ProAdaFS: Probabilistic and Adaptive Feature Selection in Deep Recommendation Systems

2024 International Conference on Information Networking (ICOIN) | 979-8-3503-3094-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICOIN59985.2024.10572163

1st Hyston Kayange
School of Computer Science and Engineering
Soongsil University
Seoul, South Korea
hyston@soongsil.ac.kr

2nd Jonghyeok Mun
School of Computer Science and Engineering
Soongsil University
Seoul, South Korea
jonghyeokmun@soongsil.ac.kr

3rd Yohan Park
School of Computer Science and Engineering
Soongsil University
Seoul, South Korea
imjin3027@soongsil.ac.kr

4th Jongsun Choi
School of Computer Science and Engineering
Soongsil University
Seoul, South Korea
jongsun.choi@ssu.ac.kr

5th Jaeyoung Choi
School of Computer Science and Engineering
Soongsil University
Seoul, South Korea
choi@ssu.ac.kr

***Abstract*—Deep recommender systems are essential for providing personalized recommendations in various domains, such as e-commerce, social media, and entertainment. In deep recommender systems, feature selection plays a vital role as it identifies the features that are the most informative for predicting user preferences. However, most existing deep recommender systems are designed without a systematic approach to feature selection. They typically feed all available features into their sophisticated neural networks, or experts choose features manually or employ existing feature selection algorithms. These approaches might potentially undermine the accuracy and effectiveness of recommender systems, since they execute feature selection separately from the subsequent model of the recommender system, without considering the model's prediction behavior. Moreover, existing feature selection methods tend to select a fixed set of features, which is not adaptable to the dynamic and complex environments of practical recommender systems, where the importance of a specific feature can vary across user-item interactions. To address these challenges, we propose a novel adaptive feature selection framework, Probabilistic and Adaptive Feature Selection in Deep Recommendation Systems (ProAdaFS), for deep recommender systems. ProAdaFS leverages the power of two existing adaptive feature selection techniques (AdaFS and AutoField) with significant modifications to enhance feature selection. To identify the most informative features corresponding to a subsequent recommendation task model, we design a network controller that dynamically and adaptively adjusts the probability of selecting a feature field, generates scores and re-evaluates feature fields to identify informative features. Our experiments were conducted on two real-world e-commerce recommender systems datasets. The experimental results demonstrate the effectiveness of ProAdaFS in improving the feature selection process in deep recommender systems.**

Keywords—Feature Selection, Recommender Systems, AutoML

I. INTRODUCTION

"*People don't know what they want until you show it to them*" ~ *Steve Jobs*. The quote best describes human behavior in terms of how we humans perceive products and consume them. In the same scenario, electronic retailers and content providers offer a wide selection of products, to match consumers with the most appropriate products for user satisfaction and maintaining loyalty. As another way of boosting their sales and accumulating views on their content, more retailers have become interested and are working tirelessly in developing and engineering sophisticated recommendation system architectures. Some of the leading e-commerce platforms such as Amazon.com, Netflix, YouTube, and other social media platforms such as X, formerly known as Twitter, and TikTok, have integrated recommendation systems to add a dimension to user experience, particularly suggesting products, services, and content to users based on their preferences and behaviors.

The phrase "Garbage in, garbage out," commonly used in machine learning, emphasizes that the model's performance depends on the quality of input features provided. This holds true for deep recommender systems (DRS), where feature quality significantly influences recommendation performance. Despite extensive research in DRS, the majority focuses on engineering complex neural architectures, often overlooking feature selection—a crucial process to enhance the model's performance [2]. In reality, electronic retail and content platforms collect a broad range of user features on their websites and applications. These encompass user demographics (e.g., id, gender, age), item or content preferences (e.g., category, brand), user behaviors (e.g., clicks, views, likes, purchases), and contextual information (e.g., time, location). However, not all these features should be input to the network, because irrelevant features can hamper recommendation performance, slow down model optimization, and increase computational costs [3].

Several classical feature selection methods (hand-crafted, wrapper, filter, embedded) [4,5,6,12,13], have shown effectiveness, but they fall short in deep recommender systems, because their selection process is independent from the subsequent DRS model, disregarding the model's prediction model behavior [17] . Recent methods [2,3,17,20] have utilized the AutoML approach to identify and automatically select the

Authorized licensed use limited to: Soongsil University. Downloaded on August 06,2024 at 05:22:49 UTC from IEEE Xplore. Restrictions apply.

most predictive features for DRS models and their performance has been convincing.

Adaptive feature selection (AdaFS) [2] and AutoField [3] being the state of art methods. AdaFS adaptively selects significant features for each data instance across user-item interactions making it suitable for dynamic and complex environments of practical recommender systems as it improves the performance over the classical methods. AutoField is able to automatically adjust the probability of selecting a particular feature field. Both methods employ a controller network that serves as the primary mechanism for the generation of feature importance and the regulation of feature probabilities.

To contribute to the development of adaptive methods for DRS. This study introduces a method called Probabilistic and Adaptive Feature Selection (ProAdaFS). An AutoML framework for automating adaptive feature selection for varying user-item interactions. Leveraging AutoField's probabilistic guidance and AdaFS adaptive hard selection with some significant modifications, to enhance feature selection for more accurate and tailored recommendations in deep recommendation models.

II. FRAMEWORK

In this section, we discuss our method approach (ProAdaFS). We will detail and highlight the overview of our framework including other important modules, and its optimization process.

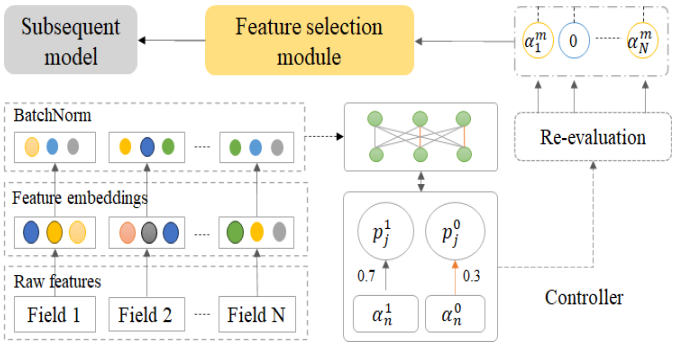

Fig. 1. Overview of the ProAdaFS's Framework

A. Framework Overview

To enhance the process of adaptive and dynamic feature selection, we propose a framework that automatically scores, re-values, and selects the optimal subset of features that are further utilized in a subsequent DRS network. We have illustrated the framework architecture in Fig 1.

The framework follows the fundamentals of the DRS network in which we have included the embedding and MLP components. We propose a DRS framework with a controller network based on AutoField and AdaFS. The controller scores, weighs, assigns, and adjusts probabilities to each feature field according to their perceived importance in correspondence with the user-item interaction. We then perform hard selection to reserve the top k features.

B. Deep Recommendation Architecture

In the following subsections, we discuss the basic components that build our model, the MLP and embedding component, and the modifications we have made to improve the feature adaptive selection process.

Embedding component. Deep recommender systems typically use categorical input features, that are sparse and high-dimensional. To address this, binarization and projection can be used to convert these features into a lower-dimensional representation.

Binarization is a technique for converting categorical features into binary vectors. Each unique value within a categorical feature field corresponds to a binary representation, with the number of dimensions determined by the number of unique values. For example, the categorical feature field "Preferred Payment Method" with three unique values ("Credit Card", "PayPal", and "Bitcoin") could be binarized as follows: [1,0,0], [0,1,0], and [0,0,1].

Projection is a mathematical transformation that projects binary vectors into a lower-dimensional feature space. This is achieved by multiplying each binary vector by a learnable weight matrix: The binary representation of N features can be obtained by concatenation; for example, x = [x_1 , x_2 …x_N], x_n $\in \mathbb{R}^{D_n}$ x_n representing a binary vector (where n is the index of the feature field) by a learnable weight matrix A_n. The matrix A_n has dimensions d × D_n where d is the predefined embedding size for the projection space and D_n is the dimension of the original binary vector for the n^{th} feature field. We can represent it as:

$$\boldsymbol{e_n = A_n x_n} \quad (1)$$

Suppose that the input dataset has N features and a batch size of M, we can denote the projection of the final embedding of the user-item interaction as :

$$\boldsymbol{E = [e_1^m, e_2^m, \dots, e_N^m]} \quad (2)$$

MLP Component. Multi-layer perceptrons (MLPs) are common components in deep recommender systems. They are used to further process and extract features from the dense embedding obtained from the embedding component. MLPs consist of fully connected layers with non-linear activation functions to capture high-order feature interactions. The output layer of an MLP can be equipped with a specific activation function tailored for a specific task, such as prediction or classification.

An MLP comprises L hidden layers, and we can detail each hidden layer as h_l where l ranges from 1 to L as follows: h_l is computed as a non-linear transformation of the weighted sum of inputs:

$$h = \varphi\ (W_l\ h_{l-1} + b_l\ ) \quad (4)$$

Where h_l represents the output of the l^{th} hidden layer, W_l is the weight matrix specific to the l-the hidden layer. h_{l-1} is the output of the previous $l-1^{th}$ hidden layer or the input $\boldsymbol{E}$ for l = 1. b_l is the bias vector for the l-the hidden layer. φ (·) is the activation function (e.g., ReLU) applied elementwise. The first

Authorized licensed use limited to: Soongsil University. Downloaded on August 06,2024 at 05:22:49 UTC from IEEE Xplore. Restrictions apply.

hidden layer h_0 is initialized with the dense feature embeddings $\boldsymbol{E}$ obtained from the Embedding component:

$$h_0 = \boldsymbol{E} \tag{5}$$

The output layer of an MLP can be equipped with a specific activation function tailored for a specific task, such as prediction or classification. For example, the sigmoid function is commonly used for regression tasks, and the Softmax function is commonly used for multi-class classification tasks. The general prediction function can be represented as follows:

$$\hat{\boldsymbol{y}} = \sigma(W_0 h_L + b_0) \tag{6}$$

Where $\hat{\boldsymbol{y}}$ is the prediction, W_0 is the weight matrix for the output layer. b_0 is the bias vector for the output layer $\sigma(\cdot)$ is the task-specific activation function.

III. Method Approach

As mentioned earlier ProAdaFS is based on two existing algorithms, with modifications to apply towards adaptive feature selection. The following are main contributions:

- ProAdaFS utilizes Gumbel-Softmax [9] instead of Softmax to generate probabilities for feature fields based on their importance. Unlike Softmax, which can output the same probability for different input values, Gumbel-Softmax preserves the diversity and distinctiveness of the features by adding random noise to the input values. This is one method of preventing the controller from exhibiting bias towards dominating feature patterns. In practice, we have applied different techniques to achieve this.
- ProAdaFS implements a reevaluation process in which it incorporates a function with a threshold to adjust the weights of feature fields under the guidance of their final allocated probabilities prior to feature selection. This is to enable the controller to adjust to variations in user-item interaction and avoid missing predictive features.

Controller. Before discussing feature selection with ProAdaFS network controller, it is necessary to process the embedded data. The embedding size of the feature fields varies significantly, which can reduce the reliability of the probabilities and weights calculated by the controller. To address this, we use BatchNorm [7] to ensure that the controller generates reliable probabilities and weights. We can present BatchNorm as:

$$\hat{e}_n^m = \frac{e_n^m - E_B^n}{\sqrt{Var_B^n + \epsilon}} \tag{7}$$

where n belongs [1, N] representing the n^{th} feature and m is of the m^{th} data example in the input batch, E_B^n calculates the mini-batch mean and Var_B^n gives the value of the min-batch variance for feature embeddings in n^{th} feature filed. To control for exceedingly small values by Var_B^n we add a constant to the variance before calculating the standard deviation (ε). We can denote the feature embeddings after this process as:

$$\hat{E} = [\hat{e}_1^m, \hat{e}_2^m, \ldots, \hat{e}_N^m] \tag{8}$$

Feature selection Module: Our feature selection module combines a directed acyclic graph (DAG) [8] and Gumbel-Softmax [9] to generate probabilities for each feature field, considering both feature dependencies and importance scores. The controller uses these probabilities to draw binary decisions for each feature field, with the threshold values acting as weights for control purposes.

According to the controller's parameter pair (α_m^1, α_m^0) each feature field will be allocated a probability for either to be selected or dropped in the final stage of hard selection. Here, α_m^1 signifies the probability of selecting a feature, α_m^0 represents the probability of dropping it. Our sampling can be formulated as:

$$\hat{e}_n^m = (\alpha_m^1 \cdot v_1 + \alpha_m^0 \cdot v_0) \cdot e_n^m \tag{9}$$

$$\hat{E} = [\hat{e}_1^m, \hat{e}_2^m, \ldots, \hat{e}_N^m] \tag{10}$$

where e_n^m is the embedding of the n^{th} feature field. v_1 and v_2 are vectors with the same length of e_n^m acting as a threshold in which we assume v_1 is close to 1 and v_0 is close to 0, and their weighted sum embedding of ($\alpha_m^1 v_1 e_n^m + \alpha_m^0 v_0 e_n^m$) is 1. And $\hat{e}_n^m$ is of soft selection of the n^{th} feature field as in (10), in which $\hat{E}$ is the soft selection of the feature embeddings E of (2).

The formula for Gumbel-Softmax is as follows.

$$p_m^j = \frac{\exp((\log \alpha_m^j + g_j)/\tau}{\exp((\log \alpha_m^1 + g_1)/\tau + \exp((\log \alpha_m^0 + g_0)/\tau} \tag{11}$$

where p_m^j is the final allocated probability to the n^{th} feature field in the m^{th} data example, α_m^j is the importance score of n^{th} feature field in the m^{th} data example g_j is a random variable sampled from the Gumbel distribution, and τ is a temperature parameter that controls the smoothness of the approximation.

The final allocated probability can be formulated as:

$$\hat{e}_n^m = (p_m^1 \cdot v_1 + p_m^0 \cdot v_0) \cdot e_n^m \tag{12}$$

The binary decision nodes are represented as two-dimensional vectors, each feature field consisting of two parameters (α_m^1, α_m^0), that initiate the feature selection process. We begin by assigning an equivalent pair of values (α_n^1, α_n^0) to each feature field, initializing them at $\alpha_n^1 = \alpha_n^0 = 0.5$. During training, the parameters α_n^1 increase while α_n^0 decreases according to their set threshold. Gumbel-Softmax adjusts these parameters accordingly to ensure an unbiased distribution, diversity, and distinctiveness of the feature fields.

In Equation (10), soft selection assigns probabilities to feature embeddings based on their importance, but it does not eliminate the impact of irrelevant features on the final recommendation. To address this, we implement hard selection using k-max pooling after re-evaluation process by the controller. During re-evaluation (Fig 2), we reactivate the controller to score feature importance incorporating a threshold mentioned earlier to control Gumbel Softmax noise, we take $\hat{e}_n^m$ of n^{th} feature field with a probability of p_m^j to calculate its feature weight α_n^m.

For hard selection (Fig 3), we use the k-max pooling technique from AdaFS [2], selecting the top k features based on

Authorized licensed use limited to: Soongsil University. Downloaded on August 06,2024 at 05:22:49 UTC from IEEE Xplore. Restrictions apply.

their weighted sums (α_n^m) and masking the rest as zeros. We then reweighted the selected weights to ensure that their sum equals 1 , maintaining reliability and reasonability. For instance, if we have feature weights for the n^{th} feature in m^{th} batch as: [0.7, 0.0, 0.4, 0.5,0.0 and predefine k as k=3. the selection module performs 3 max pooling, resulting in new feature weights [0.4375, 0.0, 0.25, 0.3125, 0.0] by reweighting them accordingly based on the sum of the new weights (1.6). We can denote this process as.

$$\hat{e}_n^m\ (p_m^j) = (p_j^1 \tilde{\alpha}_1^m \hat{e}_1^m\ .\ 1 + p_j^0\ \tilde{\alpha}_1^m \hat{e}_n^m\ .\ 0).\ e_n^m = p_j^1 \tilde{\alpha}_n^m \hat{e}_n^m \quad (13)$$

where 1 and 0 are all-one and all-zero vectors, respectively, with the same length of e_n^m. On the basis of (13) , we can obtain the $\hat{E}$ as in (8) and then replace the embeddings **E** of (2) by $\widehat{\boldsymbol{E}}$ to perform feature selection with our proposed controller. After this, selected features are fed into a subsequent model as shown in Fig. 1 for the final prediction.

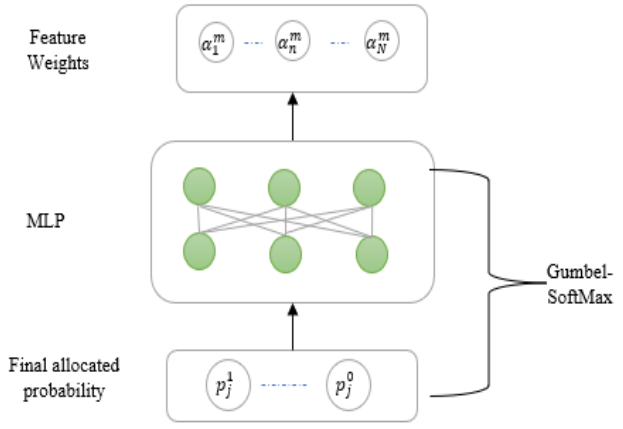

Fig. 2. The controller network and re-evaluation process

IV. AN OPTIMIZATION METHOD

In this section, we explain our optimization approach for updating the framework parameters. $\boldsymbol{W}$ denotes parameters for the DRS network (including Embedding and MLP components), while **θ** represents the controller network parameters crucial for feature selection.

Optimizing **W** and **θ** simultaneously on the same training batch can lead to overfitting. To address this issue, we employ the improved differentiable architecture search (I-DARTS) technique [11]. We alternate updates for W and **θ** on different dataset partitions — switching between training and validation data. $\boldsymbol{W}$ is updated using training loss L_{train} to optimize the DRS parameters, while **θ** is updated using validation loss L_{val}, to optimize the controller parameters. This alternating update, called bi-level optimization, prevents overfitting and enhances the optimization of both the basic DRS and the controller network. We can denote this as:

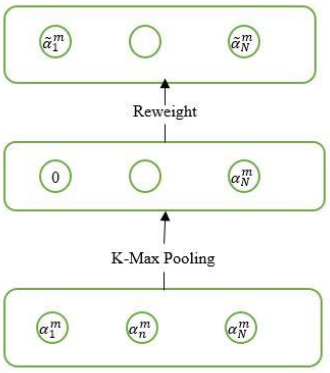

Fig. 3. The hard selection process

$$\min_{\theta} L_{val}(W^*(\boldsymbol{\theta}), \boldsymbol{\theta})$$
$$\text{s.t } \boldsymbol{W}^*\ (\boldsymbol{\theta}) = \arg\min_{W} L_{train}(\mathbf{W}, \boldsymbol{\theta}^*) \quad (14)$$

Algorithm 1 Optimization Algorithm of ProAdaFS Framework

Input: Raw features $\{e_n\}$, number of selected feature fields K, training epoch (t) ground-truth labels y.

Output: K selected features for final recommendations

1. Initialize DRS (**W**) and Controller ($\boldsymbol{\theta}$) parameters.
2. **for** i=0; I < t; i++ **do**
3. Sample a min-batch of the training set.
4. Update W according to Equation (11)
5. **end for**
6. **while** not converged do
7. Sample a mini batch of the validation set.
8. Update **θ** according to equation (11)
9. **end while**

V. EXPERIMENT

In this section, we first describe our experimental setup and then evaluate the performance of the proposed framework.

A. Dataset

To evaluate our method, we used two publicly available datasets: Criteo[1] and Avazu[2]. These datasets are used for online advertising in click-through rate prediction tasks and, they have 39 and 23 features, respectively.

B. Evaluation metrics

Logloss and AUC were used to evaluate our method, because these are commonly used metrics for CTR prediction. A higher AUC and a lower Logloss indicate better performance. It is important to highlight that even **a 0.001-level** improvement in either metric is considered significant [1].

[1] https://ailab.criteo.com/resources/

[2] https://www.kaggle.com/c/avazu-ctr-prediction

Authorized licensed use limited to: Soongsil University. Downloaded on August 06,2024 at 05:22:49 UTC from IEEE Xplore. Restrictions apply.

Table 1: Comparison of performance of feature selection methods. * Indicates the statically significant improvements (i.e., p ≤ 0.05 for the two-sided t-test over the best baseline.

Dataset	Model	No Selection		AutoField		AdaFS		OptFS		ProAdaFS	
		AUC ↑	Logloss ↓	AUC ↑	Logloss ↓	AUC ↑	Logloss ↓	AUC ↑	Logloss ↓	AUC ↑	Logloss ↓
Avazu	DeepFM	0.7813	0.3790	0.7821	0.3784	0.7831	0.3781	0.7847	0.3771	**0.7843**	**0.3779**
	DCN	0.7789	0.3787	0.7791	0.3786	0.7799	0.3777	0.7837	0.3751	**0.7867***	**0.3731***
	W&D	0.7781	0.3806	0.7785	0.3804	0.7797	0.3801	0.7763	0.3815	**0.7787**	**0.3803**
Criteo	DeepFM	0.8057	0.4451	0.8059	0.4439	0.8079	0.4438	08043	0.4463	**0.8088***	**0.4429***
	DCN	0.8063	0.4455	0.8061	0.4452	0.8074	0.4434	0.8065	0.4449	**0.8076**	**0.4436**
	W&D	0.8051	0.4462	0.8063	0.4451	0.8057	0.4464	0.8060	0.4454	**0.8086***	**0.4433***

C. Performance comparison and DRS models

We compare the performance of ProAdaFS with the following state-of-the-art baselines (i). AdaFS [2] which adaptively select predictive feature fields of each data instance via a controller, (ii) AutoField [3] which selects global informative feature fields using neural network architecture search techniques via the drive of controller generated feature fields probabilities and third (iii) we compare it with OptFS [17] which also globally select informative feature fields in accordance with feature interactions. We evaluate its performance on existing DRS models for recommendation tasks: W&D [1], DCN [18], and DeepFM [19].

D. Implementation details

Our approach is implemented using the PyTorch Public Library for recommendation models[3] and we utilize the official implementation for AdaFS [4] and AutoField [5]. We set the embedding size for the feature fields to 16 and used a two-layer MLP with sizes [16, 8], employing ReLU as the activation function. The controller involved generating four pairs of probabilities through a combination of DAG and Gumbel-Softmax alternatively (α_m^1 and α_m^0 , p_m^j (p_j^1 and p_j^0 equation (13)). We utilized k-Max pooling for hard selection post-re-evaluation. Additional parameters included a batch size of 2048, a learning rate of 0.0001, a dropout rate of 0.2, temperature (τ) of 0.01-1, and a GPU GEFORCE RTX 3080 for running experiments.

VI. Overall Performance

Table 1 illustrates the performance of existing various state-of-the-art feature selection methods on three different DRS models. It can be observed that almost all methods are superior to No selection portraying performance gains to some extent. In comparison with ProAdaFS, ProAdaFS can be considered to show significant improvements over other baselines, though there is a limited improvement, but ProAdaFs is covering some of the limitation (i) AutoField and OptFS have a limitation of globally fixed feature selection in which it does not consider varying feature importance for each data instance [20], (ii) AdaFS performs better, however its controller and the feature selection process could easily lead to bias to dominant features, whilst ProAdaFS can maintain, revaluate and manage feature dependency which make it robust to variations and biasness in the selection process.

VII. Analyzing ProAdaFS

Table 2: Transferability of ProAdaFs on Avazu.

Model	No Selection		ProAdaFS	
	AUC ↑	Logloss ↓	AUC ↑	Logloss ↓
DeepFM	0.7813	0.3790	**0.7843***	**0.3779***
DCN	0.7789	0.3787	**0.7867***	**0.3731***
W&D	0.7781	0.3806	**0.7787***	**0.3803***

"*" Indicates the statically significant improvements (i.e., p ≤ 0.05 for the two-sided t-test over No Selection.

Transferability Study analysis. In this section, we conduct an analysis to validate the transferability of ProAdaFS. Following AdaFS [2], we freeze the parameters of a trained controller, to utilize it for training popular existing DRS models (W&D, DeepFM and DCN). From the results in Table 2, it can be noted that there is a significant improvement in all models, indicating that ProAdaFS can consistently select the most optimal predictive features for different DRS models in which we can draw conclusions that ProAdaFS can be implemented in real world recommender systems.

Parameter analysis. In Fig. 4 we demonstrate the effectiveness of ProAdaFs with a varying K which is a crucial hyperparameter towards the performance of ProAdaFS. Other hyperparameter are not changed. K represents the number of feature fields to be selected. The experiment is conducted on Avazu dataset, which has 22 features. We set K to [7, 9, 11,13, 15,18], Looking at ProAdaFS method with different K values, we notice some important patterns in how the number of selected features affects the model's performance. According to Fig 4 we can observe that the performance is better when K =9 and K= 11, meaning that having a moderate number of features i.e. 40%-50% leads to the best results but again we can notice from K= 15 and K=18 the models performance begins to pick up from a downgraded performance of K=13 of which we can conclude that ProAdaFS can handle varying feature scenarios better by adapting to the right balance between few and many features. From this performance we can agree to consider that a larger K value can either improve model performance or downgrade it by introducing irrelevant features and a smaller value might miss the predictive ones. In short, it is essential to

[3] https://github.com/rixwew/pytorch-fm
[4] https://github.com/Applied-Machine-Learning-Labs/AdaFS
[5] https://github.com/fuyuanlyu/autofs-in-ctr

Authorized licensed use limited to: Soongsil University. Downloaded on August 06,2024 at 05:22:49 UTC from IEEE Xplore. Restrictions apply.

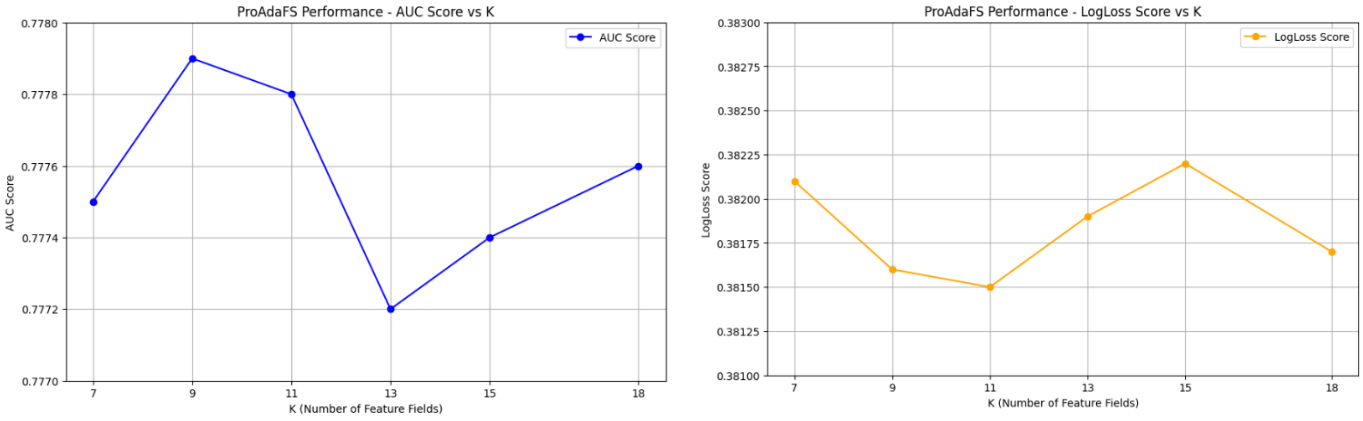

Fig 4. K parameter analysis, comparison of various K values. From Left AUC and Right Logloss results of ProAdaFS on Avazu.

choose the right K value. Finally, it is worth mentioning that K value is manually chosen.

VIII. CONCLUSION

In this study, we introduce a feature selection method and an AutoML-based feature selection model, ProAdaFS, to enhance adaptive feature selection in DRS Models. Overall, our work highlights improving dynamic and adaptive feature selection in deep recommender systems for various recommendation tasks. ProAdaFS approach leverages probability and adaptive techniques to efficiently select the most predictive features. Through extensive experiments on two benchmark datasets, we demonstrate our approach's capability to enhance dynamic and adaptive feature selection. ProAdaFS contributes to the gap between classical methods and adaptive methods. While parameter fine-tuning posed a challenge, we acknowledge the need for further studies, particularly in exploring dynamic K parameter analysis.

IX. ACKNOWLEDGEMENTS

This work was supported by an Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No.2022-0-00218).

REFERENCES

[1] H.-T. Cheng *et al.*, "Wide & Deep Learning for Recommender Systems," Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp.7-10, Sept. 2016.

[2] W. Lin, X. Zhao, Y. Wang, T. Xu, and X. Wu, "AdaFS: Adaptive Feature Selection in Deep Recommender System," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Aug. 2022, pp. 3309–3317.

[3] Y. Wang, X. Zhao, T. Xu, and X. Wu, "AutoField: Automating Feature Selection in Deep Recommender Systems," in *WWW 2022 - Proceedings of the ACM Web Conference 2022*, Association for Computing Machinery, Inc, Apr. 2022, pp. 1977–1986.

[4] R. Kohavi and G. H. John, "Wrappers for feature subset selection.", Artificial Intelligence, Vol.97, pp.273-324, December 1997.

[5] Huan Liu and Rudy Setiono, "Chi2: Feature selection and discretization of numeric attributes," Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence, November 1995.

[6] R. Tibshiranit, "Regression Shrinkage and Selection via the Lasso,", Journal of the Royal Statistical Society, Vol.58, pp 267-288, January 1996.

[7] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", Proceedings of the 32nd International Conference on Machine Learning, pp.448-456, 2015.

[8] A. Nilsson, C. Bonander, U. Strömberg, and J. Björk, "A directed acyclic graph for interactions," *Int J Epidemiol*, vol. 50, no. 2, pp. 613–619, April 2021.

[9] E. Jang, S. Gu, and B. Poole, "Categorical Reparameterization with Gumbel-Softmax," arxiv:1611.01144, Nov. 2016.

[10] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient Neural Architecture Search via Parameter Sharing," Proceedings of the 35th International Conference on Machine Learning Vol.80, pp.4095-4104, Feb2018,

[11] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search," arxiv:1806.09055, Jun. 2018.

[12] S. Wold, K. Esbensen, and P. Geladi, "Principal Component Analysis." Chemometrics and Intelligent Laboratory Systems, Vol.2, pp. 37-52, August 1987.

[13] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Front Neurorobot*, vol. 7, Dec 2013.

[14] Y. Zhai, "Grouping features in big dimensionality," Nanyang Technological University, 2016.

[15] X. Zhao, "Adaptive and automated deep recommender systems," *ACM SIGWEB Newsletter*, vol. 2022, no. Spring, pp. 1–4, May 2022.

[16] L. Pack Kaelbling, M. L. Littman, A. W. Moore, "Reinforcement Learning: A Survey," Journal of Artificial Intelligence Research, Vol.4, May 1996.

[17] F. Lyu, X. Tang, D. Liu, L. Chen, X. He and X. Liu: "Optimizing Feature Set for Click-Through Rate Prediction", in Proceeding of the ACM web Conference 2023, Apr.2023, pp. 3386-3395.

[18] R. Wang, B. Fu, G. Fu, M. Wang: "Deep & Cross Network for Ad Click Predictions ". In Proceedings of the ADKDD'17, Aug. 2017, Article.:12, pp.1-7.

[19] H. Guo, R. Tang, Y. Ye, Z. Li, X. He: "DeepFM: A Factorization-Machine based Neural Network for CTR Prediction", in proceedings of the 26th international Joint Conference on Artificial Intelligence, Mar. 2017, pp. 1725-1731.

[20] Y. Lee, Y. Jeong, K. Park, S. Kang: "MvFS: Multi-View Feature Selection for Recommender System", in Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, Oct. 2023, pp. 4048-4052.

Authorized licensed use limited to: Soongsil University. Downloaded on August 06,2024 at 05:22:49 UTC from IEEE Xplore. Restrictions apply.