ON THE IRRELEVANCE OF MACHINE LEARNING ALGORITHMS AND THE IMPORTANCE OF RELATIVITY

Carlos Huertas, Qi Zhao

Amazon Science
{carlohue,qqzhao}@amazon.com

ABSTRACT

Information explosion has brought us a wide range of data formats and machine learning keeps in constant evolution to develop mechanisms to extract knowledge from them. Modern models in the Deep Learning space have proven to be very successful in multiple applications, yet in the tabular space they fail to provide consistent competitive performance. However, in this work we claim model selection can become irrelevant as the key tends to lie in data processing. In this paper we introduce the concept of relativity in feature engineering, a powerful methodology to boost any classifier performance and we provide over 30 different configurations of models and feature engineering designs to prove we can bias any result to help an arbitrary model score best. Our results attribute 600% more value to feature engineering than model selection. In order to validate the effectiveness of our approach, we submitted our work to a live machine learning competition with outstanding results regardless of our model of choice.

Index Terms— Feature Engineering, GBDT, DNN, Tabular Data

1. INTRODUCTION

As humans, we are exposed to multiple ways to explore the world that surrounds us, therefore for real-world applications it is very common to have access to a wide variety of data sources and when combined it gives origin to multimodal tasks which have been a recent field of study for the machine learning community[1]. Two of the most well studied fields are Natural Language Processing (NLP) and Computer Vision (CV), both with groundbreaking developments like the introduction of transformers[2] and the easy access to large-scale pre-trained models like DeBERTa[3] have pushed the state-of-the-art (SOTA) and gave deep learning (DL) a huge positional advantage when it comes to model selection. The CV field follows a very similar trend with convolutional neural networks (CNN), providing outstanding performance in many applications, and although computational expensive, architectures like EfficientNets[4] have found great success balancing the performance-vs-cost trade-off. Vision Transformers (ViT) have found their way into CV as well[5] with recent deliveries like Next-ViT[6] aiming to bridge the gap that still separates ViT from CNN in terms of efficiency in the latency/accuracy trade-off.

From the current SOTA review, we would argue that deep learning has an edge for unstructured-homogeneous data (like text and images), however multimodal data can come in all sort of sources, including tabular which continues to be the most popular form of data[7, 8]. For tabular data (TD), deep learning has not been able to overcome the success of gradient boosted decision trees (GBDT) [9, 10, 11] and although neural networks (NN) provide additional theoretical properties over GBDT[12], and in some cases can provide the best performance, they hardly provide *consistent* improvements, with bias detected on independent research where DL solutions only tend to outperform when tested on their own specific datasets[13].

The case of tabular data is peculiar as feature engineering (FE) is not as well defined unlike NLP or CV. Even if tools exist to automatically build features[14], the importance that proper feature engineering causes on a system can shift dramatically the performance of any classifier, hence, allowing potential bias in model evaluation comparison, as FE for TD can take a multitude of forms.

In this work, we introduce the concept of relativity and how to leverage it in the process of FE. We further present how powerful state-of-the-art auto feature engineering may fail to uncover this property and the impact it brings to a variety of classifiers. Our extensive experimentation includes performance summaries for over 30 configurations of models & feature sets but more important, it shows the relative irrelevance of classifier selection when compared to the value FE brings to a system.

2. RELATED WORK

The importance that feature engineering (FE) plays in a ML system has been studied in the past and there has always been an interest to understand how a given model can change its performance depending on the feature-space, this brought us the field of Feature Selection (FS) that dates back at least three decades, with particular booms around late 90's [15, 16]. The

studies on FS have shown that there is a direct-link between performance and features, especially in the presence of redundant and noisy features that tend to lead a loss in generalization, in order to improve on this, techniques like wrapper approaches were developed[16] but they tend to be too expensive for modern standards, and usually filter methods are preferred[17]. With the recent advances in technology, modern algorithms like GBDT have feature selection built-in[10] giving the capability for algorithms to pick the relevant features on their own[18].

Even if the interest on FS has faded over the years, the study of features and their contribution to a better model remains very much alive, and one of the reasons is the curse of dimensionality [19], as the data we collect is pushing us to automate the process of FE which has traditionally been manual[20]. Domain expertise is usually required to properly build features, and it continues to provide very powerful gains compared to naive model fitting[21], i.e., model trained on raw features. In many cases these improvements are way beyond what model selection could have achieved; this might suggest selecting the right features might be more important that picking a model. In recent studies[22], the contribution of FE has been compared to other properties of model building like training size, with features having over 2x the contribution achieved by data increase, these findings keep supporting the idea that FE is a critical and irreplaceable step in model building.

Although there are many FE tools, some are restricted to specific applications, notably FeatureTools[23] which is mostly designed for relational data aggregations, or TFresh[24] that focuses on time series, in this work we focus on flat data for its generalization, e.g., after transformations we could turn any data into a flat-format. To this end, two main works are of particular relevance: AutoFeat[25] combines both feature generation and selection in a single package and while it has shown to improve the performance of linear classifiers, it is unable to bring value when compared to stronger models as stated by their own researchers, thus rendering this into a simplification tool rather than a performance option, on the other hand, OpenFE[14] focus strongly in performance, claiming their methodology outperforms even carefully crafted human engineering. In our experiments we dive further on the ability of OpenFE to find meaningful relations in the presence of relativity, which we will introduce in the following section.

3. PROPOSED METHODOLOGY

Tabular data, unlike other forms like images and audio tends to be agnostic to position, i.e., there is no spatial information, this is usually true in most tabular datasets. However, in cases where there is an underlying logic behind each column (feature), traditional Feature Engineering (FE) fails to extract the proper information out of the data, as such information is relative to the observer, this concept we denote later as "relativity".

In order to formalize our proposal, we start from the concept of influential features[26], an understanding of how a shift in a given feature has an impact on the test loss, this is shown to have a closed-form as follows:

$$I_{up,loss}(z, z_{test}) = -\nabla_\theta L(z_{test}, \hat{\theta}) \top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \quad (1)$$

Where the influence (I) of upweighting($_{up}$) z on the loss (L) at a test point z_{test} is defined give the parameters θ and Hessian H. Based on this definition, we can then approximate the entire validation set impact as follows[27]:

$$\sum_{j=1}^{m} L(z_j, \hat{\theta_{\delta i}}) - \sum_{j=1}^{m} L(z_j, \hat{\theta}) \approx [\sum_{j=1}^{m} \phi(z_i, z_j)]\delta_i \quad (2)$$

Where m is the validation set size and δ represents a given perturbation. This leads to the possibility of defining instance-wise influential features[27] which can effectively be used as a mechanism to quantify the real overall power of a given variable. However, in this work we present an extension to generalize further the concept of influential features that takes into account the particular available projections a feature might have.

In order to understand further multi projections, we now introduce the concept of *relativity*, defined as: *the capacity of a given feature to be projected under different observers*. Each observer is an arbitrary reference point to compare a give feature. Although there might be an unlimited set of theoretical observers, in this work we focus on two main ones:

- Positional Observer: aims to exploit the information behind a known underlying logic of the positions of features in a given matrix. This type of relativity is usually exploited by re-arranging and/or grouping features as shown in Table 1 and 2.
- Fractional Observer: seeks to represent data as fractional value of a reference, usually such reference can be another feature in our matrix but not required. This relativity is usually easy to find by a brute-force approach, e.g., by iterating over all possible pair combinations. However, feature selection is critical to remove all the generated noise. In Table 3 and 4 we provide an common example of this transformation.

A practical example for positional observer transformation could be to leverage temporal patterns e.g., a range of sensors that trigger in a particular order given an event E. We could format this as $sensor_1$ to $sensor_i$ in a tabular space, where i indicates the sensor number, yet, this can also be represented as $trigger_1$ to $trigger_n$ where n represents the order in which a given sensor trigger. This can be seen visually in

Tables 1 & 2, both tables present the same information, only changing the $observer$.

Table 1. Sensor trigger with sensor observer

SensorOne	SensorTwo	SensorThree
1	3	2
3	1	2
2	3	1

While Table 1 uses a $sensor$ observer, which looks at each sensor to record in which order they triggered, Table 2 has access to the same original data just with a different $observer$ at the $trigger$ level to add additional spatial information to our matrix. This tends to improve overall model performance by extracting different information from the same original data.

Table 2. Sensor trigger with trigger observer

TriggerOne	TriggerTwo	TriggerThree
$Sensor_1$	$Sensor_3$	$Sensor_2$
$Sensor_2$	$Sensor_3$	$Sensor_1$
$Sensor_3$	$Sensor_1$	$Sensor_2$

Unlike positional observers which might require more involvement, fractional observers are much easier to automate and popular tools like OpenFE[14] are useful for this technique. In its most simple form it involves taking a reference point as a fixed denominator over a series of columns. In Table 3 we show a practical example of a bank loan application where the *loan amount* might be particular complex to digest for some models as the value itself might actually be irrelevant without the income context, this is easily resolved by building ratios as shown in Table 4.

Table 3. Loan info with amount and year observer

Loan($)	Income($)	Year
$\$26,100$	$\$100,000$	2021
$\$64,500$	$\$200,000$	2022
$\$105,300$	$\$400,000$	2023

However, while certain transformations might result in more relevant information (e.g., loan to income ratio), others might provide pure noise as shown by *Year* column, which although numeric, the division by income is unlikely to bring any predictive value. This sort of processing should be paired with feature selection, but even then, it can lead to severe overfitting.

Table 4. Loan info with income observer

Loan($)	Income($)	Year
0.261	1	0.020
0.322	1	0.010
0.263	1	0.005

Although the process of FE can be very vast and covering all possibilities is intractable, in this work we focus on the presented transformations. This ability to change the projection of signals we define as *relativity*, which is exploited then by multiple set of *observers* also known as reference points. Following this principle, the optimization for δ_i is dependent to the observer used to project the data, leading to the optimization problem:

$$\delta^*_{(i,\beta)} = \arg\min \sum_{j=1}^{m} L(z_{(j,\beta)}, \hat{\theta}_{\delta(i,\beta)}) \tag{3}$$

Where the optimal perturbation δ^* is a function of the perturbation itself and the observer β used in the projection, leading to different optimal depending on β in order to minimize the loss L, which can be seen as a double optimization to determine the best $observer$ leading to the minimum loss. This leads to Algorithm 1.

Algorithm 1: Optimal Observer Detection

Input: A dataset D with features i and samples M
Input: A set of candidate projection observers β
Output: Optimal Observer
`// Initialize states`
$minimum\_loss \leftarrow None$
$best\_observer \leftarrow None$
`// Evaluate projections`
foreach $observer\beta$ **do**
  `// Compute validation loss`
  $loss \leftarrow \sum_{j=1}^{m} L(z_j, \hat{\theta}_{\delta i}) - \sum_{j=1}^{m} L(z_j, \hat{\theta})$
  `// Update best observer`
  if $loss < minimum\_loss$ **then**
    $minimum\_loss \leftarrow loss$;
    $best\_observer \leftarrow observer$;
  end
end
return $best\_observer$

The effect of an optimal $observer$ in the FE process, can dominate the entire solution rendering other aspects like model selection and hyper-parameter tuning as a relative irrelevant process, in other words, in the presence of $relativity$, a simple model with the proper $observer$ could outperform more complex architectures if those were to use naive FE.

4. EXPERIMENTS

The efficacy of our process has been benchmarked in a live-competition as part of *IEEE International Conference on Multimedia and Expo (ICME) 2023: Predicting Frags in Tactic Games*, where this work has achieved 2nd place. Participants are tasked to determine if a *frag* (i.e., soldier kill) would happen given a set of soldier characteristics (like position & health), map properties (e.g. objects around) and videogame screenshot. We believe this setup provides unbiased results as (1) test-set labels are hidden and (2) solutions can be compared relative to other participants. In the following subsections we provide the details of our experiments and solution design.

4.1. Baseline: Main Tabular Data Only

The challenge dataset is an ideal candidate for information fusion for its rich content of multimedia, including tabular, metadata and images. In the spirit of simplicity and low latency, we first provide a baseline relying purely on tabular data (TD) and further expand the solution complexity. The TD in this competition has strong relativity properties and it allows data to be projected from different observers (e.g., soldiers in the field) presenting a great opportunity to exploit this property as part of the FE process.

We benchmarked our proposal against raw data (e.g., as provided in competition website), showing a reference of the performance that can be achieved without FE. In addition, we leverage OpenFE to build features automatically. We finally compared both prior solutions against our FE exploiting relativity across multiple classifiers. All data is transformed into numerical features with dimensions as shown in Table 5.

Table 5. Feature space for transformed datasets

Dataset	Samples	Features
Raw (original)	50k	39
OpenFE (automatic FE[14])	50k	2039
Observer Based FE (this work)	50k	122

Our 122 features are the result of projecting the raw data but changing the observer from a set of candidates as follows:

- Active Player: rearrange soldiers properties such as its non-ambiguous which unit might be able to move
- Distance: encode positional coordinates as distances to multiple reference points.
- Health: compute relative ratios for all units.
- Unit Types: aggregate values at the unit type level.

These main projections allow to exploit the relativity in the data, other projections were explored but discarded following Algorithm 1. Our results are summarized in Table 6, with all models running optimized parameters through a 10-fold stratified validation with mean area under the ROC Curve (ROC AUC) reported.

Table 6. AUC scores for benchmark configurations

Algorithm	Raw	OpenFE	Observers
LGBM[10]	0.757	0.781	0.807
Xgboost[9]	0.761	0.781	0.803
Catboost[11]	0.752	0.781	0.806
HistGradientBoosting	0.745	0.777	0.795
RandomForest	0.713	0.732	0.798
ExtraTrees	0.707	0.725	0.797
DeepCrossNet[28]	0.663	0.696	0.780
TabNet[29]	0.650	0.660	0.790
MLP	0.678	0.626	0.788
DeepFM[30]	0.670	0.629	0.782
WideDeep[31]	0.672	0.614	0.786
LogisticRegression	0.580	0.645	0.781

Our results are consistent with previous observations[7] that deep neural network (DNN) algorithms might offer competitive performance only after careful FE. When the data was provided as-is (raw, but normalized for stable gradient propagation), LightGBM outperformed the best *overall* DNN (DeepCrossNet) solution by a staggering relative 58% $(0.757 - 0.5)/(0.663 - 0.5)$.

Model selection, does indeed play a role in overall performance when data is poorly prepared. Auto-FE can help to reduce the impact of models, with larger improvements (81% for LogReg) observed for weaker models, but providing small or no benefit for others. In average, we managed to improve performance by 6.7% thanks to OpenFE. However, in the presence of relativity in the data, all classifiers exposed to observer-based feature engineering outperformed any combination of Algorithm+OpenFE, with a simple Logistic Regression (using observers) matching the best solution (using OpenFE) at 0.781 AUC.

We quantified the effect of observer-based feature engineering in the overall solution against the value it brings to invest in model selection. Our results show the investment of FE brings around 600% the returns $(\frac{1}{12}\sum_{k=1}^{12}[(Observers_k - 0.5)/(Raw_k - 0.5)])$ compared to the time spent benchmarking different models or tuning architectures and parameters $((0.807 - 0.5)/(0.781 - 0.5))$. Specifics values shown in Figure 1.

4.2. Additional Metadata: Map Specific Features

The concept of relativity is not exclusive to tabular data, although in other forms, such as images, powerful DNN models

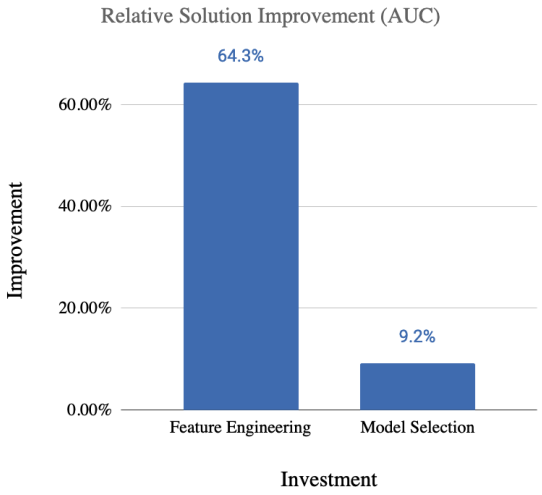

Fig. 1. Relative Contribution of Investments

might have a better chance to approximate these projections, unlike for structured data. In this particular challenge a set of map properties are provided, however, the map metadata itself tends to be irrelevant. It is the *relative position* of a particular unit in the map that matters. Essentially changing the *observer* to be each *unit* (instead of the map masks) multiple features can be derived from the exact same map, e.g., region properties close to each unit coordinates. This *relativity* concept to exploit reference points in a cartesian plane can be extended to any problem as its data agnostic.

An additional 85 features ($85+122=207$) provided consistent gains across top GBDT classifiers, with observed improvements around 6%. For simplicity and given our findings that FE gives much better return, we did not continue benchmarking other models to reduce invested time. Our results as shown in Table 7 further demonstrate model selection can play such as small role at the top with all GBDT performing virtually the same.

Table 7. AUC gains with metadata features

Algorithm	Baseline	Base+Maps	Improvement
LGBM	0.807	0.826	+6.18%
Catboost	0.806	0.823	+5.55%
Xgboost	0.803	0.822	+6.27%

4.3. Additional Images: Gameplay Screenshots

Our solution refrained from using images in the spirit of simplicity. But we acknowledge information fusion could lead to greater results as DNN can extract important signals. This simple design also makes our solution flexible and portable as its compatible with future game maps that do not exist as of today.

5. CONCLUSION

Algorithms have evolved to a great extent. For most applications seeking max performance, simple algorithms like Logistic Regression could handicap the result if no feature engineering is done. This could give the false impression that more powerful/complex models should be the key to improve a system. Deep learning in particular has had a tremendous success with automatic feature engineering in fields like CV, however, for tabular data and based on our results, DNN still heavily lags decision trees(DT) algorithms in performance. Even the weakest of our DT-based models (ExtraTrees) outperforms all DNN solutions and this is without considering NN applications are slower and usually require special data processing. The importance of models might look conflicting with our earlier claims on the irrelevance of model selection. This is a relative perception, models are undoubtedly important, however, the impact of powerful FE outshines any gains from models per-se. We firmly believe this is a generic pattern for any tabular-rich application and not exclusive to our case study. For industry applications, projects tend to be time & cost bounded and it's important to invest resources where it matters the most. Data and FE are your most valuable resource for return on investment, and model selection, although important, might be the least of your problems.

6. REFERENCES

[1] Jabeen Summaira, Xi Li, Amin Muhammad Shoib, and Songyuan Li andJabbar Abdul, "Recent advances and trends in multimodal deep learning: A review," *CoRR*, vol. abs/2105.11087, 2021.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[3] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen, "Deberta: Decoding-enhanced BERT with disentangled attention," *CoRR*, vol. abs/2006.03654, 2020.

[4] Mingxing Tan and Quoc V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. 2021, OpenReview.net.

[6] Jiashi Li, Xin Xia, Wei Li, Huixia Li, Xing Wang, Xuefeng Xiao, Rui Wang, Min Zheng, and Xin Pan, "Next-

vit: Next generation vision transformer for efficient deployment in realistic industrial scenarios," 2022.

[7] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci, "Deep neural networks and tabular data: A survey," *CoRR*, vol. abs/2110.01889, 2021.

[8] Dugang Liu, Pengxiang Cheng, Hong Zhu, Xing Tang, Yanyu Chen, Xiaoting Wang, Weike Pan, Zhong Ming, and Xiuqiang He, "Diwift: Discovering instance-wise influential features for tabular data," 2022.

[9] Tianqi Chen and Carlos Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016.

[10] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. 2017, vol. 30, Curran Associates, Inc.

[11] Anna Veronika Dorogush, Andrey Gulin, Gleb Gusev, Nikita Kazeev, Liudmila Ostroumova Prokhorenkova, and Aleksandr Vorobev, "Fighting biases with dynamic boosting," *CoRR*, vol. abs/1706.09516, 2017.

[12] Tomaso Poggio, Andrzej Banburski, and Qianli Liao, "Theoretical issues in deep networks," *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30039–30045, 2020.

[13] Ravid Shwartz-Ziv and Amitai Armon, "Tabular data: Deep learning is not all you need," 2021.

[14] Tianping Zhang, Zheyu Zhang, Zhiyuan Fan, Haoyan Luo, Fengyuan Liu, Wei Cao, and Jian Li, "Openfe: Automated feature generation beyond expert-level performance," 2022.

[15] Avrim L. Blum and Pat Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1, pp. 245–271, 1997, Relevance.

[16] Ron Kohavi and George H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[17] Carlos Huertas, Reyes Juárez-Ramírez, and Christian Raymond, "Heat map based feature ranker: In depth comparison with popular methods," *Intell. Data Anal.*, vol. 22, pp. 1009–1037, 2018.

[18] Zhixiang Eddie Xu, Gao Huang, Kilian Q. Weinberger, and Alice X. Zheng, "Gradient boosted feature selection," 2019.

[19] Michel Verleysen and Damien François, "The curse of dimensionality in data mining and time series prediction," in *Proceedings of the 8th International Conference on Artificial Neural Networks: Computational Intelligence and Bioinspired Systems*, Berlin, Heidelberg, 2005, IWANN'05, p. 758–770, Springer-Verlag.

[20] Michael R. Anderson and Michael Cafarella, "Input selection for fast feature engineering," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 577–588.

[21] Slater Stefan, S. Baker Ryan, and Yeyu Wang, "Iterative feature engineering through text replays of model errors," in *The 13th International Conference on Educational Data Mining (EDM 2020)*, 2020.

[22] Franz M. Rohrhofer, Santanu Saha, Simone Di Cataldo, Bernhard C. Geiger, Wolfgang von der Linden, and Lilia Boeri, "Importance of feature engineering and database selection in a machine learning model: A case study on carbon crystal structures," 2021.

[23] James Max Kanter and Kalyan Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Paris, France, October 19-21, 2015*. IEEE, 2015, pp. 1–10.

[24] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[25] Franziska Horn, Robert Pack, and Michael Rieger, "The autofeat python library for automated feature engineering and selection," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2019, pp. 111–120.

[26] Pang Wei Koh and Percy Liang, "Understanding black-box predictions via influence functions," 2017.

[27] Dugang Liu, Pengxiang Cheng, Hong Zhu, Xing Tang, Yanyu Chen, Xiaoting Wang, Weike Pan, Zhong Ming, and Xiuqiang He, "Diwift: Discovering instance-wise influential features for tabular data," 2022.

[28] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang, "Deep and cross network for ad click predictions," in *Proceedings of the ADKDD'17*, New York, NY, USA, 2017, ADKDD'17, Association for Computing Machinery.

[29] Sercan O. Arik and Tomas Pfister, "Tabnet: Attentive interpretable tabular learning," 2019.

[30] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, Xiuqiang He, and Zhenhua Dong, "Deepfm an end-to-end wide and deep learning framework for ctr prediction," 2018.

[31] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah, "Wide and deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, New York, NY, USA, 2016, DLRS 2016, p. 7–10, Association for Computing Machinery.