Predicting Potential Customers using Boosting Methods with Bag-of-values Encoding

沈信憲、李文先、周文揚、巖昀叡、任翊瑄

Chapter 1: Introduction

Precision marketing has become a critical focus in recent years as companies aim to allocate resources effectively to target customers. To address challenges in this domain, we referenced the study *Generalize for Future: Slow and Fast Trajectory Learning for CTR Prediction*, which explores methods for improving click-through rate (CTR) prediction. For this study, we utilized the customer dataset from Huawei[1], which is no longer available now.

While the original paper uses embedding as the encoding method for categorical variables, this approach has notable limitations, including high computational costs and reduced interpretability of the relationship between categorical features and the target variable. To overcome these challenges, we propose an alternative approach by leveraging XGBoost and CatBoost combined with bag-of-values encoding. Bag-of-values encoding offers two primary advantages: it enhances training efficiency by reducing computational complexity and improves model interpretability by explicitly capturing the statistical relationships between features and the target. This methodological adjustment demonstrates the potential for achieving both efficiency and transparency. In addition, since we focus on precision marketing, we aim to predict potential customers instead of predicting CTR mentioned in the paper.

Chapter 2: EDA

Data Overview

Data Sources

- Advertisement Dataset: train_data_ads.csv containing features such as user ID, task ID, and labels.
- User Browsing Dataset: train_data_feeds.csv.

Data Scale

Advertisement data includes various features:

- Task ID Distribution:

[1] https://developer.huawei.com/consumer

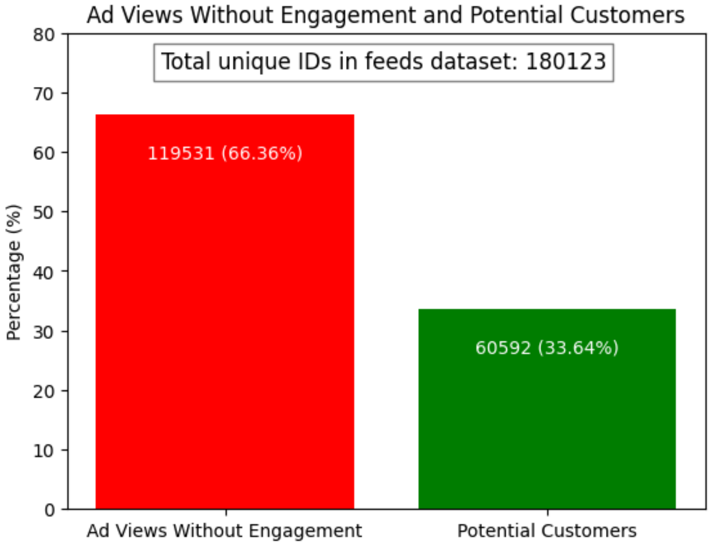
- Average samples per task: 0.015591.
- Standard deviation: 0.014868.
- Minimum value: 0.000912.
- Maximum value: 0.109355.

User Segmentation Analysis

User Classification

Based on data analysis, users were classified into:

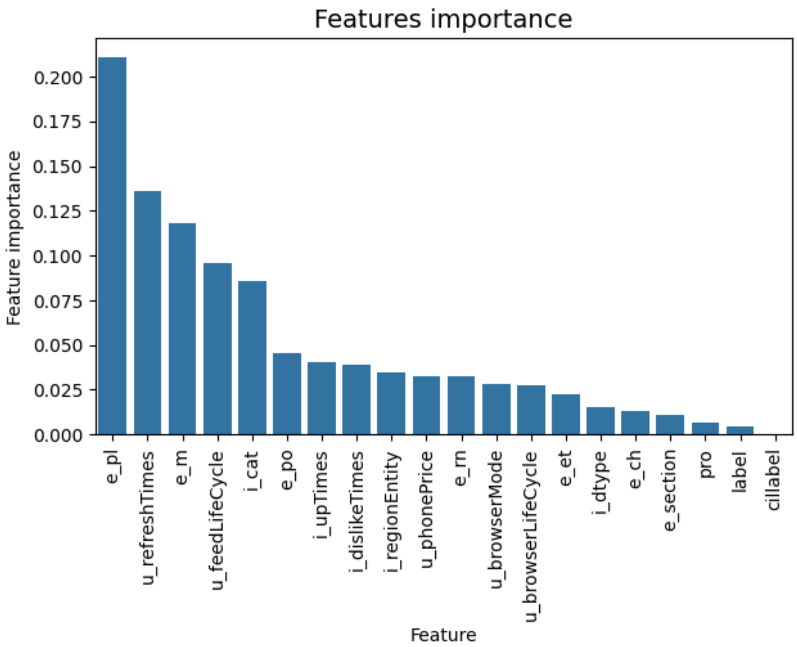
- Non-Engaged Users: Users who view ads but show no interaction.
- Potential Customers: Users likely to engage.



Feature Importance Analysis

Using a Random Forest classifier, the most influential features were identified:

1. e_pl (most important).
2. u_refreshTimes.
3. e_section.
4. u_browserLifeCycle.
5. u_feedLifeCycle.



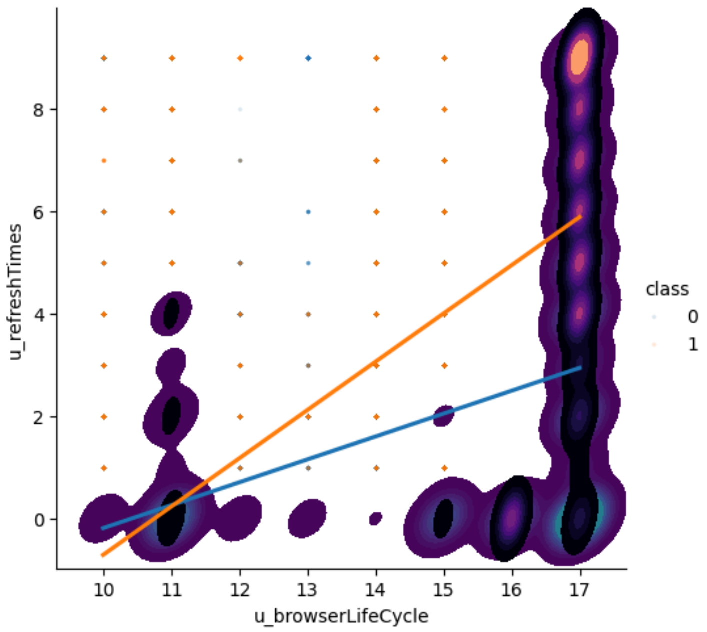
Behavioral Feature Analysis

Time-Dimension Analysis

• KS Test and T-Test were applied to analyze behavioral differences between user groups.
• Findings:
- Significant differences in browsing time (e_et) between groups.
- Time density distribution revealed more concentrated active periods for potential customers.

Interaction Behavior Analysis

• Key Discoveries:
1. Positive correlation between u_browserLifeCycle and u_refreshTimes.



2. Higher page interaction for potential customer groups.
3. Significant relationship between content type (i_dtype) and page position (e_section).

Statistical Significance Analysis

Key Statistical Test Results

- Page Position (e_pl):
  - T-Test showed significant differences between user groups.

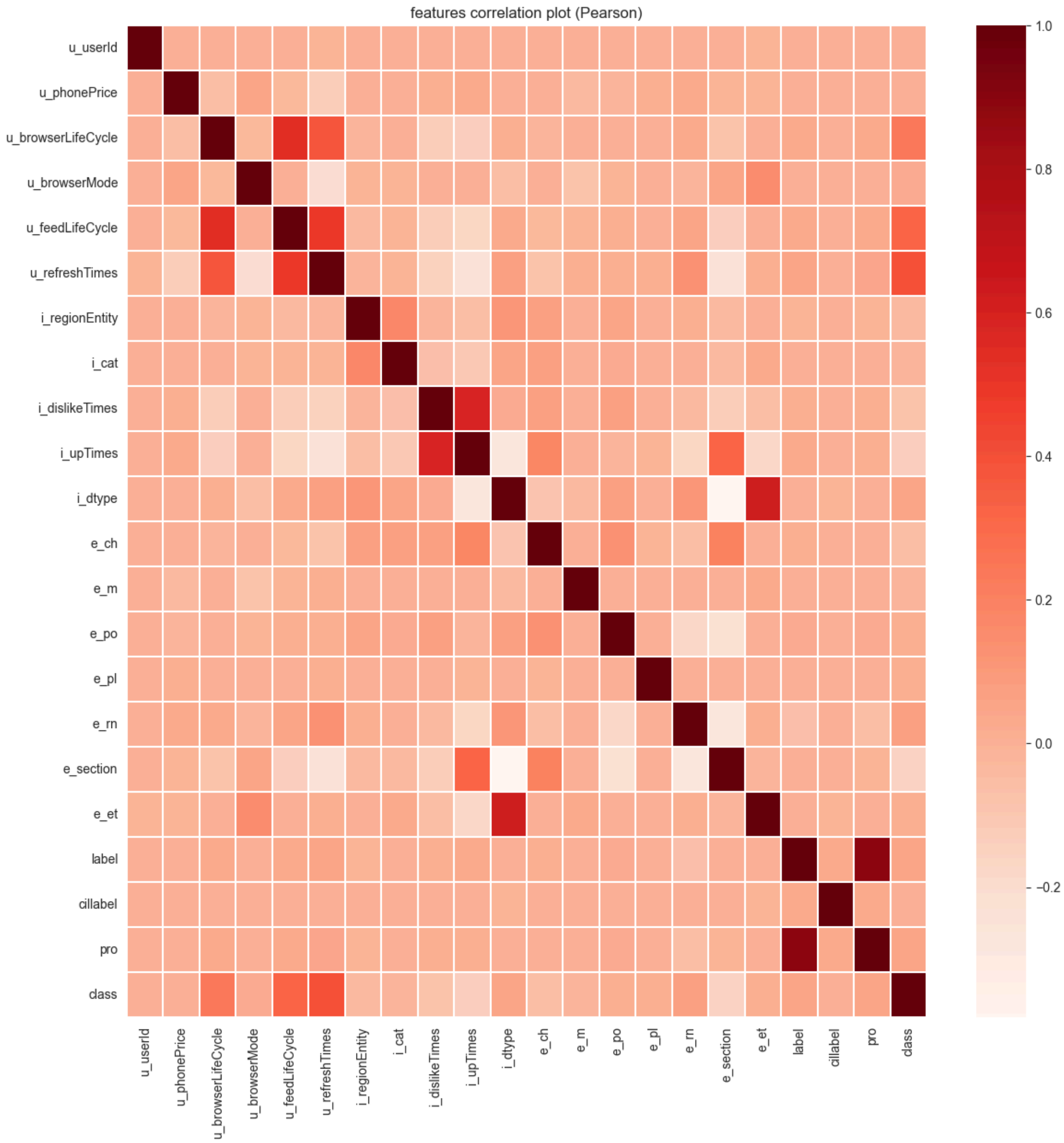
```
KstestResult(statistic=0.014521468302746754,
pvalue=4.08897297790306e-108, statistic_location=1358,
statistic_sign=1)
```

- Refresh Times (u_refreshTimes):
  - Statistical significance confirmed via KS Test.

```
KstestResult(statistic=0.4219041262761701, pvalue=0.0,
statistic_location=3, statistic_sign=1)
```

Correlation Analysis

- Heatmap analysis showed:
  - Strong correlations between user behavior features.
  - Time-related features strongly associated with interaction behaviors.



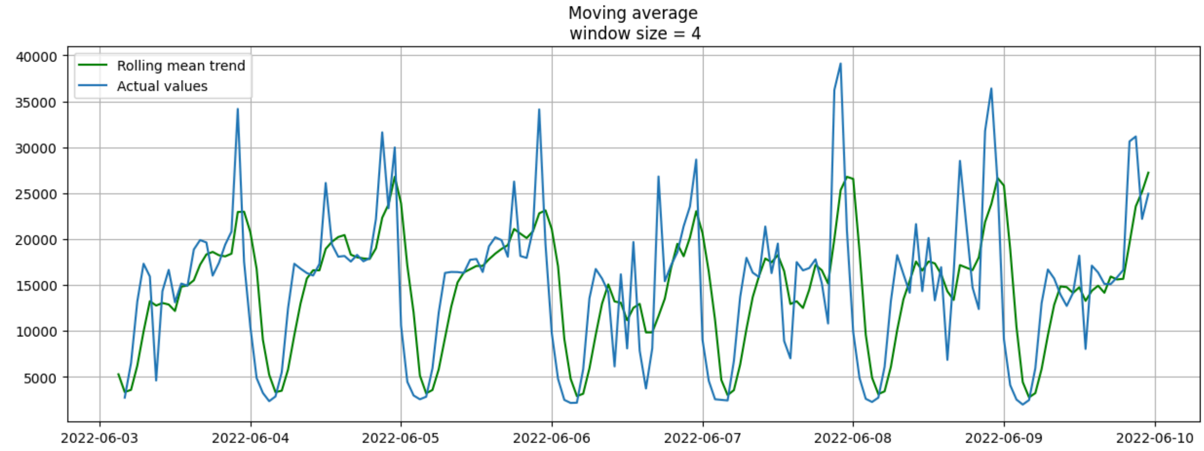
Key Metrics Analysis

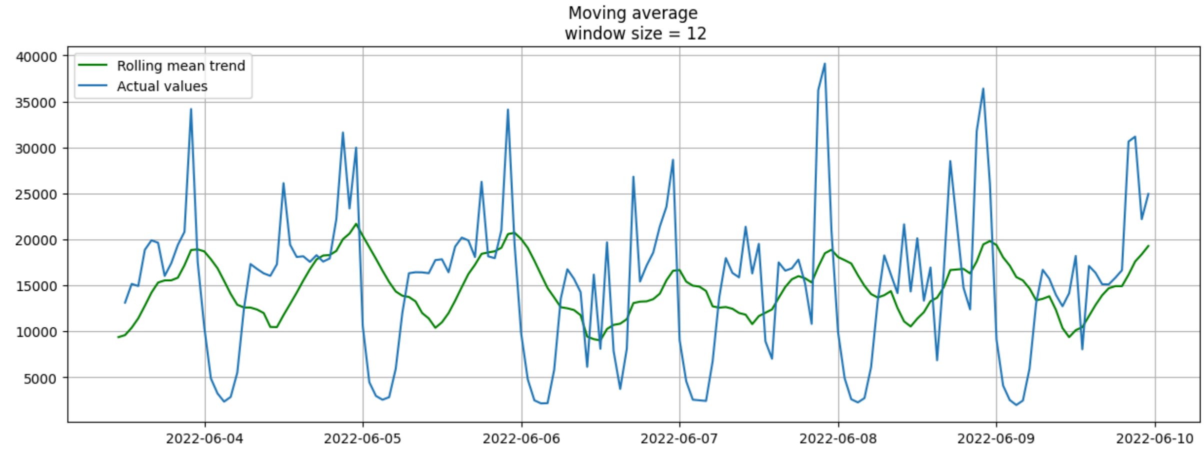
Click-Through Rate (CTR) Analysis

- Overall CTR: ~1.55%.
- Last-touch CTR: 12.2%, significantly higher than the average.
- Time-decayed weighted click analysis indicated temporal dependencies in user behavior.

Content Delivery Analysis

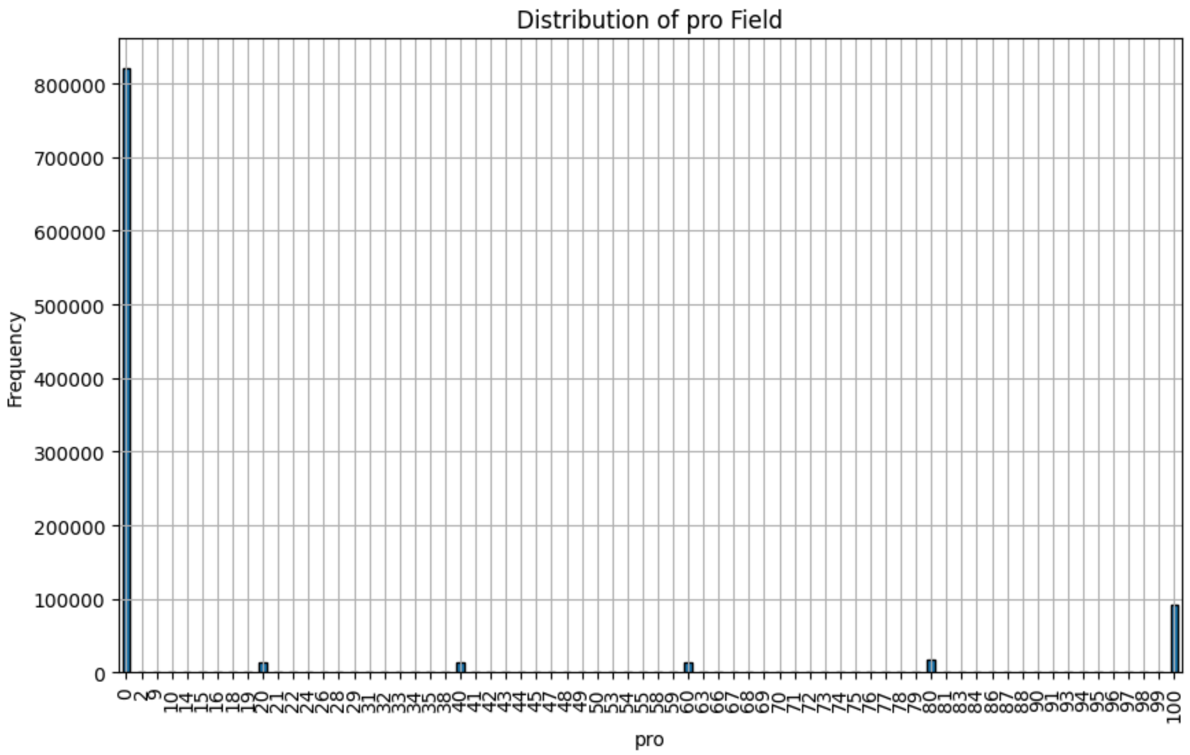
- Users received 0-130 content pieces, with a clear 24-hour periodicity.
- Moving average analysis showed stable delivery trends.

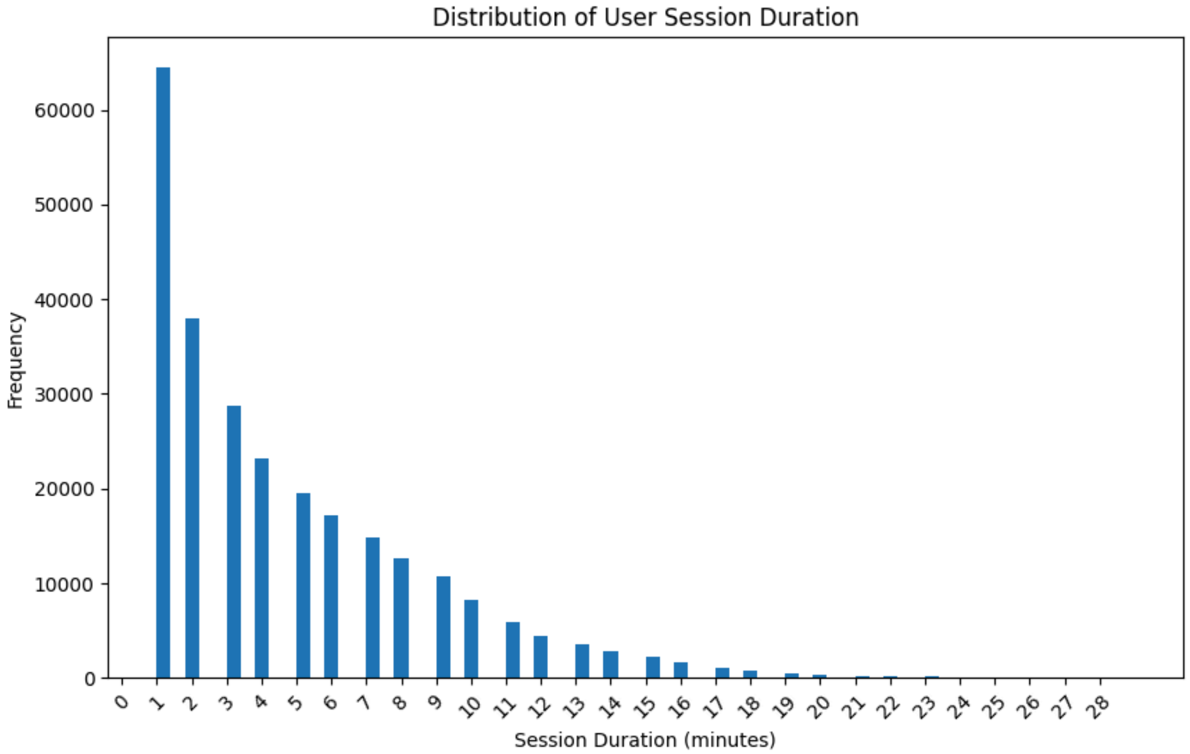


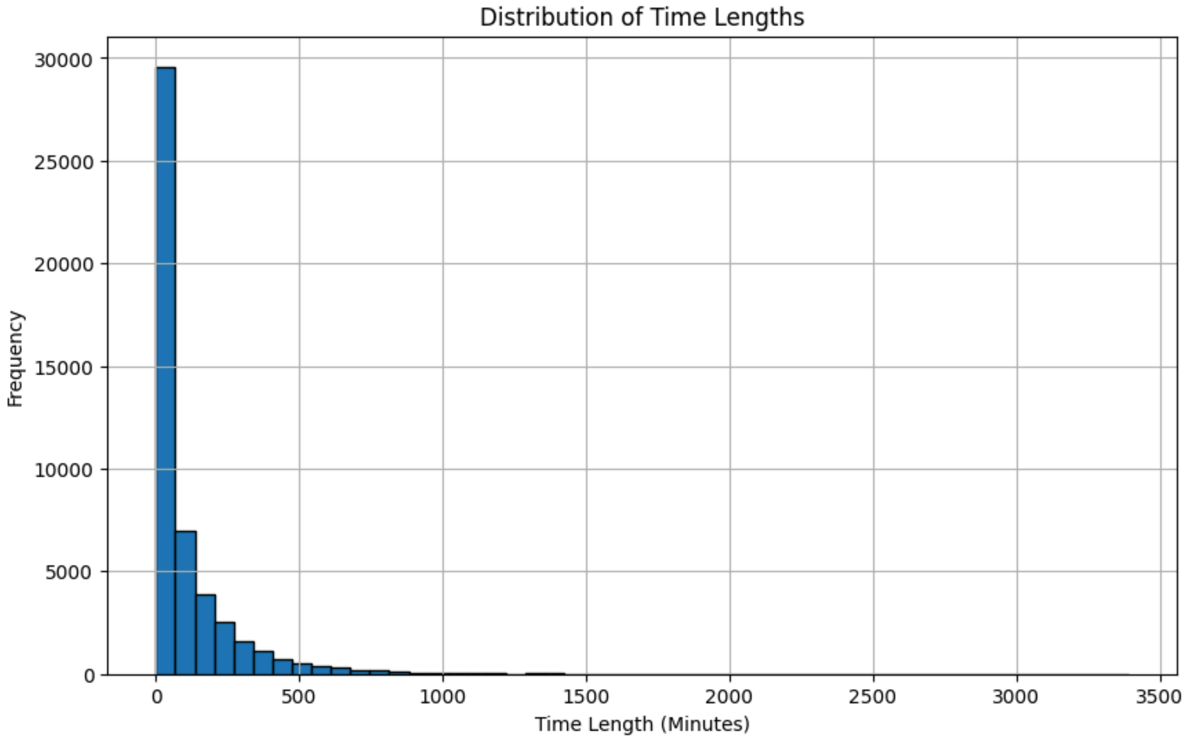


User Behavior Timing Analysis

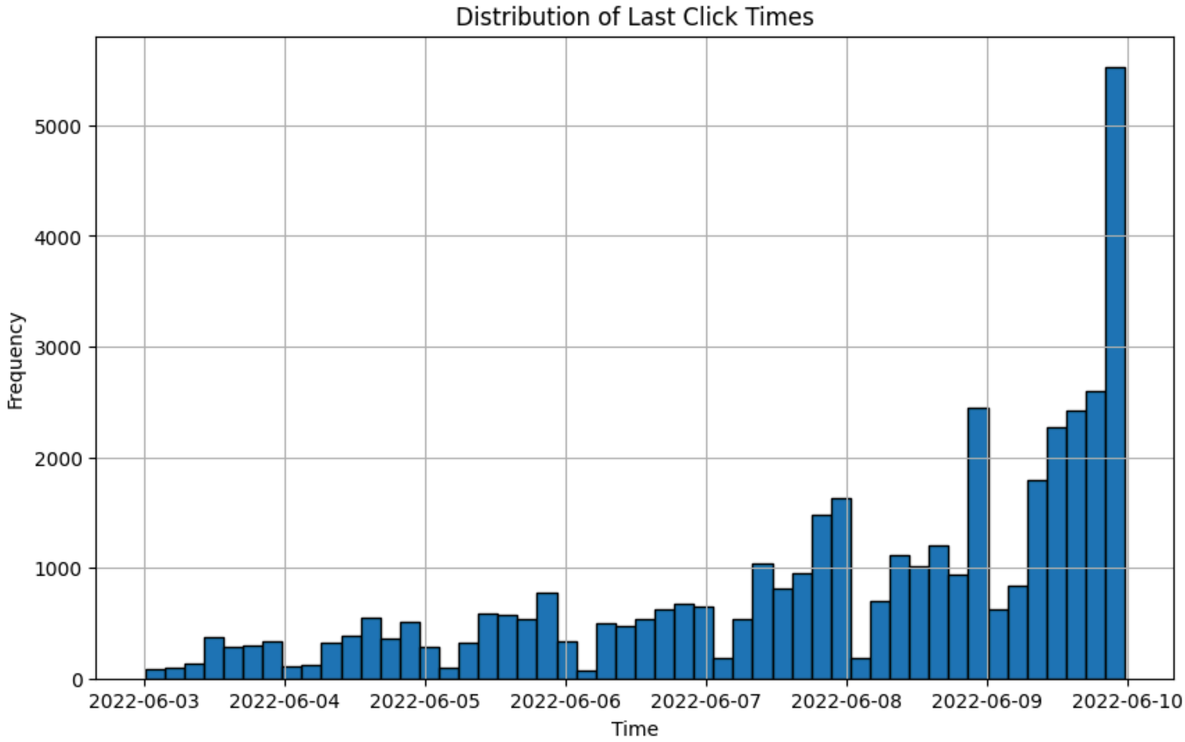
- We use pro (browsing progress) to filter out unreasonable usage durations.
- We find that average time spent (by id) is 104 min

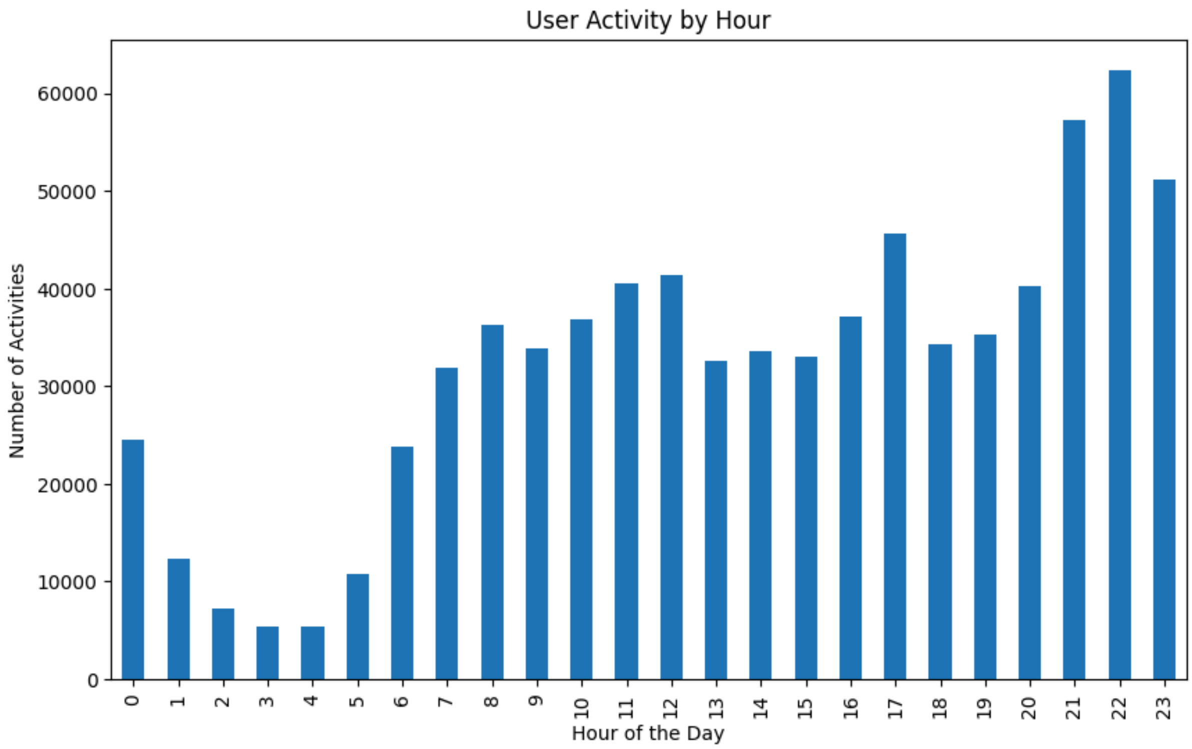

Distribution of pro Field
Frequency
800000
700000
600000
500000
400000
300000
200000
100000
0
0 2 9 10 14 15 16 18 19 20 21 22 24 26 28 29 31 32 33 34 35 38 40 41 42 43 44 45 47 48 49 50 53 54 55 58 59 60 63 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 83 84 86 87 88 90 91 93 94 95 96 97 98 99 100
pro


Distribution of User Session Duration
Frequency
60000
50000
40000
30000
20000
10000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
Session Duration (minutes)



- Click behaviors showed distinct peaks during specific time intervals.





Time Series Analysis

Trend Analysis

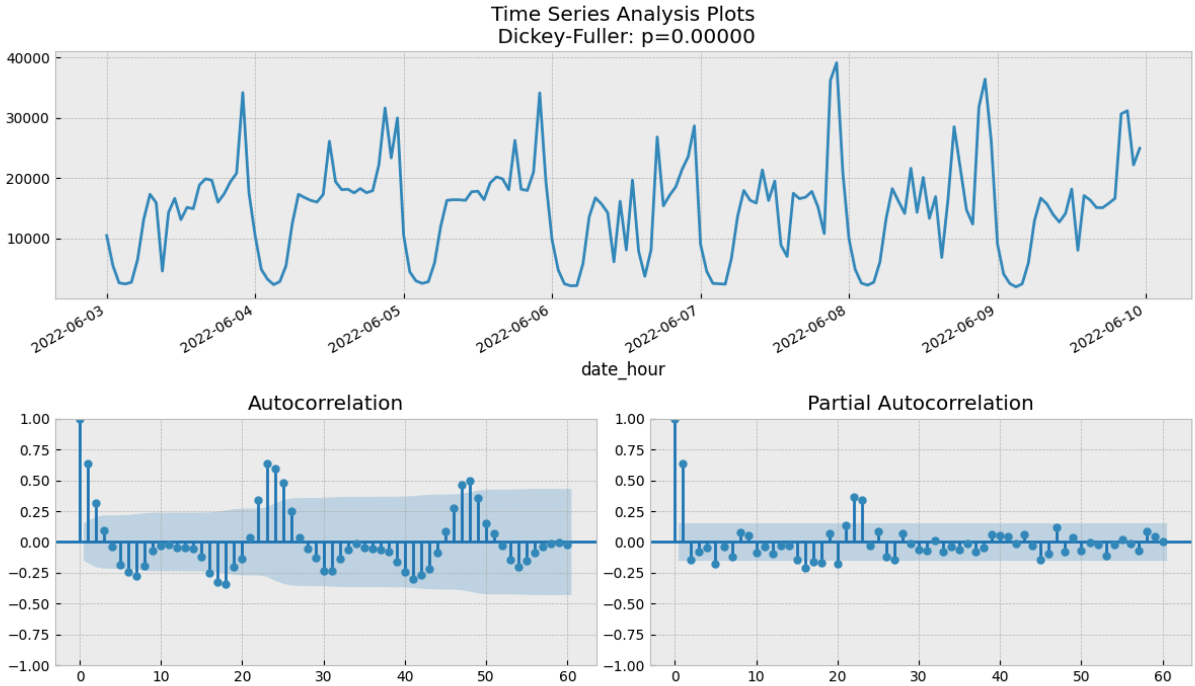
- Exponential Smoothing revealed clear trends in user activity.
- ACF and PACF analyses indicated strong autocorrelations in time series data.
- Data was stabilized using 24-hour differencing to mitigate seasonality.

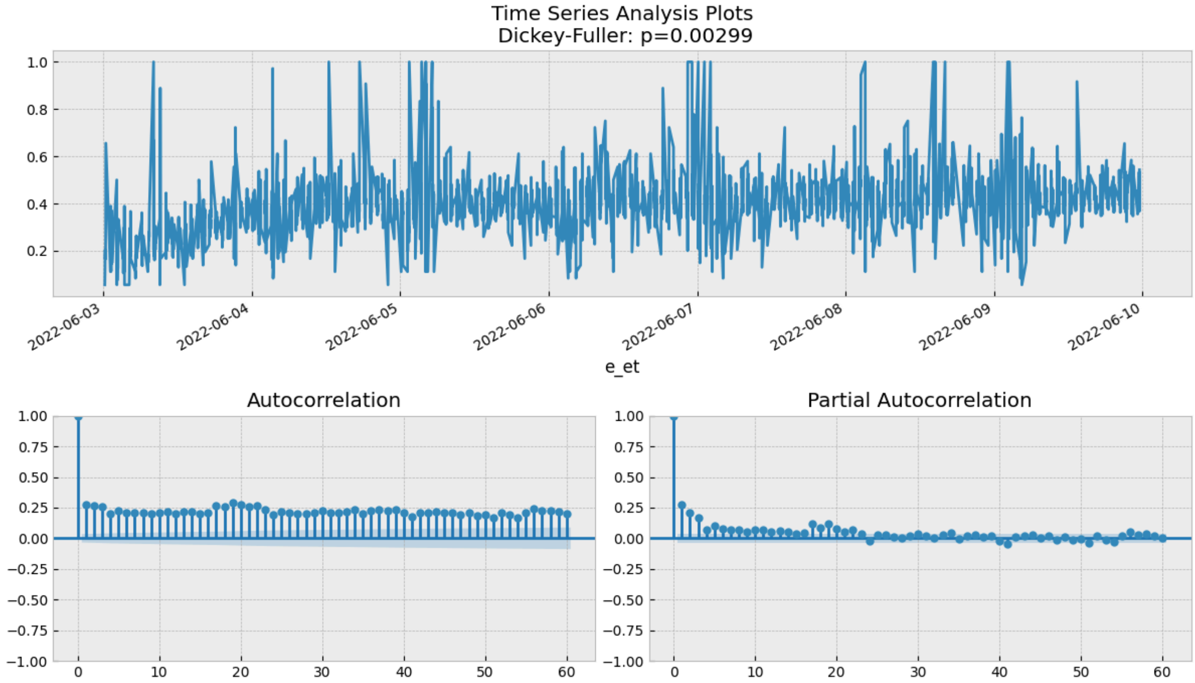
Anomaly Detection

- Moving average analysis identified several anomalous delivery periods.
- High CTR periods were often correlated with anomalous delivery volumes.

Dickey-Fuller Test

- The Dickey-Fuller test was applied to test for stationarity.
- Findings:
  - The null hypothesis (presence of a unit root) was rejected.
  - This indicates that the time series data is stationary and suitable for further analysis.


Time Series Analysis Plots
Dickey-Fuller: p=0.00000
40000
30000
20000
10000
2022-06-03
2022-06-04
2022-06-05
2022-06-06
2022-06-07
2022-06-08
2022-06-09
2022-06-10
date_hour
Autocorrelation
Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50
60


Time Series Analysis Plots
Dickey-Fuller: p=0.00299
1.0
0.8
0.6
0.4
0.2
2022-06-03
2022-06-04
2022-06-05
2022-06-06
2022-06-07
2022-06-08
2022-06-09
2022-06-10
e_et
Autocorrelation
Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50
60

Chapter 3: Methodology

We first evaluate classical statistical learning models — Logistic Regression, Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA). These methods offer interpretable and computationally efficient baselines. They allow for a clearer understanding of data relationships, providing a benchmark to assess the performances of XGBoost and CatBoost.

Baseline Methods from Textbook: Logistic, LDA, and QDA

- **Logistic Regression**
  Logistic regression is a classification algorithm for binary outcomes. It models the log-odds of the target class as a linear combination of input features.
- **Linear Discriminant Analysis (LDA)**
  LDA is a method that finds a linear combination of features to separate classes. It works well when predictor distributions are normal and class covariances are similar
- **Quadratic Discriminant Analysis (QDA)**
  QDA extends LDA by allowing different covariance structures for each class, making it better suited for more complex, non-linear class separations.

After establishing benchmarks with these textbook models, we introduce XGBoost and CatBoost, which leverage gradient boosting on decision trees for enhanced predictive performance.

XGBoost, CatBoost and Bag-of-Values Encoder

1) **XGBoost**

   XGBoost (eXtreme Gradient Boosting) is a machine learning algorithm based on Gradient Boosted Decision Trees. It is one of the boosting ensemble methods, designed to combine hundreds or thousands of tree models into a highly accurate predictive model. The algorithm works iteratively, generating new trees to correct errors from previous ones and improving the overall performance.

   XGBoost has several important features that make it very efficient. It utilizes a gradient boosting framework to iteratively build new decision trees to correct errors of previous models and improve accuracy. To prevent overfitting, it combines L1 and L2 regularization, while its parallelization capabilities ensure efficient training. In addition, XGBoost automatically handles missing values and intelligently decides how to deal with incomplete data. It provides tree pruning and depth control to keep model complexity manageable, and supports predefined and customizable loss functions to provide flexibility for a variety of tasks.

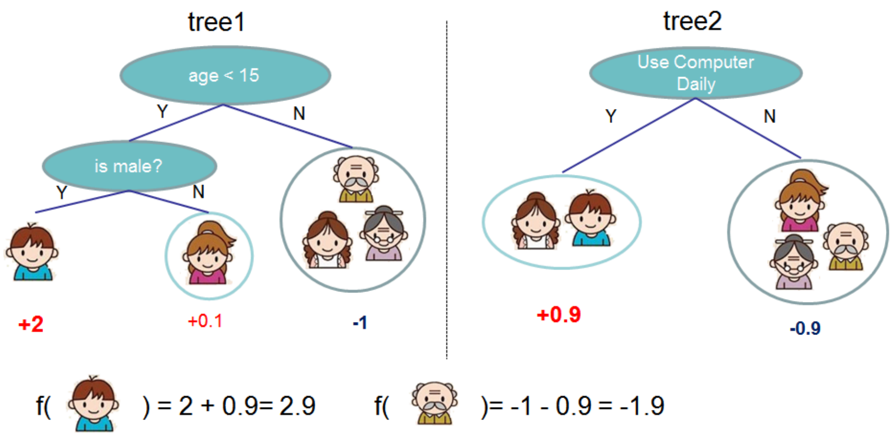


Figure: concept of XGBoost (references:https://ithelp.ithome.com.tw/articles/10301273)

As shown in the figure above, XGBoost can be thought of as multiple CARTs for fusion. We can add the outputs of the two trees to get the final output.

2) CatBoost

The "Cat" in CatBoost's name stands for Categorical, because it is an algorithm designed specifically for processing categorical data. However, CatBoost is not only suitable for classification problems, but can also be applied to regression problems. The core of CatBoost is that it uses symmetric decision trees (Symmetric or Oblivious Trees). This tree structure performs level-wise segmentation through binary segmentation, that is, all nodes in each layer use the same feature as the segmentation benchmark. , and the segmentation result only has two choices: "yes" and "no". This makes the structure of the decision tree very simple and regular, making it more suitable for dealing with categorical problems.

In addition, CatBoost has an Ordered Boosting mechanism to prevent overfitting. It randomly arranges and combines category features to increase the diversity of statistical calculations, thereby effectively avoiding target leakage and overfitting problems.

CatBoost offers several advantages, including native support for categorical features, simplified data preprocessing, and more. Its symmetrical decision tree structure simplifies the model and improves efficiency and consistency. In particular, the Ordered Boosting mechanism effectively alleviates overfitting by avoiding target leakage and enhancing generalization.

3) Bag-of-Values Encoder

The Bag-of-Values Encoder is an encoding technique used to handle categorical features where each data point contains multiple categorical values. Unlike traditional single-label encoding methods (e.g., one-hot encoding), this approach effectively transforms multi-valued categorical variables into numerical representations. This encoding is particularly beneficial in fields such as recommendation systems, natural language processing (NLP), and any domain involving multi-label or multi-tagged data.

For instance, in our news recommendation system, a user's interests might be represented as "*sports^travel^music*", where each category holds equal importance. Simply treating this feature as a single categorical value would lose granular information. To resolve this, the Bag-of-Values Encoder transforms multi-valued features into a format suitable for machine learning models by creating numerical vectors that retain the original information structure.

Chapter 4: Data Analysis

Baseline Methods (nonsampling)

- Logistic Regression: 81.5%
- LDA: 81.1%
- QDA: 79.7%

Boosting Methods

We set early stopping to avoid overfitting, which monitors the validation data accuracy. To compare models with more metrics, we record testing data accuracy and the best iteration. The result from different sampling methods are as follows:

	XGBoost	**CatBoost**
nonsampling	Accuracy: 0.883 Iteration: 10,000	Accuracy: 0.868 Iteration: 3,422
oversampling	Accuracy: 0.880 Iteration: 10,000	Accuracy: 0.868 Iteration: 10,000
undersampling	Accuracy: 0.857 Iteration: 6,546	Accuracy: 0.776 Iteration: 3

Table: Result from different sampling methods

From the table above, we can find that nonsampling has best prediction performance. Therefore, we optimize our models under non-sampling conditions.

In the previous model, we excluded object-type variables for simplicity. However, recognizing their potential importance, we applied bag-of-values encoding to these variables. Results comparing the use and non-use of the bag-of-values encoder under non-sampling conditions are shown as follows:

	XGBoost	**CatBoost**
without bag-of-values encoding	Accuracy: 0.883 Iteration: 10,000	Accuracy: 0.868 Iteration: 3,422
with bag-of-values encoding	Accuracy: 0.956 Iteration: 8,863	Accuracy: 0.916 Iteration: 6,552

Table: Comparison between use and non-use of the bag-of-values encoding

As shown in the table above, we found bag-of-values encoding improves model performance significantly, and the learning curves of the two models with bag-of-values encoding show as below:

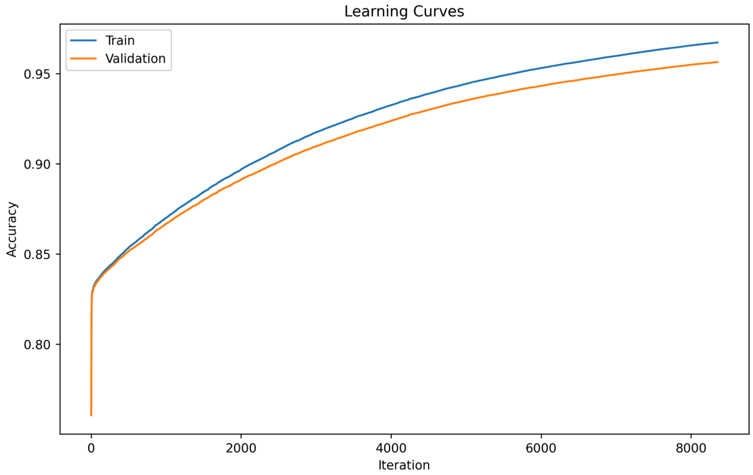


Figure: XGBoost with bag-of-values encoding

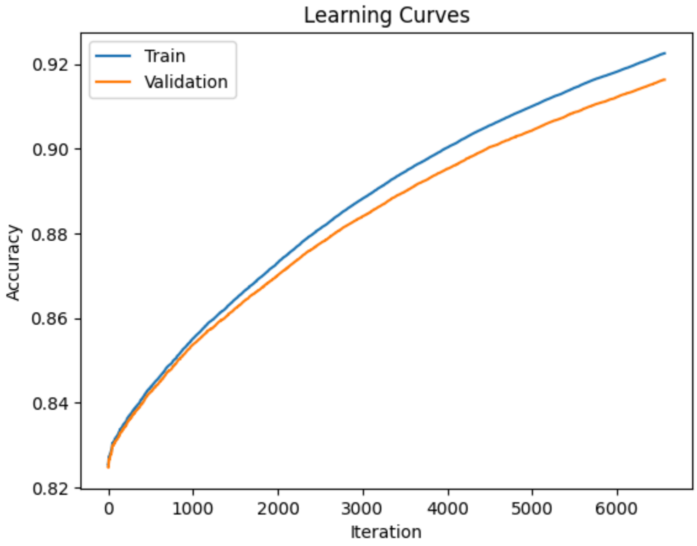


Figure: CatBoost with bag-of-values encoding

From the two figures above, XGBoost shows rapid initial improvement, followed by a plateau. This suggests that the model quickly captures the key patterns in the data, and further progress slows as it learns the more challenging aspects. The plateau may also be due to overfitting risks.

In contrast, CatBoost maintains a steady upward trend, indicating more stable learning. Its consistent performance improvement is likely due to its effective handling of categorical variables and techniques that reduce overfitting, allowing for gradual refinement of predictions.

To know which features are important for potential customer prediction, we record the feature importance with the most accurate models. The result shows as below:

	XGBoost with bag-of-values encoder		**CatBoost with bag-of-values encoder**	
	feature	**importance**	**feature**	**importance**
1	u_newsCatInterestsST_feature_0	182.81	e_m	11.13
2	u_feedLifeCycle	141.39	u_refreshTimes	9.44
3	u_newsCatDislike_feature_2	126.08	e_pl	7.37
4	u_refreshTimes	85.36	u_phonePrice	3.11
5	u_newsCatDislike_feature_64	55.83	u_newsCatInterests_feature_96	2.58

Table: Comparison between feature importance (only listed the top 5 most important features)

The feature with '_feature_number' in its name is generated from an object feature using bag-of-values encoding. From the tables above, we can find that some features derived from object features are important, so if we don't use bag-of-values encoding, we will lose the important information thus decreasing prediction accuracy.

In addition, the feature *u_refreshTimes* is regarded as important in both methods, which represents the average number of the user's valid news feeds updates per day. Since valid news feeds updates indicate the user's engagement and loyalty, it's reasonable that the feature is important for potential customer prediction. The features *u_newsCatInterestsST* and *u_newsCatDislike* are also important in XGBoost, indicating that the users' behaviors are also helpful for predicting potential customers.

In conclusion, we recommend that the company use XGBoost with bag-of-values encoding to predict potential customers. Besides, the company can think about how to change the user's behavior to increase his/her loyalty, thus becoming a potential customer.

Chapter 5: Conclusion

In this report, we predict potential customers for precision marketing using boosting methods (XGBoost and CatBoost) combined with bag-of-value encoding for categorical features.

Three Key Findings

- **Model Performance Across Sampling Methods**
  XGBoost consistently outperformed CatBoost in terms of accuracy under various sampling conditions.
- **Impact of Bag-of-Values Encoder**
  The bag-of-values encoder significantly improved model performance under non-sampling conditions
    - XGBoost accuracy increased from 0.8826 to 0.9559.
    - CatBoost accuracy increased from 0.8682 to 0.9163.
- **Feature Importance**
  XGBoost, features like *u_newsCatInterestsST_feature_0* and *u_feedLifeCycle* were the most influential in predicting potential customers; for CatBoost, however, features such as *e_m* and *u_refreshTimes* are of greatest importance.

Future Directions

In our current work, we focused on building the most accurate models to predict potential customers. However, in real-world scenarios, the cost of misclassification varies:

- **False Negatives** (missed potential customers) result in significant revenue loss.
- **False Positives** (incorrectly identified potential customers) only incur small advertising or marketing costs.

Since missing a potential customer is far more costly, future work should focus on misclassification cost weighting. By assigning higher penalties to false negatives, we can prioritize recall to capture more potential customers while still managing precision to minimize unnecessary costs.

Additionally, we can extend our work by integrating **time-series models** into the existing framework to better capture temporal patterns in user behavior. While XGBoost and CatBoost have demonstrated strong predictive performance, they are inherently static models and do not account for sequential dependencies in time-sensitive features such as *u_refreshTimes* and *e_et*. Combining these two approaches can create a more robust and dynamic prediction pipeline.

Chapter 6: Contribution

	Team Members
Chapter 1. Introduction	沈信憲、李文先、周文揚、巖昀叡、任翊瑄
Chapter 2. EDA	巖昀叡
Chapter 3. Methodology	XGBoost: 沈信憲、李文先 CatBoost: 周文揚、巖昀叡、任翊瑄
Chapter 4. Data Analysis	XGBoost: 沈信憲、李文先 CatBoost: 周文揚、巖昀叡、任翊瑄
Chapter 5. Conclusion	沈信憲、李文先、周文揚、巖昀叡、任翊瑄

Appendix

- **References**

  - Zhu, J., Liu, C., Jiang, X., Peng, C., Lin, Z., & Shao, J. (2024). Generalize for future: Slow and fast trajectory learning for CTR prediction. In Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24).
  - Mumtaz, S., & Giese, M. (2022). Hierarchy-based semantic embeddings for single-valued & multi-valued categorical variables. *Journal of Intelligent Information Systems*, 58, 613–640.
  - Kwok, T. S. T., Wang, C. H., & Cheng, G. (2024, November 5). DEREC-SIMPRO: Unlock language model benefits to advance synthesis in data clean room.

- **Codes**
  https://colab.research.google.com/drive/1bEmWzJoSbKHK9K0XiFX0Z9I-M4DdjxaI?usp=drive_link