Pessimistic Rescaling and Distribution Shift of Boosting Models for Impression-Aware Online Advertising Recommendation

Paolo Basso
paolo3.basso@mail.polimi.it
Politecnico di Milano
Italy

Arturo Benedetti
arturo.benedetti@mail.polimi.it
Politecnico di Milano
Italy

Nicola Cecere
nicola.cecere@mail.polimi.it
Politecnico di Milano
Italy

Alessandro Maranelli
alessandro.maranelli@mail.polimi.it
Politecnico di Milano
Italy

Salvatore Marragony
salvatore.marragony@mail.polimi.it
Politecnico di Milano
Italy

Samuele Peri
samuele.peri@mail.polimi.it
Politecnico di Milano
Italy

Andrea Riboni
andrea.riboni@mail.polimi.it
Politecnico di Milano
Italy

Alessandro Verosimile
alessandro.verosimile@mail.polimi.it
Politecnico di Milano
Italy

Davide Zanutto
davide.zanutto@mail.polimi.it
Politecnico di Milano
Italy

Maurizio Ferrari Dacrema
maurizio.ferrari@polimi.it
Politecnico di Milano
Italy

ABSTRACT

In this paper, we provide an overview of the approach we used as team Gabibboost for the ACM RecSys Challenge 2023, organized by ShareChat and Moj. The challenge focused on predicting user activity in the online advertising setting based on impression data, in particular, predicting whether a user would install an advertised application using a high-dimensional anonymized feature vector. Our proposed solution is based on an ensemble model that combines the strengths of several machine learning sub-models, including CatBoost, LightGBM, HistGradientBoosting, and two hybrid models. Our proposal is able to harness the strengths of our models through a distribution shift postprocessing and fine-tune the final prediction via a custom build pessimistic rescaling function. The final ensemble model allowed us to rank 1st on the academic leaderboard and 9th overall.

KEYWORDS

ACM RecSys Challenge 2023; Recommender systems, Machine Learning, Online Advertising

ACM Reference Format:
Paolo Basso, Arturo Benedetti, Nicola Cecere, Alessandro Maranelli, Salvatore Marragony, Samuele Peri, Andrea Riboni, Alessandro Verosimile, Davide Zanutto, and Maurizio Ferrari Dacrema. 2023. Pessimistic Rescaling and Distribution Shift of Boosting Models for Impression-Aware Online Advertising Recommendation. In *ACM RecSys Challenge 2023 (RecSysChallenge '23), September 19, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3626221.3627288

RecSysChallenge '23, September 19, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-1613-3/23/09.
https://doi.org/10.1145/3626221.3627288

1 INTRODUCTION

The ACM RecSys Challenge 2023, organized by ShareChat and Moj, focuses on the role of impressions in the online advertising setting [5]. Impressions, also called exposure data or past exposures, contain the previous recommendations to users, meaning the items or products displayed on users' screens [15, 16]. Impressed items usually come from the existing recommendation system, a search function, or business rules. The goal of the challenge is to predict whether a user will install an advertised application or not, given a feature vector representing information about the user and the advertisement. The proposed final solution is an ensemble to combine the predictions of several models. The source code of our final model and the documentation are publicly available on GitHub.[1]

The paper is structured as follows. In Section 2 we provide an overview of the problem, analyze the dataset, and the evaluation metrics used in the challenge. In Section 3 we describe how we split the dataset to exploit the temporal information and how we preprocessed the features. In Section 4 we describe the models that we employed as well as the ensembling technique. In Section 5 we discuss the results and draw conclusions.

2 PROBLEM FORMULATION AND DATA ANALYSIS

In this section, we formulate the competition task and describe the data analysis we performed.

[1] https://github.com/recsyspolimi/recsys-challenge-2023-sharechat

2.1 Problem formulation

The dataset for this edition of the ACM RecSys Challenge 2023 was issued by Sharechat. The dataset contains 3.485.852 impressions, coming from the Sharechat and Moj users, sampled from a period of 22 days. Each impression represents an application advertisement shown to a user and is characterized by 79 features and two binary labels, one indicating whether the advertisement was clicked or not (`is_clicked`) and the other whether the application was installed or not (`is_installed`). The challenge task is binary classification: given the impressions of the first day after the training data (the 23rd day) the goal is to predict whether the associated application was installed or not. A secondary goal is to predict whether an advertisement was clicked or not. The evaluation metric is binary cross-entropy, normalized by a constant that was not disclosed. We will refer to the negative or positive class with respect to the `is_installed` label.

2.2 Data analysis

In the dataset each impression has 79 features, we refer to them as `f_i` with `i` being the feature identifier. The meaning of each feature was not disclosed, except for the date. The lack of information on the feature semantics did not allow us to perform a traditional feature engineering process based on domain knowledge. We only know the semantics of `f_1`, which represents the number of the day. The features can be grouped in three: from `f_2` to `f_32` they have categorical values, from `f_33` to `f_41` they have boolean values and from `f_42` to `f_79` they have numerical values. Considering, in particular, the label `is_installed`, the data is strongly unbalanced towards negative samples, only 17% of the impressions are associated with the application being installed. Our analysis focused on five main aspects:

- **Understanding the meaning of the features:** we looked for patterns in features that could inform us about their meaning. The only interesting patterns we discovered were for `f_9` and `f_11`. Feature `f_9` turned out to be periodical every 7 days, so it likely represents the day of the week in which the impression was shown; feature `f_11` consists of 24 distinct values with a periodic pattern, so it likely represents the hour of the day in which the impression was shown. Feature `f_7` has a constant value for every impression, therefore it is not informative and we decided to remove it.
- **Analysis of the value distribution:** we obtained useful insights from a mathematical analysis of the numerical features, by relying on the probability density distributions and on statistical properties like mean and covariance. An example is feature `f_64`, which has a huge range of values, with a maximum value over 10^{12}, a mean value over 10^{9}, and a very large standard deviation.
- **Counter features:** looking at the values of the numerical features, we discovered that many of them likely represent counters.[2] By normalizing these features based on the smallest non-zero value we obtained natural numbers. The majority of values is zero, with the distribution of the others following a decreasing exponential trend. This suggests those features represent counters and therefore we processed them in a different way compared to the other numerical features.
- **Behavior during different days:** to assess whether the behavior of the features varies we plotted their value distribution on different days. We discovered a weekly periodicity, which we used to define the data splitting, as described in Section 3.1. We were also able to identify days with anomalous behaviors, for example, day 52, in which the mean and variance of many numerical features differ greatly from those of the overall training data.
- **Correlation analysis:** this was the most useful analysis and allowed us to highlight important similarities among features. For numerical features we compared correlation matrices obtained using 3 different correlation coefficients: Pearson correlation [18], Kendall Tau [13], and Spearman rank correlation [19]. All these methods showed similar results, highlighting clusters of features with very high correlation. We identified two features that carried the same information, `f_43` and `f_66`, of which we kept only one. For categorical features, we used Cramer's V correlation [7], which highlighted a very strong correlation among many features. In particular, we observed that all the features from `f_22` to `f_29` carried either the same or very similar information. In particular, within this group of features, we identified many nested relationships. For instance, the categories of `f_23` are sub-categories of `f_25`, which are sub-categories of `f_26`, and similarly for other couples of features.

[2] In particular: `f_42`, `f_44` to `f_50`, `f_52` to `f_57`, `f_60` to `f_63` and `f_71` to `f_79`.

3 DATA PREPROCESSING AND FEATURE ENGINEERING

In this section, we describe the data splitting and preprocessing adopted, based on the findings of the data analysis.

3.1 Data split

The impressions in the data have a temporal order, therefore performing a training-validation split with random sampling is not ideal. Given that the task of the challenge is to classify impressions from the day after the training data and assuming that the impressions within a day are distributed in a very similar way, we sampled the data by holding-out entire days to prevent overfitting and encourage the model's generalization. We identified two different splitting strategies that we applied to different methods. First, assuming that the closer two days are in time, the more they will be characterized by a similar feature distribution, we used the last available day in the training set as validation. Second, we hypothesize that the data distribution exhibits a periodic behavior within a week, with each day being similar to the corresponding day in different weeks. For this reason in this second split, the validation data is the same weekday of the test set but one week prior. Finally, since a single day may not contain enough data for a consistent validation set, we decided to also include the day preceding the selected one.

3.2 Preprocessing

Based on the data analysis we applied the following preprocessing:

Outliers removal: In numerical features that did not contain counters, we removed the outliers and replaced them with the mean feature-wise using the Mean and Standard Deviation Method [3].

Cyclic encoding of the date: We mapped the feature representing the date (`f_1`) into two new features by applying the sine and the cosine functions. This encoding technique allows to capture the cyclic nature of the data while maintaining a continuous numerical representation and keeping an equal relative distance between the original points. We decided to repeat the cyclic sequence every seven days.

Preprocessing of `f_9`: This feature shows cyclic behavior every seven days. Hence, assuming it is time-dependent and that the difference between users is only due to their different time zones, to better capture its semantics for each training day we changed its value with the mode computed on the same day.

Feature selection: We decided to remove features that were constant or highly correlated, as described in Section 2.2, and we performed backward feature selection to remove detrimental features.

Anomalous days: As stated in Section 2.2, some days exhibit different behavior compared to others; in particular days 48, 50, and 52. Therefore, we removed them from the training data.

Hand-crafted features: We created new features to use during the training phase of the models by performing pairwise multiplication within a group of correlated features, found with the correlation analysis. The idea is to better exploit the 'correlation' information and capture more nuanced patterns.

4 PROPOSED SOLUTION

In this section, we describe the structure of the final model, all its components, and the techniques we used to merge them to obtain the final result. In addition, we also describe some of the ideas that proved unsuccessful to provide a more comprehensive overview of our work. We start with the description of the models that were combined in our final ensemble, which we refer to as sub-models, and then describe the final ensemble.

4.1 CatBoost

CatBoost is a gradient boosting algorithm [17] particularly effective when working with datasets that contain a mix of numerical and categorical features. The preprocessing used are *Cyclic encoding of the date* and *Outliers removal* (see Section 3). Additionally, we sorted the training impressions by date because CatBoost can leverage the temporal order of the dataset.[3] After the hyperparameter tuning CatBoost reached the best score among all the sub-models. Notice that some noisy features were dropped following the feature selection method as described in Section 3.2.

4.2 LightGBM and HistGradientBoosting

LightGBM and HistGradientBoosting are gradient boosting frameworks that use tree-based learning algorithms [11, 12]. Both methods generally allow to obtain accurate models with fast training times [6, 8, 10]. The training data was preprocessed as follows: *Preprocessing of* `f_9` , *Feature selection*, *Anomalous days*, *Hand-crafted features* (see Section 3.2).

LightGBM. Despite the noise removal attempts, *LightGBM* showed a huge variance in its performance. To address this issue we increased the regularization of the model to reduce the impact of noise in categorical features.[4] Good results on the validation data have also been achieved by optimizing the focal loss function [14] instead of the normalized entropy in the training phase; unfortunately a similar improvement did not materialize on the public leaderboard.

HistGradientBoosting. We applied an additional preprocessing for this method by considering as categorical features only those having less than a certain number of distinct values which was treated as a hyperparameter. [5]

4.3 Ensemble Models

The ensemble models are built using the available features and include, as additional ones, the predictions computed by some of the sub-models. To this end, the following models were trained:

- A *LightGBM* containing as features both *LightGBM* and *HistGradientBoosting* predictions;
- A *HistGradientBoosting* containing as features both *LightGBM* and *HistGradientBoosting* predictions.

The design of these ensemble models introduced several challenges, such as the different behavior of the sub-models between training set and test set. Consequently, it was quite difficult for the hybrid models to accurately discern the precise and imprecise predictions of the sub-models. Another challenge was the choice of the training-validation split to first train the sub-models, then generate the predictions on which to train the ensemble models while avoiding overfitting. We adopted the following approach:

(1) We trained a different *LightGBM* and *HistGradientBoosting* model on the whole training set, excluding a single day that we used to compute the predictions, starting from the first day. In this way, to predict the labels of the impressions belonging to a given day, the models were trained on all the past and future days.
(2) We repeated this procedure for all the existing training days, introducing two new features containing the predictions of each of the two models.

In the end, the two ensemble models were trained in the same way as described in Section 4.2, using the same hyperparameters used for the sub-models.

4.4 Distribution Shift Postprocessing

We observed a particular issue with some models such that the predictions generated did not have the appropriate magnitude, which resulted in decreased normalized cross entropy. However, these models were remarkably accurate in establishing an order of the samples based on the predictions. Essentially, these models excelled

[3] This was done using the hyperparameter `has_time` [1].

[4] This was done using the hyperparameter `cat_smooth` [2].

[5] This was done using the hyperparameter `max_bins`.

at arranging predictions in ascending order from the least confident to the most confident. On the other hand, their proficiency in assigning probabilities that minimize the normalized entropy was comparatively less robust. Based on this observation, we decided to take the ordering produced by a given model A but shift its predictions by replacing them with the ones produced by our best-performing model, *CatBoost*. We applied the following steps:

(1) Use both model A and *CatBoost* to compute the predictions for the same batch of impressions;
(2) Order each model prediction values in ascending order;
(3) Assign to the ordered samples of the model A the ordered predictions of *CatBoost*.

This allowed us to exploit the diversity of the models and leverage their respective strengths: the capability of model A to establish a successful order of the samples and the capability of *CatBoost* to estimate a sound probability distribution.

4.5 Unsuccessful Ideas

Not all the models we explored met our expectations and, indeed, some yielded unsatisfactory results. We report the two most interesting ones because we believe they can provide useful information.

Clustering. Various clustering algorithms were investigated, but their effectiveness was hindered primarily by the extensive computational time they required. For instance, we experimented with the *k-prototypes* algorithm, which can handle a combination of categorical and numerical features. Unfortunately, the processing time was excessive. Similar issues arose with the DBSCAN and KNN algorithms when the categorical features were encoded as numerical values. Among the clustering algorithms we tested, only *k-means* did not encounter this problem. We observed that some of the clusters identified in the training set were interesting, exhibiting either a high or low percentage of positive samples. However, these findings did not carry over to the public leaderboard. In fact, the test samples were assigned to less informative clusters which was not helpful for the final classification task.

Autoencoder for anomaly detection. The idea was to train an autoencoder [9] with only samples from the negative class, i.e., the impressions with no interactions, which is the most common. This method exploits the fact that autoencoders learn to reconstruct normal ('negative') data with a lower error rate compared to abnormal ('positive') data, given that they have not been trained on the latter. Assuming there is a significant difference between the negative and positive classes, the reconstruction error for samples belonging to the positive class should be noticeably higher. Thus, we introduced a threshold value t for the cosine similarity c calculated between the original sample and its reconstructed counterpart. When the cosine similarity of a sample exceeds the threshold ($c > t$), the sample is classified as belonging to the negative class, otherwise, it is classified as positive.

4.6 Final Ensemble and Interval Rescaling

The final ensemble is built as depicted in Figure 1, both by weighting the predictions and postprocessing them. The weights have been chosen empirically, proportionally with respect to the performance of the single models. The ensemble combines five different models: *CatBoost* (see Section 4.1), *LightGBM* and *HistogramGradientBoosting* (see Section 4.2), two ensembles using *LightGBM* and *HistogramGradientBoosting* (see Section 4.3). The distribution shift postprocessing (see Section 4.4) is applied to the last four models,

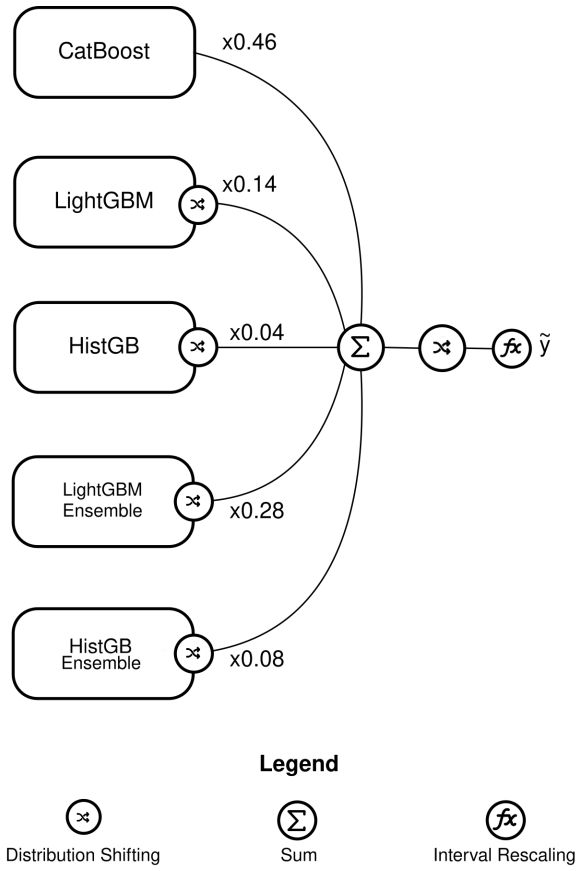

Figure 1: High-level diagram of the final ensemble model. Note that *HistGB* indicates the *HistGradientBoosting* method.

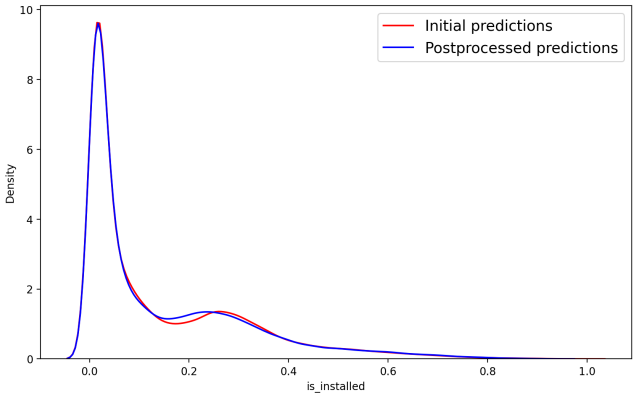

Figure 2: Prediction values for LightGBM before and after the distribution shift postprocessing.

Model	Cross-entropy
CatBoost	6.085
LightGBM	6.114
HistGradientBoosting	6.178
LightGBM-Ensemble	6.095
HistGradientBoosting-Ensemble	6.151
Final Ensemble	6.056
Final Ensemble with Interval Rescaling	**6.053**

Table 1: Model results obtained on the public leaderboard.

before combining the predictions given by all the models through a weighted sum, as well as on the result of the final linear combination of models. The final step in computing the overall model predictions involves an interval rescaling function, detailed in Equation 1. This function serves to adjust the model prediction p using two thresholds a and b that can be tuned as hyperparameters. In practice, we found that this operation makes predictions more pessimistic.

$$\text{interval rescaling}(p) = \begin{cases} 0 & p < a \\ \frac{1}{b-a}p + \frac{a}{a-b} & a \leq p \leq b \\ 1 & p > b \end{cases} \quad (1)$$

5 RESULTS AND CONCLUSIONS

In this section, we describe the experimental protocol and the results obtained on the public leaderboard.

Experimental protocol. The validation of the models was carried out using our local resources and Amazon AWS services, using a validation set that for some models consisted of two days from the week preceding the test set day, the first one being exactly seven days before and the second being its previous day, while for other models was the last two days of the training data. All the hyperparameters, both of the sub-models and of the postprocessing strategies, have been tuned on the validation set with Optuna [4], which proved an effective tool in previous challenges [6, 8], exploring a number of cases in the order of thousands.

Results. Table 1 reports the normalized cross-entropy obtained by all models on the public leaderboard. We can see that the final ensemble provides an improvement compared to all individual sub-models taken separately. Nonetheless, the results show how the boosting algorithms alone are able to obtain effective performance. We used only models based on gradient-boosted decision trees. As expected, their performance is very similar. *Catboost* is the model that obtains the best results standalone. This can be explained by the fact that the dataset is a mix of categorical, binary, and numerical features, and the categorical features proved to be the most meaningful to predict the label. Indeed, by analyzing the feature importance of the 15 most important features we can see that the 6 most important ones are categorical, and a total of 10 out of 15 are categorical. Therefore *Catboost*, which was specifically created to better handle categorical features, slightly outperforms the other models. We can also see that the ensembling is able to improve the performance of both LightGBM and HistGradientBoosting slightly. Considering the postprocessing, Figure 2 shows the difference within the prediction distributions of LightGBM before and after applying the distribution shift postprocessing. We can see how the score distribution shifts slightly for predictions around 0.2 while remaining mostly unchanged for high and low values. Regarding the interval rescaling postprocessing, the hyperparameter tuning found the following values $a = 0.0012, b = 1.0060$, meaning that the rescaling acts in a *pessimistic* way by sightly reducing all the model predictions and achieving a final improvement over the ensemble model. Overall in our ensemble method, we were effectively able to combine the predictions coming from different models by introducing tailored postprocessing steps. This ensemble method allowed us to reach the 1st position on the academic leaderboard and the 9th overall.

ACKNOWLEDGMENTS

We would like to thank Prof. Paolo Cremonesi for his support.

REFERENCES

[1] 2023. Catboost documentation. https://catboost.ai/en/docs/references/training-parameters/common.
[2] 2023. LightGBM documentation. https://lightgbm.readthedocs.io/en/stable/Parameters.html.
[3] 2023. Outlier detection method. https://docs.oracle.com/cd/E40248_01/epm.1112/cb_statistical/frameset.htm?ch07s02s10s01.html.
[4] 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.
[5] Sarang Brahme, Rahul Agarwal, Liu Yong Abhishek Srivastava, and Athirai Irissappane. 2023. Recsys Challenge 2023: Challenge task. http://www.recsyschallenge.com/2023/.
[6] Luca Carminati, Giacomo Lodigiani, Pietro Maldini, Samuele Meta, Stiven Metaj, Arcangelo Pisa, Alessandro Sanvito, Mattia Surricchio, Fernando Benjamín Pérez Maurera, Cesare Bernardis, and Maurizio Ferrari Dacrema. 2021. Lightweight and Scalable Model for Tweet Engagements Predictions in a Resource-constrained Environment. In *RecSysChallenge '21: Proceedings of the Recommender Systems Challenge 2021 (RecSysChallenge 2021), October 1, 2021, Amsterdam, Netherlands*. ACM, 8 pages. https://doi.org/10.1145/3487572.3487597
[7] HARALD CRAMÉR. 1999. *Mathematical Methods of Statistics (PMS-9)*. Princeton University Press. http://www.jstor.org/stable/j.ctt1bpm9r4
[8] Nicola Della Volpe, Lorenzo Mainetti, Alessio Martignetti, Andrea Menta, Riccardo Pala, Giacomo Polvanesi, Francesco Sammarco, Fernando Benjamin Perez Maurera, Cesare Bernardis, and Maurizio Ferrari Dacrema. 2022. Lightweight Model for Session-Based Recommender Systems with Seasonality Information in the Fashion Domain. In *Proceedings of the Recommender Systems Challenge 2022* (Seattle, WA, USA) *(RecSysChallenge '22)*. Association for Computing Machinery, New York, NY, USA, 18–23. https://doi.org/10.1145/3556702.3556829
[9] Raja Giryes Dor Bank, Noam Koenigstein. 2003. Autoencoders. https://arxiv.org/pdf/2003.05991.pdf
[10] Nicolò Felicioni, Andrea Donati, Luca Conterio, Luca Bartoccioni, Davide Yi Xian Hu, Cesare Bernardis, and Maurizio Ferrari Dacrema. 2020. Multi-Objective Blended Ensemble For Highly Imbalanced Sequence Aware Tweet Engagement Prediction. In *RecSys Challenge '20: Proceedings of the Recommender Systems Challenge 2020, Virtual Event Brazil, September, 2020*. ACM, 29–33. https://dl.acm.org/doi/10.1145/3415959.3415998
[11] Aleksei Guryanov. 2019. Histogram-based algorithm for building gradient boosting ensembles of piecewise linear decision trees. In *Analysis of Images, Social Networks and Texts: 8th International Conference, AIST 2019, Kazan, Russia, July 17–19, 2019, Revised Selected Papers 8*. Springer, 39–50.
[12] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).
[13] M. G. Kendall. 1938. A New Measure of Rank Correlation. *Biometrika* 30, 1/2 (1938), 81–93. http://www.jstor.org/stable/2332226
[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.
[15] Fernando Benjamín Pérez Maurera, Maurizio Ferrari Dacrema, and Paolo Cremonesi. 2022. Towards the Evaluation of Recommender Systems with Impressions. In *RecSys '22: Sixteenth ACM Conference on Recommender Systems, Seattle, WA, USA, September 18 - 23, 2022*, Jennifer Golbeck, F. Maxwell Harper, Vanessa Murdock, Michael D. Ekstrand, Bracha Shapira, Justin Basilico, Keld T. Lundgaard, and

Even Oldridge (Eds.). ACM, 610–615. https://doi.org/10.1145/3523227.3551483

[16] Fernando Benjamín Pérez Maurera, Maurizio Ferrari Dacrema, Lorenzo Saule, Mario Scriminaci, and Paolo Cremonesi. 2020. ContentWise Impressions: An Industrial Dataset with Impressions Included. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (Eds.). ACM, 3093–3100. https://doi.org/10.1145/3340531.3412774

[17] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems* 31 (2018).

[18] Philip Sedgwick. 2012. Pearson's correlation coefficient. *BMJ* 345 (07 2012), e4483–e4483. https://doi.org/10.1136/bmj.e4483

[19] C. Spearman. 1904. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology* 15, 1 (1904), 72–101. http://www.jstor.org/stable/1412159