DIFER : Differentiable Automated Feature Engineering

Guanghui Zhu [*], **Zhuoer Xu**[*], **Xu Guo**, **Chunfeng Yuan** and **Yihua Huang**
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
{ganghui.zhu, zhuoer.xu, guoxu}@smail.nju.edu.cn, {cfyuan, yhuang}@nju.edu.cn

arXiv 2020

Total cites : 0

2021.05.25

최영제

Index

1 Introduction

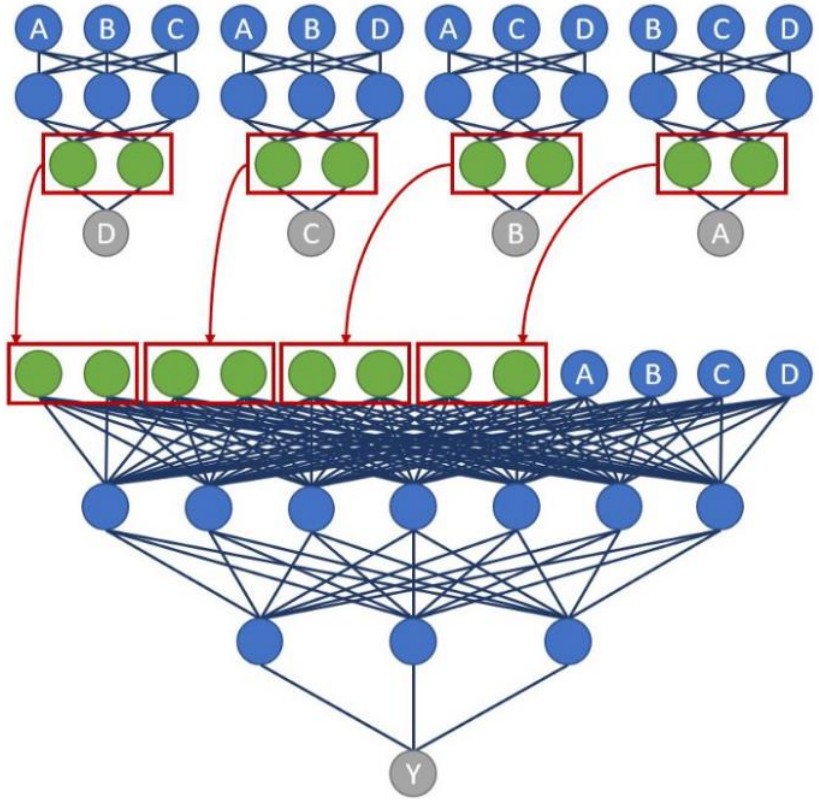
Introduction & related works

- A crucial step of machine learning for improve model performance
- In recent years, automated feature engineering (AutoFE) to replace expensive human labor
- More data beats clever algorithms, but better data beats more data
- DIFER outperforms the state-of-the-art AutoFE approach in terms of both model performance and computational efficiency.

- Heuristic approaches
  - Deep feature synthesis (DFS, 2015)
  - One buttong machine (2017)
  - Cognito (2016)
  - AutoFeat (2019)
- Learning based approaches
  - Learning feature engineering … (LFE, 2017)
  - TransGraph (2018)
- NAS based approaches
  - Neural architecture for … (NFS, 2019)

Introduction & related works

- Auto feature engineering example



Example Full feature network diagram by author

Model	RMSE	Cross Entropy	Error
Auto-feature	0.4314893	0.55944	37.2%
MLP - emb	0.4315610	0.55960	37.6%
XGBoost	0.4316558	0.55982	37.6%

Accuracy scores by model

출처 : https://towardsdatascience.com/automated-feature-engineering-using-neural-networks-5310d6d4280a

2 Methods

Problem formulation

- Dataset with a target vector (y) and raw features (F) → $D = \langle F, y \rangle \quad F = \{f_1, \cdots, f_d\}$
- Performance of machine learning model (M) learned from datasets and measured by an evalutation metric (L) → $L_M(F, y)$
- Composition of transformations for extraction new features → $t \in \mathcal{R}^n \times \cdots \times \mathcal{R}^n \rightarrow \mathcal{R}^n$
- Let o denote the arity of the transformation t extracted new feature → $f_i = t(f_{i_1}, \cdots, f_{i_o})$ f_{ij} denotes the j-th input of t to extract f_i
- A set of transformations with different arities → $T = \{t_1, \ldots, t_m\}$
- For examples,

$$T = \{square, devide\} \quad \rightarrow \quad f_i = t(height, weight) \quad \rightarrow \quad f_i = divide(weight, square(height))$$

Given a set of transformations with different arities $T = \{t_1, \cdots, t_m\}$, we define the feature space F^T as follows: For $\forall f_i \in F^T$, f_i satisfies any of the following conditions:

- $f_i \in F$
- $\exists t \in T,\ f_i = t(f_{i_1}, \cdots, f_{i_o})$, where $f_{i_1}, \cdots, f_{i_o} \in F^T$

Formally, let $\alpha(f_i)$ denote the *order* of the feature $f_i \in F^T$, $\alpha(f_i)$ can be defined as:

$$\alpha(f_i) = \begin{cases} 1 + \max_j \alpha(f_{i_j}) & f_i = t(f_{i_1}, \cdots, f_{i_o}) \\ 0 & f_i \in F \end{cases} \quad (1)$$

$$F^* = \arg\min_{\hat{F}} L_M(F \cup \hat{F}, y), \text{ s.t. } \hat{F} \subset F^T \quad (2)$$

In practice, we limit the order of features and search in a smaller feature space $F_k^T = \{f \mid f \in F^T \wedge \alpha(f) \leq k\}$ since the size of the original space is infinite (i.e., $|F^T| = \aleph_0$). We explore the feature space F_k^T and search for top features ranked by the performance metric $L_M(F \cup \{f\}, y)$ as F^*.

Methods

Overview of DIFER

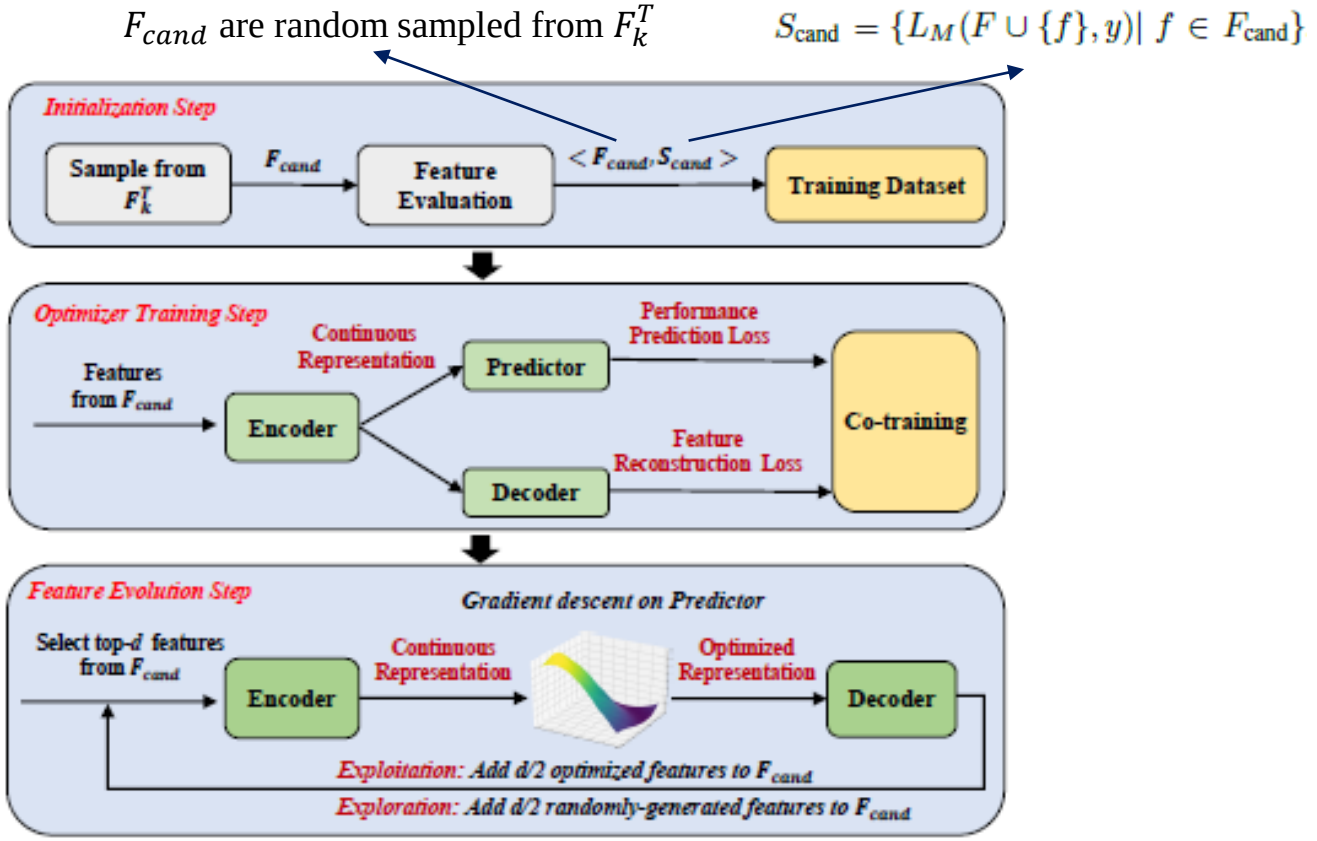


Figure 1: Overview of DIFER

2 Methods

Optimizer training step

- Encoder의 역할은 post-order traversal strings를 continuous representation으로 변환하는 것 $\psi_e \in \mathcal{X} \to \mathcal{E}$
- Post-order traversal strings란 사용된 변수, 연산의 종류만을 표현하여 순서에 상관없는 string을 말함
- Can be viewed as a way of data augmentation
- x_r denote each token in the strings

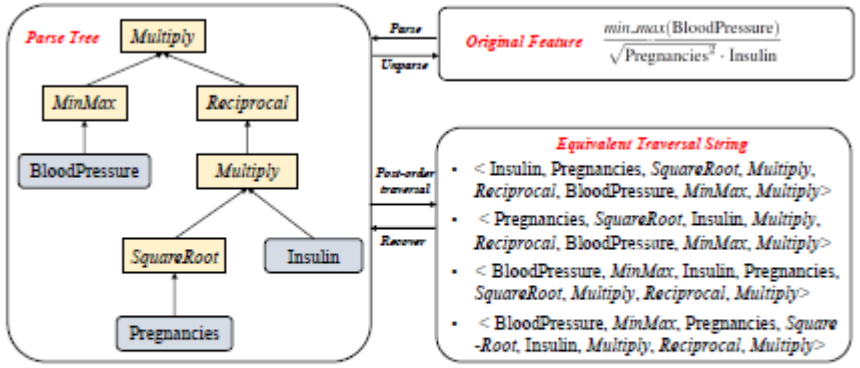
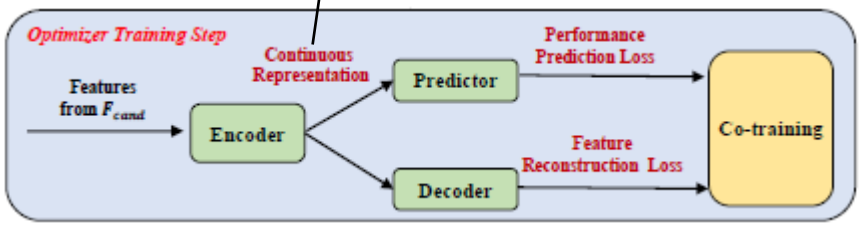


Figure 2: Parse tree and post-order traversal strings of the feature $\frac{min\_max(BloodPressure)}{\sqrt{Pregnancies^2} \cdot Insulin}$ in the *PimaIndian* dataset.

Sum aggregate each hidden state

$$H_x = \{h_1, h_2, \cdots, h_{|x|}\}$$



Methods

Optimizer training step

- Predictor는 target을 예측하는 것이 아닌 기존 feature의 예측 성능이였던 s_x를 예측함

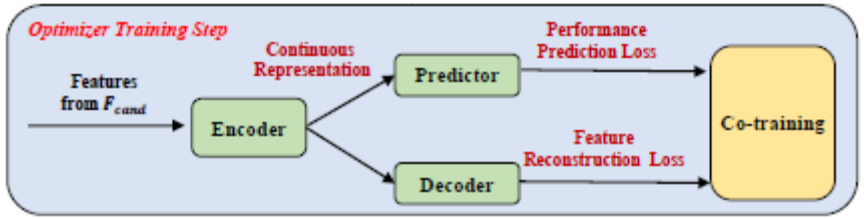
$$\psi_p \in \mathcal{E} \rightarrow \mathcal{R} \qquad \text{score } s_x \text{ measured by } L_M(F \cup \{f\}, y). \qquad \mathcal{L}_{pp} = \sum_x (s_x - \psi_p(\psi_e(x)))^2$$

- Decoder는 입력 string을 복원함

$$\psi_d \in \mathcal{E} \rightarrow \mathcal{X} \qquad \mathcal{L}_{rec} = -\sum_x \sum_{r=1}^{|x|} \log P_{\psi_d}(x_r | \psi_e(x))$$

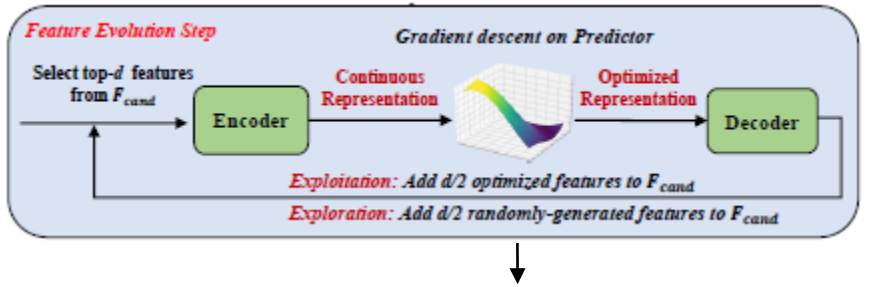
- 최종 loss는 다음과 같음

$$\mathcal{L} = \lambda \mathcal{L}_{pp} + \mathcal{L}_{rec} \qquad \lambda = \sum_{t=1}^{5} \mathcal{L}_{rec} / \sum_{t=1}^{5} \mathcal{L}_{pp}.$$



Feature evolution step

Based evaluation performance $\{L_M(F \cup \{f\}, y))$



d/2 = 최적화된 기존 feature

d/2 = 최적화되지 않은 다른feature

The process of feature evolution is repeated until a maximum number of iterations is reached

2 SIL Methods

Feature optimization

- After the convergence of the feature optimizer, we directly optimize the feature embedding in the continuous space by performing gradient descent and then decode the optimized embedding into a new feature
- Starting from the extracted feature x, we optimize e_x to get a new embedding in the continuous space along the gradient direction induced by ψ_p

$$e_{x'} = \sum_{h_r \in H_x} \left(h_r - \eta \frac{\partial \psi_p}{\partial h_r} \right)$$

- However, due to the nature that the corresponding parse tree of a feature may have several equivalent post-order traversal strings $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$, the strings in X are highly similar in the continuous space

$$\exists \epsilon,\ \forall x^{(i)} \in X, \| e_{x^{(i)}} - \frac{1}{n} \sum_{x^{(j)} \in X} e_{x^{(j)}} \|_2 \leq \epsilon$$

- After one step of gradient descent, $e_{x'}$ may still be in the neighborhood of X
- To address this problem, apply the process of optimization in Equation (6) multiple times with a small learning rate until we get new parse trees

Experiments and Results

Experimental setting

- Use 23 public datasets from OpenML, UCI repository, and Kaggle
- 13 classification (C) datasets / 10 regression (R) datasets / numbers of features (5 to 57) / instances (100 to 30000)
- Set the max order k to 5 except in RQ3 and utilize 9 transformation functions totally
  - Unary transformation: *logarithm*, *square root*, *min-max normalization* , and *reciprocal*
  - Binary transformation: *addition*, *subtraction*, *multiplication*, *division*, and *modulo*
- evaluate the AutoFE method, we use the performance metric $(1 - (relative\ absolute\ error))$ and $f1 - score$

- **RQ1**: How effective is the proposed DIFER approach?
- **RQ2**: How efficient is DIFER to explore the feature space?
- **RQ3**: Howe effective are the high-order features extracted by DIFER?
- **RQ4**: Can DIFER improve the performance of different machine learning algorithms?

3 SIL Experiments and Results

Results – RQ1

5-fold cross-validation

Dataset	Source	C/R	Instances\Features	Base	Random	DFS†	AutoFeat†	NFS†	DIFER
Openml_586	OpenML	R	1000\25	0.6617	0.6511	0.6501	0.7278	0.7401	**0.7683**
Openml_589	OpenML	R	1000\25	0.6484	0.6422	0.6356	0.6864	0.7141	**0.7727**
Openml_607	OpenML	R	1000\50	0.6344	0.6285	0.6388	0.6699	0.6870	**0.6918**
Openml_616	OpenML	R	500\50	0.5747	0.5714	0.5717	0.6027	0.5915	**0.6554**
Openml_618	OpenML	R	1000\50	0.6267	0.6167	0.6343	0.6324	0.6400	**0.6603**
Openml_620	OpenML	R	1000\25	0.6336	0.6178	0.6263	0.6874	0.6749	**0.7442**
Openml_637	OpenML	R	1000\25	0.5136	0.5268	0.5191	0.5763	0.5693	**0.6343**
Bikeshare DC	Kaggle	R	10886\11	0.8200	0.8436	0.8214	0.8498	0.9746	**0.9813**
Housing Boston	UCIrvine	R	506\13	0.4336	0.4446	0.3412	0.4688	**0.5013**	0.4944
Airfoil	UCIrvine	R	1503\5	0.4962	0.5733	0.4346	0.5955	0.6163	**0.6242**
PimaIndian	UCIrvine	C	768\8	0.7566	0.7566	0.7501	0.7631	0.7839	**0.7865**
SpectF	UCIrvine	C	267\44	0.7750	0.8277	0.7906	0.8161	0.8501	**0.8612**
German Credit	UCIrvine	C	1001\24	0.7410	0.7550	0.7490	0.7600	**0.7818**	0.7770
Ionosphere	UCIrvine	C	351\34	0.9233	0.9344	0.9175	0.9117	0.9516	**0.9770**
Credit Default	UCIrvine	C	30000\25	0.8037	0.8060	0.8059	0.8060	0.8049	**0.8096**
Messidor_features	UCIrvine	C	1150\19	0.6584	0.6878	0.6724	0.7359	0.7461	**0.7576**
Wine Quality Red	UCIrvine	C	999\12	0.5317	0.5641	0.5478	0.5241	**0.5841**	0.5824
Wine Quality White	UCIrvine	C	4900\12	0.4941	0.4930	0.4882	0.5023	0.5150	**0.5155**
SpamBase	UCIrvine	C	4601\57	0.9102	0.9237	0.9102	0.9237	0.9296	**0.9339**
Credit-a	UCIrvine	C	690\6	0.8377	0.8449	0.8188	0.8391	0.8652	**0.8826**
Fertility	UCIrvine	C	100\9	0.8530	0.8300	0.7500	0.7900	0.8700	**0.9098**
Hepatitis	UCIrvine	C	155\6	0.786	0.8300	0.8258	0.7677	0.8774	**0.8839**
Megawatt1	UCIrvine	C	253\37	0.8890	0.8973	0.8773	0.8893	0.9130	**0.9171**
Avg. Eval. Num.								160,000	**4,096**

Table 1: Comparison between DIFER and the existing AutoFE methods († the results obtained using the open-sourced code)

Dataset	Instances\Features	LFE*	NFS†	DIFER
Credit-a	690\6	0.771	0.8652	**0.8653**
Feritility	100\9	0.873	0.87	**0.9098**
Hepatitis	155\6	0.831	0.8774	**0.8839**
Ionosphere	351\34	0.932	0.9516	**0.9770**
Megawatt1	253\37	0.894	0.913	**0.9171**
SpamBase	4601\57	**0.947**	0.9296	0.9339

Table 2: Comparison between DIFER, LFE, and NFS (* the results reported in the paper)

Experiments and Results

Results – RQ2

Performance when limiting the number of variables

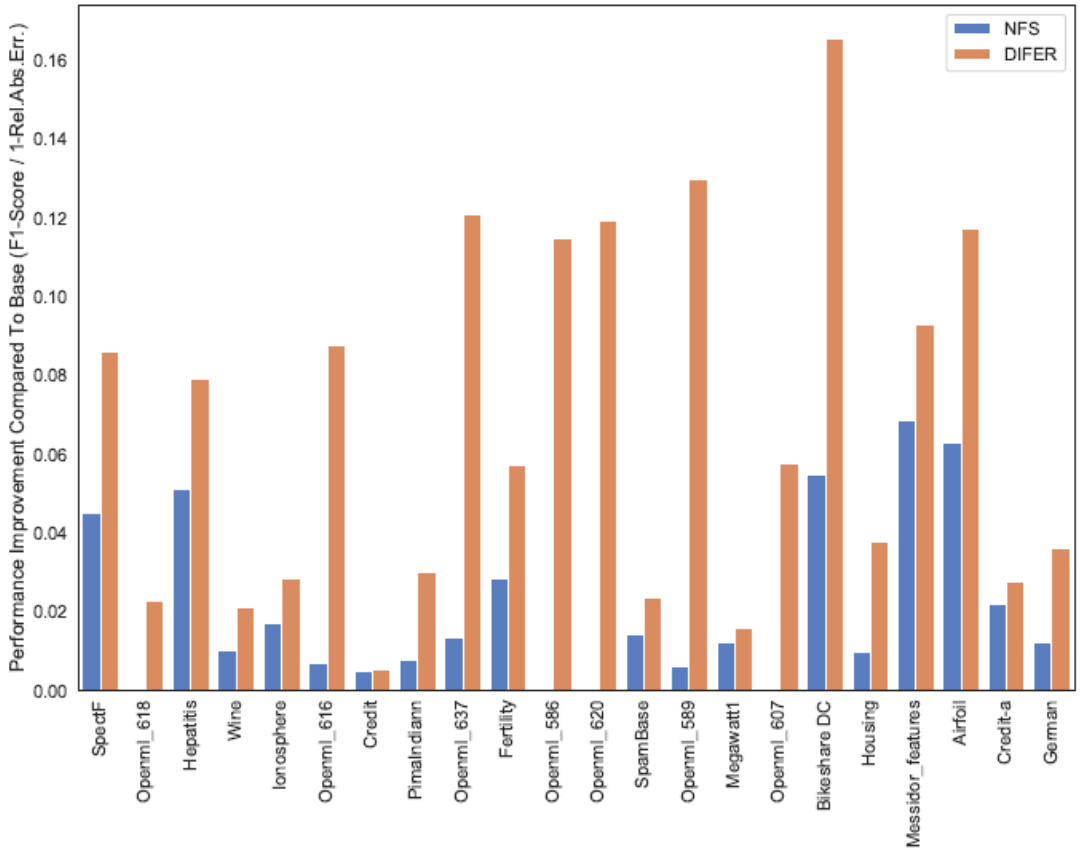


Figure 3: Comparison results between NFS and DIFER. The number of feature evaluations is restricted to 3500.

3 Experiments and Results

Results – RQ3

- High-order features means that they have large k

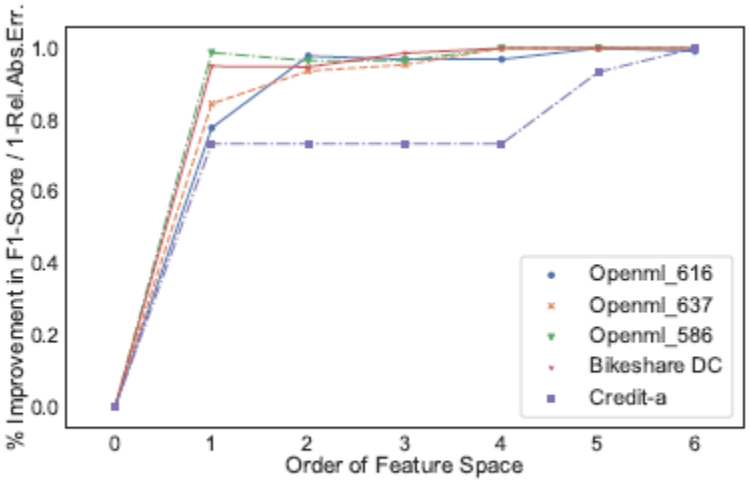


Figure 5: Effect of the high-order feature space

3 Experiments and Results

Results – RQ4

Task	Algorithm	Avg Impr	Min/Max Impr
C	LogisticRegression	5.94%	1.01% / 15.94%
	LinearSVC	13.98%	3.17% / 22.32%
	XGBoost	6.85%	0.30% / 27.98%
	LightGBM	7.69%	0.16% / 32.63%
R	LassoRegression	14.59%	1.22% / 66.66%
	LinearSVR	32.74%	13.21% / 96.98%
	XGBoost	13.44%	3.20% / 67.06%
	LightGBM	15.48%	4.75% / 71.92%

Table 3: Statistics on performance of DIFER with different machine learning algorithms on classification (C) and regression (R) tasks. The metrics for classification and regression tasks are F1-Score and Relative Absolute Error.

Experiments and Results

Future work

For future work, we plan to automatically search for transformations based on feature types

Q&A