Predicting Conversion Rate in Advertising Systems: A Two-Stage Approach with LightGBM

Lulu Wang
wanglulu2@myhexin.com
Hithink RoyalFlush Information Network Co.,Ltd.
Hangzhou, China

Yu Zhang
zhangyu6@myhexin.com
Hithink RoyalFlush Information Network Co.,Ltd.
Hangzhou, China

Huayang Zhao
zhaohuayang@myhexin.com
Hithink RoyalFlush Information Network Co.,Ltd.
Hangzhou, China

Zhewei Song
songzhewei@myhexin.com
Hithink RoyalFlush Information Network Co.,Ltd.
Hangzhou, China

Jiaxin Hu
hujiaxin@myhexin.com
Hithink RoyalFlush Information Network Co.,Ltd.
Hangzhou, China

ABSTRACT

This paper presents the solution designed by team "RoyalFlush" for the ACM RecSys Challenge 2023, organized by Mohalla Tech Pvt. Ltd. ("ShareChat/MTPL") in the online advertising domain. The task was to predict whether users would install an app given an ad impression. We propose a two-stage approach to improve the accuracy of conversion rate estimation. First, we construct two prediction models: $Model_{ts}$, which splits the conversion rate estimation problem into two sub-tasks, modeling the probability that a user likes an ad and the probability that a user installs an app given that they like the ad, and $Model_{e2e}$, which directly models the probability of user installation end-to-end. In the second stage, we use ensemble learning and calibration methods to improve the accuracy of the prediction. Our model achieved strong results, placing 4th on the final leaderboard. To facilitate reproducibility, we open-source our materials: https://github.com/colorblank/recsys-challenge-2023.

CCS CONCEPTS

• **Information systems** → **Computational advertising**; • **Theory of computation** → *Boosting*.

KEYWORDS

recommender systems, neural networks, recsys challenge, calibration, feature engineering

ACM Reference Format:
Lulu Wang, Yu Zhang, Huayang Zhao, Zhewei Song, and Jiaxin Hu. 2023. Predicting Conversion Rate in Advertising Systems: A Two-Stage Approach with LightGBM. In *RecSysChallenge23: 17th ACM Conference on Recommender Systems, September 18th–22nd, 2023, Singapore*. ACM, New York, NY, USA, 7 pages. https://doi.org/1xxxxx

RecSysChallenge23, September 18th–22nd, 2023, Singapore

ACM ISBN xxxxxx…$15.00
https://doi.org/1xxxxx

1 INTRODUCTION

Online advertising has become a rapidly growing industry, and effective personalization techniques such as click-through rate (CTR) prediction and conversion rate (CVR) prediction are crucial for ensuring efficient traffic distribution and reducing unpleasant user experiences. As part of the RecSys Challenge 2023 organized by ShareChat, a leading social media company in India, the task is to predict whether a user will install an app given their features, ad features, and historical interactions.

Modeling high-order interactions among raw features is a major challenge in the task due to the lack of clarity regarding the fundamental meaning of anonymous features in the raw data set. This challenge is distinct from practical scenarios. Additionally, user actions tend to follow a sequential pattern of "impression" → "click" → "conversion (install)," highlighting the importance of modeling both click and install objectives.

To tackle these challenges, in this paper we propose a two-stage solution with LightGBM for the RecSys Challenge 2023, taking the 4th place on the final leaderboard. More specifically, we initially conducted an extensive data analysis to gain insights into the core essence of the unprocessed anonymous features, i.e., 'date' and 'f_76/f_77' field, and explore the correlations among them. The latter one is for sake of subsequently find-grained feature engineering. Then, we construct two prediction models. Inspired by multi-task learning, we present a two-step approach to conduct $Model_{ts}$. We split the conversion rate estimation problem into two sub-tasks: modeling the probability that a user likes an ad and modeling the probability that a user installs an app given that they like the ad. $Model_{e2e}$ directly models the probability of user installation in an end-to-end manner. Last but not least, calibration and ensemble methods, are employed to further seek the predictive performance.

Roadmap. The rest of the paper is organized as follows: We conduct a detailed analysis of the dataset and propose a method for dividing it into training and validation sets in Sec.2. Feature engineering is detailed in Sec.3. After we elaborate the details of our proposed solution in Sec.4. The experimental results are provided in Sec.5. Finally, Sec.6 presents our conclusions.

2 DATA ANALYSIS AND PRE-PROCESSING

The ACM RecSys Challenge 2023 provides a dataset of anonymous online impressions sampled over 22 consecutive days. The first 21 days are used for training, while the 22nd day forms the test set. The dataset includes user and ad features (categorical, binary, numerical) and whether the user generates a click and/or install. The challenge task is to predict the is_installed label for the test set records.

2.1 Data Analysis

In-depth data analysis is crucial for competition success. We conducted four types of analysis: feature column analysis, missing value analysis, feature distribution, and label conflict analysis.

2.1.1 Feature Column Analysis. Advertising recommendation tasks typically involve both the advertisement and the user. Ad features appear repeatedly in multiple interaction data, while user features can differ greatly due to diverse sources. The main challenge of the task is the lack of clarity regarding the fundamental meaning of anonymous features in the raw data set. Based on our experience, we made initial guesses and classifications of features based on their source, as shown in Table 1.

Table 1: We made initial guesses about the domains of some anonymous feature columns, omitting the 'f_' prefix.

Guessing	Categorical	Numerical
User	5, 8, 10~18, 32	/
Ad	2~4, 6, 19~22	43, 51, 58, 59, 64~70
Interaction	/	42, 44~50, 52~57, 60~63, 71~79

Furthermore, we discovered that certain numerical features can be transformed into technical features based on specific base values. We calculated the base value for each field by dividing its maximum value by its minimum value. This calculation can provide a better understanding of the data, as demonstrated in Table 2.

Table 2: We guessed the base values of numerical feature columns and restored them to integers by performing divisional operations.

Feature Columns	Base Value
42, 52~57, 74~76	0.03856407
44~50, 71~73	0.57112147
60	8.07946039
61	0.147850899
62	0.129299709
63	0.355221093
77~79	37.38457512

2.1.2 Missing Value Analysis. Missing values can have a significant impact on model robustness and hinder its ability to be quickly transferred to other models such as XgBoost or DNN. To address this issue, we experimented with several approaches, including constant filling (0, -1, -99), statistical filling (mean, mode), and automated missing value imputation by LightGBM. Our results showed that constant filling with 0 consistently outperformed the other methods in terms of processing speed and model performance. Therefore, we recommend using constant filling with 0 for addressing missing values in this task.

2.1.3 Feature Distribution Analysis. To enable effective model training and ensure consistency between online and offline performance, we carefully examined the distribution of features in the training, validation, and test sets. We plotted the kernel density estimation (KDE) for each individual feature across the three datasets and categorized the feature distributions into three types:

Consistent: At least 70% (56/79) of the feature distributions were consistent, indicating that the model possessed certain learning conditions.

Inconsistent: Some features, such as f_11, f_43, f_51, and f_64, had inconsistent distributions, especially f_51 and f_64. By observing the KDE plots, we found that in certain ranges of the distribution, the training, validation, and test sets were almost completely inconsistent. Further analysis revealed that in the test set, some "Ad" features had never appeared in the training or validation sets, accounting for as much as 29.5% (from the perspective of f_51 and f_64).

Anomalous: This includes two types: 1) completely inconsistent distribution; 2) high homogeneity with minimal information content. For example, f_7 is a constant column, f_9 rotates between two values every day (with an unknown meaning), and f_23 is almost all zeros.

2.1.4 Label Conflict Analysis. To ensure data quality, we investigated the consistency between the input features and the "is_installed" labels. We employed a simple approach to identify instances where the input features were completely consistent, but the "is_installed" labels were inconsistent. Additionally, we analyzed the presence of duplicate data in the dataset. However, in our actual experiments, we found that attempting to resolve conflicts in the labels could lead to overfitting. Therefore, we ultimately did not take any measures to address conflicts in the samples.

Table 3: We performed a quality check for duplicate data and label conflicts. Note that the duplicate statistics did not consider the "is_clicked" label.

	Num. of Duplicated	Num. of Conflicts
Training	20017	1138
Validation	69	8
Leaderboard	163	/

2.2 Data Split

To avoid data leakage and overfitting, we used a heuristic approach to select day 66 as the validation set from the date range [45, 66] of the training data. This approach maximizes the use of all available data and ensures greater distribution consistency between the validation and test sets. Our experiments showed that other approaches, such as selecting earlier dates as the validation set or

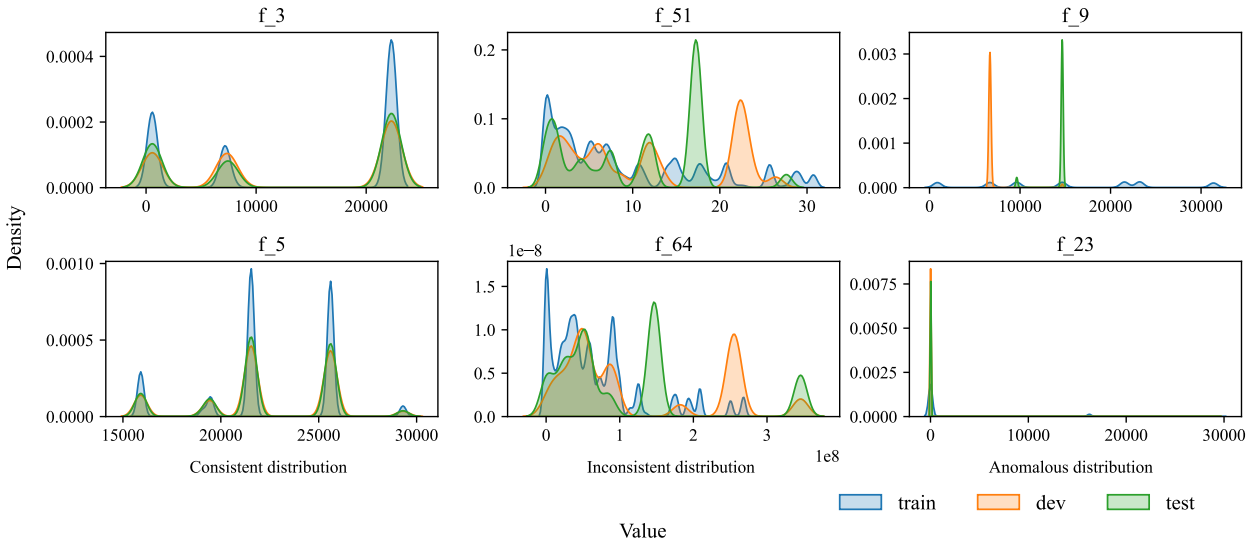

Figure 1: We plotted single feature KDE plots for three distribution types: consistent, inconsistent, and anomalous. Note that we filtered out training data where the f_64 feature was greater than 3e8 for plotting, which accounts for approximately 2.5% of the training set and is not present in the validation or test sets.

Table 4: Data splits used in our experiments

Split	Date Interval	Num. of Records
Training	45 - 65	3387157
Validation	66	97962
Leaderboard	67	160973

adding some day 66 data to the training set, resulted in significant performance degradation.

3 FEATURES

The quality and quantity of data available ultimately determine the performance of machine learning models. However, the effectiveness of feature engineering largely determines the degree to which we can reach this performance limit. Feature engineering helps identify relevant features, reduce noise and redundancy, and improve the accuracy and generalization ability of models. Therefore, successful feature engineering strategies can often help achieve better performance with limited data resources in machine learning competitions. In this section, we discuss commonly used feature engineering strategies [1].

3.1 Feature Selection

Based on the feature distributions, we first removed features with anomalous distributions, such as f_23. In its training set, the high-frequency item "16234" (6.8% proportion) does not exist in the test set, and the proportion of other non-zero values is only 0.02%. In the test set, the proportion of non-zero values is also only 0.03%. This indicates that the information carried by this feature is almost negligible. After considering feature importance, we ultimately removed the following feature columns: [f_9, f_23, f_24, f_47, f_49].

[1]The complete list of features used in our proposed approach can be found in the source code.

3.2 Feature Encoding

To begin with, our primary consideration is encoding categorical data into numerical values for computational efficiency. The given tabular dataset comprises 31 columns of multicategory features, with some categories having a maximum count of thousands. These rich categories encompass user information, item information, and their interaction information. The conventional one-hot encoding [6] method would result in a loss of similarity and correlation between categories. Hence, given the high cardinality of the categories, we have opted for Beta Target Encoding [7] to encode categorical features.

Suppose we have a dataset D that includes a binary target variable y and a categorical feature x. Our goal is to convert the categorical feature x into a continuous feature z that better reflects the relationship between x and y in a binary classification problem.

First, we compute the prior probability distribution $P(y)$ of the target variable y by calculating its prior mean μ, which is the overall proportion of positive samples in the dataset D. Here, $P(y)$ follows a Beta distribution $\text{Beta}(\alpha_0, \beta_0)$ where $\alpha_0 = \mu N_0$ and $\beta_0 = (1-\mu)N_0$, and N_0 is the number of samples in dataset D. This represents the basic relationship between the categorical feature x and the target variable y:

$$P(y) \sim \text{Beta}(\alpha_0, \beta_0), \quad \text{where} \quad \alpha_0 = \mu N_0, \quad \beta_0 = (1-\mu)N_0. \quad (1)$$

Next, we calculate the number of positive samples n and the total number of samples N for each category feature x. These are used to compute the posterior probability distribution $P(y|x)$, which is

also a Beta distribution Beta(α, β):

$$P(y \mid x) \sim \text{Beta}(\alpha, \beta), \quad \text{where} \quad \alpha = \alpha_0 + n, \quad \beta = \beta_0 + N - n. \tag{2}$$

To prevent a denominator of zero, we set a threshold $N_{\min}$ such that when $N < N_{\min}$ for a category x, we set $N = N_{\min}$ and α and β are set to α_0 and β_0.

Finally, we calculate the posterior probability distribution for each categorical feature x based on the desired statistics type (such as mean, variance, skewness and kurtosis) and use it as the new continuous feature z. If a category x does not appear in the training set, we set $P(y|x)$ to the median of $P(y)$.

In the competition, we use features generated from all the mentioned statistics type. Given the parameters α and β of a Beta distribution, we provide the following formulas to compute each of these measures:

Mean: The mean of the distribution can be calculated as:

$$P(y)_{\mu} = \frac{\alpha}{\alpha + \beta}. \tag{3}$$

Mode: The mode of the distribution can be calculated as:

$$P(y)_{Mo} = \frac{\alpha - 1}{\alpha + \beta - 2}. \tag{4}$$

Median: The median of the distribution can be calculated as:

$$P(y)_{Med} = \frac{\alpha - \frac{1}{3}}{\alpha + \beta - \frac{2}{3}}. \tag{5}$$

Variance: The variance of the distribution can be calculated as:

$$P(y)_{\sigma^2} = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}. \tag{6}$$

Skewness: The skewness of the distribution can be calculated as:

$$P(y)_{\gamma_1} = \frac{\alpha - \beta}{\sqrt{\alpha\beta(\alpha + \beta + 1)}}. \tag{7}$$

Kurtosis: The kurtosis of the distribution can be calculated as:

$$P(y)_{\gamma_2} = \frac{\alpha\beta(\alpha + \beta + 1)}{(\alpha + \beta + 2)(\alpha + \beta)(\alpha\beta)} - 3. \tag{8}$$

These formulas can be used to efficiently compute these statistical measures for any given Beta distribution, providing valuable insights into the shape and characteristics of the distribution.

3.3 Feature Augmentation

Temporal Feature. According to the challenge provided, the feature column named f_1 represents a date feature. This feature is an integer value ranging from 45-67, with 67 only appearing in the test set. In order to capture cyclic variations in the data, we create a new date feature column by taking the modulo 7 of f_1, representing one day within a 7-day period. This preprocessing step allows us to better analyze and model the cyclic nature of the data.

Categorical Feature & Categorical Feature. The dataset documentation did not specify the significances of individual features. However, through exploratory data analysis, we identified interdependencies between some feature pairs. For instance, the categorical features f_4 and f_6 exhibited high collinearity, where f_4 could only take one value given f_6. To address such feature coupling, we concatenated the strings of correlated categorical features to create a new merged category and employed a label encoding scheme to map the concatenated string to an integer value. This approach mitigated the effects of multicollinearity while preserving the intrinsic relationship between the correlated features. The transformed combined feature captured more information than the original separate features, resulting in improved model performance and generalization.

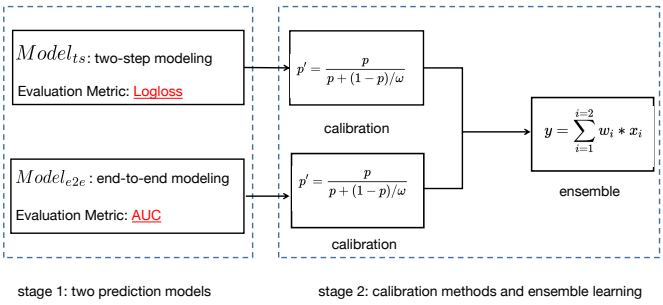

Figure 2: The Overall Proposed Solution Architecture.Our proposed solution architecture consists of two prediction models: $Model_{ts}$ and $Model_{e2e}$. $Model_{ts}$ splits the conversion rate estimation problem into two sub-tasks, namely, modeling the probability that a user likes an ad and modeling the probability that a user installs an app given that they like the ad. On the other hand, $Model_{e2e}$ directly models the probability of user installation end-to-end. In the second stage, we employ ensemble learning and calibration methods to improve the accuracy of the prediction.

Categorical Feature & Numerical Feature. The entire production process can be summarized into three steps: grouping, statistics, and transformation. Assuming A represents categorical data and B represents numerical data. The process can be summarized as follows.First, all data (columns A and B) are grouped according to the specified categories of column A. Column B data within each group is then subjected to statistical analysis to compute metrics including but not limited to mean, variance and median. The statistical metrics for each group are concatenated to form new derived feature columns.

4 PROPOSED SOLUTION

Our solution, depicted in Figure 2, involved two stages: optimizing two prediction models for conversion rate estimation, and then applying calibration and ensemble methods to enhance prediction accuracy. Finding the optimal hyperparameters for each model is a critical step to maximize prediction accuracy. To achieve this, we utilized Bayesian optimization with the Optuna framework [1] for all prediction models. Our results demonstrate that consistently using the same feature engineering methods across all prediction models can significantly improve their performance, which has practical implications for applying machine learning in real-world scenarios.

4.1 Prediction Models

4.1.1 Evaluation Metric. In real-world scenarios, the accuracy of predicted order and precision of predicted values are crucial for effective decision-making [8]. In this paper, as depicted in Figure 2, we underscore the significance of evaluating both order accuracy and value precision in prediction models. We propose two prediction models that optimize different evaluation metrics, namely AUC and logloss. By utilizing these distinct evaluation methods, we can assess model performance from two different perspectives. Our findings reveal that these two evaluation metrics provide complementary information about model performance. Therefore, we recommend that future research and practical applications of prediction models consider both evaluation metrics to ensure the overall efficacy of the models in real-world scenarios.

4.1.2 Two-step Modeling. The dataset was annotated with two binary fields, is_clicked and is_installed, indicating whether a user clicked an ad and installed the app. We can partition the entire dataset into four non-overlapping subsets based on the values of these fields as follows:

(1) (0,0) corresponds to impressions with neither clicks nor installs.
(2) (1,0) corresponds to impressions with clicks but no installs.
(3) (0,1) corresponds to impressions with no clicks but installs[2].
(4) (1,1) corresponds to impressions with both clicks and installs.

In this paper, we introduce a new label called "is_like", derived from the binary variables "is_clicked" and "is_installed". The "is_like" label is calculated using the boolean OR operation. That is, if a user has either clicked or installed, the "is_like" label is set to 1; otherwise, it is set to 0. By introducing the "is_like" label, we can model the probability of installation given user and ad as a product of two conditional probabilities: the probability of "is_like" given user and ad, and the probability of installation given user, ad, and "is_like".

$$p(is\_installed|user, ad) = p(is\_like|user, ad) * p(is\_installed|user, ad, is\_like) \quad (9)$$

Inspired by ESMM [5], we propose a two-step modeling approach using LightGBM to improve prediction accuracy. In the first step, we train a LightGBM model to predict the probability of "is_like" given user and ad. In the second step, we train another LightGBM model to predict the probability of installation given user, ad, and "is_like". This approach allows us to capture complex relationships between user, ad, and installation behavior, and improve prediction accuracy. Although it would be possible to use a deep neural network to simultaneously optimize the two probabilities, we were unable to implement this approach due to time constraints during the competition. However, we provide a detailed design plan for this approach in Appendix A.

4.2 Ensemble and Calibration Method

Calibration is a crucial aspect of predictive modeling as it ensures that predicted probabilities are consistent with the true probabilities. In this study, we utilize a scaling formula for calibration proposed in [3], given by $p^{'} = \frac{p}{p+(1-p)/\omega}$, where p is the prediction and ω is a single parameter that can alleviate bias caused by distribution shifts. To calculate the scaling parameter, we use the formula $\omega = \frac{p_{real}}{p_{pred}}$, where p_{real} is the true installation rate in the validation set, and p_{pred} is the average predicted installation rate. This enables us to adjust predicted probabilities to match true probabilities and reduce bias due to global data distribution shifts, ensuring proper calibration of our models.

[2]It should be noted that in the case of online advertising, it is possible that a user has viewed an ad and then didn't click on the ad and directly installed the underlying application. The implication of this behavior is that we have records where there is no click but there is an install.

5 EXPERIMENTS

We evaluated the performance of different models on the dataset. Table 5 summarizes the results of our experiments, including the leaderboard score and ranking of each model.

We compared the performance of two different models, $Model_{ts}$ and $Model_{e2e}$, which were constructed to optimize different evaluation metrics, AUC and logloss, respectively. We also evaluated the performance of calibrated versions of both models, denoted as $Model_{ts}$-calibration and $Model_{e2e}$-calibration. Finally, we assessed the performance of an ensemble model, denoted as *Model*-ensemble, which combined the predictions of $Model_{ts}$-calibration and $Model_{e2e}$-calibration using a weighted average with weights of 0.5 and 0.5.

Our results show that the *Model*-ensemble achieved the best performance, with a leaderboard score of 5.958641 and a ranking of 4. This indicates that the ensemble of the two models can effectively leverage the complementary information provided by the two evaluation metrics and improve the accuracy of prediction. Moreover, the calibrated versions of the models significantly improved their performance, demonstrating the effectiveness of the calibration method in improving the accuracy of prediction.

In conclusion, our experiments demonstrate the effectiveness of the proposed two-stage modeling approach for predicting the probability of installation given user and ad. The ensemble of the two models achieved the best performance, highlighting the importance of considering both AUC and logloss in evaluating model performance. Our work provides valuable insights and practical guidance for applying machine learning methods in real-world scenarios.

Table 5: Performance comparison of different models.

Model	LB Score	LB Ranking
$Model_{ts}$	6.039694	6
$Model_{e2e}$	6.01	4
$Model_{ts}$-calibration	6.024205	6
$Model_{e2e}$-calibration	6.003765	4
Model-ensemble	**5.958641**	**4**

6 CONCLUSION

In conclusion, this paper presents the two-stage approach developed by the team "RoyalFlush" for the ACM RecSys Challenge 2023, organized by Mohalla Tech Pvt. Ltd. The competition focused on predicting whether a user will install an app or not for a given ad impression in the online advertising domain. This study demonstrates the effectiveness of capturing the underlying mechanisms

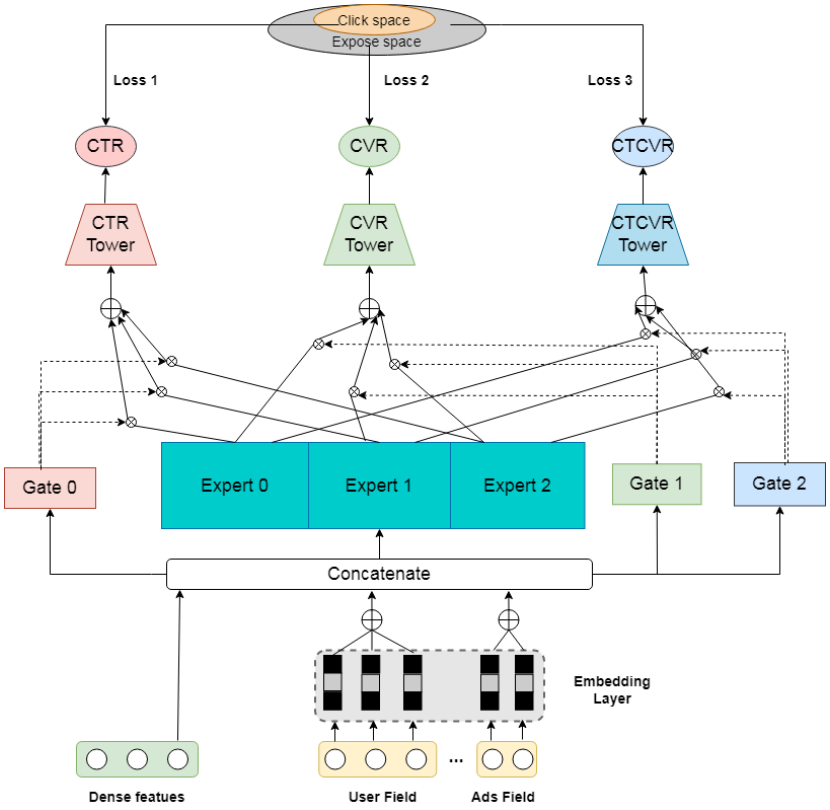

Figure 3: Our multi-task deep model

of user behavior and applying calibration methods in online advertising conversion rate estimation. The results of this study provide important insights for future research in online advertising. Future research could explore more advanced deep learning techniques and deeper analysis of user behavior to further improve the accuracy of conversion rate estimation.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[2] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-Task Learning for Deep Text Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM.

[3] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the eighth international workshop on data mining for online advertising*. 1–9.

[4] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling Task Relationships in Multi-Task Learning with Multi-Gate Mixture-of-Experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. 1930–1939.

[5] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1137–1140.

[6] Kedar Potdar, Taher S. Pardawala, and Chinmay D. Pai. 2017. A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. *International Journal of Computer Applications* 175 (2017), 7–9.

[7] Pau Rodríguez, Miguel A. Bautista, Jordi Gonzàlez, and Sergio Escalera. 2018. Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing* 75 (2018), 21–31. https://doi.org/10.1016/j.imavis.2018.04.004

[8] Xiang-Rong Sheng, Jingyue Gao, Yueyao Cheng, Siran Yang, Shuguang Han, Hongbo Deng, Yuning Jiang, Jian Xu, and Bo Zheng. 2022. Joint Optimization of Ranking and Calibration with Contextualized Hybrid Model. *arXiv preprint arXiv:2208.06164* (2022).

[9] Yuhao Wang, Ha Tsz Lam, Yi Wong, Ziru Liu, Xiangyu Zhao, Yichao Wang, Bo Chen, Huifeng Guo, and Ruiming Tang. 2023. Multi-Task Deep Recommender Systems: A Survey. arXiv:cs.IR/2302.03525

Appendix A MULTI-TASK DEEP MODEL

To accurately predict the probability of user installation, we proposed a multi-task modeling approach that simultaneously modeled the click-through rate (CTR), the conversion rate (CVR), and the conversion-to-click-through rate (CTCVR) based on users' behavior of clicking on advertisements.In this paper, we use CTR to refer to the estimated probability of a user liking an ad, which is equivalent to the is_like metric used in the experiments.Our objective was to facilitate knowledge transfer from a user's preceding behavior to their subsequent behavior through joint training. To effectively capture useful information and relevance among tasks, we incorporated the multi-task learning paradigm into our multi-task deep model (MTDM). The MTDM leverages the ability of deep neural networks to learn high-order feature interactions and model complex user-item interaction behaviors [2]. However, the MTDM faced two challenges: effectively capturing task information and accounting for the unique sequential dependency that arises from the sequential pattern of user actions across tasks. In this paper, we present

our strategy for the MTDM, depicting it through two aspects: task relation and parameter sharing mode, as shown in Figure 3.

Appendix A.1 Task Relation

Our MTDM can be categorized into three paradigms depending on the relation of tasks: parallel, cascaded, and auxiliary with main tasks [9]. We calculated the CTR, CVR, and CTCVR independently, based on the is_like and is_installed labels in the dataset, without the sequential dependency of their results, similar to the parallel modeling paradigm. As samples that are exposed but not clicked violate the definition of CVR, the objective function of the CVR task was defined as the weighted sum of losses without samples where the click label equals 1, while the other two targets considered the entire exposed sample space.

Compared with tackling multiple tasks separately and cascadedly, our modeling paradigm offered two main benefits. Firstly, by exploiting data and knowledge across multiple tasks, it achieved mutual enhancement among the tasks. For example, the installation task had fewer positive samples, and if trained separately, the shared bottom layer would not be sufficiently trained. Secondly, CVR was optimized in the sample space where the click label equals one and was not affected by samples that are exposed but not clicked. In serial modeling, CVR was a hidden target and indirectly optimized while CTCVR was being optimized. We could use the model's output of CTCVR as the installation probability or multiply the model's outputs of CVR and CTR manually as the installation probability.

Appendix A.2 Parameter Sharing

MTDM's MMOE [4] structure uses different expert networks to handle different tasks, allowing each task to focus on its specific area of expertise. When processing each sample, the MMOE structure dynamically adjusts the degree of parameter sharing based on the current sample and task, so that each task can update its own parameters without interfering with other tasks.

For the CTR and CTCVR tasks, all exposed samples are used to update the parameters because both tasks need to consider both exposures and clicks. In contrast, for the CVR task, only samples with is_clicked = 1 are used for parameter updates because the CVR task only needs to consider clicks and non-clicked samples do not fit the definition of CVR.

Therefore, in the MMOE structure, each task updates its own parameters based on the weights of its corresponding expert network, and does not interfere with the parameters of other tasks. This approach allows each task to focus on its own objectives and improve the model's performance and accuracy.

In summary, the MTDM approach leverages the MMOE structure to enable each task to update its own set of parameters based on the relevant expert networks, reducing interference between tasks. While all exposed samples are used to update the parameters for CTR and CTCVR tasks, only samples with is_clicked = 1 are used for CVR updates. This enables each task to focus on its specific objectives and improves the overall model performance.