MACFE: A Meta-learning and Causality Based Feature Engineering Framework

Ivan Reyes-Amezcua[1], Daniel Flores-Araiza[2], Gilberto Ochoa-Ruiz[2], Andres Mendez-Vazquez[1], and Eduardo Rodriguez-Tello[3]

[1] Department of Computer Science, CINVESTAV – Guadalajara, Mexico.
[2] Tecnológico de Monterrey, School of Engineering and Sciences, Mexico.
[3] Cinvestav - Tamaulipas, Km. 5.5 Carretera Victoria - Soto La Marina, 87130 Victoria Tamps., Mexico

Abstract. Feature engineering has become one of the most important steps to improving model prediction performance, and producing quality datasets. However, this process requires non-trivial domain knowledge which involves a time-consuming task. Thereby, automating such processes has become an active area of research and interest in industrial applications. In this paper, a novel method, called Meta-learning and Causality Based Feature Engineering (MACFE), is proposed; our method is based on the use of meta-learning, feature distribution encoding, and causality feature selection. In MACFE, meta-learning is used to find the best transformations, then the search is accelerated by preselecting "original" features given their causal relevance. Experimental evaluations on popular classification datasets show that MACFE can improve the prediction performance across eight classifiers, outperforms the current state-of-the-art methods on average by at least 6.54%, and obtains an improvement of 2.71% over the best previous works.

Keywords: automated feature engineering, automated machine learning, causal feature selection.

1 Introduction

Extracting features from raw data and transforming them into formats that are appropriate for Machine Learning (ML) models is what is known as feature engineering [12]. This task is usually carried out by a data scientist with good domain knowledge and the data sources of the task at hand [19, 21, 33]. Generally, feature engineering entails the daunting manual labor of designing, selecting, and evaluating features where even a great intuition is needed [6, 18]. This is due to the fact that the performance of most machine learning algorithms heavily relies on the training data quality. These type of datasets usually consists of a large collection of different formats that need to be curated to be exploited by machine learning algorithms [6]. Therefore, by using feature engineering, we can select and obtain novel features from the raw data that would better represent the problem.

However, most of the existing automated feature engineering proposals perform this task by applying the *expansion-reduction* method [17], which is the process of trying a predefined set of transformation functions applied to raw features. Then, those transformed features are selected based on the improvement of model performance or some evaluation metric [21]. However, *expansion-reduction* leads to an exponential growth in the space of constructed features, which is known as the *feature explosion* problem [5]. In addition, extracting novel features without a proper and systematic method can lead to an unnecessary increase in the dimensionality of the data, and hence a poor performance in the learning process of the model [3]. Thus, the *curse of dimensionality* arises [20], which is the potential of high-dimensional data to be more complicated to process than low-dimensional data [8].

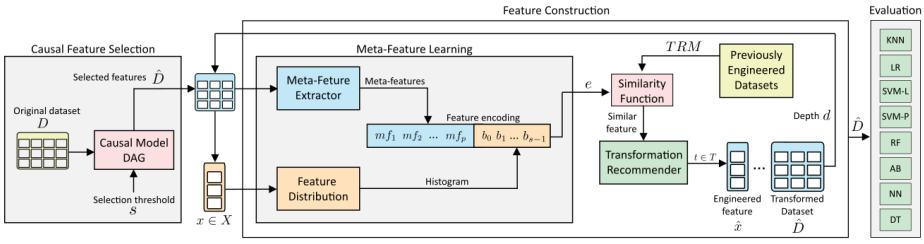

Fig. 1: The framework of our method. MACFE extracts meta-features from dataset $\boldsymbol{D}$ and a frequency table for each feature $\boldsymbol{x} \in \boldsymbol{X}$. Then, an encoding $\boldsymbol{e}$ is generated by the meta-features and feature distribution. Next, we search for the most similar encoding on the Transformation Recommendation Matrix ($\boldsymbol{TRM}$) in order to recommend a useful transformation from it. The transformed dataset $\hat{\boldsymbol{D}}$ is built from the constructed novel features and the original ones selected by the Directed Acyclic Graph (DAG) causal model.

It is crucial to realize that there are dozens of types of machine learning models, and each has its peculiarities and needs [19]. For instance, some models neither work with highly correlated features nor with highly multi-collinearity. Additionally, other models have trouble dealing with missing, noisy, or irrelevant features. Furthermore, since data and models are so diverse, it is difficult to generalize the practice of feature engineering across projects [33]. Thus, finding a proper process to treat data agnostically from a specific learning algorithm can help to choose transformations that better suit the learning process. To tackle this issue, a possibility is to incorporate only the generated features that have more appropriate knowledge about the data. For this, we present MACFE, a novel meta-learning and causality approach for automated feature engineering for classification problems using tabular data. The main contributions of this paper are briefly described as follows:

- We present a causality-based method for feature selection on the original dataset. For this, we use the mean magnitude effect of the features on the target to rank and select a subset of them.
- We propose a novel meta-learning generation for unary, binary, and high-order features based on non-linear transformations. This approach addresses the feature explosion problem by only searching for feature transformations that were found useful in past experiences.

In order to evaluate the proposed method, we designed a series of experiments on fourteen popular public classification datasets with relatively small dimensions to evaluate the feature generation and selection performance of MACFE. The results are obtained from eight machine learning models: Logistic Regression *(LR)*, K-Nearest Neighbors *(KNN)*, Lineal Support Vector Machine *(SVC-L)*, Polynomial Support Vector Machine *(SVC-P)*, Random Forest *(RF)*, AdaBoost *(AB)*, Multi-layer Perceptron *(MLP)* and Decision Tree *(DT)*. As illustrated in Fig. 1, our approach is divided into three phases. In the first one, the feature selection is carried out by using a Structural Causal Model (SCM) [22] for choosing the most promising features. Then, we move to a meta-learning phase (the second one), where meta-features are extracted from datasets and feature distributions to create encodings for each attribute. Then, we lookup for feature transformation on similar previously engineered datasets. Finally, in the third phase, we evaluate the engineered features among eight machine learning models and obtain the mean accuracy of stratified 5-Fold Cross Validation in order to assess the quality of the feature engineering method. Experimental results show that our proposal is effective in surpassing the scores of state-of-the-art feature engineering methods by achieving a mean accuracy of 81.83% across the fourteen testing datasets and the eight machine learning models evaluated.

The rest of this paper is organized as follows. In Section 2, we review the state of the art in automated feature engineering. In Section 3, we elaborate on the need for automated methods like ours. In Section 4, we introduce our proposed method MACFE, whereas Section 5, we show in detail our evaluation results, and finally, in Section 6 the conclusions drawn in this research work are given.

2 Related Work

In recent years, many automated feature engineering methods have been proposed using different methodologies. For instance, Data Science Machine (DSM) [14] is an automated feature engineering approach for structured and relational data. DSM proposed a Deep Feature Synthesis (DFS) method, which searches for relations and transformations across features in databases. They include a depth hyper-parameter d for setting the maximum composition, which recursively enumerates all possible transformations. In addition, DSM generates a large novel feature space, which is reduced by using Singular Value Decomposition (SVD) based feature selection. However, DSM is only suitable for relational data and

it could take high computational times due to all the transformation functions used for processing all the original feature sets.

The data-driven approach presented in FCTree [9], creates novel features from sequential transformations of the original space by employing decision trees and then selecting the best features with the aid of information gain. The method in [25], known as the TFC framework, presents an iterative feature generation algorithm. The method applies feature transformation across all the features, and then it selects the best features based on information gain. Nevertheless, the generated feature space grows in a combinatorial way, leading to feature explosion. AutoFeat [13] and AutoLearn [16] are also data-driven methods. They can generate large transformations of features, selecting useful features by using regularized regression models for each pair of features. However, these methods require training a regression model, which can be time-consuming. Also, they both suffer from the feature explosion problem. Label based Regression (LbR) [30] is another method for generating novel features by using Ridge Regression and Kernel Ridge Regression. This method selects features based on the Distance Correlation Coefficient and the Maximum Information Coefficient (MIC) for each feature pair, which leads to discriminate features that are useful in combination with others.

2.1 Meta-learning for Feature Engineering

Recently, meta-learning has been proposed as a means of improving the quality of the generated features [21]. Meta-data can be simply defined as data about data [31]. For this work, meta-features are used to characterize and identify features with attributes in the context of meta-learning [1, 4, 10, 28]. Some examples of meta-features are: a) *General*, such as the number of samples, features or classes, etc. b) *Statistical* like standard deviation, correlation coefficients, etc. c) *Information-theoretic* such as entropy, mutual information, noise ratio, etc. d) *Model-based* describes some characteristics of models such as Decision Trees, Bayesian Networks, SVMs, etc.

ExploreKit [15] is an example of a method that uses meta-learning for ranking and selecting the most promising generated features. ExploreKit does this by applying all possible transformations on features, suffering from the feature explosion problem. Furthermore, Learning Feature Engineering (LFE) is an approach that also uses meta-learning for recommending useful features for classification problems. The transformation recommender in LFE is based on the construction of a meta-feature vector based on the feature values associated with a class label. However, LFE can recommend only unary and binary transformations, lacking high-order transformations.

2.2 Causality Feature Selection

Classical feature selection approaches to leverage the correlations between features and class variables but lack taking advantage of the causal relationships

between them. In contrast, knowing the causal relationship implies the underlying mechanism of a dataset [32], and thus causal variables are expected to be persistent across different settings or environments.

Hence, basing the feature engineering on relevant causally related features to the class of interest ideally should provide a more rich and robust output of engineered features. Consequently, if we work only with causally related variables to the target variable, independently of the type of relationship, it should be possible to be learned by an ML model, which additionally implies it facilitates, at some level, the efficacy of applying feature engineering to causally related variables.

3 Problem Definition

Let $D = \{\boldsymbol{X}, \boldsymbol{Y}\}$ be a dataset of input-output pairs, $\boldsymbol{X}$ a collection of n features $\{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}\}$, and m labels $\boldsymbol{Y} = \{y_1, ..., y_m\}$. A machine learning algorithm L (e.g. SVM, Logistic Regression, or Random Forest), and an evaluation metric E (e.g. accuracy, F1-score).

We refer to a transformation $t \in \boldsymbol{T}$ as a function $t(\boldsymbol{x})$ that takes a feature as an argument, and maps it to a transformed feature output $\hat{\boldsymbol{x}} \in \boldsymbol{X'}$. Where $\boldsymbol{T}$ is our set of transformations $\{t_1, t_2, ..., t_k\}$ that can be unary or binary, depending on the number of given arguments. Here, a high-order transformation is a composition of unary and binary transformations. Over each feature it is possible to define a series of non-linear transformations, $t_i : \boldsymbol{x_i} \rightarrow \hat{\boldsymbol{x_i}}$ that allow to extract as much intra and inter information from the "original" data. The goal of feature engineering is thus to transform $\boldsymbol{X}$ into $\boldsymbol{X'}$ by applying $\boldsymbol{T}$ such that $\boldsymbol{X'}$ maximizes the evaluation metric E of a machine learning algorithm L. The search for new transformed features and their combinations grows exponentially, and the feature explosion problem arises. MACFE, our proposed feature engineering approach, was devised to help mitigate this problem by employing meta-features to guide the search for transformations on features.

3.1 Meta-learning and Meta-features

A formal definition of meta-features was proposed in [28], in which meta-features are a set of q values extracted from a dataset D by a function f.

$$f(D) = \sigma(\mu(D, h_\mu), h_\sigma), \tag{1}$$

Where $f : D \mapsto \mathbb{R}^q$ is the extraction of q values from dataset D, $\mu : D \mapsto \mathbb{R}^{q'}$ is a characterization measure, $\sigma : \mathbb{R}^{q'} \mapsto \mathbb{R}^q$ can be a summarization function such as: mean, minimum, maximum, etc. Moreover, h_μ and h_σ are hyperparameters for μ and σ, respectively. Thus, the function f is built by measuring some characteristic from D by μ, and a summarizing function defined by σ.

Here, meta-features describe features using meta-data. An example is the mean or median, as they are features that provide extra information about the underlying data distribution. In particular, the core of this work is meta-learning applied to the identification of data through meta-features.

4 Proposed Approach

In the following sections, we describe the dataset preprocessing along with the construction of the meta-feature vector and encodings for features. Also, we present the training of our method including the Meta-learning and Causal Selection phases.

4.1 Datasets

Preprocessing. MACFE is guided by meta-feature learning based on past experience to create novel features. Our method is trained with M random datasets $\boldsymbol{D_{train}} = \{\boldsymbol{D_1}, \boldsymbol{D_2}, ..., \boldsymbol{D_M}\}$ collected from Open ML [29], which have a structured format and a classification task related to the data. First, the preprocessing and cleaning of data are performed for each dataset by removing non-numerical features and imputing missing values with the feature mean. Next, a meta-feature *extractor* is used to obtain meta-data about the datasets. Let $\boldsymbol{mf}$ be a meta-feature vector composed by the main characteristics of a given dataset $\boldsymbol{D_i} \in \boldsymbol{D_{train}}$. Thus, a meta-feature vector for a dataset $\boldsymbol{D_i}$ is defined as:

$$\boldsymbol{mf} = [mf_1, mf_2, ..., mf_p], \tag{2}$$

Where each mf_i is a meta-feature value extracted from the data, and p is the size of the extracted meta-features.

However, describing datasets by mapping their main characteristics can be a challenging task. A full set of estimators and metrics can be extracted from a dataset, e.g., the number of classes or instances in a dataset can be a meta-feature value from such a dataset. For this, we use the approach of [24] to perform the automatic meta-feature extraction process. The extraction of meta-features is divided into five categories proposed by Rivolli *et al.* [28]: simple or general, statistical, information-theoretic and model-based, and landmarking. In order to automate the process of extracting meta-features from datasets, we use the framework Meta-feature Extractor (MFE) [1] for each training datasets $\boldsymbol{D_i} \in \boldsymbol{D_{train}}$, which implements the standard meta-feature extraction described above.

Next, we treat each feature $\boldsymbol{x} \in \boldsymbol{D_i}$ as follows:

1. We create a frequency table with a fixed number of *buckets* or *bins* b, for each feature $\boldsymbol{x}$
2. A range r is calculated on the feature values given by the *upper* and *lower* bounds of the feature.
3. We generate s disjoint sets or bins b with uniform width w. Thus, each bin range b_i is a bucket in which values that are in the bin range lie. Each b_i range starts with the lower bound of $\boldsymbol{x}$ *plus* i times the width w, and ends with the lower bound of $\boldsymbol{x}$ *plus* $i + 1$ times the width w.
4. Finally, each frequency table or histogram is normalized in the range [0,1].

Thus, we obtain an encoding $\boldsymbol{e} \in \mathbb{R}^{1\times\eta}$ for each feature $\boldsymbol{x} \in \boldsymbol{D_i}$, composed by the meta-feature vector $\boldsymbol{mf}$ of the dataset and the feature distribution as follows:

$$\boldsymbol{e} = [mf_1, mf_2, \ldots, mf_p, b_0, b_1, ...b_{s-1}] \tag{3}$$

4.2 Model Training

Meta-learning Phase. The meta-learning phase is described as follows. The unary, binary, and scaling feature transformations $t \in \boldsymbol{T}$ are applied to the original features $\boldsymbol{X}$. Then, an evaluation is performed on both original features and the generated features $t(\boldsymbol{X})$. For this, we use the Maximal Information Coefficient (MIC) [27], which measures the strength of the linear or non-linear relationships between two variables. MIC generates values between 0 and 1, where 0 means statistical independence and 1 stands for a noiseless statistical relationship between variables. Thus, we get the set of selected transformations $\boldsymbol{T_{sel}}$ for each original feature in $\boldsymbol{x} \in \boldsymbol{X}$ with the maximum score as follows:

$$\boldsymbol{T_{sel}} = \underset{t\in\boldsymbol{T}}{\operatorname{argmax}}\ g_t\Big(MIC(t(\boldsymbol{x})) - MIC(\boldsymbol{x})\Big). \tag{4}$$

Finally, the selected transformations $t \in \boldsymbol{T_{sel}}$ are stored in the Transformation Recommendation Matrix ($\boldsymbol{TRM}$) for each $\boldsymbol{x} \in \boldsymbol{D_{train}}$ represented by its corresponding encoding $\boldsymbol{e}$. Thus, $\boldsymbol{TRM}$ is represented as follows (Fig. 2).

$$\boldsymbol{TRM} = \begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,\eta} & t_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ e_{N,1} & e_{N,2} & \cdots & e_{N,\eta} & t_N \end{bmatrix}$$

Fig. 2: $\boldsymbol{TRM}$ Matrix, where the i^{th} row in the matrix is the feature $\boldsymbol{x} \in \boldsymbol{D_{train}}$, and the j^{th} column is the encoding value of $\boldsymbol{e}$ (Eq. 3). N is the size of all the features in $\boldsymbol{D_{train}}$, and η is the size of encoding $\boldsymbol{e}$ composed by the meta-feature vector $\boldsymbol{mf}$ (Eq. 2) and feature histogram. The last column stands for the transformations $t \in \boldsymbol{T}$ with the resulting highest MIC score for the given features (Eq. 4).

In Alg. 1 the training procedure to learn the most appropriated unary $\boldsymbol{T_{un}}$ and binary $\boldsymbol{T_{bin}}$ transformations is presented. This process is done for each

feature in a given dataset $\boldsymbol{D}$. Similarly, high-order transformations are built by combining several unary or binary transformations one after the other (Alg. 2).

Algorithm 1 Training $\boldsymbol{TRM}$

Input: Structured Dataset D
Output: $\boldsymbol{TRM}$
D = preprocess(D)
for each $\boldsymbol{x_i} \in D$ **do**
  $\boldsymbol{e_i}$ = encode_feature($\boldsymbol{x_i}$)
  for each $t \in \boldsymbol{T_{un}}$ **do**
    $\hat{\boldsymbol{x}}_{\boldsymbol{i}} = t(\boldsymbol{x_i})$
    s.append($MIC(\hat{\boldsymbol{x}}_{\boldsymbol{i}}) - MIC(\boldsymbol{x_i})$)
  end for
  t_{top} = argmax(s)
  $\boldsymbol{TRM}$.append($\boldsymbol{e_i}$, t_{top})
  for each $\boldsymbol{x_j} \in D | j > i$ **do**
    $\boldsymbol{e_j}$ = encode_feature($\boldsymbol{x_j}$)
    for each $t \in \boldsymbol{T_{bin}}$ **do**
      $\hat{\boldsymbol{x}}_{\boldsymbol{i,j}} = t(\boldsymbol{x_i}, \boldsymbol{x_j})$
      $s_i = MIC(\hat{\boldsymbol{x}}_{\boldsymbol{i,j}}) - MIC(\boldsymbol{x_i})$
      $s_j = MIC(\hat{\boldsymbol{x}}_{\boldsymbol{i,j}}) - MIC(\boldsymbol{x_j})$
      if $s_i > 0$ and $s_j > 0$ **then**
        $\boldsymbol{TRM}$.append($\boldsymbol{e_i}$, $\boldsymbol{e_j}$, t)
      end if
    end for
  end for
end for

Algorithm 2 Data Transformation

Input: D, d, s
Output: $\hat{D}$
$\hat{D}$ = preprocess(D)
$\hat{D}$ = causal_selection($\hat{D}$, s)
for 1 to d **do**
  for each $\boldsymbol{x_i} \in \hat{D}$ **do**
    $\boldsymbol{e_i}$ = encode_feature($\boldsymbol{x_i}$)
    t_{un} = Similarity($\boldsymbol{TRM}$, $\boldsymbol{e_i}$)
    $\hat{\boldsymbol{x}}_{\boldsymbol{i}} = t_{un}(\boldsymbol{x_i})$
    $\hat{D}$.append($\hat{\boldsymbol{x}}_{\boldsymbol{i}}$)
    for each $\boldsymbol{x_j} \in \hat{D}$, $\boldsymbol{x_i} \neq \boldsymbol{x_j}$ **do**
      $\boldsymbol{e_j}$ = encode_feature($\boldsymbol{x_j}$)
      t_{bin}=Similarity($\boldsymbol{TRM}$,$\boldsymbol{e_i}$,$\boldsymbol{e_j}$)
      $\hat{\boldsymbol{x}}_{\boldsymbol{i,j}} = t_{bin}(\boldsymbol{x_i}, \boldsymbol{x_j})$
      $\hat{D}$.append($\hat{\boldsymbol{x}}_{\boldsymbol{i,j}}$)
    end for
  end for
end for
$e_{\hat{D}}$ = encode_dataset($\hat{D}$)
t_{scaler} = Similarity($\boldsymbol{TRM}$, $e_{\hat{D}}$)
$\hat{D} = t_{scaler}(\hat{D})$

The order value of the transformation function is related to the number of times a feature is processed by a transformation, e.g., an input feature $\boldsymbol{x_1}$ is given as an argument of the *log* function, so $f_1(\boldsymbol{x_1}) = log(\boldsymbol{x_1})$. Then, the resulting feature is combined with another feature $\boldsymbol{x_2}$, lets say a multiplication, thus, $f_2(f_1(\boldsymbol{x_1}), \boldsymbol{x_2}) = mult(log(\boldsymbol{x_1}), \boldsymbol{x_2})$. Finally, the output feature is given to the unary function *square*. Thus, the final transformed feature $\hat{\boldsymbol{x}}$ has an order of 3, and can be seen as follows:

$$\hat{\boldsymbol{x}} = f_3(f_2(f_1(\boldsymbol{x_1}), \boldsymbol{x_2})) = square(mult(log(\boldsymbol{x_1}), \boldsymbol{x_2}))) \tag{5}$$

Hence, we look for the underlying information about data through the extraction of more complex features. This gives us the capability of creating novel features from raw features that apparently do not have any predictive power, but in combination with high-order functions can have suitable predictive power for some machine learning models.

Causal Feature Selection Phase. Once the $\boldsymbol{TRM}$ is trained, MACFE is ready to recommend useful transformations for new datasets and features. For this, we start selecting the most promising original features, a causality-based

feature selection is performed on the features. A DAG Classifier is trained to discover a causal graph from data. For this, we use the implementation of CausalNex [2]. This graph underlies the causal relationship between features and a target variable. The mean identified causal magnitude effect of the features on the target is used to rank the features. Then, a given threshold hyperparameter s determines the top k selected features. The resulting subset of selected features are processed to obtain an encoding $\boldsymbol{e}$ (Eq. 3).

Then, for a given feature encoding $\boldsymbol{e}$, we search for a transformation in $\boldsymbol{TRM}$ by retrieving the most similar feature encoding using the *cosine distance* as a similarity measure. We benefit from this measure for ranking the most similar feature-vectors in the range 1.0 for identical feature-vectors and 0.0 for orthogonal feature-vectors [26]. Next, the most similar feature transformation is applied to the feature. The process is followed by the binary transformations and iterating over the features in the dataset (Alg. 2). Furthermore, a depth d hyperparameter is set to look for the maximum transformation order across *unary* and *binary* functions. Lastly, for the *Scaling transformations* we refer to those transformations on features that change the scale on a standard range. Many machine learning algorithms struggle to find patterns in data when features are not on the same scale. For this, having scaled features can help gradient descent to converge faster towards a minimum.

We scale features as follows. For a given feature $\boldsymbol{x} \in \boldsymbol{X}$, the following scaling functions can be applied. *Normalization*, also called *Min-Max Scaler*, is a method that scales each feature value to the range [0,1]. *Standardization*, this method scales each feature value so that the mean is 0 and the standard deviation is 1. *Robust Scaler*, this scaler is useful when the input feature has a lot of *outliers*. The Robust Scaling is done by calculating the median (50^{th} percentile), and also the 25^{th} and 75^{th} percentiles. Then, each feature value is subtracted from the median, and divided by the Interquartile Range (IQR). In order to learn and recommend which scaler is appropriate for a given dataset, we follow a series of data testings. First, we test the features to know the proportion of outliers. If this proportion is larger than a certain threshold γ, then a Robust Scaler is applied to the features. Secondly, if the data follows a normal distribution, then we use a Standard Scaler. In particular, we use a *Shapiro-Wilk* test [11] to evaluate the normality of data. Then, if the test value is greater than 0.05 we consider the data is normally distributed. Finally, if none of the above tests is true about the data, then we use a Min-Max Scaler on the features. The resulting scaling method is saved in TRM according to the dataset encoding.

5 Experimental Results

For the evaluation of MACFE, first, we describe the evaluation details, such as the case study datasets and learning algorithms. Next, we briefly describe each of the implementation details of the classifiers and evaluation methods. Finally, a comparison with previous work is done and a discussion is presented by analyzing the characteristics of datasets and algorithms where MACFE is convenient.

5.1 Evaluation Details

Table 1: Statistics of 14 Case Study Datasets

ID	Dataset	Source	Labels	Features	Instances
1	*Pima Diabetes*	UCI ML	2	8	768
2	*Sonar*	UCI ML	2	60	208
3	*Ionosphere*	UCI ML	2	34	351
4	*Haberman*	UCI ML	2	3	306
5	*Fertility*	UCI ML	2	9	100
6	*Wine*	UCI ML	3	13	178
7	*E.coli*	UCI ML	8	7	336
8	*Abalone*	UCI ML	29	7	4177
9	*Dermatology*	UCI ML	4	34	366
10	*Libras*	UCI ML	15	90	360
11	*Optical*	UCI ML	10	64	5620
12	*Waveform*	OpenML	3	21	5000
13	*Fourier*	OpenML	10	76	2000
14	*Pixel*	OpenML	10	240	2000

The evaluation of MACFE as an automated feature engineering method is performed on a set of fourteen classification datasets and eight machine learning algorithms commonly cited in the literature [15, 16, 30]. These datasets are from different areas, such as medical, physical, life, and computer science. In addition, these datasets are publicly available in the UCI ML Repository [7] and OpenML Repository [29]. The main statistics of these datasets are shown in Table 1.

5.2 Implementation Details

For our experiments, we tested the following learning algorithms: Logistic Regression *(LR)*, K-Nearest Neighbors *(KNN)*, Linear Support Vector Machine *(SVC-L)*, Polynomial Support Vector Machine *(SVC-P)* and Random Forest *(RF)*, AdaBoost *(AB)*, Multi-layer Perceptron *(MLP)* and Decision Tree *(DT)*. The scoring method for the evaluations is the mean accuracy of stratified 5-Fold Cross Validation on each dataset. Same as the state-of-the-art methodology for scoring. Each algorithm is used with scikit-learn [23] default parameters. This is because our objective is to enhance the accuracy of a model by improving the data through our automated feature engineering process, MACFE.

5.3 Comparison with previous works

The comparison of our proposal takes into account the same scenario conditions of the results presented in recent feature engineering proposals such as TFC [25],

Table 2: Mean accuracy results in 5-fold cross validation among original datasets (ORIG) and consulted state-of-the-art (TFC [25], FCTree [9], ExploreKit [15], AutoLearn (AL) [16], LbR [30]) and MACFE (ours). The best performing approach is shown in bold, each dataset is shown with its corresponding ID from Table 1.

D. ID	CLF	ORIG	TFC [25]	FCT [9]	EK [15]	AL [16]	LbR [30]	MACFE
1	KNN	71.48	72.42	73.52	73.6	68.36	72.13	**75.12**
	LR	76.55	75.92	76.52	73.9	74.99	71.86	**77.47**
	SVM-L	65.23	62.71	72.52	73.7	74.85	75.22	**77.34**
	SVM-P	64.89	65.71	70.52	72.6	76.32	**78.32**	78.12
	RF	75.37	72.42	73.52	74.0	73.05	72.47	**78.12**
	AB	74.34	74.08	74.08	74.3	72.52	73.01	**76.29**
	NN	64.32	64.12	64.22	67.3	72.39	72.50	**77.86**
	DT	**72.38**	70.23	70.46	70.9	71.05	71.12	71.74
2	KNN	78.35	81.48	82.70	82.4	83.19	**83.33**	81.27
	LR	77.42	78.12	78.72	78.7	79.00	**90.47**	86.05
	SVM-L	73.54	74.54	75.75	76.1	77.30	**90.47**	86.06
	SVM-P	53.36	58.41	66.44	33.6	81.71	80.95	**86.57**
	RF	73.55	81.00	82.54	47.4	77.87	76.19	**85.62**
	AB	80.74	80.00	81.04	54.0	78.83	**85.71**	83.69
	NN	80.30	81.07	82.00	82.4	84.09	85.71	**88.03**
	DT	75.01	74.23	74.52	75.0	75.02	**83.33**	75.53
3	KNN	84.31	84.66	84.87	86.0	83.46	**92.95**	89.74
	LR	87.44	87.26	87.39	87.7	87.95	**95.77**	93.44
	SVM-L	87.44	86.71	87.78	88.0	84.30	90.14	**92.58**
	SVM-P	64.10	70.16	71.45	72.6	74.63	78.87	**93.72**
	RF	93.15	91.65	93.16	94.0	92.30	91.54	**95.44**
	AB	92.02	90.94	90.12	90.3	92.43	90.14	**94.01**
	NN	93.14	92.45	92.13	93.6	92.29	**97.18**	92.58
	DT	88.32	87.12	88.04	88.1	88.59	88.73	**94.87**
4	KNN	71.89	70.00	71.28	72.3	68.68	70.36	**76.14**
	LR	74.19	72.07	73.96	74.5	76.16	**76.50**	74.51
	SVM-L	74.18	73.97	74.18	75.4	75.82	**76.01**	73.53
	SVM-P	74.18	73.98	74.81	75.1	**75.52**	**75.52**	74.51
	RF	68.63	68.91	69.07	70.0	65.34	70.17	**72.22**
	AB	70.25	71.19	71.57	72.2	69.93	**73.05**	71.89
	NN	73.19	69.02	71.19	72.2	70.91	75.02	**76.13**
	DT	66.65	66.09	66.79	67.2	66.34	67.74	**73.86**
5	KNN	85.00	86.00	86.00	87.0	87.00	88.00	**88.95**
	LR	88.00	88.00	89.00	88.0	87.00	88.00	**89.95**
	SVM-L	85.00	87.00	88.00	87.0	87.00	87.00	**89.95**
	SVM-P	88.00	87.00	87.00	88.0	88.00	88.00	**88.89**
	RF	82.00	87.00	87.00	**90.0**	84.00	88.00	89.89
	AB	79.00	83.00	84.00	83.0	79.00	85.00	**87.84**
	NN	88.00	88.00	88.00	88.0	85.00	88.00	**90.00**
	DT	80.00	84.00	84.00	85.0	85.00	**88.00**	87.89
6	KNN	67.93	74.89	79.93	83.4	93.84	95.49	**97.22**
	LR	95.52	96.89	97.24	95.1	98.30	**99.44**	98.87
	SVM-L	83.03	88.14	89.94	90.8	98.31	**98.87**	98.32
	SVM-P	96.65	96.68	96.65	92.1	92.68	94.74	**99.43**
	RF	96.07	96.68	97.12	90.0	97.20	**98.89**	97.19
	AB	85.82	88.12	**91.23**	62.8	84.71	83.03	89.27
	NN	42.73	46.23	49.56	64.6	97.19	**98.87**	98.32
	DT	91.57	91.79	92.01	92.5	93.22	93.37	**95.49**
7	KNN	86.59	**88.42**	87.56	88.4	84.82	85.39	87.81
	LR	75.88	78.23	79.24	82.8	87.19	87.19	**87.73**
	SVM-L	85.71	85.71	85.71	86.3	86.30	86.80	**88.87**
	SVM-P	56.54	59.32	62.14	72.3	80.33	81.59	**93.72**
	RF	82.73	83.46	83.76	85.1	86.59	84.80	**95.44**
	AB	62.47	63.54	64.37	65.8	65.75	63.06	**93.44**
	NN	78.28	80.37	81.97	83.7	86.90	86.89	**92.30**
	DT	79.74	76.32	77.67	80.3	76.40	82.11	**94.87**
8	KNN	**23.27**	21.64	22.60	23.1	22.71	21.69	22.62
	LR	24.61	23.69	23.60	24.8	26.50	25.25	**26.84**
	SVM-L	25.71	25.64	25.72	25.7	26.07	25.23	**26.57**
	SVM-P	19.46	17.64	22.12	21.4	23.77	23.98	**26.33**
	RF	22.91	18.78	23.02	23.2	22.21	24.15	**25.52**
	AB	20.61	19.10	19.97	21.1	20.61	21.01	**21.45**
	NN	27.53	26.32	26.41	27.1	27.81	26.40	**28.27**
	DT	19.27	19.00	19.13	19.3	19.41	19.42	**20.13**
9	KNN	89.11	90.46	92.89	91.0	96.09	96.66	**97.82**
	LR	97.21	97.76	97.97	97.6	**98.61**	97.77	97.81
	SVM-L	97.21	96.02	96.27	96.3	96.92	**98.32**	97.54
	SVM-P	94.41	94.00	94.12	92.0	93.56	**98.04**	95.90
	RF	96.92	96.45	96.61	95.5	95.81	98.04	**98.09**
	AB	54.13	57.12	61.00	57.3	54.96	54.13	**75.67**
	NN	98.04	97.13	97.22	97.7	**98.22**	97.77	98.09
	DT	95.24	95.06	94.96	95.4	94.68	96.08	**96.45**
10	KNN	70.00	71.00	71.18	73.7	69.44	70.27	**75.28**
	LR	60.27	64.68	67.12	71.7	70.00	68.88	**79.72**
	SVM-L	68.61	69.88	70.83	70.4	67.22	68.61	**82.22**
	SVM-P	2.22	36.68	47.97	47.8	49.44	50.13	**85.83**
	RF	71.94	72.12	73.07	77.6	70.22	72.50	**86.11**
	AB	8.05	10.12	13.11	16.9	**18.05**	14.57	15.28
	NN	71.66	72.35	74.24	75.7	78.33	**85.56**	83.06
	DT	62.50	62.64	63.12	63.7	65.55	65.27	**73.06**
11	KNN	98.77	97.20	98.02	98.0	97.03	**99.03**	98.74
	LR	96.49	96.40	96.40	97.0	95.83	94.82	**97.88**
	SVM-L	94.89	94.12	95.17	95.1	94.01	94.71	**98.49**
	SVM-P	99.09	99.03	99.03	99.1	96.20	**99.21**	99.06
	RF	96.38	96.36	96.91	97.3	96.57	92.68	**98.26**
	AB	68.65	67.62	68.35	69.7	73.78	**75.46**	68.17
	NN	98.02	95.62	95.37	96.5	96.93	96.77	**98.33**
	DT	89.90	88.00	88.46	90.4	90.41	87.42	**91.10**
12	KNN	**82.48**	81.28	82.00	82.1	81.14	81.54	81.44
	LR	86.58	86.72	**87.18**	86.9	85.12	87.14	86.90
	SVM-L	86.90	84.54	86.23	86.9	84.40	**87.18**	86.66
	SVM-P	81.70	81.62	82.54	80.4	85.42	**86.18**	83.84
	RF	82.10	81.45	82.04	82.1	81.12	80.90	**86.12**
	AB	83.62	82.54	82.84	83.0	**83.78**	83.04	83.34
	NN	85.84	82.31	3.10	84.7	83.72	83.94	**86.26**
	DT	75.04	72.46	73.00	73.2	73.06	76.60	**78.72**
13	KNN	83.85	82.17	83.82	**84.0**	83.55	82.17	82.85
	LR	79.45	79.97	80.00	82.2	83.15	**84.03**	82.20
	SVM-L	81.45	81.15	82.86	82.5	83.05	83.05	**84.40**
	SVM-P	8.70	42.25	57.97	66.7	82.30	81.10	**85.10**
	RF	79.90	78.90	79.16	80.8	79.31	81.85	**84.45**
	AB	48.65	46.66	49.29	50.0	**50.40**	48.65	43.75
	NN	81.90	82.34	83.12	83.4	85.50	**86.90**	83.40
	DT	74.00	74.00	74.00	74.1	74.35	74.50	**75.40**
14	KNN	97.75	**98.12**	97.23	98.0	97.95	97.45	97.75
	LR	94.35	94.22	94.28	95.5	95.75	94.95	**96.75**
	SVM-L	92.90	92.57	93.26	94.3	94.27	93.45	**97.70**
	SVM-P	98.35	98.22	**98.66**	**98.7**	97.25	**98.66**	98.10
	RF	95.50	94.26	95.12	96.5	94.20	95.50	**97.60**
	AB	54.05	54.00	54.86	55.3	55.60	54.05	**65.10**
	NN	97.15	97.15	97.15	97.2	97.15	**97.90**	97.20
	DT	87.30	86.12	86.78	86.6	87.65	87.90	**88.55**

FCTree [9], ExploreKit [15], AutoLearn [16] and LbR [30]. In Table 2 are shown the scores achieved by our proposal compared against the scores obtained by other approaches in the state-of-the-art. The best scores are shown in bold, each dataset is represented by its ID defined in Table 1. The improvement among algorithms and datasets is notable: as shown in Fig. 3 we achieve an average accuracy of 81.83% across all tested datasets and classifiers, outperforming TFC, FCT, ExploreKit, AutoLearn (AL), LbR, by 6.54%, 5.99%, 5.63%, 3.95%, and 2.71%, respectively.

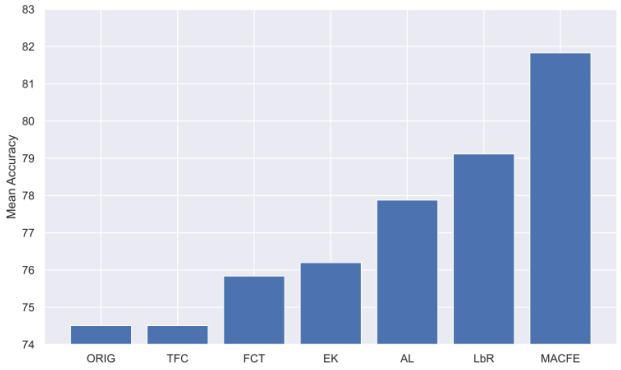

Fig. 3: Mean accuracy of state-of-the-art methods and MACFE (ours) across fourteen case study datasets and eight machine learning models.

5.4 Discussion

The transformation recommendation procedure of this method is agnostic of the learning algorithm. But, some transformations can be more appropriate for a certain algorithm. Therefore, MACFE achieves 100% of efficacy in terms of improving at least one model for each dataset. The depth hyperparameter d of MACFE can generate different orders of complex features to improve the model performance. A high value in d can result in too complex novel features, thus the algorithm cannot learn from the data. In contrast, a small value of the hyperparameter s can lead to a small subset of the original features, thus not finding good relationships between features. Hence, it is recommended a grid search to find the optimal values of hyperparameters.

6 Conclusions and Future Work

In this paper, we presented a causality-based feature selection to reduce the feature space search for feature transformations. Also, a meta-learning-based method for automated feature construction, on which the number of transformations executed on features depends on the number of useful transformations found on historical past similar features. In particular, this method has the capability of constructing novel features from raw data that are informative and useful for a learning algorithm. Hence, MACFE can automatically create features by applying selected transformations to the data, either unary, binary, or high-order, instead of applying all possible combinations of those. Hence, the feature explosion problem is minimized. However, MACFE has a fixed set of unary, binary, and scaling transformations. In future work, we intend to increase this set by adding more transformation functions, leading to the construction of more informative features from raw features. In addition, the causal selection of

features could be improved, since it is applied equally to all datasets but different datasets can be expected to satisfy different causal assumptions, which produces different levels of efficacy when selecting the features to be engineered. To improve this, better methods of general causal discovery are needed.

Acknowledgments

The authors wish to thank the CINVESTAV, the AI Hub, and the CIIOT at ITESM, for their support and the use of the DGX for running the experiments in this paper.

References

1. Alcobaça, E., Siqueira, F., Rivolli, A., Garcia, L.P., Oliva, J.T., de Carvalho, A.C., et al.: Mfe: Towards reproducible meta-feature extraction. Journal of Machine Learning Research **21**(111), 1–5 (2020)
2. Beaumont, P., Horsburgh, B., Pilgerstorfer, P., Droth, A., Oentaryo, R., Ler, S., Nguyen, H., Ferreira, G.A., Patel, Z., Leong, W.: CausalNex (10 2021), https://github.com/quantumblacklabs/causalnex
3. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Learnability and the vapnik-chervonenkis dimension. Journal of the ACM (JACM) **36**(4), 929–965 (1989)
4. Brazdil, P., Carrier, C.G., Soares, C., Vilalta, R.: Metalearning: Applications to data mining. Springer Science & Business Media (2008)
5. Chen, X., Lin, Q., Luo, C., Li, X., Zhang, H., Xu, Y., Dang, Y., Sui, K., Zhang, X., Qiao, B., et al.: Neural feature search: A neural architecture for automated feature engineering. In: 2019 IEEE International Conference on Data Mining (ICDM). pp. 71–80. IEEE (2019)
6. Domingos, P.: A few useful things to know about machine learning. Communications of the ACM **55**(10), 78–87 (2012)
7. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
8. Duda, R.O., Hart, P.E., et al.: Pattern classification. John Wiley & Sons (2006)
9. Fan, W., Zhong, E., Peng, J., Verscheure, O., Zhang, K., Ren, J., Yan, R., Yang, Q.: Generalized and heuristic-free feature construction for improved accuracy. In: Proceedings of the 2010 SIAM International Conference on Data Mining. pp. 629–640. SIAM (2010)
10. Filchenkov, A., Pendryak, A.: Datasets meta-feature description for recommending feature selection algorithm. In: 2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT). pp. 11–18. IEEE (2015)
11. Hanusz, Z., Tarasinska, J., Zielinski, W.: Shapiro-wilk test with known mean. REVSTAT-Statistical Journal **14**(1), 89–100 (2016)
12. Heaton, J.: An empirical analysis of feature engineering for predictive modeling. In: SoutheastCon 2016. pp. 1–6. IEEE (2016)
13. Horn, F., Pack, R., Rieger, M.: The autofeat python library for automated feature engineering and selection. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 111–120. Springer (2019)

14. Kanter, J.M., Veeramachaneni, K.: Deep feature synthesis: Towards automating data science endeavors. In: 2015 IEEE international conference on data science and advanced analytics (DSAA). pp. 1–10. IEEE (2015)
15. Katz, G., Shin, E.C.R., Song, D.: Explorekit: Automatic feature generation and selection. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). pp. 979–984. IEEE (2016)
16. Kaul, A., Maheshwary, S., Pudi, V.: Autolearn—automated feature generation and selection. In: 2017 IEEE International Conference on data mining (ICDM). pp. 217–226. IEEE (2017)
17. Khurana, U., Samulowitz, H., Turaga, D.: Feature engineering for predictive modeling using reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
18. Khurana, U., Turaga, D., Samulowitz, H., Parthasrathy, S.: Cognito: Automated feature engineering for supervised learning. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW). pp. 1304–1307. IEEE (2016)
19. Kuhn, M., Johnson, K.: Feature engineering and selection: A practical approach for predictive models. CRC Press (2019)
20. Kuo, F.Y., Sloan, I.H.: Lifting the curse of dimensionality. Notices of the AMS **52**(11), 1320–1328 (2005)
21. Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E.B., Turaga, D.S.: Learning feature engineering for classification. In: IJCAI. pp. 2529–2535 (2017)
22. Pearl, J.: Causality: Models, Reasoning and Inference. Cambridge University Press, 2nd edn. (2009)
23. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
24. Pinto, F., Soares, C., Mendes-Moreira, J.: Towards automatic generation of metafeatures. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 215–226. Springer (2016)
25. Piramuthu, S., Sikora, R.T.: Iterative feature construction for improving inductive learning algorithms. Expert Systems with Applications **36**(2), 3401–3406 (2009)
26. Qian, G., Sural, S., Gu, Y., Pramanik, S.: Similarity between euclidean and cosine angle distance for nearest neighbor queries. In: Proceedings of the 2004 ACM symposium on Applied computing. pp. 1232–1237 (2004)
27. Reshef, D.N., Reshef, Y.A., Finucane, H.K., Grossman, S.R., McVean, G., Turnbaugh, P.J., Lander, E.S., Mitzenmacher, M., Sabeti, P.C.: Detecting novel associations in large data sets. science **334**(6062), 1518–1524 (2011)
28. Rivolli, A., Garcia, L.P., Soares, C., Vanschoren, J., de Carvalho, A.C.: Towards reproducible empirical research in meta-learning. arXiv preprint arXiv:1808.10406 pp. 32–52 (2018)
29. Vanschoren, J., Van Rijn, J.N., Bischl, B., Torgo, L.: Openml: networked science in machine learning. ACM SIGKDD Explorations Newsletter **15**(2), 49–60 (2014)
30. Wang, M., Ding, Z., Pan, M.: Lbr: A new regression architecture for automated feature engineering. In: 2020 International Conference on Data Mining Workshops (ICDMW). pp. 432–439. IEEE (2020)
31. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Practical machine learning tools and techniques. Morgan Kaufmann **578**, 1 (2005)
32. Yu, K., Guo, X., Liu, L., Li, J., Wang, H., Ling, Z., Wu, X.: Causality-based feature selection: Methods and evaluations. ACM Computing Surveys (CSUR) **53**(5), 1–36 (2020)
33. Zheng, A., Casari, A.: Feature engineering for machine learning: principles and techniques for data scientists. " O'Reilly Media, Inc." (2018)