient process. Data scientists normally spend over half of their analytical time on preprocessing, nonetheless non-experts [10]. Therefore, data scientists are keen to develop a technique to automate this procedure.

Data preprocessing includes a variety of tasks such as cleaning, encoding, scaling and dimensionality reduction [11]. The task of resolving data issues is referred to as data cleaning. Typical data issues include missing values, inaccurate datatypes, and repeated rows. Data cleaning is intended to clean the data with missing values, inconsistencies, and noisy data [12]. Data preprocessing is also intended to perform feature encoding, scaling, and Dimensionality reduction.

1) DETECTION OF DATA TYPES

Detection of data types beforehand, can help to analyze and evaluate preprocessing and encoding options for each respective feature. Despite the diversity of data types, all the datatypes are not equally significant in the field of ML. The statistical data types within a data contain a wealth of information that is particularly valuable in the context of ML. Several feasible techniques are present to detect data types from a data. Some methods are straightforward and might only need few heuristics or statistics. But some complex and advanced techniques are also available which employ machine learning models for datatypes detection.

Messytables [13], is a python package uses brute force to predict data types. Brute force guessing extract a random sample from an attribute and then attempt to convert each value in the sample to every possible data type. The successful transformations rate is calculated for each data type, and the most likely data type for that column is determined by a majority vote. Messytables consider the following data types: Integer, String, Decimal, Data, and Bool. This method is adaptable and simple to implement.

2) MISSING DATA IMPUTATION

In practice, missing data is a frequent occurrence because of manual data entry systems, incorrect measurements, equipment malfunctions, intentional omissions et cetera. A few missing instances in some features can significantly reduce the sample size. Subsequently, the efficiency and precision of data analysis can be compromised, weaken the statistical power, and the parameter estimation could be biased because of the differences between complete and missing data [14]. In the field of ML, the missing values can increase the classifier error rate [15]. Therefore, missing data must be addressed before employing learning models.

Whether data comes from experiments, surveys, or secondary sources, missing data is abundant. But what effect does this have on statistical analysis results? That is contingent upon two factors: the mechanism that caused the data to

be missing and how the data analyst handles it [16]. In any type of study, data may be missing because of an accident or a data entry error. A meticulous understanding of data enables us to ascertain the mechanism of missing data. Missing data Mechanisms can be divided into three categories which are, MCAR [17], MAR and MNAR [18].

Collectively there are numerous viable techniques to deal missing data. Different methods are appropriate for different conditions. Deletion is effective only in MCAR without producing a significant bias [19]. Statistical information-based imputation is essentially effective to MCAR only because they estimate data without contemplating the relationship between attributes, MICE, matrix factorization, and KNN are mostly effective in MAR [20].

Some other alternative missing imputation strategies are also available, including missing indicator and maximum likelihood. Those approaches are not explained here because either they are too complex to automate or can only be useful to MCAR.

3) QUALITATIVE DATA ENCODING

ML algorithms expect all inputs and output attributes to be numerical [21]. This means if a dataset has categorical data, first encode that into numerical format before employing a ML algorithm. Encoding is a mandatory preprocessing stage when working with qualitative data for ML models and there exists a spectrum of methods for categorical data encoding [22]. The appropriate method can have a substantial effect on the performance of a model. We have determined encoding methods at one side, and conversely, there also exist some advanced algorithmic techniques. One way to identify a determined technique is it will generate the same encoded values every time we employ it, contrary to algorithmic methods [23]. Additionally, these methods have a minimal complexity in terms of run time.

Label encoding is essentially the process of allocating a numeral value to each potential value of a categorical attribute. Duan [24] compares the ability of several classifiers discover that to encode qualitative features, one-hot encoding provides satisfactory outcomes in the implementation of a neural network, which surpass the other ML algos. One-hot encoding approach needs very little effort to implement but drawback of this technique is, if we store encoded values directly, it uses a lot of storage resources. For large cardinality, the feature space can soon explode, and you are forced to combat with the curse of dimensionality, but the advantage of One-hot encoding is that it is easy to employ and has an effective running time [25].

From Saleem and Naseer's [26], we find out about the Leave-one-out encoding method for qualitative data. Leave One Out encoding determines the mean of the target variables for all records that contain the same value for the categorical feature variable in concern. An immature strategy of computing the mean value of labels for each row by traversing the full dataset is an O(n2) process that is prohibitive for large datasets. Similarly, The Hash encoder encodes categorical variables with the new dimensions just as one hot encoding. The user can here limit the number of dimensions after processing, by passing the component number as parameter. Chollet mentions hashing and exclaims that you can use hashing instead of one-hot encoding in case the number of potential values that you want to encode is so big that it is not practicable to create mappings of values to one-hot vectors.

4) FEATURE SCALING

It is common for real-world datasets to contain features that vary in units, magnitude, and range. As a result, feature scaling is required in order for ML models to comprehend these variables on the same scale [27]. Some machine learning algorithms are sensitive to feature scaling while others are completely insensitive to it. Scaling data is required for machine learning methods such as logistic regression, linear regression, and neural networks that use gradient descent as an optimization technique. Because scaled features can benefit the gradient descent to converge more swiftly towards the minima. Similarly, the range of features has the greatest effect on distance-based algorithms such as K-means, KNN, and SVM [28]. This is because they are determining the similarity of data points using distances between them. As a result, before utilizing a distance-based method, scaling the data is essential to ensure that all attributes contribute equally to the final outcome. On the other hand, tree-based algorithms are rather invariant to feature scaling.

Ambarwari *et al.* [29] discovered that feature scaling methods like standardization and Min-Max normalization have a considerable effect on final outcomes. The experiments were performed by using different ML algorithms including Naïve Bayesian, KNN, SVM, and ANN. Another study [30] undertaken by Balabaeva *et al.*, analyzed the impact of several scaling strategies on cardiac disease prediction dataset. Ahsan *et al.* [27] study concluded that the experimental outcomes for various scaling techniques may not always be satisfactory. For instance, where the majority of prior researchers found that scaling algorithms such as Normalization, Min-max, and StandardScaler improved SVM performance, this study discovered that SVM performance was dramatically reduced.

5) DIMENSIONALITY REDUCTION

Dimensionality reduction is the process of converting a high-dimensional data representation into a low-dimensional representation [31]. With the tremendous increase in high-dimensional data, the usage of several dimensionality reduction methods has prevalent in a wide range of applications. Furthermore, new modern ways are constantly appearing. Dimensionality reduction strategies take a high-dimensional dataset and convert it into a low-dimensional dataset while retaining as much of the original meaning of the data as possible. The representation of the original data in a low-dimensional format contributes to resolving the dimensionality-curse problem. Low-dimensional data is also easy to analyze, process, and visualize [32].

Numerous advantages can be achieved by employing dimensionality-reduction techniques on a dataset. (i) Data storage space can be lowered when the number of dimensions decreases. (ii) It requires significantly less computation time. (iii) Irrelevant, noisy, and redundant data can be eliminated. (iv) The quality of data can be enhanced. (v) High-dimensional data can cause the decline in the performance of many algorithms. Thus, dimensions reduction enables an algorithm to perform more efficiently and accurately. (vi) The visualization of high-dimension data is challenging and reduction in dimensions can help us to analyze or dissect patterns easily. (vii) It also simplifies the prediction process and increase efficiency [33].

In general, dimensionality-reduction methods can be divided into two categories, or in other words, dimensionality-reduction can be accomplished by two distinct techniques: feature selection and feature extraction [34]. Feature selection is used to reduce the data dimensionality by finding the features subset that best defines the data. From the original data it selects the features that are crucial and relevant to the ML task, and it removes the features that are irrelevant and redundant [94]. It is helpful for identifying a good subset of features that are relevant to the task at hand [35]. In feature selection, the primary goal is to generate a subset of features that is as small as feasible while yet accurately representing the key properties of the entire input [36]. Feature selection reduces the data size, decrease the storage requirements, improves prediction accuracy, avoids overfitting, and reduces execution and training time. The feature selection algorithm has two phases — Subset Generation and Evaluating Subsets. Subset Generation requires us to construct a subset from the input dataset, Subset Evaluations examine whether the resulting subset is optimal [37].

B. AUTOMATED MACHINE LEARNING (AutoML)

Several notable Machine learning libraries and tools have been developed from 1990 to 2020, comprising Weka in the 1990s, RapidMiner in 2001, Scikit-learn in 2007, H2O in 2011, MLlib in 2013, and many others [4]. Additionally, DNN platforms have also acquired fame over recent years. The TensorFlow, MXNet, Keras, and PyTorch have contributed significantly to the adoption of DL models [3]. And after this period, it became evident to Researchers and ML professionals that human expertise plays a substantial role in accomplishing the best results from ML models.

The initial attempt of AutoML was ''Auto-Weka'' in 2013 from the University of British Colombia [38]. Next came the ''Auto-Sklearn'' from the University of Freiburg in 2014 [39]. Similarly, In 2015, the University of Pennsylvania developed the ''TPOT'' [40]; Next came the ''Auto-ml'' in 2016. Texas A&M University released the ''Auto-Keras'' in 2017 [41], which utilizes the Keras, Scikit-learn, and TensorFlow. ''MLjar'' was released in 2018, based on TensorFlow and Scikit-learn. All the previously mentioned platforms and libraries draw attention to the various parts of the AutoML space. For instance, DataRobot, H2O-DriverlessAI, and In Auto-Keras, the preprocessing and feature generation tasks are not available; this technique only accepts data in model acceptable form, so the users must clean or preprocess data manually first. Nevertheless, upon receiving the manually prepared dataset, AutoKeras do recommend the best learning model. Darwin offers services about time-series data. Categorical Ensemble offered by auto-ml. Auto-Keras and Google AutoML perform Neural Network search. In short, each of them has its different strengths and shortcomings.

C. DATA PREPROCESSING IN AUTOML

In the previous section, we perceived the background of AutoML and some of the notable work achievements so far in this area. This section acknowledges how the various AutoML pipelines address the data preprocessing.

Data preprocessing is a fundamental stage in the Machine Learning pipeline. At Present, no AutoML technique is mature enough to handle this task accurately and demands significant human intervention. The schema or data types of detection is essential in this task, which in most cases are not supported by AutoML. Nevertheless, on identifying data types, AutoML can handle the feature engineering task for the next stage in the pipeline. TransmogrifAI is one step ahead in identifying the basic (numeric or categorical) and advanced data types (phone, address, currency). Even though TransmogrifAI has this functionality, but it is not stable on many datasets. Data preprocessing is handled by default methods, but the users can customize the parameters with other available options.

H2ODriverlessAI, and H2O-Automl [42] also have the functionality to detect the datatypes, but this is limited to only basic categorical, time-series, and numeric data. Many transformation methods are up to users to select according to requirements. For instance, H2O-AutoML has many options (none, normalize, descale, or standardize) to transform the numeric columns. However, there is no assistance provided to choose the best fitting one. Similarly, DataRobot, Darwin, and MLjar can only identify the basic data types and convert the dataset automatically according to user-specified techniques without providing any assistance for this task [3].

In Autosklearn [39], users have to manually specify the data types of the columns to perform feature engineering. Moreover, Autosklearn only accepts the numeric data, so all the categorical fields must be converted manually into numbers before any further transformation. Hence, this service cannot perform data cleaning but can implement Feature engineering on the numeric dataset. Auto-ml and AzureML are also just supporting the feature engineering, and data cleaning is up to users to perform manually.

In AutoWeka, user-assistance is available only in model selection like all other techniques. The system recommends the best learning-algorithm to implement with its suitable parametrization without considering the preprocessing step.

In TPOT [40], users need to handle preprocessing tasks manually because this system also does not support the data preprocessing stage in its AutoML pipeline. The only purpose

TABLE 1. Data preprocessing in AutoML.

Technique	Data Pre-Processing	Input Data Source		Data type's Detection				Feature Engineering				ML Task		Reference
		Spread Sheet	Text Mining	Numeric	Categorical	Date Time	Advance	Categorical Processing	Missing Value	Feature Redaction, Extraction	Advance Feature Extraction	Supervised Learning	Unsupervised Learning	
1.Transmogrif AI	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	[3]
2.Auto-Sklearn	N	Y	N	N	N	N	N	N	Y	Y	Y	Y	N	[39]
3.Auto-ML	N	Y	N	N	N	N	N	Y	Y	Y	Y	Y	N	[3]
4.Aut -Keras	N	Y	Y	N	N	N	N	N	Y	Y	N	Y	N	[41]
5 Aut Weka	N	Y	N	Y	Y	N	N	N	Y	Y	N	Y	N	[38]
6.TPOT	N	Y	N	N	N	N	N	N	Y	N	Y	Y	N	[40]
7.H2O-Auto ML	Y	Y	N	Y	Y	Y	N	Y	Y	Y	N	Y	N	[42]
8. H2O DriverlessAI	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	[42]
9. Darwin	Y	Y	N	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	[43]
10. DataRobot	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	[43]
11. MLjar	Y	Y	N	Y	Y	Y	N	Y	Y	N	N	Y	N	[43]
12. Google AutoML	Y	N	Y	.	.	.	.	.	N	Y	Y	Y	Y	[44]
13. Azure ML	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	N	[45]

of this system is to find and recommend the best model, and for pipeline optimization, genetic programming is used. The detailed comparison is illustrated in (Table 1).

III. METHODOLOGY

In this research, we develop a Python- based method to provide automatic, interactive, and data-driven service for users to perform data preprocessing efficiently. The developed method's primary responsibility is to preprocess data by deciding and applying the most suitable transformer methods based on the dataset's features. The method aims to improve the quality of data in order to get better performance from machine learning models. We employ pre-existing approaches to address a wide range of data cleaning problems and feature engineering tasks. Notably, our main focus is placed on automatic detection of data type, automatic imputation of missing value, and automatic features encoding, dimensionality reduction and feature scaling.

The most difficult challenge in designing this method was integrating all of the data preprocessing steps and their respective transformer methods in such a way that we could achieve our goal. The first and most evident approach one can think of is to permutate all of the techniques on the datasets and then evaluate all of the combinations to select the best performing one. However, this approach is not only exhaustive (which makes it time-consuming and computationally complex), but we also lack a set of norms or standards for evaluating these combinations.

Therefore, our proposed method must handle each task independently and be competent in making rational decisions along the way based on the given dataset. In subsequent sections, we discuss each subtask individually in detail and present how we deal with each particular problem.

A. AUTOMATIC DATA TYPES DETECTION

In order to do effective data-driven science and computation, it is necessary to understand how data is stored and manipulated. We already know from the literature review that statistical datatype (category, discrete, and continuous) of features are more significant in the field of machine learning, as mentioned in Section 2.2. Therefore, the ultimate goal of this function is not only to identify basic datatypes, but also to distinguish statistical datatypes. To do this, we suggest a strategy that blends the forthright logic approach with the Pandas library. Pandas can be used to detect the general data types for each feature automatically. The datatype inference of Pandas only considers int, float and object which is unfortunate. In practice, we can see that Pandas classifies a large number of features as 'objects.' For instance, it won't convert the datetime columns automatically unless you explicitly give it a list of the datetime columns.

Consequently, our proposed solution is to divide the task into two steps. In the first step, we discover the basic data types (integer, float, object) by using the simple inference approach of Pandas. Then in step 2, We recheck the type of each element in the object type feature, as inspired by Brute Force Guessing and Anjelo's work [46]. Date, time, bool, category, and string are the data types that will be inferred in this stage. We identify the additional datatype of the features using the following simple logical rules:

1. Bool: a feature with precisely two distinct values.
2. Date: up to ten characters, including the '−' or '/' symbol
3. Time: Contains ':' symbol and is limited to 8 characters (e.g., hh:mm:ss)
4. Category: A finite collection of text values
5. String/object: anything that's left after above rules.

Following that, we will go over how to detect each of these categories. Starting with the most fundamental datatype, bool, we compute the number of unique values in the sample before confirming the type of each element. If there are precisely two distinct values, the feature will be directly assigned the bool type. As we can observe, the datatypes are prioritized. If a feature is determined to be bool, we do not further investigate whether it is a category, datetime or a string. The second datatype to be determined is date. Dates can be encoded in a variety of ways. To be considered a date, the elements should adhere to the two conditions outlined below.

1. An entry can only be up to ten characters long.
2. An entry must contain '−' or '/' symbol

To infer dates, we must first handle the fact that dates come in a variety of formats, such as mm/dd/yyyy, dd/mm/yyyy, yyyy/mm/dd, mm-dd-yyyy, dd-mm-yyyy, yyyy-mm-dd, and so on. It is now difficult to consider all of the forms while performing brute force guessing. So, to detect and extract information from dates, we use pandas' datetime64[ns] datatype. We begin by attempting to convert each column to datetime, and then we process the successful conversions to extract the given information in separate columns (i.e., day, month, year). As a result, in this scenario, we are handling two jobs at the same time: datatype discovery and feature engineering. In next step, the features with a limited number of distinct values are classified as category type, we will discuss this type more in categorical encoding section.

Finally, on the basis of given rules, we evaluate if a numerical feature is discrete or continuous by counting the number of unique values:

1. Discrete: Finite number of unique values
2. Continuous: Infinite number of unique values

B. AUTOMATIC MISSING VALUE IMPUTATION

To deal with missing values, we must first discover them and then choose a suitable method for cleaning them. As discussed in literature review, no algorithm is infallibly superior to others. And the performance of an algorithm is highly dependent on the missing mechanism [17]. Therefore, we have divided this task into several subtasks, which are: discover missing values, visualize missing data, drop features with high missing-data percentage, missing values imputation with several techniques, and the selection of most suitable technique.

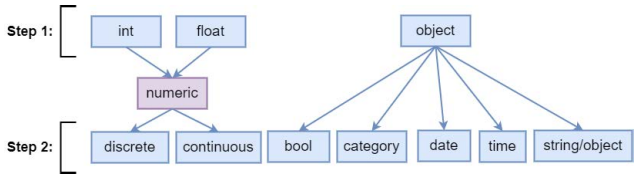

FIGURE 1. Workflow of automatic datatypes detection.

The workflow is depicted in Figure 2 to provide a high-level overview. We begin by identifying the missing values. Then, to help the user comprehend the missing ratio and missing mechanism we present them in effective graphs. Following that, we employ all the candidate techniques and recommend the best performing one.

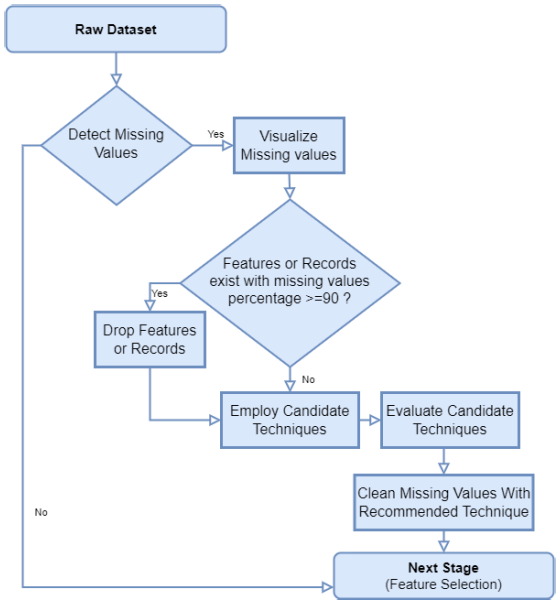

FIGURE 2. Workflow of automatic missing values handling.

1) DISCOVER MISSING VALUES

The first challenge to handle the missing data automatically is to detect the presence of missing Values. Missing values may be represented in the dataset using a variety of alternative formats, including '?' and 'nan'. We have categorized the formats into three types for better understanding. (I) The representations which can be treated by default as null values, such as: 'NAN', 'n/a', and Empty cells are Standard Missing Values. (II) Some datasets tend to have non-standard representation of missing values as well, such as: '—', '-', 'na', or '?', etc. (III) Depending on the context of the dataset, certain datasets represented missing values in an unexpected

manner. For example, if a feature's range is 0 to 100, then 9999 may indicate the missing value.

We've constructed our unique missing values list as default for the first two scenarios, but for the third scenario, let's say, we cannot take 9999 as missing/null for every dataset. As a result, it would be preferable to identify missing values in an interactive manner. To be more particular, in addition to the most common missing characters, such as 'nan,' we ask the user every time before the detection whether they want to add any additional specific value to be identified as missing.

2) MISSING MECHANISM

We have already established from literature that there exist three kinds of missing mechanisms: MAR, MCAR, and MNAR. Unfortunately, the hypothesis in case of MNAR cannot be tested because the necessary information is unavailable. We may have good cause to believe that the prospect of missingness is influenced by the values that are missing from the data set. For instance, some people may be less prone to record their incomes if they have a high salary, but there's no way to know from the dataset if this is true or not [47]. In case of MAR and MCAR, we can identify if any association exist among the missing values of one variable or feature with some other features. We may not detect any association between the two features, yet a MAR may nonetheless exist because a value might perhaps be absent as a function of several other features.

The Pearson correlation coefficient (PCC) is a measure of the linear-correlation between two features. The presence of missing values is denoted by the number 1 and the absence by the number 0, and then using the coordinates (1, 1), (1, 0), (0, 1), (0, 0), we may represent each pair of features as illustrated in Figure 3. The coordinates (1, 1) and (0, 0) should show frequently if there is a missing dependence among the respective two features, and a linear-correlation will be discovered.

Income	Profession
NAN	Manger
2000	Staff
1800	NAN
NAN	NAN

Income	Profession
1	0
0	0
0	1
1	1

FIGURE 3. Pearson correlation coefficient (PCC).

This is a symmetrical correlation and correlation is inversely proportional to the strength of the missing dependency. If the PCC is higher than 0.8 between two features, the missing mechanism will presumably be MAR.

3) VISUALIZE MISSING VALUES

The primary objective of this step is to help the user understand the missing data and mechanism in dataset more precisely.

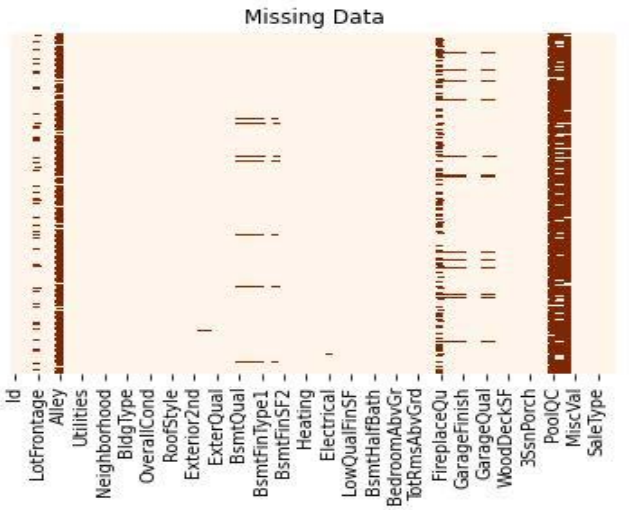

FIGURE 4. Seaborn heatmap.

I. seaborn heatmap: This heatmap not only depicts the features and their proportion of missing values, but also indicates instantly whether the occurrence of missing values is sparsely distributed or concentrated in a large chunk.

I. missingno matrix: Matrix charts are largely identical to the previous chart. It makes use of a density display to highlight a dataset's completeness. In comparison to the preceding illustration, a matrix is more effective at identifying missing patterns. For instance, it is evident that some missing relation exists in features 26–27, or 31–34. When one of them is absent, the others are also absent. The matrix pattern can aid users in comprehending the data and provide insight into the missing process. For instance, the first situation is more likely to be MCAR, but the second case is more likely to be MAR. missingno heatmap: A heatmap is used to show how deeply the absence or presence of one feature affects the absence or presence of another feature. The missing correlation is estimated using the Pearson Correlation Coefficient (PCC) — which we have discussed previously. Because PCC is symmetric, so is the heatmap. The range of Correlation is -1 to 1. When correlation is -1, one variable is present while the other is absent. A 0 correlation means that one variable does not affect the other. If the correlation is 1, then both variables must be present. This heatmap ignores features that have no missing data. Hence, the visualizations are intended to aid users in comprehending data so that they can infer the missing mechanism on their own.

4) ACCEPTABLE PERCENTAGE OF MISSING DATA

We may notice that the values of particular records or features are substantially absent after detecting missing values in the dataset. In this circumstance, these features or data provide

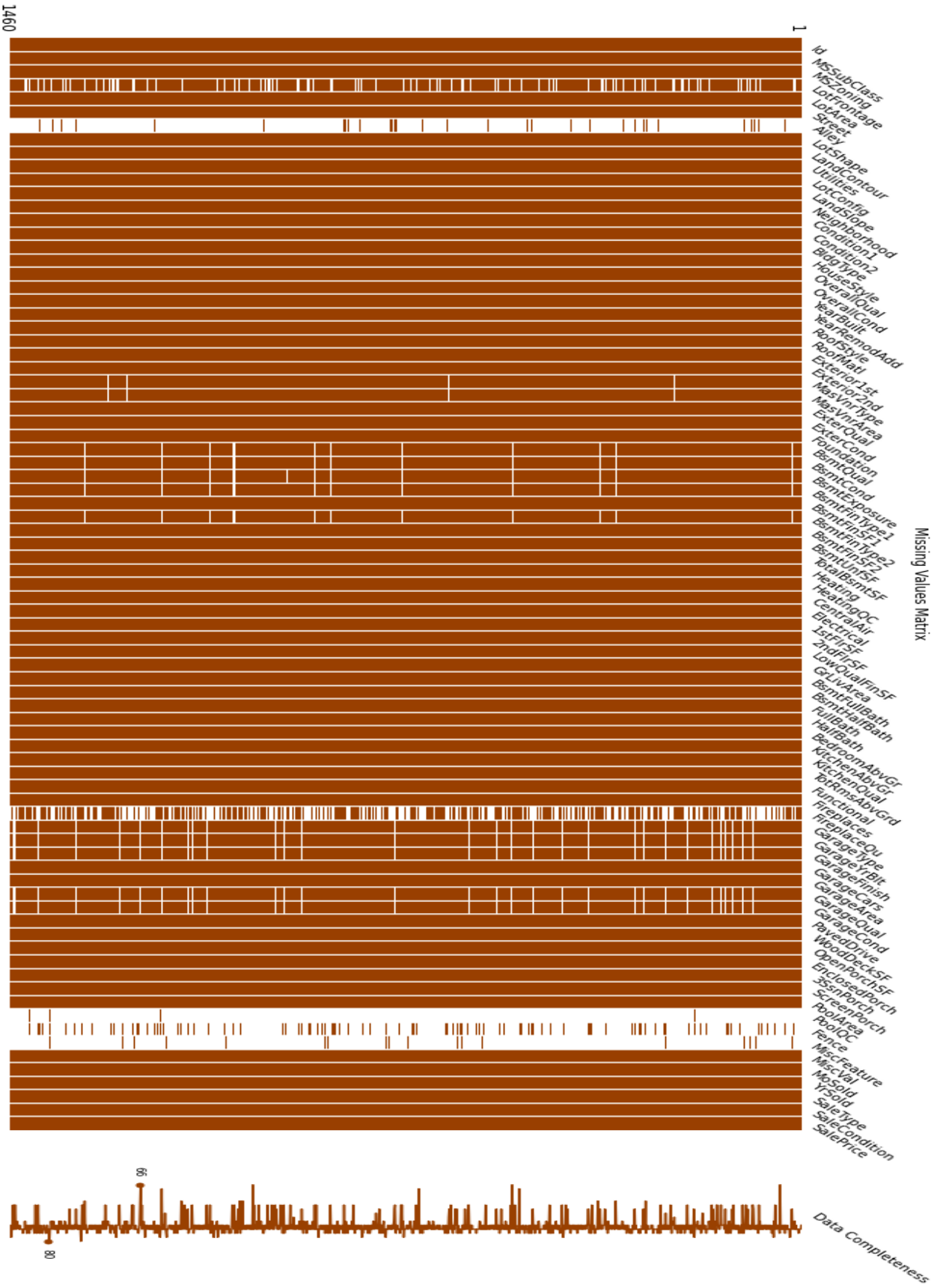

FIGURE 5. Missingno matric.

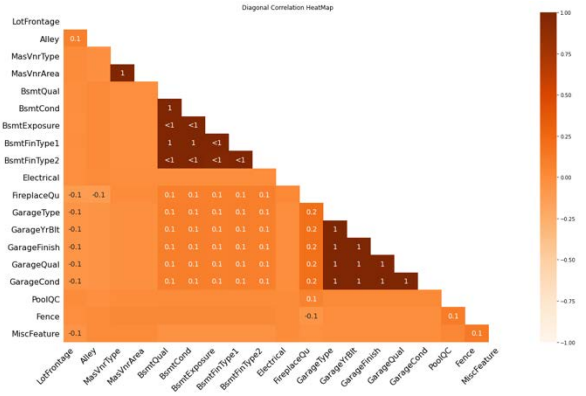

FIGURE 6. Missingno heatmap.

no information on the machine learning model while training. The percentage of missing data relates directly to the validity of statistical inferences. However, there is no agreed literature criterion for an acceptable proportion of missing data in the dataset for meaningful statistical inferences. Schafer *et al.* [48], for example, stated that a missing rate of 5 percent or less is insignificant. Bennett *et al.*[49] claimed that when more than 10% of data is missing, statistical analysis is likely to be skewed or biased. Furthermore, a researcher's assessment of the missing data problem is not solely based on the amount of missing data. Missing data methods and patterns, according to Tabachnick *et al.* [50], have a higher impact on research results than the quantity of missing data. Recently [51], established that even with a huge percentage of missing data (up to 90% in their simulated study), unbiased findings can be obtained if the imputation model is reasonably specified, and data are MAR.

As a result, in our proposed strategy, we allow variables with less than 90% missing values percentage and drop features with a greater missing rate than the default threshold. However, the default limit can be changed by the user.

5) IMPUTE MISSING VALUES

The several approaches to clean missing values may be perplexing to a novice user. As a result, this proposed method must use the most efficient strategy to cleaning missing values from each feature. Diverse techniques are appropriate for various missing mechanisms. The most frequently used techniques are addressed in this proposed method: mean, median, mode, most frequent, and k neighbor (KNN), or interpret missing values as a separate category and multiple imputation. These approaches are implemented using scikit learn.

Our candidate techniques for each mechanism are summarized as follows:

1. MCAR: mean, median mode, most frequent, k-nearest neighbors (KNN)
2. MAR: k-nearest neighbors (KNN)
3. MNAR: Treat as a separate category, Multiple Imputation

MCAR is not generally the case, but if it is a valid assumption, then there is a plethora of convenient solutions for dealing with missing data. All of the strategies presented above is considered.

The majority of research papers make the assumption of MAR. In this instance, there are significant relationships between variables/features. As a result, the statistical procedures mean, and mode should not be employed because they are just generating data without taking into account the feature dependency.

MNAR is by far the most difficult situation to deal with. Theoretically, data must be cleansed manually in this scenario, and deductive procedures should be used to accomplish this. Example: A person with two children in 2014, no children in 2015, and two children in 2016 may have two children in 2015 as well. This type of deductive assertion usually requires some context to be applied to it. But, since we are attempting to automate the whole process, we choose the multiple-imputation method as the sole viable option because it can still obtain acceptable results even in the MNAR situation [18], or we just can construe the missing values as containing information (i.e., missing for a reason).

In the preceding part, we classified the techniques according to the presence or absence of a missing mechanisms. However, as we have found with the home prices dataset, a dataset can contain numerous mechanisms at the same time. And we cannot just presume which technique will yield the most information or will be the most advantageous for a particular dataset to use. As a result, to select an approach, we must first anticipate the performance of each of the alternative approaches. A more formal evaluation of the performance of a computing approach must be made by comparing the missing values ground truth with imputed values. In reality, however, finding the ground truth for missing data is impractical in reality. Furthermore, it is challenging to evaluate in case of datasets that contain a variety of different sorts of missing mechanisms. In Literature, mostly the imputation approaches are evaluated by employing a classifier after computing missing values in order to determine whether or not the performance of a classifier is improved [12]. In this type of assessment, list deletion frequently outperforms imputation approaches. Suppose a dataset only have one complete record and after applying list deletion only that record will remain. Furthermore, there is no requirement for classification at this time. Evidently, this is not a reasonable position. As a result, list deletion is not considered in our method.

In order to impute or anticipate the performance of different imputation techniques, our approach first iterates over each column/feature and determines whether or not the data has any missing data. If there are no null values, it passes over this step; if there are any null values, we apply candidate techniques for imputation. After imputation, we employ some basic classifiers or regressors (according to the task type), and compute mean accuracy as the imputation score for each method. The classifiers and regressors used for evaluation are detailed below.

I. Naïve-Bayes Learner

Naive Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem.

$$p(X|Y) = \frac{p(Y|X).p(X)}{p(Y)}$$

Here, p(X|Y) represents the posterior-probability and p(X) represents the prior-probability. The assumption that all attributes of a dataset under consideration are independent of each other is what makes a naïve Bayes classifier naive.

II. Decision Tree Learner

This classifier is intent to build a model that predicts the value of a target variable using basic decision rules derived from data attributes. The classifier utilizes the information gain metric, which reveals the degree to which an attribute is informative in relation to the classification job by calculating its entropy. Higher the variance in features, the greater the information gain from the attribute. This learner prefers the feature that provides the greatest amount of information. Then, it constructs a single-node decision tree that has the selected feature as a split node.

III. Linear Discriminant Analysis

A classifier with a linear decision boundary that was created by fitting class conditional densities to data and applying Bayes' rule. The model assumes that all classes have the same covariance matrix and hence fits a Gaussian density to each class.

The Regressors employed are as follows:

I. Linear Regression

By fitting a linear equation to observed data, linear regression seeks to model the relationship between two variables. One variable is regarded as an explanatory variable, while the other is regarded as a dependent variable. Linear regression is depicted in the following way.

$$y = b^{*}x + c.$$

In the preceding equation, the independent variable is denoted by 'y', while the dependent variable is denoted by 'x'. When plotting linear regression, the slope of the line that produces the output variables is denoted by 'b', while the intercept is denoted by 'c'. The linear regression techniques make the assumption that the input and output have a linear relationship.

II. Support Vector Machine

Support Vector Machines can also be utilized as a regression technique, retaining all of the algorithm's primary characteristics (maximal margin). Support Vector Regression (SVR) is a regression model that is similar to the Support Vector Machine (SVM), with a few small modifications. To begin, because output is a continuous number, it becomes extremely difficult to forecast the data at hand, which contains an unlimited number of options. In the case of regression, a tolerance margin (epsilon) is specified in order to approximate the SVM. However, the fundamental concept remains constant: to minimize error by personalizing the hyperplane that maximizes the margin, while keeping in mind that some error is acceptable.

III. Random Forest Regressor

A Random Forest is an ensemble technique that combines several decision trees plus an approach called Bootstrap and Aggregation, more generally referred to as bagging, to solve both regression and classification problems. The underlying concept is to combine several decision trees in order to determine the final output, which in the case of regression is the mean prediction of the individual trees, rather than depending on individual decision trees.

Rather of cleaning data with the highest-scoring strategy, we display the scores of each possible technique and recommend the highest-scoring approach to the user. The user has the option of following the advice or pursuing alternative strategies. The interactive method of clearing missing values is depicted in following Figure.

```
Imputation Score of mean is 0.47668544869547275
Imputation Score of median is 0.5086872710941149
Imputation Score of mode is 0.4111797546099269
Imputation Score of KNN is 0.52319411565120855
Imputation Score of Multiple imputation is 0.339889744760498
Imputation Score of Null values as Separate Category is 0.5525134511452191

The most suitable Approach is Separate Category
Do you want to apply recommended approach [Y/N] N

1.mean 2.median 3.mode 4.KNN 5.multiple imputation 6.seperate category
Please Select the approach you want to apply [1/2/3/4/5/6] 4

Applying KNN ...

Missing Values Cleaned
```

FIGURE 7. Automatic missing values imputation.

C. AUTOMATIC QUALITATIVE DATA ENCODING

After Cleaning the missing values, the next stage comes the proper encoding to Qualitative features present in the dataset. We have used the word qualitative here instead of categorical because in statistics, a categorical variable is one that has a finite, and typically fixed number of potential values, allocating each individual or other unit of observation to a specific group or nominal category based on some qualitative attribute.

When working with qualitative data for machine learning algorithms, encoding is a necessary pre-processing step. All input and output variables in machine learning models must be numeric. This means that if your data contains qualitative data, you must first convert it to numeric data before training and evaluating a model.

We divided the qualitative data into bool (features have exactly two distinct values), categorical data (i.e., features with a category datatype) and object data (features that take on from infinite options of strings e.g., email and address). Additionally, categorical data can also be divided into nominal and ordinal categories.

1. Nominal attributes (Categorical): Features that derive their value from a finite set of discrete values that carry no relation to one another in terms of order.

2. Ordinal attributes: Features taking values from a finite set of discrete values that have a ranked ordering.

It is extremely difficult to tell the difference between nominal and ordinal data because the presence of an order in the data makes sense only in context. While colors in confectionery normally do not convey an order, but traffic light colors definitely do. And, as a result of our literature review, we determined that the label or ordinal encoding is more appropriate for ordinal attributes. While label encodings for nominal attributes assume a natural ordering between categories and may result in poor performance or unexpected results, which makes the one-hot encoding more advantageous in this case.

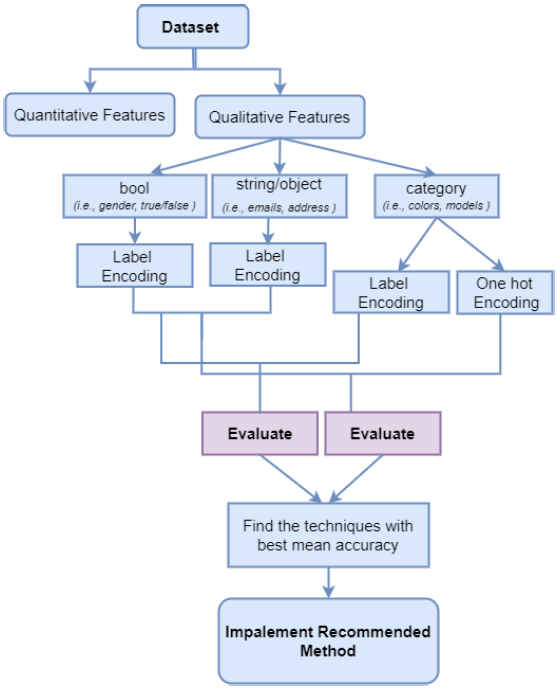

FIGURE 8. Workflow of automatic qualitative features encoding.

Additionally, we must bear in mind that for our previous task (i.e., evaluating missing value imputation techniques), we were required to convert categorical values prior to passing them through classifiers or regressors. And for the sake of simplicity, we used a label encoder at that stage. Now all that remains is to implement one-hot encoding on features with category datatype only and determine whether or not it improves the model's accuracy. We intentionally left the bool and object datatype features in place because no changes will happen to bool datatype features and features with object datatypes may cause the curse of dimensionality. In summary, our method evaluates label and one-hot encoding techniques and selects the one that provides the highest level of accuracy. Figure 8 depicts the entire workflow.

D. AUTOMATIC FEATURE SELECTION

When developing a machine learning model, it is critical to pick only those predictors that are required. Assume our dataset has 100 features but that doesn't mean we have to include all 100 features in our model. This is due to the fact that not all 100 attributes will have a substantial impact on the model. However, this does not imply that it will be true in all circumstances. It is entirely dependent on the information or dataset we have at our disposal.

There are several methods for determining which features have significant impact on the model and which can be removed from the dataset. For feature selection in this work, we used the Backward Elimination procedure. Fig 9. presents the visual representation of all the stages of this process.

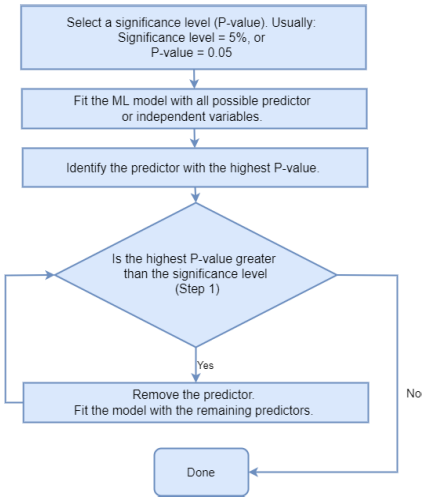

FIGURE 9. Backward elimination steps.

Step 1: The initial stage in backward elimination is rather straightforward; you simply choose a significance level or P-value. Typically, and in the majority of circumstances, a 5% significance level is chosen. This results in a P-value of 0.05.

Step 2: In this step simply fit your machine learning model with all of the features you've chosen. So, if there are 100 predictors, you incorporate them all in your model and test it on your test dataset.

Step 3: Determine predictor or column with the highest P-value.

Step 4: This is a substantial step as we make decisions here. We identified the columns with the highest P-value in the preceding phase. If the P-value for this column exceeds the significance level we specified in the first stage, we delete it from our dataset. If the P-value for this feature, which is the most significant in the collection, is less than the significance level, we can skip to Step 6, indicating that we are finished. Remember, if the greatest P-value exceeds the significance level, that characteristic should be removed.

Step 5: This stage will identify the feature that needs to be deleted from the dataset. As a result, we delete the feature from the data and re-fit the model with the new data. We'll return to step 3 after fitting the model to the new dataset. This method is repeated until the greatest P-value obtained from all remaining columns in the dataset is less than the significance determined in step 1. This indicates that we iterate from

step 3 - 5 and repeat until the dataset's greatest P-value is less than 0.05.

Step 6: Once we reach step 6, the feature selection procedure is complete. We were successful in filtering out features that were not significant enough for our model using backward elimination.

E. AUTOMATIC FEATURE SCALING

It is common for real-world datasets to contain features that vary in units, magnitude, and range. As a result, feature scaling is required in order for ML models to comprehend these variables on the same scale. The importance of comparing apples to apples has long been recognized in the scientific community. Despite this fact, many users (especially beginners) have a proclivity to exclude feature scaling as a part of data preprocessing for machine learning, which can result in models making inaccurate predictions.

But as we have discussed in literature review, that not all machine learning models require feature scaling for generating better results. Where certain machine learning algorithms are sensitive to feature scaling while others are completely insensitive to it.

Scaling data is required for machine learning methods such as logistic regression, linear regression, and neural networks that use gradient descent as an optimization technique. Because scaled features can benefit the gradient descent to converge more swiftly towards the minima. Similarly, the range of features has the greatest effect on distance-based algorithms such as K-means, KNN, and SVM. This is because they are determining the similarity of data points using distances between them. As a result, before utilizing a distance-based method, scaling the data is essential to ensure that all attributes contribute equally to the final outcome.

On the other hand, tree-based algorithms are rather invariant to feature scaling. Consider that a decision tree is simply a method for node splitting based on a single feature. The decision tree split a node based on a property that increases the node's homogeneity. This feature-specific divide is unaffected by other features. Thus, the remaining features have little effect on the split. This is why they are invariant with respect to the scale of the features.

In our method, we have considered the two mostly used methods for feature scaling (i.e., normalization and standardization). Based on the literature review, the models that perform best under normalization and standardization are detailed below.

I. Normalization: Better performs for Distance-based algorithms such as K-means, KNN, and SVM.
II. Standardization: Performs best in case of logistic regression, linear regression, and neural networks that use gradient descent as an optimization technique

As we have explained, the choice of feature scaling approach is entirely dependent on the type of machine learning model, and we cannot determine which strategy produces the best results merely on the basis of data. Given that we want our automatic method to be interactive as well, for this task, we have allowed the user to select the technique he or she want to utilize on datasets while keeping in mind the model he or she is going to implement. The user can choose Feature Scaling parameter to be ''Normalize'', ''Standardize'', or ''False'' option.

F. AUTOMATIC FEATURE EXTRACTION

Feature extraction is a method that generates new features that are dependent on the original input feature set in order to reduce the feature vector's high dimensionality. The transformation is carried out algebraically and in accordance with specified optimization criteria. On the other hand, the data description is sometimes lost after the feature extraction, and in several datasets the cost of this transformation has been found to be prohibitively high. For feature extraction our approach uses PCA, but its solely depends on the users if they want to implement PCA for dimensionality reduction. But by default, our approach does not apply PCA because firstly it cannot be necessary for every situation, secondly as literature revealed it can be proved costly. More significantly, after running PCA on the dataset, your original features will be transformed into Principal Components — the linear combination of your original features. Original features are more legible and interpretable than Principal Components. Furthermore, while Principal Components attempt to cover the greatest amount of variance among the features in a dataset, if the number of Principal Components is not carefully chosen, it may lose some information when compared to the original list of features. As a result of these considerations, feature extraction is an optional element of our method that the user can implement if desired.

Following the completion of all processes, our data is now efficiently prepared to be retrieved by any machine learning model.

IV. EVALUATION AND RESULTS

A. DATASETS

Due to the fact that our approach is capable of performing a wide variety of data preparation tasks, we select datasets with the goal of covering as many scenarios as feasible. We choose the most frequently used datasets from different repositories such as UCI3, OpenML, and Kaggle. The aim was to select datasets based on the variety of feature, data size and data challenges in order to validate Auto-preprocessing method in different situation. Table 2 summarizes the essential characteristics of selected datasets.

B. MACHINE LEARNING MODELS

Our primary objective for evaluation is to compare the accuracy of the model trained on auto preprocessed data to the accuracy of the same model trained on manually preprocessed data. Given that we are not concerned with the complexity and intricacy of the models, we employ simple and well-known machine learning algorithms for classification and regression.

TABLE 2. Dataset details.

Dataset Name	Number of Features	Number of Instances	ML Task
Ecommerce Customers	8	501	Regression
AirQualityUCI	15	9358	Regression
House Prices - Advanced Regression Techniques	81	2921	Regression
Automobile	26	205	Regression
Weather in Szeged 2006-2016	12	96454	Regression
Cancer detection	32	569	Binary Classification
Auto Insurance Claims Data	40	1001	Binary Classification
Tic-Tac-Toe Endgame	9	958	Binary Classification
Seeds	7	210	Multinomial Classification
Steel Plates Faults	28	1941	Multinomial Classification
Musk	168	6598	Binary Classification

I. Linear Regression

For regression tasks, we yet again select linear regression (LR) because of its straightforwardness and wide use in machine learning specially for regression tasks. By fitting a linear equation to observed data, linear regression seeks to model the relationship between two variables. One variable is regarded as an explanatory variable, while the other is regarded as a dependent variable. Linear regression is depicted as y = b*x + c.

In the preceding equation, the independent variable is denoted by 'y', while the dependent variable is denoted by 'x'. When plotting linear regression, the slope of the line that produces the output variables is denoted by 'b', while the intercept is denoted by 'c'. The linear regression techniques make the assumption that the input and output have a linear relationship.

II. Support Vector Machine

The ''Support Vector Machine'' (SVM) is a supervised machine learning algorithm that is frequently used for Classification and Regression problems. Here it is, however, employed for classification tasks.

The SVM algorithm's objective is to find the optimal line or decision boundary that partitions n-dimensional space into classes, allowing us to easily classify new data points in the future. This optimal decision boundary is referred to as a hyperplane. SVM selects the extreme points/vectors that aid in the formation of the hyperplane. These extreme points are known as support vectors, and so the method is named as Support Vector Machine.

III. Decision Tree

A Decision Tree is another straightforward way of classification. It is a form of Supervised Machine Learning in which data is continually partitioned according to a particular parameter. This classifier is intent to build a model that predicts the value of a target variable using basic decision rules derived from data attributes. The classifier utilizes the information gain metric, which reveals the degree to which an attribute is informative in relation to the classification job by calculating its entropy. Higher the variance in features, the greater the information gain from the attribute. This learner prefers the feature that provides the greatest amount of information. Then, it constructs a single-node decision tree that has the selected feature as a split node

C. PERFORMANCE METRICS FOR CLASSIFICATION

Numerous measures can be used to evaluate the performance of an ML algorithm. First, we will explore various performance measures that are used to evaluate predictions for classification problems and how they can be applied in practice

1) CONFUSION MATRIX

In a classification problem where the output may be divided into two or more types of classes, confusion matrix is the most straightforward method of evaluating the problem's performance. A confusion matrix is just a table with two dimensions, ''Actual'' and ''Predicted'', and each dimension contains ''True Positives (TP)'', ''True Negatives (TN)'', ''False Positives (FP)'', and ''False Negatives (FN)'', as illustrated below:

The following terms are defined in relation to the confusion matrix:

1. True Positives (TP): This is the case when both the actual and expected class of a data point are 1.
2. True Negatives (TN): This is the case when both the actual and expected class of a data point are 0.
3. False Positives (FP): This is the case when both the actual class is 0 and expected class is 1.
4. False Negatives (FN): This is the case when both the actual class is 1 and expected class is 0.

2) CLASSIFICATION ACCURACY

For classification algorithms, it is the mostly used performance metric. It can be defined as the ratio of correct predictions to total predictions — How frequently does the classifier make an accurate classification. With the help of the following formula, we can simply compute it using the confusion matrix.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

3) RECALL

The recall score of a classifier is the ratio of the number of accurately predicted positive observations divided by the total number of observations in the positive class.

$$\text{Re}call = \frac{TP}{TP + FN}$$

4) F1 SCORE

The F1 Score is calculated as the weighted average of the Recall and Precision. As a result, this score accounts for both

TABLE 3. Auto preprocessing on classification dataset.

		Data Exploration / Detecting Datatypes of Features						Detect and Impute Missing Values			Encode Categorical Features	Feature Reduction	Feature Scaling
	Dataset	Total Number of Features	Discrete Numeric Features	Continuous Numeric Features	Qualitative Features	Date Time Feature	Duplicate data Available	Numeric Missing Values	Categoric Missing Values	Missing Values Imputation	Encode Categorical Features	Features Selection	
1.	Cancer detection	32	0	31	1	0	0	0	0	No	Yes	Yes	None
2.	Auto Insurance Claims Data	40	9	11	18	2	0	1000	881	Yes	Yes	Yes	Standardization
3.	Tic-Tac-Toe	10	1	0	9	0	0	0	0	No	Yes	Yes	None
4.	Wheat-Seeds	8	1	7	0	0	0	0	0	No	No	Yes	Normalization
5.	Steel Plates Faults	24	4	24	0	0	8	80	0	Yes	No	Yes	None
6.	Musk	170	1	168	2	0	0	0	0	No	Yes	Yes	None

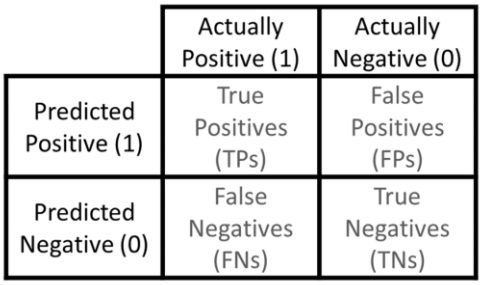

FIGURE 10. Confusion matrix.

false negative and false positives. While F1 is not as intuitive as accuracy, it is frequently more useful, especially when the class distribution is unequal.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

D. CLASSIFICATION RESULTS

The Auto-Prep method is used to preprocess the six different classification datasets stated in Table 2. The first four datasets are binary classification datasets; however, they differ in terms of feature size and data quality. The final two datasets are associated with multiclass classification problems. The operations performed by Auto-preprocessing algorithms on each dataset are summarized in Table 3.

Support vector classifiers are used to evaluate the findings on cancer diagnosis, insurance claims, and tic-tac-toe datasets. On the Musk dataset, a decision tress classifier is used. The final two datasets, wheat-seed and steel-plates are predicted using support vector machine and random forest classifiers, respectively. All of these classifiers are applied once to auto-preprocessed data and once to manually-preprocessed data. Finally, the models' performance in both scenarios is compared. The confusion matrices for each dataset are shown in figure 11 to figure 16. The ones on the left side are the result in case of auto preprocessing, while the ones on the right side are the result in case of manually preprocessing of the datasets.

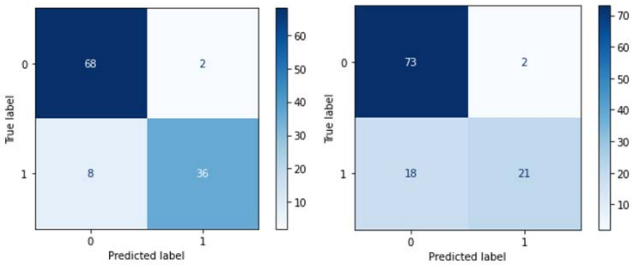

FIGURE 11. Confusion matrices of cancer detection datasets (a) in case of auto preprocessing, and (b) in case of manually preprocessing.

Following confusion matrix, accuracy is the next performance metric. Table 4 summarizes the classification accuracy of classifiers for each dataset. As can be seen from the table, the accuracy of models used on auto-preprocessed datasets is either enhanced or remains close to the accuracy of models used on manually preprocessed datasets. The results validate the main objective of this research.

Presented in Table 5 is the precision score of each classifier in both cases. In simple words this score represents

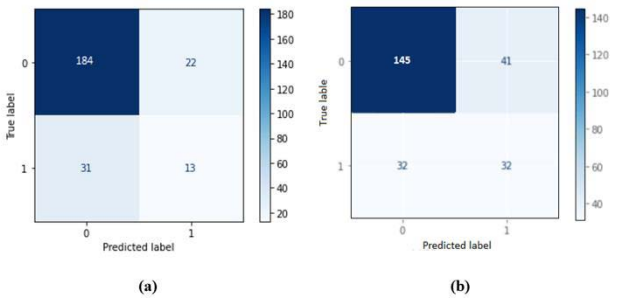

FIGURE 12. Confusion matrices of auto insurance claims dataset (a) in case of auto preprocessing, and (b) in case of manually preprocessing.

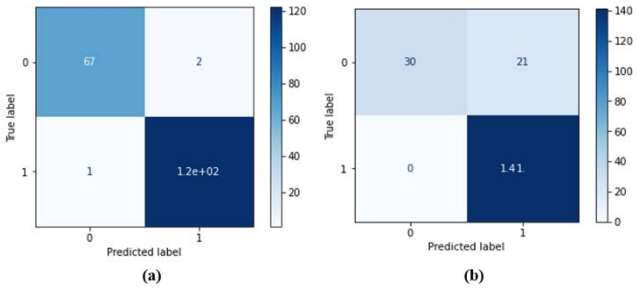

FIGURE 13. Confusion matrices of Tic-Tac-Toe dataset (a) in case of auto preprocessing, and (b) in case of manually preprocessing.

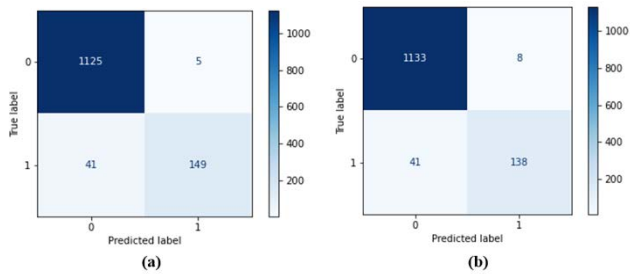

FIGURE 14. Confusion matrices of musk dataset (a) in case of auto preprocessing, and (b) in case of manually preprocessing.

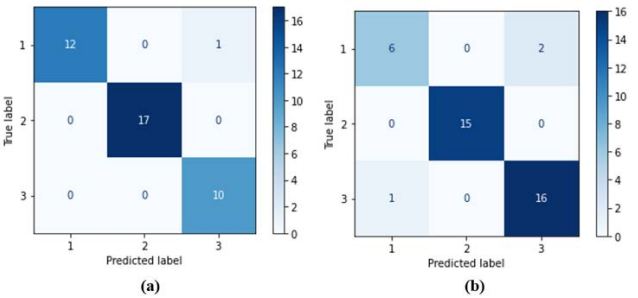

FIGURE 15. Confusion matrices of wheat seed dataset (a) in case of auto preprocessing, and (b) in case of manually preprocessing.

the classification model's ability to detect only relevant data points. In both situations, the precision scores for Auto-insurance claims are low as compared to other datasets, indicating that this dataset may require more preprocessing to improve the data's quality.

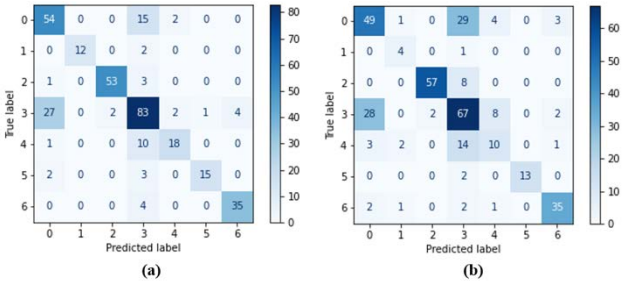

FIGURE 16. Confusion matrices of steel plate faults dataset (a) in case of auto preprocessing, and (b) in case of manually preprocessing.

TABLE 4. Accuracy of classifiers.

	Dataset Name	Results after Automatic Data Preprocessing		Results after Best Manual Preprocessing	
		ML Algorithm/ Model	Accuracy	ML Algorithm/ Model	Accuracy
1.	Cancer detection	SVC	0.91	SVC	0.82
2.	Auto Insurance Claims Data	SVC	0.78	SVC	0.70
3.	Tic-Tac-Toe	SVC	0.98	SVC	0.89
4.	Musk	DT	0.96	DT	0.96
5.	Wheat-Seeds	SVM	0.97	SVM	0.95
6.	Steel Plates Faults	Random Forest	0.77	Random Forest	0.67

Tables 6 and 7 show the recall score, and F1 score for each dataset in both cases.

E. PERFORMANCE METRICS FOR REGRESSION

The performance measurements used to evaluate the regression model are as follows:

I. R Squared

Additionally, it is referred to as the coefficient of determination. This metric indicates the degree to which a model fits a given dataset, or it represents the degree to which the regression line is congruent with the original data values.

II. Mean Absolute Error

MAE calculates the average of the errors from each sample in a dataset.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

In the following formula:

$\boldsymbol{n}$ = total no. of data points

$\boldsymbol{Y_i}$ = observed values

$\hat{\boldsymbol{Y}}_{\boldsymbol{i}}$ = predicted values

TABLE 5. Precision score of classifiers.

	Dataset Name	Results after Automatic Data Preprocessing		Results after Best Manual Preprocessing	
		ML Algorithm/ Model	Precision	ML Algorithm/ Model	Precision
1.	Cancer detection	SVC	0.947	SVC	0.91
2.	Auto Insurance Claims Data	SVC	0.43	SVC	0.37
3.	Tic-Tac-Toe	SVC	0.98	SVC	0.87
4.	Musk	DT	0.96	DT	0.94
5.	Wheat-Seeds	SVM	0.977	SVM	0.95
6.	Steel Plates Faults	Random Forest	0.78	Random Forest	0.68

TABLE 6. Recall ccore of classifiers.

	Dataset Name	Results after Automatic Data Preprocessing		Results after Best Manual Preprocessing	
		ML Algorithm/ Model	Recall	ML Algorithm/ Model	Recall
1.	Cancer detection	SVC	0.818	SVC	0.53
2.	Auto Insurance Claims Data	SVC	0.56	SVC	0.29
3.	Tic-Tac-Toe	SVC	0.99	SVC	1.0
4.	Musk	DT	0.78	DT	0.77
5.	Wheat-Seeds	SVM	0.97	SVM	0.95
6.	Steel Plates Faults	Random Forest	0.773	Random Forest	0.673

$\boldsymbol{y_i}$ = predicted value
$\boldsymbol{x_i}$ = actual value
$\boldsymbol{n}$ = total no. of data point

III. Mean Squared Error

The MSE is calculated by averaging the square of the difference between the data's original and predicted values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

TABLE 7. F1 score of classifiers.

	Dataset Name	Results after Automatic Data Preprocessing		Results after Best Manual Preprocessing	
		ML Algorithm/ Model	F1 Score	ML Algorithm/ Model	F1 Score
1.	Cancer detection	SVC	0.87	SVC	0.67
2.	Auto Insurance Claims Data	SVC	0.46	SVC	0.32
3.	Tic-Tac-Toe	SVC	0.98	SVC	0.93
4.	Musk	DT	0.86	DT	0.84
5.	Wheat-Seeds	SVM	0.975	SVM	0.95
6.	Steel Plates Faults	Random Forest	0.77	Random Forest	0.675

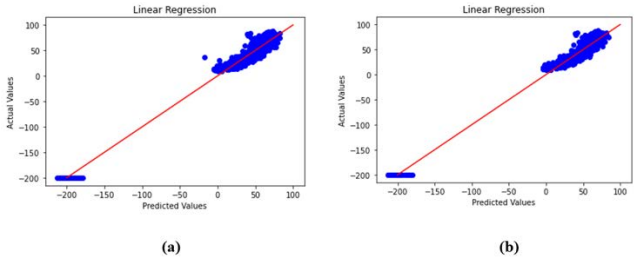

FIGURE 17. Scatter plot of air quality datasets (a) after auto preprocessing of dataset, and (b) after manually preprocessing of dataset.

In the following formula of MSE:

$$RMSD = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}}$$

IV. Root Mean Squared Error

The root mean square error (RMSE) represents the standard deviation of the errors that occur when a prediction is made on a dataset. This is identical to MSE, except that the root of the number is considered when calculating the model's accuracy.

F. REGRESSION RESULTS

Similarly, the Auto Preprocessing method is applied to preprocess the five different regression datasets listed in Table 2. Each dataset has a distinct domain, feature size, preprocessing required, and data quality. Table 8 summarizes the operations performed on each dataset by automatic preprocessing method.

TABLE 8. Auto preprocessing on regression datasets.

		Data Exploration / Detecting Datatypes of Features						Detect and Impute Missing Values			Encode Categorical Features	Feature Reduction	Feature Scaling
	Database	Total Number of Features	Discrete Numeric Features	Continuous Numeric Features	Categorical Features	Date Time Feature	Duplicate data Available	Numeric Missing Values	Categoric Missing Values	Missing Values Imputation	Encode Categorical Features	Features Selection	
1.	Ecommerce Customers	8	0	5	3	0	0	0	56	Yes	Yes	Yes	None
2.	AirQualityUCI	15	0	13	0	2	0	0	0	No	No	Yes	Standardization
3.	House Prices - Advanced Regression Techniques	81	17	21	43	0	0	348	6617	Yes	Yes	Yes	Normalization
4.	Automobile	26	2	14	10	0	2	49	2	Yes	Yes	Yes	None
5.	Weather in Szeged 2006-2016	12	1	7	3	1	24	0	517	Yes	Yes	Yes	Normalization

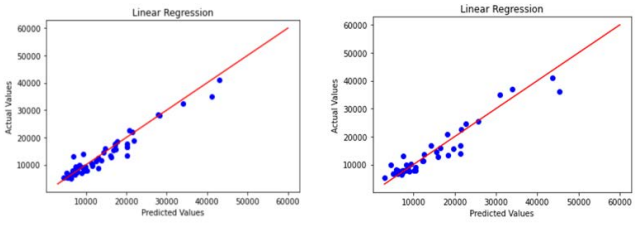

FIGURE 18. Scatter plot of automobile dataset (a) after auto preprocessing of dataset, and (b) after manually preprocessing of dataset.

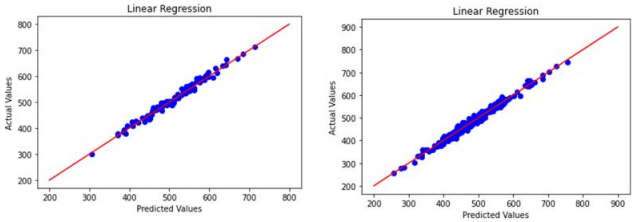

FIGURE 19. Scatter plot of ecommerce customers dataset (a) after auto preprocessing of dataset, and (b) after manually preprocessing of dataset.

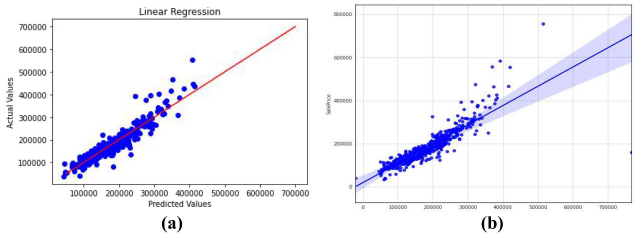

FIGURE 20. Scatter plot of house prices dataset (a) after auto preprocessing of dataset, and (b) after manually preprocessing of dataset.

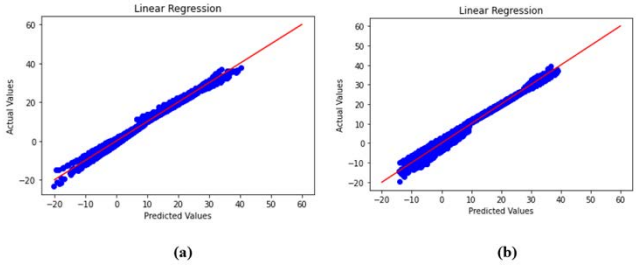

FIGURE 21. Scatter plot of weather in Szeged 2006-2016 dataset (a) after auto preprocessing of dataset, and (b) after manually preprocessing of dataset.

Each regression dataset is evaluated using a linear regression model. The model is applied to both automatically preprocessed and manually preprocessed data. Finally, the models' performance is compared in both cases. Scatter plots of each dataset are depicted in figures 17 to 21. The ones on the left represent the outcomes of automatic preprocessing, while the ones on the right represent the outcomes of manual preprocessing.

Following Scatter Plots, Table 9 to Table 12 summarizes the other performance metrics results of linear regression model for each dataset. MAE is less susceptible to outliers

TABLE 9. Regression models test score.

	Dataset Name	Results after Automatic Data Preprocessing		Results after Best Manual Preprocessing	
		ML Algorithm/ Model	R^2 Score	ML Algorithm/ Model	R^2 Score
1.	Ecommerce Customers	LR	0.983	LR	0.981
2.	AirQualityUCI	LR	0.977	LR	0.974
3.	House Prices - Advanced Regression Techniques	LR	0.87	LR	0.74
4.	Automobile	LR	0.91	LR	0.88
5.	Weather in Szeged 2006-2016	LR	0.98	LR	0.97
6.	Ecommerce Customers	LR	0.983	LR	0.981

TABLE 10. MAE score.

	Dataset Name	Results after Automatic Data Preprocessing		Results after Best Manual Preprocessing	
		ML Algorithm/ Model	Mean Absolute Error	ML Algorithm/ Model	Mean Absolute Error
1.	Ecommerce Customers	LR	7.98	LR	8.50
2.	AirQualityUCI	LR	5.97	LR	6.11
3.	House Prices - Advanced Regression Techniques	LR	0.048 (log scale)	LR	20825
4.	Automobile	LR	1588.1	LR	2237
5.	Weather in Szeged 2006-2016	LR	0.11	LR	0.83
6.	Ecommerce Customers	LR	7.98	LR	8.50

TABLE 11. MSE score.

	Dataset Name	Results after Automatic Data Preprocessing		Results after Best Manual Preprocessing	
		ML Algorithm/ Model	Mean Squared Error	ML Algorithm/ Model	Mean Squared Error
1.	Ecommerce Customers	LR	98.8	LR	110.11
2.	AirQualityUCI	LR	55.6	LR	57.7
3.	House Prices - Advanced Regression Techniques	LR	0.0040 (Log scale)	LR	1564098374. 18
4.	Automobile	LR	4923428.5	LR	9262710.6
5.	Weather in Szeged 2006-2016	LR	0.020	LR	1.13
6.	Ecommerce Customers	LR	98.8	LR	110.11

TABLE 12. RMSE score.

	Dataset Name	Results after Automatic Data Preprocessing		Results after Best Manual Preprocessing	
		ML Algorithm/ Model	Root Mean Absolute Error	ML Algorithm/ Model	Root Mean Absolute Error
1.	Ecommerce Customers	LR	9.94	LR	10.49
2.	AirQualityUCI	LR	7.46	LR	7.59
3.	House Prices - Advanced Regression Techniques	LR	28609 Or 0.063 (Log scale)	LR	39548
4.	Automobile	LR	2218.88	LR	3043.47
5.	Weather in Szeged 2006-2016	LR	0.14	LR	1.06
6.	Ecommerce Customers	LR	9.94	LR	10.49

than MSE because it does not penalize large errors. It is typically utilized while measuring performance on continuous variable data. Larger errors are given a higher weight in RMSE. This suggests that RMSE is significantly more

beneficial when substantial errors are present and have a significant impact on the model's performance. The lower the values, the more efficient the model. In case of house prices prediction, the left side metrics are calculated as the difference between the logarithm of the predicted price and the logarithm of the actual price (Using logs ensures that errors in estimating expensive and inexpensive houses have an equal impact on the final results). The results indicate that the proposed technique performed admirably and is quite effective.

The findings indicated that the proposed strategy was certainly capable of simplifying and automating the complete preprocessing stages, as well as improving the performance of the machine learning model. In some circumstances, the model's performance is not entirely adequate, owing to the fact that data is a complicated entity that occasionally requires more cleaning than the offered method provides. As of now, this approach is incapable of dealing with outliers and unbalanced data. This observation points us in the right direction for future enhancements.

V. CONCLUSION

In this work, we automate the data preprocessing tasks. We explored typical data challenges as well as existing approaches for resolving them. To aid the research, an automated, data-driven, and interactive system is being developed to identify potential flaws in the data and report results and recommendations to the user. The tool is intended for use in the field of machine learning, and the following components are meaningfully automated: data type detection, missing values imputation, qualitative data encoding features scaling, feature selection and extraction. We use six regression and 5 classification datasets with diverse features to evaluate our method. For evaluation we compare the accuracy ratings of the model trained on automated preprocessed data to the model trained on manually preprocessed data. We see a considerable improvement in the model's accuracy when trained on datasets prepared using the Auto-Preprocessing approach. As a result, our devised approach not only aided the user in doing the tedious preprocessing operation, but also improved the model's accuracy. Additionally, Auto-Prep method offers a great deal of room potential for future expansion.

VI. FUTURE WORK

Data preparation is a broad task with many different components to consider, such as the algorithm, visualization, or a particular subtask. Each subtask or data challenge can be studied independently as a separate project. Our data preprocessing method attempts to address as many data issues as possible, leaving much space for improvement in a variety of areas. Following that, we'll get more specific.

1) At the moment, Auto-Preprocessing is capable of discovering several statistical data kinds. But some more kinds remain undefined: real-valued, interval, and ordinal. The Bayesian model presented in the literatures might be implemented for real-valued variables and can be expanded by including ordinal and interval likelihood functions for detecting respective datatypes.
2) We forecast the performance of a strategy for missing value detection and features encoding by assessing it against several fundamental machine learning classifiers and regressors. This does not imply, however, that the recommended strategy is also the ideal one for the user's classifier. We can allow the user to specify their classifier interactively and then immediately evaluate the approaches based on this user-specified classifier.
3) While our method does not yet deal with outliers, appropriate detection and management strategies for evaluation and recommendation can also be introduced.
4) We provide multiple options for users to visualize data, yet this may still be insufficient given the complexity and high-dimensionality of the data. An interactive visualization is preferable; the user can manipulate the visualization directly to view what they want to see.
5) Auto-Prep is restricted to supervised classification and regression, although it is extensible to other problems such as clustering. In this situation, the evaluation of null values imputation procedures must be altered to take into account clustering algorithms such as DBSCAN and k-means. Likewise, for other subtasks.

REFERENCES

[1] R. Budjač, M. Nikmon, P. Schreiber, B. Zahradníková, and D. Janácová, "Automated machine learning overview," *Vedecké Práce Materiálovotechnologickej Fakulty Slovenskej Technickej Univerzity v Bratislave so Sídlom v Trnave*, vol. 27, no. 45, pp. 107–112, 2019.
[2] H. J. Escalante, "Automated machine learning—A brief review at the end of the early years," 2020, *arXiv:2008.08516*.
[3] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools," in *Proc. IEEE 31st Int. Conf. Tools Artif. Intell. (ICTAI)*, Oct. 2019, pp. 1471–1479.
[4] R. Elshawi, M. Maher, and S. Sakr, "Automated machine learning: State-of-the-art and open challenges," 2019, *arXiv:1906.02287*.
[5] Q. Yao, M. Wang, Y. Chen, W. Dai, Y.-F. Li, W.-W. Tu, Q. Yang, and Y. Yu, "Taking human out of learning applications: A survey on automated machine learning," 2018, *arXiv:1810.13306*.
[6] B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel, "Automated data pre-processing via meta-learning," in *Automated Data Pre-Processing Via Meta-Learning*. New York, NY, USA: Springer, 2016, pp. 194–208.
[7] D. Wang, J. D. Weisz, M. Müller, P. Ram, W. Geyer, C. Dugan, Y. Tausczik, H. Samulowitz, and A. Gray, "Human-AI collaboration in data science: Exploring data scientists' perceptions of automated AI," *Proc. ACM Hum.-Comput. Interact.*, vol. 3, pp. 1–24, Nov. 2019.
[8] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data Mining Knowl. Discovery*, vol. 2, no. 1, pp. 9–37, Jan. 1998.
[9] S. F. Crone, S. Lessmann, and R. Stahlbock, "The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing," *Eur. J. Oper. Res.*, vol. 173, no. 3, pp. 781–800, Sep. 2006.
[10] M. A. Munson, "A study on the importance of and time spent on different modeling steps," *ACM SIGKDD Explor. Newslett.*, vol. 13, no. 2, pp. 65–71, Dec. 2011.
[11] S. A. Alasadi and W. S. Bhaya, "Review of data preprocessing techniques in data mining," *J. Eng. Appl. Sci.*, vol. 12, no. 16, pp. 4102–4107, 2017.
[12] J. Brownlee, *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*. Melbourn, VIC, Australia: Jason Brownlee, 2020.
[13] Y. Xian, H. Zhao, T. Y. Lee, S. Kim, R. Rossi, Z. Fu, M. G. De, and S. Muthukrishnan, "EXACTA: Explainable column annotation," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2021, pp. 3775–3785.

[14] R. J. Little and D. B. Rubin, *Statistical Analysis With Missing Data*. Hoboken, NJ, USA: Wiley, 2019.

[15] L. A. Hunt, "Missing data imputation and its effect on the accuracy of classification," in *Data Science*. New York, NY, USA: Springer, 2017, pp. 3–14.

[16] J. Kaiser, "Dealing with missing values in data," *J. Syst. Integr.*, vol. 5, no. 1, pp. 1–10, 2014.

[17] A. P. Barata, F. W. Takes, H. J. van den Herik, and C. J. Veenman, "Imputation methods outperform missing-indicator for data missing completely at random," in *Proc. Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2019, pp. 407–414.

[18] K. M. Lee, R. Mitra, and S. Biedermann, "Optimal design when outcome values are not missing at random," *Statistica Sinica*, vol. 28, no. 4, pp. 1821–1838, 2018.

[19] T. B. Pepinsky, "A note on listwise deletion versus multiple imputation," *Political Anal.*, vol. 26, no. 4, pp. 480–488, Oct. 2018.

[20] C. Curley, R. M. Krause, R. Feiock, and C. V. Hawkins, "Dealing with missing data: A comparative exploration of approaches using the integrated city sustainability database," *Urban Affairs Rev.*, vol. 55, no. 2, pp. 591–615, Mar. 2019.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2020.

[22] W. D. McGinnis, C. Siu, and H. Huang, "Category encoders: A scikit-learn-contrib package of transformers for encoding categorical data," *J. Open Source Softw.*, vol. 3, no. 21, p. 501, Jan. 2018.

[23] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," *J. Big Data*, vol. 7, no. 1, pp. 1–41, Dec. 2020.

[24] J. Duan, "Financial system modeling using deep neural networks (DNNs) for effective risk assessment and prediction," *J. Franklin Inst.-Eng. Appl. Math.*, vol. 356, no. 8, pp. 4716–4731, May 2019.

[25] R. Garreta, G. Moncecchi, T. Hauck, and G. Hackeling, *Scikit-Learn: Machine Learning Simplified: Implement Scikit-Learn Into Every Step of the Data Science Pipeline*. Birmingham, U.K.: Packt, 2017.

[26] S. Naseer and Y. Saleem, "Enhanced network intrusion detection using deep convolutional neural networks," *KSII Trans. Internet Inf. Syst.*, vol. 12, no. 10, pp. 5159–5178, 2018.

[27] M. Ahsan, M. Mahmud, P. Saha, K. Gupta, and Z. Siddique, "Effect of data scaling methods on machine learning algorithms and model performance," *Technologies*, vol. 9, no. 3, p. 52, Jul. 2021.

[28] S. Basak and M. Huber, "Evolutionary feature scaling in K-nearest neighbors based on label dispersion minimization," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2020, pp. 928–935.

[29] A. Ambarwari, Q. Jafar Adrian, and Y. Herdiyeni, "Analysis of the effect of data scaling on the performance of the machine learning algorithm for plant identification," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 4, no. 1, pp. 117–122, Feb. 2020.

[30] K. Balabaeva and S. Kovalchuk, "Comparison of temporal and non-temporal features effect on machine learning models quality and interpretability for chronic heart failure patients," *Proc. Comput. Sci.*, vol. 156, pp. 87–96, Jan. 2019.

[31] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 2, pp. 56–70, May 2020.

[32] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *The Curse of Dimensionality in Data Mining and Time Series Prediction*. New York, NY, USA: Springer, 2005, pp. 758–770.

[33] L. Liu and M. T. Özsu, *Encyclopedia of Database Systems*. New York, NY, USA: Springer, 2019.

[34] M. Espadoto, R. M. Martins, A. Kerren, N. S. T. Hirata, and A. C. Telea, "Toward a quantitative survey of dimension reduction techniques," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 3, pp. 2153–2173, Mar. 2021.

[35] X. Huang, L. Wu, and Y. Ye, "A review on dimensionality reduction techniques," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 33, no. 10, 2019, Art. no. 1950017.

[36] S. Velliangiri, S. Alagumuthukrishnan, and S. I. Thankumar Joseph, "A review of dimensionality reduction techniques for efficient computation," *Proc. Comput. Sci.*, vol. 165, pp. 104–111, Jan. 2019.

[37] S. Visalakshi and V. Radha, "A literature review of feature selection techniques and applications: Review of feature selection in data mining," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res.*, Dec. 2014, pp. 1–6.

[38] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-WEKA: Automatic model selection and hyperparameter optimization in WEKA," in *Automated Machine Learning*. Cham, Switzerland: Springer, 2019, pp. 81–95.

[39] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter, "Auto-Sklearn 2.0: The next generation," 2020, *arXiv:2007.04074*.

[40] R. S. Olson and J. H. Moore, "TPOT: A tree-based pipeline optimization tool for automating machine learning," in *Proc. Workshop Autom. Mach. Learn.*, 2016, pp. 66–74.

[41] H. Jin, Q. Song, and X. Hu, "Auto-Keras: Efficient neural architecture search with network morphism," 2018, *arXiv:1806.10282*.

[42] E. LeDell and S. Poirier, *H2O AutoML: Scalable Automatic Machine Learning*. 2020, pp. 1–16.

[43] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*. New York, NY, USA: Springer, 2019.

[44] E. Bisong, "Google AutoML: Cloud vision," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. New York, NY, USA: Springer, 2019, pp. 581–598.

[45] J. Barnes, *Azure Machine Learning*. Washington, DC, USA: Microsoft, 2015.

[46] S. Krishnan and E. Wu, "Alphaclean: Automatic generation of data cleaning pipelines," 2019, *arXiv:1904.11827*.

[47] G. Chhabra, V. Vashisht, and J. Ranjan, "A comparison of multiple imputation methods for data with missing values," *Indian J. Sci. Technol.*, vol. 10, no. 19, pp. 1–7, 2017.

[48] J. L. Schafer, "Multiple imputation: A primer," *Stat. Methods Med. Res.*, vol. 8, no. 1, pp. 3–15, 1999.

[49] D. A. Bennett, "How can i deal with missing data in my study?" *Austral. New Zealand J. Public Health*, vol. 25, no. 5, pp. 464–469, Oct. 2001.

[50] B. G. Tabachnick, L. S. Fidell, and J. B. Ullman, *Using Multivariate Statistics*. Boston, MA, USA: Pearson, 2007.

[51] Y. Dong and C.-Y.-J. Peng, "Principled missing data methods for researchers," *SpringerPlus*, vol. 2, no. 1, pp. 1–17, Dec. 2013.

MEHWISH BILAL received the B.S. degree in software engineering from Bahria University, Islamabad, and the master's degree in computer science from the University of Okara. Currently, she is working with the Federal Investigation Agency (FIA) of Pakistan as a Software Engineer. She is also an Active Member of the Pakistan Engineering Council.

GHULAM ALI received the bachelor's degree in computer science from the University of Agriculture, Faisalabad, Pakistan, and the M.S. and Ph.D. degrees in computer science from the University of Central Punjab, Lahore, Pakistan. From September 2012 to May 2013, he has worked as a Lecturer at the Department of Computer Science, The University of Lahore, Lahore. He also worked as a Lecturer at the Department of Computer science, Government College University, from June 2013 to August 2019. He has joined the University of Okara as an Assistant Professor, in August 2019. He is the in-charge of the Department of Software Engineering, University of Okara. He has supervised 24 M.S. research students. He is currently supervising 12 M.S. research students and also co-supervising two Ph.D. student. His research interests include image processing, human emotions analysis, machine learning, video surveillance, medical image analysis, stochastic processes, intelligent agents, and formal methods.

MUHAMMAD WASEEM IQBAL received the Ph.D. degree in computer science from The Superior University, Lahore, Pakistan. He is currently working as an Associate Professor with the Software Engineering Department. He is also an Active Researcher. He has more than 75 research publications in well-reputed journals and conferences. Furthermore, he has more than 16 years of teaching and research experience in well-reputed institutions. He specializes in human–computer interaction (HCI), with special interests in adaptive interfaces (AI), user's context, UX/UI for normal, visual impaired people and user centered design (UCD), the Internet of Things (IoT), the Internet of Medical Things (IoMT), human centric artificial intelligence (HCAI), semantic relations, and ontological modeling.

MUHAMMAD ANWAR received the Ph.D. degree in computer science from the University of Technology Malaysia (UTM), in 2019. He is currently working as an Assistant Professor in information technology with the University of Education, Lahore, Pakistan. He has over 15 years of professional experience in different public and private sector ICT projects. He has various research publications in reputed journals and conferences. He is also a reviewer of various journals. His research interests include sensor networks, green computing, the Internet of Things, machine learning, and blockchain.

MUHAMMAD SHERAZ ARSHAD MALIK received the Ph.D. degree in information technology. He has more than seven years of research and industrial experience. He is currently an Assistant Professor with the Department of Information Technology, Government College University Faisalabad, Pakistan. His research interests include information visualization, temporal data, data analytics, and the Internet of Things.

RABIAH ABDUL KADIR received the Diploma and bachelor's degrees in computer science from University Putra Malaysia, in 1990 and 1993, respectively, and the master's and Ph.D. degrees in computer science from University Kebangsaan Malaysia, in 1997 and 2007, respectively. She is currently a Senior Research Fellow with the Institute of IR4.0, Universiti Kebangsaan Malaysia. She worked as a Tutor with University Putra Malaysia, from 1993 to 1997, and became a Lecturer, in 1997. After completed her master's degree, she was appointed as a Lecturer, from January 1997 to July 2008. From August 2008 to May 2014, she was appointed as a Senior Lecturer. During this period, she was seconded as an Assistant Professor at Najran University, Saudi Arabia, from September 2012 to August 2013. Since June 2014, she has been joining Universiti Kebangsaan Malaysia, as a Research Fellow. Her research interests include computational linguistics, intelligent computing, and data analytics. Currently, she is actives on big data analytics, semantic knowledge representation and extraction, medical intelligent system, and sentiment analysis. She has published more than 100 articles of academic journals, book of chapter, and proceeding on her areas. She is actives in several international conferences organized by local and international association or universities.