Team voyTECH: User Activity Modeling with Boosting Trees

Immanuel Bayer[1] and Anastasios Zouzias[2]

[1] Palaimon GmbH
Berlin, Germany
immanuel.bayer@palaimon.io
https://palaimon.io

[2] Huawei Technologies
Zurich Research Center
Switzerland
anastasios.zouzias@huawei.com

Abstract. This paper describes our winning solution for the ECML-PKDD ChAT Discovery Challenge 2020. We show that whether or not a Twitch user has subscribed to a channel can be well predicted by modeling user activity with boosting trees. We introduce the connection between target-encodings and boosting trees in the context of high cardinality categoricals and find that modeling user activity is more powerful then direct modeling of content when encoded properly and combined with a gradient boosting optimization approach.

Keywords: competition · boosting · high cardinality categoricals · target-encodings

1 The Competition

The task of the ECML-PKDD ChAT Discovery Challenge 2020 [11] is to predict whether or not a Twitch user has subscribed to a channel (binary classification task) given the list of messages he has posted on this and other channels.

The dataset consists of more than 400 million public Twitch comments taken from English channels that are published during the month of January 2020 along with metadata. The training data consists of over 29 million and the test dataset of 90,000 channel-user combinations and their comments. In more detail, each input training sample consists of the channel-id, the user-id, and a list of time-stamped comments from this user, including the specific game played in this channel at this particular time.

1.1 Competition Challenges

The ChAT competition presents two peculiarities. The first challenge is that only half of the users in the test set have no history which requires special attention when extracting users and channels features. This challenge draws similarities with the cold start problem in recommendation systems [12]. The second challenge is related to the sampling distribution of the test set (leaderboard). More precisely, the entire spectrum of user/channel activity levels (low, normal, high) is weighted equally across all groups which is vastly different than their frequencies in the training set (see Table 1). Namely, for each out of 9 combinations of user activity levels (low, normal, high) and channel activity levels (low, normal, high), 10,000 channel-user pairs are sampled uniformly (i.e., one channel-user activity group is where the user is of low activity and the channel is of normal activity). Hence, in total 90,000 test samples are generated. In Table 1, we outline statistics of the dataset within the activity groups.

Table 1. Statistics of the training dataset per channel-user activity group. 'u_low-c_normal' corresponds to low user and normal channel activity group.

group	users	channels	pairs	subscribed	% pairs	% subscribed
u_low-c_normal	141K	40K	144K	5,696	0.0049	0.0397
u_low-c_low	10K	7,7K	10K	611	0.0003	0.0595
u_normal-c_normal	480K	67K	562K	35,021	0.0190	0.0622
u_low-c_high	2,153K	34K	2,359K	181K	0.0798	0.0768
u_normal-c_low	46K	23K	47K	3651	0.0016	0.0770
u_normal-c_high	3,508K	36K	8,740K	683K	0.2958	0.0782
u_high-c_high	1,911K	36K	16,045K	1,314K	0.5432	0.0819
u_high-c_low	77K	31K	99K	8,498	0.0033	0.0858
u_high-c_normal	663K	73K	1,531K	135K	0.0518	0.0886

1.2 Contributions

The main contributions of this paper are:

– The detailed presentation of the winning solution of the Discovery Challenge 2020 including training and evaluation setup.
– Introduction of the connection between target encodings and boosting trees in the context of high cardinality categoricals.
– Additional experiments on the competition dataset that examine the critical modeling decisions of our solution.

1.3 User Activity Modeling

Our approach is based on the assumption that modeling user activity is more important than specific content (e.g. message text). User activity is modeled as interactions between the user and key objects of the system she interacts with (e.g. channels and games).

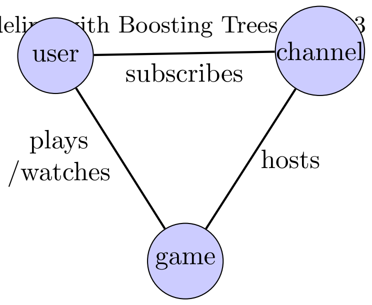


Fig. 1. Interactions

This approach naturally leads to a high dimensional categorical feature/variable representation that has been well studied in the context of recommender systems, click-through-rate predictions and similar industrial applications. It is also closely related to the concept of graph-based relational features [1].

Our experimental results (Section 3) indicate that the interactions illustrated in Figure 1 together with features describing the quantity of user activity (e.g. days active, number of frequently used channels) have strong predictive power. Introducing game-id as a high level object is especially important for the 50% test set user without history (cold-start). For a cold-start user, their most frequent game-id can effectively proxy their user-id (more details in Section 3.1). Before presenting details of our solution (Section 3.3), we first introduce the concept of target encoding to motivate our choice to combine high dimensional feature representations with boosting tree models.

2 High Cardinality Categoricals and Boosting Trees

In this section, we discuss in detail the interaction between (high cardinality) categoricals, mean target encodings and boosting trees which is at the core of our winning solution in form of the popular CatBoost library that we used to implement our models [4].

Several user and channel categorical features are present in the dataset such as which game has been played and activity levels for user, games, and channels. By computing the interaction features between user and channel representations, several categoricals with high cardinality are extracted. Due to their sparsity, such high cardinality categoricals pose several challenges in modeling and, in general, can lead to poor generalization performance. A popular class of models to handle such a semi-structured datasets containing high cardinality categoricals are Gradient Boosting Trees [8] and in particular the CatBoost library. The winning solution is based on a single CatBoost model. Model ensembles are likely to further improve our results, but were skipped due to time restrictions.

2.1 Categorical Encodings in Models

The handling of categorical features usually happens during the feature engineering phase, where the modeler has the freedom to arbitrarily transform or extract the input features before those are fed into a model. However, models exist that can handle categorical features automatically, i.e., the modeler simply specifies the features that should be handled as categoricals without any further pre-processing required. The user of such models is only able to adjust the predefined categorical encoding process with input through hyper-parameters. For example, hyper-parameters for categorical features include 'perform one-hot-encoding if cardinality of any categorical is less than a threshold', 'perform hash encoding with specified number of hashing dimensions' to name a few.

Here, we summarize a few recently proposed models that handle categorical features as part of the model definition. Two gradient boosting tree implementations, Microsoft's LightGBM [10] and Yandex's CatBoost [4], allow the user to specify which features should be handled as categoricals by the models. The *h2o.ai* implementation of Random Forests handles categoricals out of the box. Neural networks can provide an embedding layer to handle categoricals as an additional layer of a neural network, see Keras embedding layer or the so-called 'entity embeddings' [7]. LightGBM splits a categorical feature by partitioning its categories into 2 subsets. If the categorical feature has k levels, there are $2^{(k-1)} - 1$ possible partitions. However, there is an efficient $\mathcal{O}(k \log(k))$ time solution for regression trees [5]. The basic idea is to sort the categories according to the training objective at each split.

CatBoost is a gradient boosting tree implementation that applies a regularized mean target encoding on the top-level tree split as a preprocessing step. Such preprocessing could be considered sub-optimal, at least for the case of trees with large depth [3]. Although the CatBoost approach might result in sub-optimal greedy binary splits, CatBoost requires less operations per tree split and offers a very efficient and optimized implementation. The efficiency if based on the property that regularized mean target encoding values are computed only once compared to the optimal greedy approach where the mean target encodings have to be maintained or computed on every tree split.

In the following section, we provide more background on the fundamentals of CatBoost and, in particular, its connection to mean target encodings since mean target encodings are the core design principle behind CatBoost.

2.2 Target Encodings

In this section, we setup the framework of feature extraction from categoricals that is usually called *target encodings* from machine learning practitioners.

We denote m samples with n features by a $m \times n$ design matrix X with column coefficients that are either numericals (in $\mathbb{R}$) or categoricals[4]. In other words, the j-th column of X is in $\mathbb{R}^m$ or $\mathcal{C}_j^m$ for a set of elements of categoricals $\mathcal{C}_j$. Moreover, we denote by X_j the j-th column of X and by $[n]$ the set $\{1, 2, \ldots, n\}$. In addition, we denote by $\mathbf{y}$ the m-dimensional target vector. The mean value

[4] Throughout this paper, a feature is an input variable/predictor that is used for prediction. A categorical feature (or categorical) is a feature with a domain that is a fixed set without an explicit ordering. The elements of a categorical are referred as *levels*. For example, postal code, favorite color, city or country of a specific individual are examples of categoricals.

of $\mathbf{y}$, also referred to as *mean target value*, is denoted by μ. The tuple $(\mathsf{X}, \mathbf{y})$ contains all relevant information for a prediction task and we call such a tuple *design matrix pair* or for simplicity, design matrix.

We focus on the typical binary classification task, i.e., assuming an input target vector $\mathbf{y} \in \{0, 1\}^m$. The analysis can be extended directly to the regression task. Now we are ready to define target encodings.

Definition 1 (Target Encodings). *Given* $(\mathsf{X}, \mathbf{y})$ *and an integer* $j \in [n]$ *so that the* j*-th column of* X *is categorical, it follows that target encoding is a function* $f_{(\mathsf{X}_j, \mathbf{y})} : \mathcal{C}_j \to \mathbb{R}$.

From now on, we write f instead of $f_{(\mathsf{X}_j, \mathbf{y})}$ for notation convenience. It is important to note that we allow f to depend on the input dataset. Moreover, we say that f is defined (or fitted) on $(\mathsf{X}, \mathbf{y})$ to explicitly specify the input data used on the definition of f.

A very common example of target encoding is the *mean target encoding.* That is, assume that the j-th column of X is a categorical containing values/levels in $\mathcal{C}_j = \{L_1, L_2, \ldots, L_k\}$. The mean target encoding μ_j of the j-th column is defined as follows: μ_j support on $\mathcal{C}_j$ and for any $L \in \mathcal{C}_j$,

$$\mu_j(L) = \frac{1}{N} \sum_{i=1}^{m} y_i \mathbb{1}_{\mathsf{X}_{i,j} = L} \tag{1}$$

where N equals to the number of occurrences of L in the j-th column of X and $\mathbb{1}_{\text{pred}}$ is the indicator function, i.e., equals to 1 if pred is true, otherwise equals to zero. In words, mean target encodings are roughly defined as the mean target value of any level of the categorical (group).

In general, any property of the target values distribution of the group can be also extracted. For example, ML practitioners frequently use the minimum, maximum, standard deviation, kurtosis, percentiles of the target values in addition to the mean value. The main idea is to extract as much statistical information of the target distribution of the group as possible.

Regularization of Target Encodings. By definition, target encodings introduce target leakage and could lead to poor generalization performance, hence, target encoding regularization must always be used [9]. In this section, we outline several regularization methods of target encodings.

Extra caution on regularization should be given in the presence of high cardinality categoricals, i.e., categoricals with a large number of distinct levels as present in this competition. In fact, it is relatively easy to construct an example where the naive application of target encodings leads to severe overfitting. In order to exemplify this behavior, a minimal example is constructed by the authors of the ‘vtreat‘ package [15]. CatBoost provides an implementation that handles these issues automatically, however, it is important for the modeler to better understand the general approaches that we outline next.

Smoothing / Empirical Bayes / Shrinkage of Mean Target Encoding. In the presence of high-cardinality categoricals, it is quite often the case that individual categorical levels appear only in a small number of samples. In such a scenario, the estimate of the mean target encoding doesn't generalize well due to the small number of samples used to calculate the statistics. Here, smoothing or shrinkage can be applied which have a similar effect as empirical Bayesian (EB) approaches [6]. Indeed, Empirical Bayesian conditional probabilities of a categorical can be understood as mean target encodings [13].

In our notation, the EB regularized version of the mean target encoding is defined as

$$\mu_j^{\mathrm{EB}}(L) := \lambda(N)\mu_j(L) + (1 - \lambda(N))\mu \tag{2}$$

where N equals to the number of occurrences of L in the j-th column of X and $\lambda(n)$ is a monotonically increasing function on n bounded between 0 and 1. A common choice of practitioners for λ is $\lambda(n) = \frac{1}{1+\exp(-((n-l)/\sigma)}$ which is a s-shaped function with a value of 0.5 for $n = l$ and σ representing the steepness [13, Equation 4]. Thus, Equation 2 is a smoothed version of the mean target encoding.

Bootstrapping / rolling mean. Bootstrapping is another approach to regularized target encodings. A specific instance of bootstrapping and target encodings is implemented in CatBoost [4].

CatBoost uses a bootstrapping rolling mean approach to reduce overfitting while utilizing the whole training dataset for estimating the target encodings. In a nutshell, CatBoost performs a random permutation on the rows of X and for the i-th row of X (with respect to the random permutation) the mean target encoding is computed using only the rows up to the $(i-1)$-row. Namely, CatBoost averages several independent random permutations and, moreover, adds a shrinkage prior to the global mean.

To sum up, CatBoost performs categorical encoding for a level $L \in \mathcal{C}_j$ as follows. For the i-th row and a fixed permutation of the rows, CatBoost computes

$$\mu_j^{\mathrm{Cat}}(L, i) := \lambda\mu_j\left(L; (\mathsf{X}_{1:(i-1),j}, \mathbf{y}_{1:(i-1)}\right) + (1 - \lambda)\mu$$

where μ is the mean target value and λ is a smoothing hyper-parameter.

3 Winning Model and Additional Experiments

In this section, we describe the winning solution in more detail and present additional experimental results that better explain the critical choices made to achieve our result.

3.1 Feature Engineering

We have extracted a range of features from the message log representing the training data. The following features focus on different aspects of the data (time/activity, message, game).

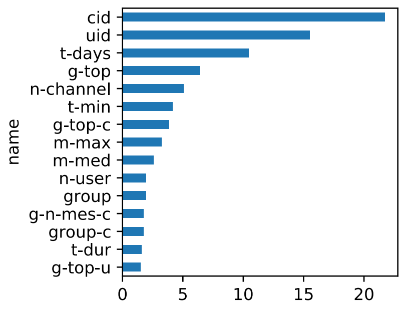
1. `t_min:`day of first message
2. `t_max:`day of last message
3. `t_days:`number of days with messages
4. `t_dur`: `t_max - t_min`
5. `t_active:t_days / t_dur`
6. `m_total:` total number of characters for all messages
7. `m_max:`max number of character per message
8. `m_med:`medium number of character per message
9. `g_n_mes:`total number of messages
10. `g_n:`number of games
11. `g_top:`game with most messages
12. `g_chat:`number of messages for game just chatting
13. `g_top_freq`: fraction of messages for `g_top`

All features have been computed as user- and channel-features. This can be done efficiently by aggregating the messages on either the user or channel id. Features such as the number of days between the first and last message (t_dur) are computed for individual channel-user combinations.

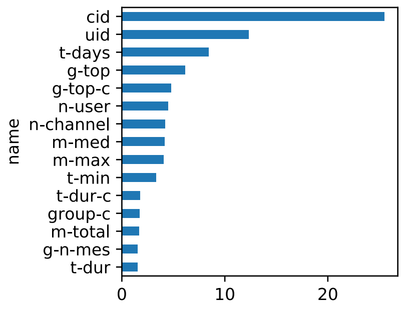
In addition the following features have been added:

1. `uid:`user id
2. `n_channel:`number of channels per user
3. `u_group`[low, normal, high]: user activity
4. `cid:`channel id
5. `n_user:`number of user per channel
6. `c_group`[low, normal, high]: channel activity

We show in Section 3.3 that only a small subset of all features is needed to achieve competitive performance.



(a) Top features (train)



(b) Top features (test)

Fig. 2. CatBoost's feature importance show clear differences between train and our constructed test data that can be attributed to the group activity shift (especially pronounced for the uid feature). The top 10 test features have been selected for a simplified model (table 3).

3.2 Best Performing Model

Model definition. The model is based on the CatBoost library version 0.23.1. The loss function is set to be *logistic loss*, also known as cross-entropy loss. Training of the model is stopped early based on the performance on our custom validation set using the autostop capabilities of CatBoost ('od_type' set to "Iter" and 'od_wait' set to 20). The best model is selected automatically by setting use_best_model=True.

The top performing submission is a single CatBoost model trained with the following hyper-parameters: 'l2_leaf_reg': 64, 'learning_rate': 0.08, 'threshold': 0.167, 'depth': 9, 'random_strength': 0.5, 'max_ctr_complexity': 2. These parameters have been manually selected on our constructed test set.

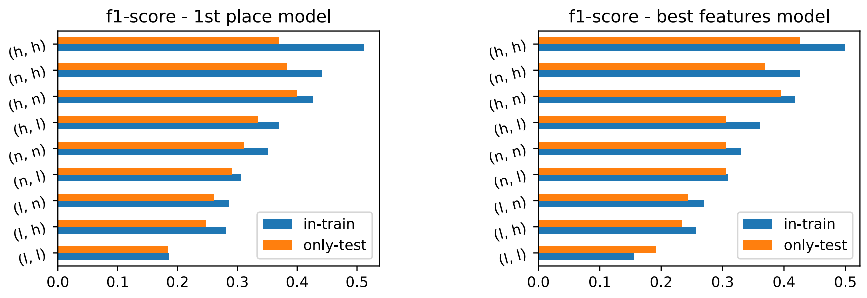
Table 2. Leaderboard results of the competition. Our submission voyTECH ranked first with a clear lead to the second best approach.

Rank	Teamname	Test F1 score
1.	voyTECH	0.3433
2.	CoolStoryBob	0.2647
3.	ItsBoshyTime	0.2593
4.	StinkyCheese	0.1422
5.	Random Baseline	0.0741

Cross validation Setup. Training and testing data come from a vastly different distribution, hence, a careful cross validation setup was crucial for model training and hyper-parameter tuning. It is given from the competition description that half of the users in the test set have no history and user-channel interactions are sampled uniformly from low, normal, and high activity levels. Therefore, our goal is to construct a validation set with similar properties. The validation set is constructed as follows: we sample in total 45,000 channel-user pairs (5,000 pairs per activity level pair group), ensuring that these pairs do not appear in the training set. These 45,000 channel-pairs are then duplicated in the validation set by modifying the user-id with an unknown identifier not present in the training set.The training dataset is sub-sampled with a 'max_per_group' parameter that restricts the number of samples per channel-user group.

3.3 Additional Experiments

We find that we can achieve strong performance (Table 3) even with a very small subset of features ('uid', 'cid', 't_days', 'g_top', 'g_top_c', 'n_user', 'n_channel', 'm_med', 'm_max', 't_min') selected by using CatBoot's feature importance (Figure 2). The sub-sampling of the training data decreases the model performance (Figure 3) and the interaction between features (max_ctr_compl ≥ 2) is important. However, the memory and run-time requirements increase dramatically when max_ctr_compl > 2. We therefore had to trade higher interactions against more training samples, which were more critical for performance.



(a) Leaderboard Model (b) Best Features Model

Fig. 3. Activity group specific F1-score. The differences between the full and reduced feature model is most pronounced for the highest (h,h) and lowest (l, l) activity groups.

Table 3. Model performance with respect to train data size, features, and hyper-parameter (random-strength=0.5, threshold=0.167, l2-leaf-reg=64 and depth=9 are the same for all rows).

f1 leaderboard	f1 mytest	train data	features	max_ctr_compl	lr
0.3433	0.3522	16m	all	2	0.15
0.3382	0.3505	8m	all	2	0.08
0.3345	0.3490	16m	top-features	2	0.15
0.3329	0.3417	full	top-features	1	0.08
0.3322	0.3421	16m	top-features	1	0.15

3.4 Conclusions & Further Work

In this work we showed that modeling user activity is more powerful than direct modeling of content when encoded properly and combined with a suitable optimization approach. We also introduced the connection between target encodings and boosting trees in the context of high cardinality categoricals and highlighted differences in the two popular boosting tree implementations CatBoost and LightGBM. We plan to conduct further experiments that compare Boosting Trees to Factorization Machines [14], a model that has been used successfully to model user activity in an earlier Discovery Challenge [2].

Acknowledgement

We would like to thank the organizers for the interesting competition and their support with the TIRA platform. From Palaimon we want to thank Phoebe Kuhn for her help with the manuscript and Alexander Pisarenko for his relentless work to provide us with a reliable and scalable infrastructure.

References

1. Bayer, I., Nagel, U., Rendle, S.: Graph based relational features for collective classification. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 447–458. Springer (2015)
2. Bayer, I., Rendle, S.: Factor models for recommending given names. ECML PKDD Discovery Challenge p. 81 (2013)
3. Breiman, L., Friedman, J., Stone, C., Olshen, R.: Classification and Regression Trees. The Wadsworth and Brooks-Cole statistics-probability series, Taylor & Francis (1984)
4. Dorogush, A.V., Ershov, V., Gulin, A.: Catboost: gradient boosting with categorical features support. In: Workshop on ML Systems at NIPS 2017 (2017)
5. Fisher, W.D.: On grouping for maximum homogeneity. Journal of the American Statistical Association **53**(284), 789–798 (1958). https://doi.org/10.1080/01621459.1958.10501479
6. Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian Data Analysis. Chapman and Hall/CRC, 2nd edn. (2004)
7. Guo, C., Berkhahn, F.: Entity embeddings of categorical variables. CoRR **abs/1604.06737** (2017)
8. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer series in statistics, Springer (2009)
9. Kaufman, S., Rosset, S., Perlich, C.: Leakage in data mining: Formulation, detection, and avoidance. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). pp. 556–563. ACM (2011)
10. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. In: Neural Information Processing Systems (NIPS), pp. 3146–3154. Curran Associates, Inc. (2017)
11. Kobs, K., Potthast, M., Wiegmann, M., Zehe, A., Stein, B., Hotho, A.: Towards predicting the subscription status of twitch.tv users — ecml-pkdd chat discovery challenge 2020. Proceedings of ECML-PKDD 2020 ChAT Discovery Challenge (2020)
12. Lam, X.N., Vu, T., Le, T.D., Duong, A.D.: Addressing cold-start problem in recommendation systems. In: Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication. p. 208211. ICUIMC 08, ACM (2008). https://doi.org/10.1145/1352793.1352837
13. Micci-Barreca, D.: A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. SIGKDD Explor. Newsl. **3**(1), 27–32 (Jul 2001). https://doi.org/10.1145/507533.507538
14. Rendle, S.: Factorization machines. In: 2010 IEEE International Conference on Data Mining. pp. 995–1000. IEEE (2010)
15. Zumel, N., Mount, J.: vtreat: a data.frame processor for predictive modeling. CoRR **abs/1706.09516** (2017)