Sapienza University of Rome

Fundamentals of Data Science - Final Project Report

Churn Prediction in Mobile Games Using Machine Learning

Group Members:

Dila Aslan (2113310)
Selin Erol (2053197)
Umut Altun (2101934)

Prof. Fabio Galasso,
Daniele Trappolini, Edoardo De Matteis, Stefano D'Arrigo

ABSTRACT

The study focused on addressing user churn in mobile games, a critical issue affecting revenue. It aimed to create a predictive model using machine learning techniques by analyzing various factors influencing churn, such as in-game activity, purchasing behavior, and demographics. Multiple models, including decision trees and logistic regression, were tested, with undersampling methods applied due to the imbalanced classification nature. The best-performing model, LGBM classifier, achieved a recall score of 0.75 and precision score of 0.25, indicating its efficacy in predicting churn. This suggests that boosting trees, particularly the LGBM classifier, could be valuable for mobile game companies seeking to pinpoint at-risk users and implement targeted retention strategies.

INTRODUCTION

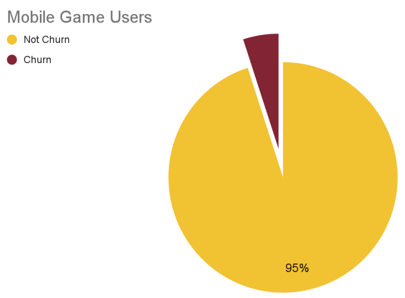
The concept of churn, the loss of customers or users, holds significance in business, impacting revenue and profitability. For mobile game developers, churn is pivotal as it directly affects game revenue and player engagement. In this study, the utilization of machine learning techniques will be applied to forecast churn within a real-world dataset of a casual mobile game. The report will explore related studies, different modeling approaches, results, evaluations, and conclusions to reveal insights into the prediction and management of churn within this gaming context.

DATASET & LABEL CHURN USERS

The dataset utilized for modeling comprises data from 6584 distinct users, encompassing session records and purchase details. Among these users, a total of 1,699,352 sessions and 236,270 purchases were recorded. The columns pertinent to these activities include user ID, date, session ID, session duration, and level completion for the session dataset; for the purchase dataset, user ID, date, and purchase price in dollars are outlined. Additionally, user attributes such as join date, country, and device operating system information are available within the dataset.

The definition of churn is tailored to suit business requirements. In the context of the mobile game under analysis in this paper, churn is characterized by a user's inactivity lasting more than 3 consecutive days, and the target variable is delineated in alignment with this criterion.

Following data labeling, an evident imbalance is observed within the dataset. Specifically, approximately 5% of users indicated churn, whereas the remaining 95% continued playing the game.



At the beginning there were those 3 data sets:

- **Users:** (contains simple details about the user)

6584 observations & 4 variables

- User_id <- Unique ID assigned to the person on the day joining the game
- Join_date <-Joining date of the user
- OS <- operating system
- Country<- User's country

- **Sessions:** (contains every session and the regarding activities of the user)

1699352 observations & 5 variables

- Session_id <- Unique session id
- Dt <- Date of the session
- User_id <- Unique ID assigned to the person on the day joining the game
- Session_duration_sec <- Duration of the related session in seconds
- Level_completed <- Level completed for session

- **Purchases:** (contains purchase amount related users in the related day)

236270 observations & 3 variables

- Used_id: User id
- Dt: Date
- Price_usd: purchase amount related users in the related day

In the initial analysis, the focus was on addressing instances of missing data (NA's). Sessions without level completion NA's were replaced with 0, and subsequently, sessions lasting less than 30 seconds were excluded from the dataset due to their lack of informativeness.

Feature Engineering:

Considering the 3-day churn horizon, the goal is to assess whether users visit within this timeframe. To accomplish this, unique rows have been created for each user and day.

For feature engineering, the features listed below were generated from the above-mentioned datasets. Here are some things to consider when creating features:

User Engagement Features:

- Session counts of a user over a specified period are indicative of their engagement level.
- Purchases made by a user over a specified period serve as a representation of their engagement with the platform.

Country-specific Features:

- Session counts in a specific country over a period reflect the platform's popularity in that particular region.
- Purchase counts in a specific country over time indicate the platform's popularity in that country.

Operating System-related Features:

- Session counts on different operating systems over a period are indicative of marketing campaigns targeted at those operating systems.
- Purchase counts on different operating systems over a period are reflective of marketing campaigns tailored to those operating systems.

Country and Operating System:

- These factors serve as key indicators in understanding user behavior and preferences.

Features derived from single-day data as well as cumulative-day data have been combined, providing a comprehensive view of user behavior.

The revenue generated by the United Kingdom and the United States collectively constitutes approximately half of the total revenue. These two countries are categorized as Tier 1, representing the highest level in the country classification. The remaining countries are then assigned to Tier 2 and Tier 3 based on a comparable methodology, where the country feature is integrated into the modeling process. This tiered classification system facilitates a nuanced understanding of revenue distribution across different countries.

After considering these factors, a total of 72 different features were obtained. The details of these features can be examined below.

No	Name	Definition	Dataset
1	session_count	number of session on that day	Session
2	level_complete_count	level completed count on that day	Session
3	session_duration	session duration in seconds on day	Session
4	days_since_join	tenure of a user	User
5	1d_ago_total_purchase_count	recent past overall purchase amount and count	Purchase
6	2d_ago_total_purchase_count		Purchase
7	3d_ago_total_purchase_count		Purchase
8	1d_ago_total_purchase_amount		Purchase
9	2d_ago_total_purchase_amount		Purchase
10	3d_ago_total_purchase_amount		Purchase
11	1d_ago_total_purchase_count_per_country	recent past purchase amount and count per country	Purchase
12	2d_ago_total_purchase_count_per_country		Purchase
13	3d_ago_total_purchase_count_per_country		Purchase
14	1d_ago_total_purchase_amount_per_country		Purchase

15	2d_ago_total_purchase_amount_per_country		Purchase
16	3d_ago_total_purchase_amount_per_country		Purchase
17	1d_ago_total_purchase_count_ratio_per_country	recent past purchase amount and count ratios among countries	Purchase
18	2d_ago_total_purchase_count_ratio_per_country		Purchase
19	3d_ago_total_purchase_count_ratio_per_country		Purchase
20	1d_ago_total_purchase_amount_ratio_per_country		Purchase
21	2d_ago_total_purchase_amount_ratio_per_country		Purchase
22	3d_ago_total_purchase_amount_ratio_per_country		Purchase
23	1d_ago_total_purchase_count_per_os	recent past purchase count per operating system	Purchase
24	2d_ago_total_purchase_count_per_os		Purchase
25	3d_ago_total_purchase_count_per_os		Purchase
26	purchase_count	purchase count on that day	Purchase
27	purchase_amount	purchase amount on that day	Purchase
28	1d_ago_purchase_amount	recent past purchase amount and count per user	Purchase
29	2d_ago_purchase_amount		Purchase
30	3d_ago_purchase_amount		Purchase
31	1d_ago_purchase_count		Purchase
32	2d_ago_purchase_count		Purchase
33	3d_ago_purchase_count		Purchase
34	l2d_ago_purchase_amount	recent past cumulative purchase amount and count per user	Purchase
35	l3d_ago_purchase_amount		Purchase
36	l5d_ago_purchase_amount		Purchase
37	l7d_ago_purchase_amount		Purchase
38	l9d_ago_purchase_amount		Purchase
39	l2d_ago_purchase_count		Purchase
40	l3d_ago_purchase_count		Purchase

41	l5d_ago_purchase_count		Purchase
42	l7d_ago_purchase_count		Purchase
43	l9d_ago_purchase_count		Purchase
44	1d_ago_session_counts		Session
45	2d_ago_session_counts	recent past session count per user	Session
46	3d_ago_session_counts		Session
47	l2d_ago_session_counts		Session
48	l3d_ago_session_counts		Session
49	l5d_ago_session_counts	recent past cumulative session count per user	Session
50	l7d_ago_session_counts		Session
51	l9d_ago_session_counts		Session
52	1d_ago_session_durations		Session
53	2d_ago_session_durations	recent past session count per user	Session
54	3d_ago_session_durations		Session
55	l2d_ago_session_durations		Session
56	l3d_ago_session_durations		Session
57	l5d_ago_session_durations	recent past cumulative session duration per user	Session
58	l7d_ago_session_durations		Session
59	l9d_ago_session_durations		Session
60	1d_ago_level_complete_count		Session
61	2d_ago_level_complete_count		Session
62	3d_ago_level_complete_count		Session
63	l2d_ago_level_complete_count	recent past cumulative level complete count per user	Session
64	l3d_ago_level_complete_count		Session
65	l5d_ago_level_complete_count		Session
66	l7d_ago_level_complete_count		Session
67	l9d_ago_level_complete_count		Session
68	Android	OS (one-hot endoded)	User
69	iOS		User
70	tier1		User
71	tier2	County Tiers (one-hot encoded)	User
72	tier3		User

IMPLEMENTATION DETAILS

All implementations were carried out in Python. For each part of data cleaning and feature engineering, step-by-step workflows were constructed. The Classifier class, which encapsulates all related methods and attributes, was devised for modeling. This class was created for several reasons, one being to apply different threshold levels, save each result to a pickle file, and ultimately compare all parameters across all thresholds. This is due to GridSearchCV's default behavior of evaluating each parameter set with the threshold for the metric set at 0.5.

From the class, the method names and explanations are as follows:

- Preprocess model data, preparing data for modeling
- Create folder, for model saving
- Get last parameter set used, to continue from the last parameter set in Grid-search
- Get used params, to get parameter sets used until that point in Grid-search
- MakeGrid, creates single parameter sets in grid-search format from a full set of parameters
- Undersample data, undersampled the majority class with a fraction
- To labels, label a single probability as 0 or 1 with a threshold
- Choose threshold, from start to end, it evaluates a model for each threshold
- Choose base learner, for a set of models, it evaluates them with default parameters
- Classify, classifies, reports and saves all results for a set of models, and given undersampling ratios
- Get best model, evaluates all models saved into pickle format and saves their results for comparison

Evaluation Details, Results and Analysis

As a business need, it is better to have a low precision and high recall compared to the opposite. Being able to catch most of those who are likely to churn and at the same time giving false signals for some is more important than catching only a fraction of them without any false signals. Thus recall (how good the model is at catching those users that are likely to churn) is set to a certain level, here a minimum of 75%.

Because after this step successful models with undersampling ratios will be fed into parameter search (grid search), time taken is also important to look at. Here, LGBM classifier ranks first due to its variance reducing algorithm which makes it faster to converge compared to other models. Random forest and LGBM classifiers with 10% undersampling ratio are taken as two enough-good examples that will be modeled with more parameters specific to each model in grid search.

	threshold	F1	recall	precision	model_name	undersample_frac	time_seconds
34	0.39	0.355516	0.752518	0.232734	RandomForestClassifier	0.1	113.41
19	0.24	0.350671	0.750252	0.228808	RandomForestClassifier	0.2	192.05
33	0.38	0.348178	0.758812	0.225920	RandomForestClassifier	0.1	113.41
18	0.23	0.343022	0.762085	0.221320	RandomForestClassifier	0.2	192.05
32	0.37	0.343008	0.767623	0.220846	RandomForestClassifier	0.1	113.41
9	0.14	0.339874	0.752266	0.219528	RandomForestClassifier	0.4	349.71
31	0.36	0.335372	0.752266	0.215787	ExtraTreesClassifier	0.1	50.08
31	0.36	0.334625	0.750000	0.215355	LGBMClassifier	0.1	50.21
31	0.36	0.336589	0.776435	0.214868	RandomForestClassifier	0.1	113.41
31	0.36	0.333426	0.751511	0.214240	XGBClassifier	0.1	50.58

Random Forest Parameter Grid Search and Evaluation Results

```
classifier.parameters = {
        'n_estimators': [150, 250, 350],
        'criterion': ['entropy', 'gini'],
        'max_depth': [None],
        'min_samples_split': [2, 3, 5],
        'warm_start': [False, True],
        'class_weight': ['balanced'],
        'max_features': [None, 'sqrt', 'log2'],
        'random_state': [42],
        'n_jobs': [-1]
    }
classifier.classify(classifier_name="RandomForestClassifier", undersample_frac=0.1)
```

	threshold	F1	recall	precision	data	classifier	index	uuid
35	0.40	0.357472	0.751259	0.234536	test	RandomForestClassifier	12	33141603888644ac8d0e87c98da4b74b
35	0.40	0.357472	0.751259	0.234536	test	RandomForestClassifier	9	4953428cfd3341aaaa397cfbfbb48c34
37	0.42	0.356617	0.750000	0.233922	test	RandomForestClassifier	53	08c05b9d70f6449b80d48d249404009c
37	0.42	0.356617	0.750000	0.233922	test	RandomForestClassifier	50	b01dc84ba55c49e79cd531548149686e
37	0.42	0.355630	0.750252	0.233049	test	RandomForestClassifier	14	315309d91cf346098fbffbc8fb61a627
37	0.42	0.355630	0.750252	0.233049	test	RandomForestClassifier	17	e9d82d1463e0440f9ea3d8a99ea4a3ed
33	0.38	0.353435	0.752518	0.230953	test	RandomForestClassifier	2	a55cab9da72544a4abcee29a89b06177
33	0.38	0.353435	0.752518	0.230953	test	RandomForestClassifier	5	b69000863cc14821b6c49a217906f08d
32	0.37	0.353427	0.753525	0.230852	test	RandomForestClassifier	57	0c130bd840014b52bfa11701857d4286

LGBM Classifier Parameter Grid Search

```
classifier.parameters = {
        'boosting_type': ["gbdt", "dart"],
        'num_leaves': [7, 15, 31, 63],
        'learning_rate': [0.3, 0.1, 0.05],
        'max_depth': [-1, 3, 4, 5, 6],
        'n_estimators': [100, 150, 250],
        'reg_lambda': [0, 0.05, 0.1],
        'n_jobs': [-1],
        'objective': ["binary"]
    }
classifier.classify(classifier_name="LGBMClassifier", undersample_frac=0.1)
```

	threshold	F1	recall	precision	data	classifier	index	uuid
54	0.59	0.374505	0.750000	0.249560	test	LGBMClassifier	680	cb4b60942eb4409c81d117f24e4a4a1c
54	0.59	0.371131	0.750252	0.246546	test	LGBMClassifier	183	fc5f6483d7ea4d7f8a7fbb67ca43503b
54	0.59	0.371053	0.750000	0.246504	test	LGBMClassifier	233	b4fecde7103e4d88ac3ff88231e1af88
53	0.58	0.370118	0.752769	0.245384	test	LGBMClassifier	836	1171c38efd31447ca2a4ce9ad89cf5b2
53	0.58	0.370118	0.752769	0.245384	test	LGBMClassifier	971	4f1abb41189149f1b93f386615c43626
53	0.58	0.369770	0.752014	0.245158	test	LGBMClassifier	864	8011f24fc1914346a22e727022a2c37f
53	0.58	0.369564	0.752014	0.244977	test	LGBMClassifier	683	9a18877204a4420eb7358fba9426f25e
53	0.58	0.369282	0.750000	0.244943	test	LGBMClassifier	717	ebc656a162404fc78285ee1fe7d0cb82
53	0.58	0.369415	0.753525	0.244686	test	LGBMClassifier	707	3a03d472671a4e7d9db340c1f8f240d9

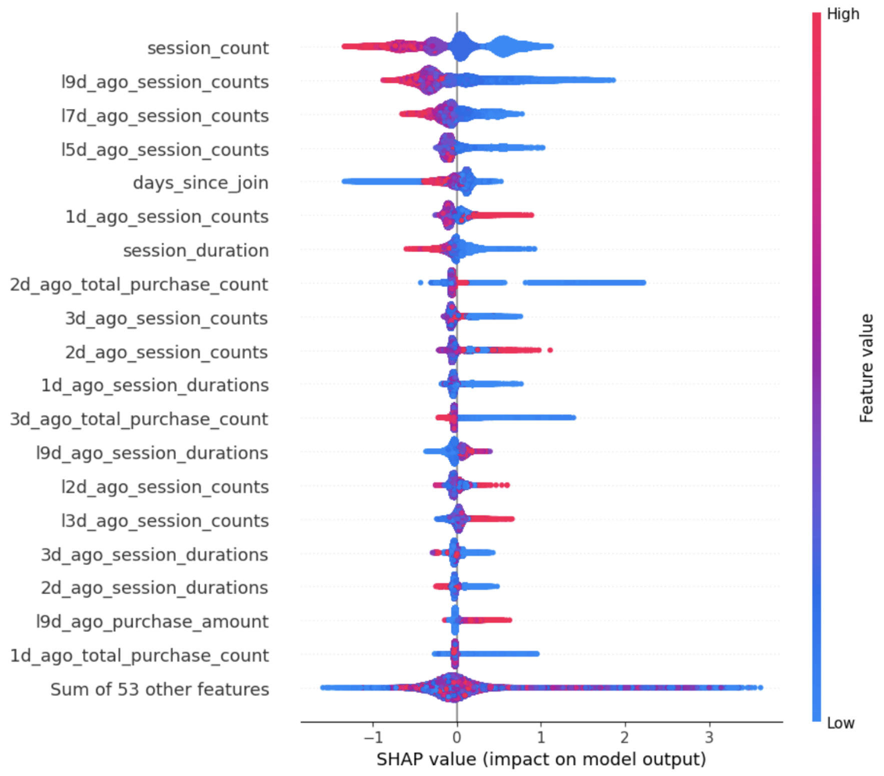
Best Model Configuration

Compared to random forest, LGBM performed better according to best F1 Score (with at least 0.75 Recall score). It can be seen that the best LGBM model parameters at below.

```
▾ GridSearchCV
GridSearchCV(cv=5, estimator=LGBMClassifier(n_jobs=-1),
             param_grid={'boosting_type': ['dart'], 'learning_rate': [0.3],
                         'max_depth': [-1], 'n_estimators': [150],
                         'n_jobs': [-1], 'num_leaves': [15],
                         'objective': ['binary'], 'reg_lambda': [0.05]},
             return_train_score=True, scoring='f1')
▾ estimator: LGBMClassifier
LGBMClassifier(n_jobs=-1)
▸ LGBMClassifier
```

Feature Importance

With this best LGBM model, SHAP feature importance values have been calculated.



- **Session Count**: More sessions correlate with lower churn risk. Encourage user engagement through regular prompts or rewards to increase session frequency.
- **Session Duration**: Optimal session lengths are associated with reduced churn. Design content to maintain user interest without becoming overly challenging or too simplistic.
- **Purchase Count**: A higher number of purchases suggests lower churn. Implement incentives for users to make in-app purchases, like discounts or limited-time offers.
- **Purchase Amount**: Higher spending is linked to increased churn, possibly due to higher user expectations. Set fair prices and ensure the perceived value justifies the cost to prevent churn.

Conclusion

In this project, this churn model is built to predict user churn using machine learning methods with a real-world data set from a mobile game. Tree-based methods outperformed a linear regression model and from the tree-based methods a leaf-wise boosting tree, Light Gradient Boosting Machine classifier, ranked first. Several undersampling and parameter search methods were performed and, based on the metrics specified such as f1 score, precision and recall, models were evaluated. Lastly, based on the feature importances, some insights derived for churn behavior of the users.

References

1. https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html
2. On Analyzing Churn Prediction in Mobile Games https://arxiv.org/pdf/2104.05554.pdf
3. Predicting Player Churn of a Free-to-Play Mobile Video Game Using Supervised Machine Learning https://www.mdpi.com/2076-3417/12/6/2795
4. Churn prediction of mobile and online casual games using play log data https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0180735
5. Predicting Customer Churn Rate in the iGaming Industry using Supervised Machine Learning https://www.diva-portal.org/smash/get/diva2:1212531/FULLTEXT02.pdf
6. Churn prediction in digital game-based learning using data mining techniques: Logistic regression, decision tree, and random forest https://www.sciencedirect.com/science/article/abs/pii/S1568494622000436
7. Churn analysis using deep convolutional neural networks and autoencoders https://arxiv.org/pdf/1604.05377.pdf