User Purchase Prediction

Olgu Altıntaş

1 Task Description

- A predictive model was developed to analyze user behavior based on the first seven days of activity. The main goal was to predict whether a user would make a purchase within 30 days of installation. The approach included data preparation, feature engineering, model selection, and evaluation.

2 Predictive Model

The initial state of the dataset was examined.

#	Column	Non-Null Count	Dtype
0	user_id	342519	object
1	platform	342519	object
2	network	342519	object
3	country	342519	object
4	age	342519	int64
5	time_spend	342519	int64
6	coin_spend	342519	int64
7	coin_earn	342519	int64
8	level_success	342519	int64
9	level_fail	342519	int64
10	level_start	342519	int64
11	booster_spend	342519	int64
12	booster_earn	342519	int64
13	coin_amount	342519	int64
14	event_participate	342519	int64
15	shop_open	342519	float64
16	d30_revenue	342519	float64

Table 1: **Data Columns Overview**

2.1 *Data Pre-Processing*

- The user_id column was removed from the dataset as it was not relevant to the task.
- Columns with an object data type were analyzed.
- Since the platform column contained only two unique values, one-hot encoding was applied.
- The network and country columns, containing 14 and 20 unique values, respectively, were encoded using label encoding.

2.2 *Feature Engineering*

- Four new columns were derived to assist with the task:

- 1-) Booster Usage Ratio → Reveals how often players rely on boosters to progress.
  2-) Average Session Time → Shows how long players stay engaged per session.
  3-) Coin Spend Ratio → Indicates spending behavior by
  4-) Level Completion Rate → Measures how often players successfully complete levels, indicating skill or difficulty.

- Based on domain expertise, it was determined that all existing rows in the dataset were relevant to the task. Statistical analyses such as correlation, which indicate relationships rather than causation, were not applied due to concerns that they could be misleading.

2.3 *Target Column*

The initial state of the target column was examined.

Statistic	**Value**
Count	342519
Mean	7.91
Std	91.52
Min	0.00
25%	0.00
50%	0.00
75%	0.00
Max	11071.19

Table 2: **Statistical Summary of the Target Column**

- This is a classification task; therefore, a binary target column is required. A thresholding technique has been applied.

- After thresholding:

Class	**Count**	**Ratio (%)**
0	315683	92.16%
1	26836	7.84%

Table 3: **Distribution of the Binary Target Column**

- Distribution after thresholding:

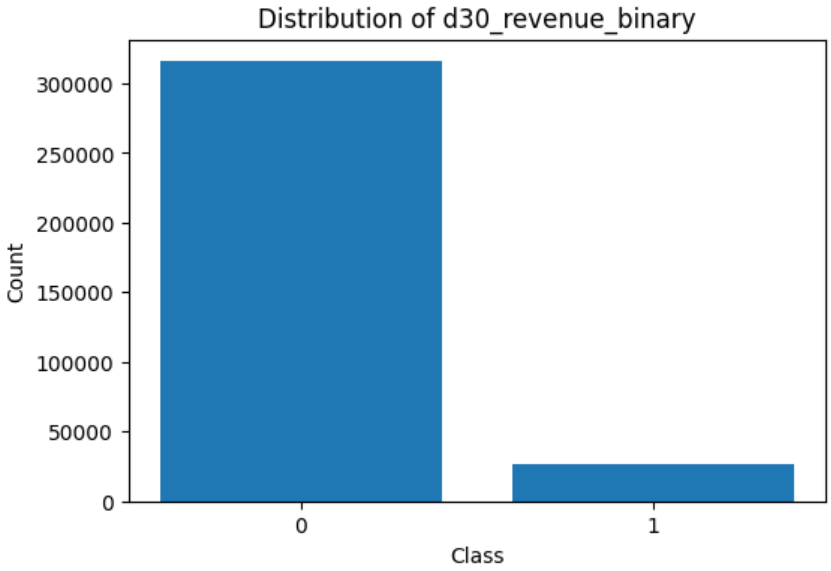

Figure 1: Distribution of d30 Revenue Binary

2.4 *Applying Models & Comparison*

- The dataset was split into training and test sets using a stratified approach to maintain the original class distribution.

Class	Original Count	Resampled Count (SMOTE)
0	252,546	252,546
1	21,469	252,546

Table 4: **Training Label Distribution Before and After SMOTE**

- The model's performance was evaluated using standard classification metrics, including accuracy, precision, recall, F1-score, ROC AUC, and PR AUC, which served as baseline metrics.

- A baseline model was trained using a Random Forest Classifier on the original training set without any resampling or class weighting.
- SMOTE (Synthetic Minority Over-sampling Technique) was applied to the training set only to address class imbalance, and the same Random Forest model was retrained.

Class	Precision	Recall	F1-score	Support
0	0.93	1.00	0.96	63,137
1	0.75	0.17	0.28	5,367
Accuracy		0.93		68,504
Macro Avg	0.84	0.58	0.62	68,504
Weighted Avg	0.92	0.93	0.91	68,504

Table 5: **Baseline RandomForest (No SMOTE)**

Metric	Value
ROC AUC	0.8412
PR AUC	0.4235

Table 6: **AUC Scores (Baseline RandomForest)**

Class	Precision	Recall	F1-score	Support
0	0.94	0.96	0.95	63,137
1	0.39	0.29	0.34	5,367
Accuracy		0.91		68,504
Macro Avg	0.67	0.63	0.64	68,504
Weighted Avg	0.90	0.91	0.90	68,504

Table 7: **RandomForest with SMOTE**

Metric	Value
ROC AUC	0.8271
PR AUC	0.3490

Table 8: **AUC Scores (RandomForest with SMOTE)**

- XGBoost was implemented without SMOTE
- XGBoost was then trained with SMOTE

Class	Precision	Recall	F1-score	Support
0	0.94	0.99	0.96	63,137
1	0.71	0.20	0.31	5,367
Accuracy		0.93		68,504
Macro Avg	0.82	0.60	0.64	68,504
Weighted Avg	0.92	0.93	0.91	68,504

Table 9: **XGBoost (No SMOTE)**

Metric	Value
ROC AUC	0.8647
PR AUC	0.4569

Table 10: **AUC Scores (XGBoost No SMOTE)**

Class	Precision	Recall	F1-score	Support
0	0.95	0.95	0.95	63,137
1	0.37	0.36	0.36	5,367
Accuracy		0.90		68,504
Macro Avg	0.66	0.65	0.66	68,504
Weighted Avg	0.90	0.90	0.90	68,504

Table 11: **XGBoost (With SMOTE)**

Metric	Value
ROC AUC	0.8283
PR AUC	0.3710

Table 12: **AUC Scores (XGBoost With SMOTE)**

- The MLP model was then trained with and without SMOTE, and the results were analyzed.
- For the MLP model trained with SMOTE, the recall for class 1 was very low, indicating that the minority class was not detected effectively.
- ADASYN and BorderlineSMOTE were then tested, and the results were slightly better, but the recall for class 1 remained low.

Class	Precision	Recall	F1-score	Support
0	0.93	1.00	0.96	63,137
1	0.85	0.14	0.24	5,367
Weighted Avg	0.93	0.93	0.91	68,504

Table 13: **MLP (No SMOTE)**

Metric	Value
ROC AUC	0.8525
PR AUC	0.4394

Table 14: **AUC Scores**

Class	Precision	Recall	F1-score	Support
0	0.95	0.94	0.95	63,137
1	0.41	0.40	0.41	5,367
Weighted Avg	0.91	0.90	0.91	68,504

Table 15: **MLP (SMOTE)**

Metric	Value
ROC AUC	0.8493
PR AUC	0.4272

Table 16: **AUC Scores**

Class	Precision	Recall	F1-score	Support
0	0.98	0.69	0.81	63,137
1	0.19	0.83	0.30	5,367
Weighted Avg	0.92	0.70	0.77	68,504

Table 17: **MLP (ADASYN)**

Metric	Value
ROC AUC	0.8471
PR AUC	0.4286

Table 18: **AUC Scores)**

Class	Precision	Recall	F1-score	Support
0	0.98	0.69	0.81	63,137
1	0.19	0.83	0.30	5,367
Weighted Avg	0.92	0.70	0.77	68,504

Table 19: **MLP (BorderlineSMOTE)**

Metric	Value
ROC AUC	0.8517
PR AUC	0.4363

Table 20: **AUC Scores**

- The models were compared using standard classification metrics, including precision, recall, F1-score, and AUC values.
- The ROC curve and Precision-Recall (PR) curve were visualized.

#	Model	Precision	Recall	F1	ROC AUC	PR AUC
0	RandomForest (No SMOTE)	0.746548	0.171232	0.278569	0.841190	0.423528
1	RandomForest (SMOTE)	0.393516	0.294019	0.336568	0.827081	0.349015
2	XGBoost (No SMOTE)	0.712938	0.197131	0.308860	0.864685	0.456902
3	XGBoost (SMOTE)	0.369686	0.358114	0.363808	0.822889	0.371016
4	MLP (No SMOTE)	0.822703	0.141792	0.241894	0.851363	0.438578
5	MLP (SMOTE)	0.423904	0.381964	0.401843	0.849883	0.431632

Table 21: **Model Performance Comparison**

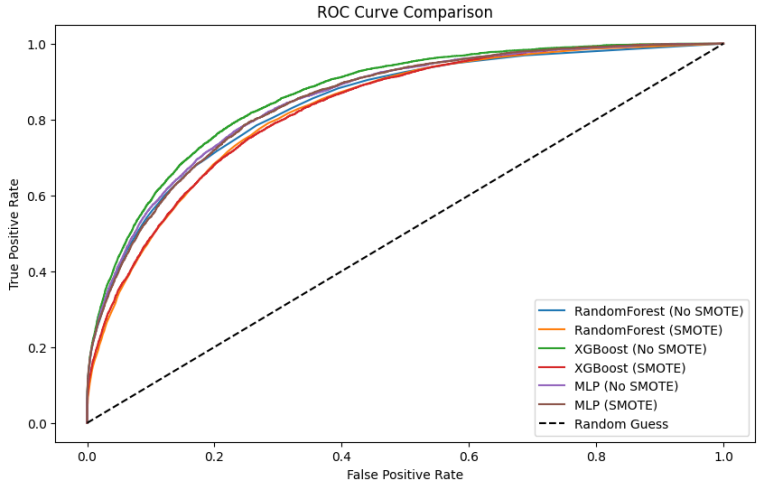

Figure 2: **ROC Curve Comparison**

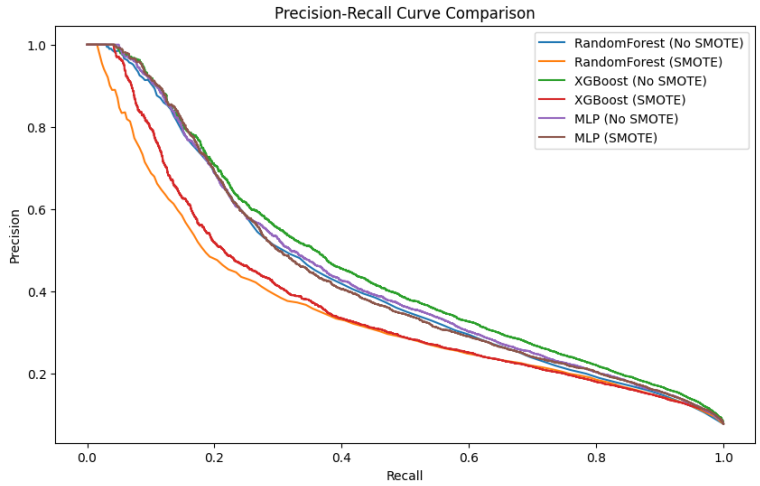

Figure 3: **Precision-Recall Curve Comparison**

2.5 *Commentation*

- XGBoost without SMOTE delivered the strongest performance, achieving a ROC AUC of 0.8647 and PR AUC of 0.4569. This suggests that the model is highly effective at ranking users based on their likelihood to make a purchase.
- Oversampling methods (e.g., SMOTE, ADASYN) occasionally observed to increase recall for the minority class but introduced overfitting concerns and did not always improve AUC scores.
- XGBoost without SMOTE was considered the best-performing approach, particularly in balancing overall predictive accuracy with stable AUC metrics.

✓ High classification metrics were recorded for the best model, suggesting that user spending is reasonably predictable from early gameplay data.

✓ SMOTE and related methods helped explore minority-class recall improvements, showcasing thorough attempts to handle class imbalance.

✗ Recall for purchasers sometimes remained low, suggesting that existing features did not fully capture key user characteristics or spending triggers.

✗ Some models showed high precision but struggled with recall when oversampling was applied incorrectly, highlighting the complexity of predicting user purchase behavior.

- **Deploying XGBoost as the primary predictive model was recommended.**