SCHOOL OF ECONOMICS AND MANAGEMENT

Master's Programme in Data Analysis and Business Economics

Click Through Rate Prediction Leveraging Machine Learning Techniques for Mobile Digital Advertisement

by Juliana Rojas Guillen

DABN01
Master's Thesis (15 credits ECTS)
May 2024
Supervisor: Simon Reese

Abstract

Predicting click-through rates (CTR) is essential for optimizing the effectiveness of mobile advertising campaigns, where accurate prediction of user interactions can significantly enhance revenue generation and ad targeting strategies. This thesis investigates the efficacy of different predictive models, using a dataset composed of impressions and interactions with mobile ads. The models examined include Logistic Regression, Random Forest, XG-Boost, CatBoost, and Feed Forward Neural Networks. Additionally, a K-Means Clustering approach was employed to segment the data into clusters prior to modeling. The findings reveal that ensemble methods, particularly CatBoost, outperformed the other tested models, delivering the lowest log-loss (0.5836) and the highest F1-score (0.7093). This superior performance highlights the robustness of gradient boosting machines in handling mobile ad data, which is often categorical and highly dimensional. Finally, features such as site_id, app_id, and device_model were identified as the most influential in the prediction of CTR using the best-performing model.

Keywords: Click-Through Rate, Machine Learning, Mobile Ads, CatBoost

Acknowledgments

I feel thankful to have achieved one of my life goals by completing my master degree presenting this thesis. I want to express my deepest appreciation to my supervisor, Simon Reese, for his constant support and valuable advice throughout this study.

I also owe a profound debt of gratitude to my family and friends. whose love and support have constantly encouraged me to pursue each of my dreams. I dedicate this milestone to each one of them.

Contents

1. Introduction....7
2. Literature Review....8
3. Data....9
3.1. Dataset Description....9
3.2. Feature Transformations....11
3.3. Handling Imbalanced Data....16
4. Methodology....17
4.1. Models....17
4.2. Evaluation Metrics....22
5. Empirical Analysis....23
5.1. Modeling settings....23
5.2. Performance Overview....27
5.3. Feature Importance....28
5.4. Discussion....29
6. Conclusion, Limitations and Future Research....30
Appendix A....37
Appendix B....43

List of Figures

Figure 3.1: Distribution and CTR of app_category..... 10
Figure 3.2: Distribution of num_impressions_user_day..... 13
Figure 3.3: Distribution of time_interval_last_visit..... 13
Figure 3.4: Distribution of num_days_user_appears..... 14
Figure 3.5: Distribution of num_previous_clicks..... 14
Figure 3.6: Cyclic Representation of Time..... 15
Figure 3.7: Imbalance Distribution of Click..... 16
Figure 3.8: Balance Distribution of Click..... 16
Figure 4.1: Training dataset grouped in clusters..... 21
Figure 4.2: Elbow Method to determine optimal number of clusters..... 21
Figure 5.1: Confusion Matrix of Logistic Regression L2..... 28
Figure 5.2: Feature Importance Bar Plot..... 29
Figure B.1: Distribution of variable app_category..... 37
Figure B.2: Distribution of variable app_domain..... 37
Figure B.3: Distribution of variable app_id..... 37
Figure B.4: Distribution of variable banner_pos..... 38
Figure B.5: Distribution of variable C1..... 38
Figure B.6: Distribution of variable C14..... 38
Figure B.7: Distribution of variable C15..... 39
Figure B.8: Distribution of variable C16..... 39
Figure B.9: Distribution of variable C17..... 39
Figure B.10: Distribution of variable C18..... 40
Figure B.11: Distribution of variable C19..... 40
Figure B.12: Distribution of variable C20..... 40
Figure B.13: Distribution of variable C21..... 41
Figure B.14: Distribution of variable device_conn_type..... 41
Figure B.15: Distribution of variable device_type..... 41
Figure B.16: Distribution of variable site_category..... 42
Figure B.17: Distribution of variable site_domain..... 42
Figure B.18: Distribution of variable site_id..... 42
Figure B.19: Distribution of variable device_id..... 43
Figure B.20: Distribution of variable device_ip..... 43
Figure E.1: Confusion Matrix Logistic Regression Dimension Reduction..... 44
Figure E.2: Confusion Matrix Logistic Regression L2..... 44
Figure E.3: Confusion Matrix Random Forest..... 45
Figure E.4: Confusion Matrix XGBoost..... 45
Figure E.5: Confusion Matrix CatBoost..... 46
Figure E.6: Confusion Matrix Feedforward Neural Network..... 46
Figure E.7: Confusion Matrix KMeans..... 47

List of Tables

Table 5.1: Parameters for Cross-Validation for Logistic Regression..... 24
Table 5.2: Parameters for Cross-Validation for Random Forest.....24
Table 5.3: Parameters for Cross-Validation for XG-Boost.....25
Table 5.4: Parameters for Cross-Validation for CatBoost..... 25
Table 5.5: Parameters for Cross-Validation for Feedforward Neural Network.....26
Table 5.6: Best Models for Clusters..... 27
Table D.1: Model Performance Results.....27

1. Introduction

Mobile devices are now integral to the daily lives of most people. As of 2023, about 68% of the global population uses mobile devices, a number expected to increase to 74% by 2030 (GSMA Intelligence, 2024). This rise in mobile usage has transformed advertising strategies, making digital platforms essential for how businesses engage with consumers, particularly through smartphones. In fact, the mobile advertising industry was valued at $175 billion in 2023, and forecasted to escalate to $1,040 billion by 2032 (Fortune Business Insights, 2024).

Click Through Rate (CTR) serves as a key metric in online advertising and digital marketing campaigns. It is defined as the ratio of users who click on a specific link or advertisement to the total number of users who view the page, ad, or any other form of digital content (Shah & Nasnodkar, 2021). A high CTR suggests that advertisements are well-targeted, resonating with viewers and often leading to higher conversion rates. Furthermore, CTR is vital for search engine advertising, influencing ad rank and affecting visibility on search engine results pages (Haans et al., 2013). This involves placing ads on search engine results pages to reach users actively searching for related products or services. Higher CTRs can result in lower costs per click (CPC), which is the amount advertisers pay each time a user clicks on their ad, consequently maximize return on investment (ROI). This underscores the importance of optimizing CTR for the success of online advertising platforms (Yang & Zhai, 2022). In this scenario, accurate prediction of CTR is crucial for advertisers as it enables more efficient allocation of advertising budgets, improves targeting strategies, and enhances overall campaign performance.

In this regard, the goal of this thesis is to explore different machine learning algorithms to predict whether a new ad will be clicked or not, using a large dataset on click-through rates in mobile ad impressions. Specifically, the study will investigate the efficacy of algorithms such as Logistic Regression, Random Forest, XGBoost, CatBoost and Feed Forward Neural Network. Additionally, this study introduces an advanced approach for preprocessing high cardinal categorical variables using a combination of feature hashing and count encoding, and a clustering approach using K-Means prior to predictive modeling was tested. By evaluating the models, this study identified the most accurate method was CatBoost, which achieved the lowest log-loss (0.5836) and the highest F1-score (0.7093) and the most significant features influencing CTR are site_id, app_id, and device_model .

The structure of this thesis is as follows: Section 2 presents a review of the relevant literature. Section 3 describes the dataset, feature transformations, and data imbalance handling. Section 4 details the methodology, including the machine learning algorithms used and the evaluation metrics. Section 5 outlines the empirical analysis, discussing modeling settings, results, and performance. Finally, Section 6 presents the conclusion, limitations, and suggests future research directions.

2. Literature Review

Click Through Rate (CTR) prediction has received considerable attention from research in the last decades where machine learning models and different approaches were continuously proposed (Yan et al., 2022). Among the relevant models reviewed in the literature for predicting CTR, classification algorithms such as Logistic Regression and ensemble methods like Random Forest, XG-Boost, and CatBoost stand out.

Investigations made by Richardson et al. (2007), Agarwal et al. (2015), and Kumar et al. (2015) concluded that Logistic Regression was effective in predicting CTR, especially in requiring minimal data preprocessing makes it suitable choices for scenarios where computational efficiency is crucial. While Logistic Regression is simple and interpretable, its limitations in capturing complex, non-linear relationships can lead to suboptimal predictive performance (James et al., 2022).

Other studies found that tree-based methods had higher performance than linear methods. For instance, Zhou (2022) compared Logistic Regression and Decision Trees to analyze user click behavior on ads, finding that Decision Trees outperformed due to their ability to handle nonlinear and complex relationships. Shi and Li (2016), who analyzed CTR and average cost per click (CPC) for keywords in Google AdWords, concluded that Random Forest delivered the most accurate predictions for both CTR and average CPC. He et al. (2014) demonstrated that a hybrid model combining Decision Trees with Logistic Regression significantly enhances click prediction accuracy emphasizing the critical role of feature selection.

In addition, the study of Çakmak et al. (2019), which aimed to predict the number of clicks for hotel advertisements in a meta-search bidding engine, concluded that XGBoost outperformed other models such as Random Forest, Gradient Boosting, Ada Boost and

Support Vector Machine (SVM), even running over ten times faster. Similarly, AlAli Moneera et al. (2021) found that XGBoost performance was superior than Logistic Regression and Random Forest in predicting CTR with reduced computational power and fewer features. On the other hand, Kulkarni (2022) highlighted CatBoost's advantages compared to XGBoost or AdaBoost, due to its capacity to handle categorical data which are very common to have in advertisement datasets that include customer demographics, interests, and behaviors. Likewise, Yi and Chang (2021) investigated the effectiveness of tabular learning models for CTR prediction in developing countries, demonstrating that CatBoost provides a cost-effective alternative to over-parameterized deep learning models.

Finally, an interesting approach using clustering was made by Kumar et al. (2020). Their study explored the use of clustering techniques, specifically comparing fuzzy c-means (FCM) and K-means clustering, to predict CTR. Their findings demonstrated that FCM, which allows ads to belong to multiple clusters, provided more accurate predictions of user clicks.

3. Data

In this section, the dataset description and feature transformations will be presented.

3.1. Dataset Description

The dataset was extracted from Kaggle[1], a data science website, which hosted a competition by Avazu, an online advertising service provider. The dataset contains 40,428,967 rows and consists of 10 days of click-through data corresponding to mobile advertisement impressions with the following 24 features.

- **id:** Unique identifier for each ad impression
- **click:** Indicates if the ad impression was clicked or not (0 for non-click and 1 for click)
- **hour:** Hour in the format YYMMDDHH, 14091123 means 23:00 on Sept. 11, 2014
- **banner_pos:** Banner position
- **site_id:** Identifier of the site anonymized

[1] *https://www.kaggle.com/c/avazu-ctr-prediction*

- **site_domain:** Domain name or URL anonymized
- **site_category:** Site category anonymized
- **app_id:** App identifier anonymized
- **app_domain:** Domain of app anonymized
- **app_category:** Category of app anonymized
- **device_id:** Device identifier anonymized
- **device_ip:** IP of the device used at the time the ad was shown to the user
- **device_model:** Model of the device anonymized
- **device_type:** Type of the device anonymized
- **device_conn_type:** Connection type anonymized
- **C1, C14-C21:** Anonymized categorical variables

In addition, due to computational limitations, this study will consider a random sample of 1 million data points. On the other hand, as part of the Exploratory Data Analysis (EDA), bar plots were created like the one shown in Figure 3.1, which also includes the CTR for each of the category types related to the specific feature. In this case, we can notice that most app categories are concentrated in five categories, and the CTR of the most concentrated category is the second highest, which might suggest that popular categories can still maintain high user engagement. Additionally, some less populated categories show high CTRs, indicating niche categories with high user interest. Similarly, the other feature distributions are shown in Appendix A, revealing that certain categories with high CTRs can be good indicators of engaging ads. Advertisers can benefit from these insights by developing strategies to improve the performance of less engaging categories.

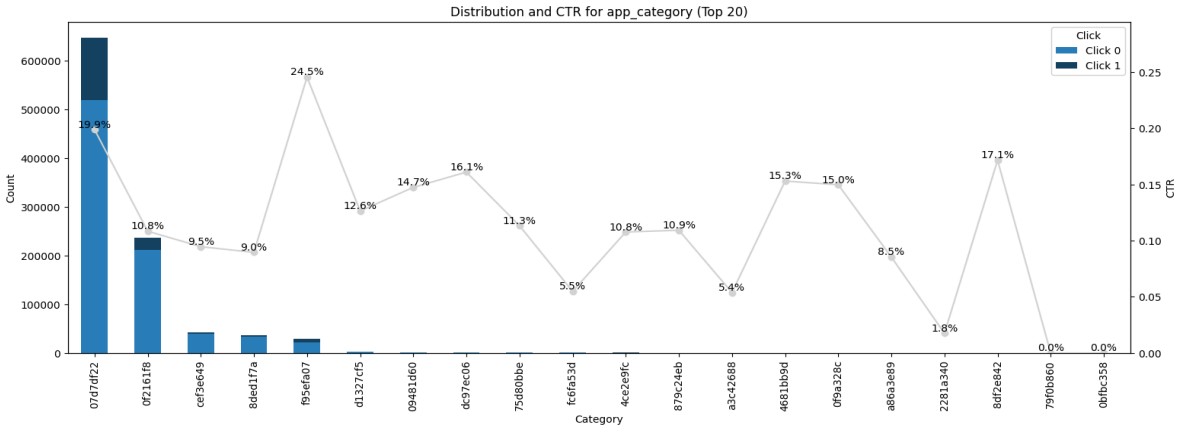


Figure 3.1: Distribution and CTR of app_category

Another insight from EDA is related to the cardinality of the categorical variables. For instance, the number of unique values for device_id is around 150 000 and for device_model, it is around 5 000. These high cardinalities are important to consider when choosing feature encoding techniques in the preprocessing part of the study. In addition, it is observed that there is a high number of data points with same device_id. However, there is no concrete information from the dataset author indicating that this high frequency might constitute outliers. While removing these data points would significantly reduce the dataset size, we opted to keep them.

3.2. Feature Transformations

This section presents the feature transformations applied to the dataset. Categorical variables were encoded to make them suitable for the applied machine learning models. Furthermore, the time-related variable 'hour' was encoded using sine and cosine transformations to capture its cyclical nature. Finally, new variables related to user behavior were created to leverage additional information.

Categorical Encoding Transformations

The dataset used in this study contains mostly categorical variables that need to be properly encoded before running the machine learning models. Different methods exist to encode categorical data, such as one-hot encoding, label encoding, count encoding, feature hashing or hashing trick, etc. These approaches offer distinct advantages and trade-offs in terms of computational efficiency, and handling of high-cardinality categorical variables. While one-hot encoding is a straightforward method that preserves the semantic meaning of categorical variables, the expansion may result in the curse of dimensionality, computational inefficiency, and increased memory requirements, particularly for large datasets with numerous categorical features (Potdar et al., 2017). Therefore, it might not be suitable to use it as the encoding method for this study, since most of the categorical variables in the dataset have high cardinality, for instance, device_id has around 150,000 unique values. Given this constraint, it might not be suitable to employ one-hot encoding in this context. Instead, two alternative methods are considered: feature hashing, and count encoding.

Feature hashing, also known as the hashing trick, is a technique that maps high-dimensional feature vectors into a lower-dimensional feature space through a hash

function. It is particularly effective in multitasking learning settings and helps manage large-scale datasets (Weinberger et al., 2009). Unlike one-hot encoding, which requires creating separate binary variables for each category, feature hashing directly computes the hash value of each category and assigns it to a pre-defined number of hash bins or buckets. This hashing process effectively reduces the dimensionality of the feature space, as the number of hash bins is typically much smaller than the total number of unique categories. Feature hashing works by applying a hash function to each feature, which ensures that similar features are likely to hash to the same bucket, while still allowing for efficient computation. The method also handles collisions by using additional information such as a second hash function or a sign function to differentiate between features that hash to the same bucket. This allows for efficient storage and retrieval of features, making it highly suitable for large-scale machine learning applications. Additionally, feature hashing is computationally efficient and can be implemented with low memory overhead, which is particularly beneficial when dealing with sparse data. For this study, the following variables have been encoded using the hashing trick: site_id, site_domain, app_id, app_domain, app_category, device_id, device_ip, device_model, C14 and C20.

In addition, the method Count Encoding is considered for its simplicity and computational constraints and can be utilized when multiple categories do not have identical counts of observations. (Zeng, 2023). This method, transforms the categorical variable into a numerical feature based on frequency of each category and can be defined with the following equation,

$$\sum_{i=1}^{n} I\,(X_i = j)$$

where I is the indicator function, X_i is the evaluated data point and j is the evaluated category type. The resulting frequency is used as the encoding numerical value for the corresponding category of the feature. In case of variables C1 and site_category of this thesis, this method was selected given that the categories have not the same counts and it is more computational efficient than feature hashing. Finally, categorical variables with numeric values such as “banner_pos”, “device_type”, “device_conn_type”, “C21” and “hour” are utilized directly in the model without need for transformation.

Creation of user behavior variables

Given that the dataset provides a time-type feature 'hour' and a feature that can approximate to identify a single user 'device_id', the following variables were created to leverage the information we can extract from the user behavior.

- **num_impressions_user_day:** This variable quantifies the total number of ad impressions received by a single user (identified by device_id) within a single day. It is created to understand the frequency of ad exposure per user per day.

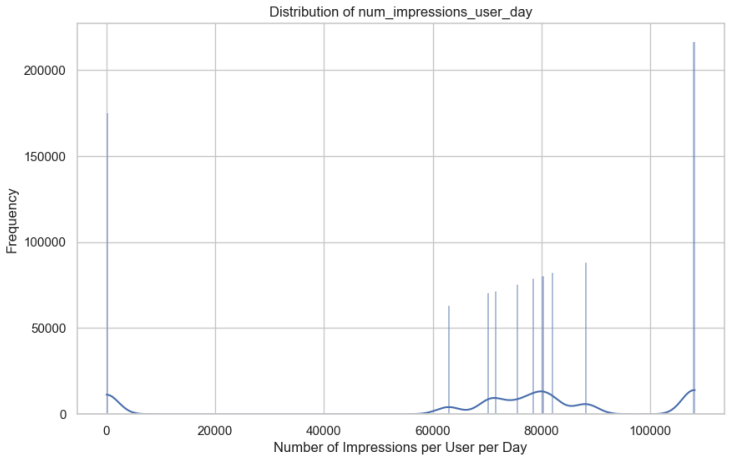


Figure 3.2: Distribution of num_impressions_user_day

- **time_interval_last_visit:** This variable represents the time between consecutive visits by the same user. It provides information on the rhythm of interactions.

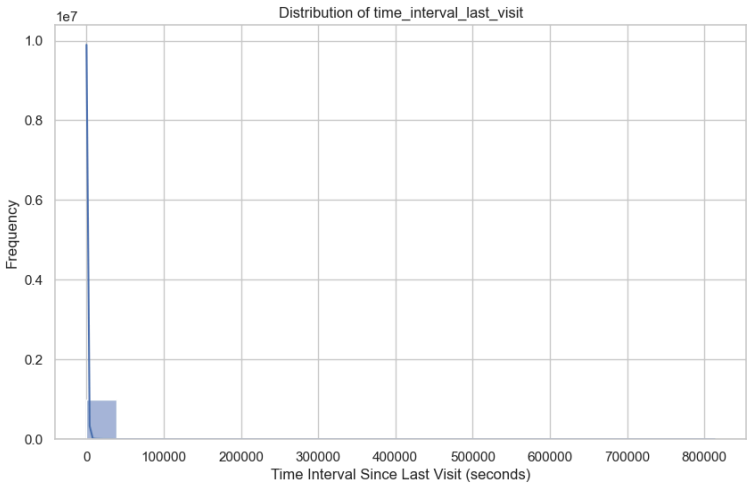


Figure 3.3: Distribution of time_interval_last_visit

- **num_days_user_appears:** It is defined as the number of distinct days a user appears in the dataset, quantifying the unique days associated with each user.

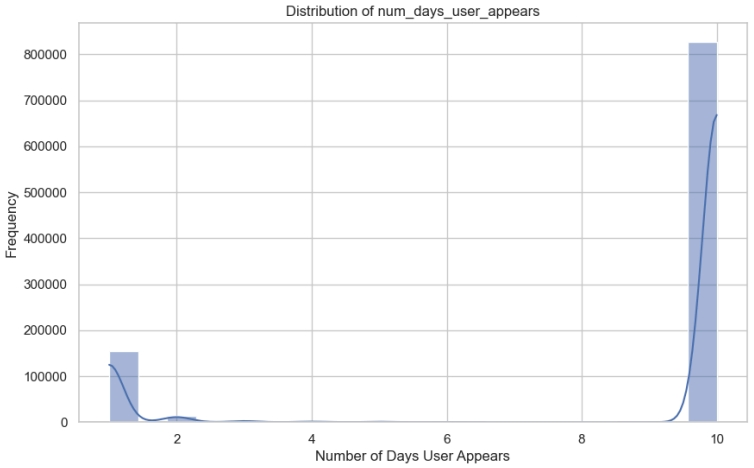


Figure 3.4: Distribution of num_days_user_appears

- **num_previous_clicks:** This feature counts the cumulative number of ad clicks by a user prior to the current impression, offering a historical perspective on the user's responsiveness to ads.

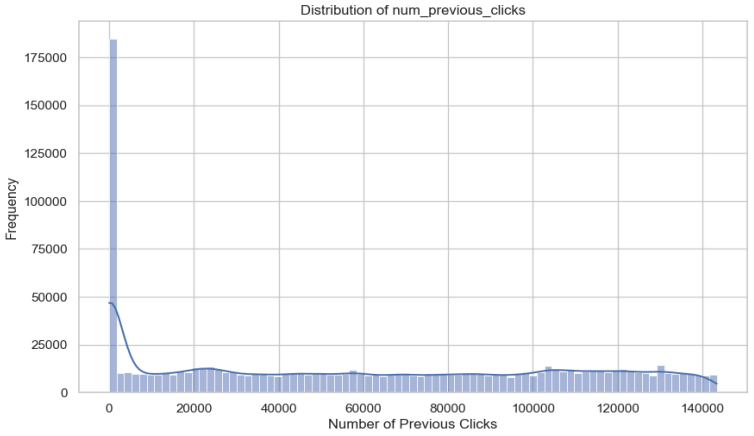


Figure 3.5: Distribution of num_previous_clicks

As we can see in the plots above, distributions of the new variables are highly skewed, which could adversely affect the performance of linear models such as logistic regression. However, tree-based methods, which are less sensitive to skewness, may still benefit from these variables, leveraging information related to user behavior.

Creation sin and cosine time features

Considering that the dataset has the feature ‘hour’, which has the date and hour of the ad impression, it is appropriate to encode this time variable, taking into account the cycle nature of time. To achieve this, sin and cosine transformations are applied. By applying sin and cos transformations, each point in time is mapped onto a unit circle, preserving the cyclic nature of time (Time-Related Feature Engineering, 2024). This transformation is crucial for handling features that exhibit regular intervals, such as hours of the day. To preserve the cycle of ad impressions, in this study, the variable ‘hour’ is transformed into variables ‘sin_time’ and ‘cos_time’, whose formulas for their creation are as follows:

$$\text{sin\_time} = sin\ (\frac{2\Pi * 'hour'}{24})\ ;\ \text{cos\_time} = cos\ (\frac{2\Pi * 'hour'}{24})$$

These transformations map each hour onto a unit circle, effectively capturing the cyclic nature of 'hour,' as shown in Figure 3.6. The plots illustrate how sine (blue curve) and cosine (red curve) transformations create smooth oscillations between -1 and 1 for existing days of ad impressions. This differs from simply extracting the hour as values from 0 to 23, which treats time as a linear progression; therefore, the difference between 23:00 and 0:00 is 23 hours, rather than 1 hour. By applying sine and cosine transformations to the hour feature, each time point is mapped onto a unit circle, maintaining the cyclicality of the hour feature and making these transformations more suitable for a model's ability to detect and utilize time-dependent patterns.

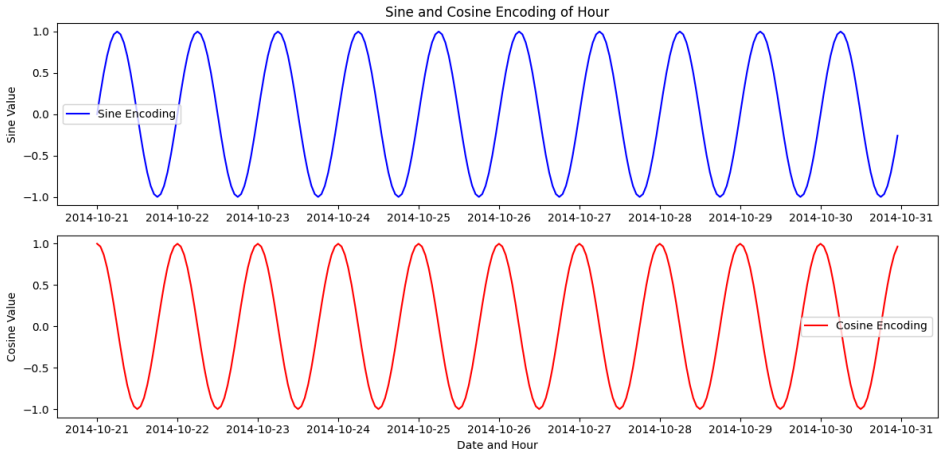


Figure 3.6: Cyclic Representation of Time

3.3. Handling Imbalanced Data

Imbalanced datasets challenge many machine learning approaches due to significant class disparity. When one class outweighs others, predictive models may become biased, showing high accuracy overall but poor performance on the minority class (Kuhn & Johnson, 2013). This is critical in fields where the minority class, like clicks on advertisements, is less frequent but more important. To address this, combining various sampling methods is effective for improving model generalizability across classes (Yap Bee Wah et al., 2016).

One technique for handling imbalanced data is downsampling, which involves randomly selecting a subset of the overrepresented class to balance the distribution. This method aims to prevent bias towards the dominant class, leading to more accurate evaluation metrics such as precision, recall, and F1-score (Kuhn & Johnson, 2013). As shown in Figure 3.8, downsampling balances the distribution of clicks compared to Figure 3.7.

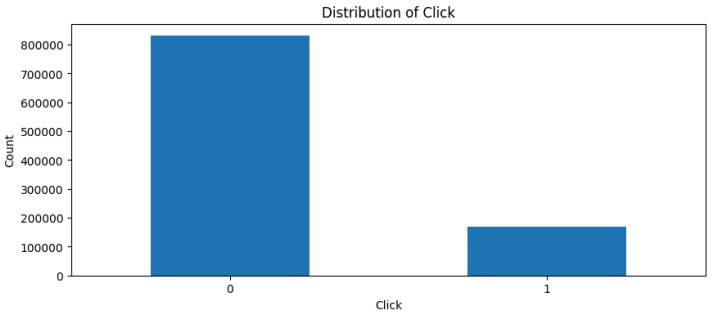


Figure 3.7: Imbalance Distribution of Click

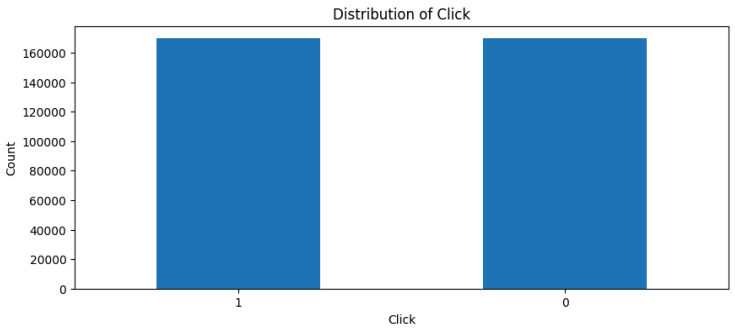


Figure 3.8: Balance Distribution of Click

4. Methodology

In this section, the employed algorithms and metrics for performance evaluation are described.

4.1. Models

In this study, the following model were chosen as relevant for the CTR prediction: Logistic Regression, Random Forest, XG-Boost, CatBoost and FeedForward Neural Network. Furthermore, a more advanced approach using cluster-specific training is employed.

Logistic Regression

Logistic Regression was chosen as a base model for this study, which was employed in several studies as reviewed in the literature. Logistic Regression models the probability that an observation belongs to one of two classes $p\,(y = 1|\,x)$ as a logistic function of a linear combination of the input features. Specifically, the probability that an input x belongs to the positive class is given by the function

$$g(x) = \frac{1}{1 + e^{\theta^T x}}$$

where θ is the parameter vector of the model. The logistic function $g(x)$ maps any real number to the (0, 1) interval, making it suitable for probability estimation. The model parameters θ are usually estimated using maximum likelihood estimation, which seeks the parameter values that maximize the likelihood of the observed data given the model (Lindholm et al., 2022).

Random Forest

Random forest is an ensemble learning method that utilizes multiple decision trees to enhance prediction accuracy and manage over-fitting. Each tree in the forest is constructed from a randomly selected subset of training data and features. This approach reduces model variance and enhances overall model reliability (James et al., 2021). Additionally, the effectiveness of random forests extends across various predictive modeling scenarios because of their ability to aggregate outcomes. For classification tasks, it uses majority voting of the outcomes (Kuhn & Johnson, 2013). For optimizing a random forest model, hyperparameter tuning plays a crucial role. Essential parameters include the number of trees, the number of

features considered at each split, the maximum depth of each tree, and the minimum number of samples required at a leaf node. Adjusting these hyperparameters aids in fine-tuning the model's complexity and predictive accuracy, ensuring it captures the essential patterns without fitting too closely to the noise in the training data (Lindholm et al., 2022).

XG-Boost

Gradient boosting and XGBoost are powerful machine learning techniques that build upon the idea of boosting to improve model predictions, particularly for complex datasets. Gradient boosting is a sequential ensemble technique that builds a series of decision trees, where each tree aims to correct the errors of the previous ones. It begins with a simple model, $F_0(x)$, which is just the mean of the target values. Then iteratively, it adds new trees that predict the residuals or errors of the current ensemble. Mathematically, it adjusts for the errors by walking in the direction that minimizes a loss function, typically using the gradient descent methodology. The update equation is:

$$F_T(x) = F_{t-1}(x) + \eta \cdot h_t(x)$$

where

$F_t(x)$ is the model at step t, η is the learning rate, and $h_t(x)$ is the new tree added at step *t* (James et al., 2021). XG-Boost builds upon the principles of gradient boosting but introduces more formalization and optimization to make the process faster and more effective. It incorporates advanced regularization (L1 and L2), which helps reduce overfitting and improves overall performance. XGBoost also optimizes computational resources by using a more efficient tree construction algorithm and handling sparse data better. The objective function of XGBoost includes a regularization term:

$$Obj = L(\theta) + \Omega(\theta)$$

where $L(\theta)$ is the differentiable loss function and $\Omega(\theta)$ is the regularization term, adding a complexity control over the model (Kuhn & Johnson, 2013; Lindholm et al., 2022).

Differences between Gradient Boosting and XGBoost include regularization—while gradient boosting primarily focuses on reducing error by fitting new predictors to the residual errors, XGBoost also penalizes the model complexity via regularization, helping to avoid

overfitting. XGBoost can handle missing data and variable sparsity through its built-in mechanisms, unlike traditional gradient boosting, which may require complete data. Furthermore, XGBoost is optimized to be more computationally efficient than standard gradient boosting, due to its use of the quantile sketch algorithm for approximate tree learning and more effective use of hardware resources.

CatBoost

CatBoost, a variant of gradient boosting developed by Prokhorenkova et al. (2019), stands out by addressing prediction shifts caused by target leakage—where training data includes information unavailable at prediction time. This issue is common in other gradient boosting frameworks, especially with categorical features. CatBoost improves categorical data handling through an ordered boosting approach, ensuring each model in the ensemble is trained on a distinct subset of data, enhancing robustness and accuracy. It converts categorical features into numerical forms reflecting their statistical relationship with the target variable, more effectively than traditional one-hot encoding. CatBoost also modifies the standard gradient boosting update formula by introducing a sequence of models adjusted based on data subsets to prevent overfitting and leakage. The core update equation in CatBoost can be described as follows:

$$F_T(x) = F_{t-1}(x) + \alpha \sum_{i=1}^{n} g_{t,i}(x)$$

where $g_{t,i}(x)$ is the gradient of the loss function with respect to the function estimate from the previous iteration, and α is the learning rate.

Moreover, CatBoost not only addresses the inefficiencies of handling high cardinality categorical features but also enhances model performance by reducing overfitting through its sophisticated data sampling and handling strategies. As a result, CatBoost outperforms other gradient boosting methods in various datasets by effectively managing categorical features and mitigating prediction shift, making it particularly advantageous in scenarios where categorical data is prevalent (Prokhorenkova et al., 2019). Additionally, for effects of this study, since CatBoost does not require categorical variables to be encoded, it will be tested both with and without encoding these variables.

Feed-Forward Neural Network (NN)

Feedforward neural networks, also known as multilayer perceptrons (MLPs), are the quintessential deep learning models. The architecture of these networks is based on a series of layers, where each layer consists of a set of neurons, and each neuron in one layer connects forward to the neurons of the subsequent layer, with no backward or lateral connections—hence the term "feedforward." As described by Goodfellow, Bengio, and Courville (2016), these connections are typically weighted, and data processing occurs in two stages through these connections: a linear summation followed by a nonlinear activation function. This process begins with the input layer, progresses through one or more "hidden" layers, and concludes at the output layer. Each hidden layer's activation function, such as sigmoid or ReLU, introduces non-linear properties that allow MLPs to learn complex functions. The training of these networks involves adjusting the weights of the connections to minimize the difference between the actual output and the target output, a process typically done through backpropagation. Goodfellow et al. (2016) detail that backpropagation efficiently computes the gradient of the loss function associated with a given state of the network by propagating error gradients backward through the network. This allows for efficient optimization of the loss function using algorithms like stochastic gradient descent. In the study, this basic deep learning approach will be tested as part of the experimentation algorithms to compare the results with the machine learning algorithms.

K-Means

Upon exploring the dataset, it was observed that the data points naturally group into clusters, as illustrated in Figure 4.1, which displays the clustering using two principal components. Consequently, a viable experimental approach is to apply K-means clustering to the training set. This clustering algorithm is widely utilized for dividing observations into a predetermined number of groups, denoted as 'k'. It aims to categorize observations into distinct groups where those within the same cluster are highly similar (resulting in high intra-class similarity), while those in different clusters are as dissimilar as possible (leading to low inter-class similarity). Each cluster in k-means is characterized by its centroid, representing the mean of the observation values assigned to that cluster.

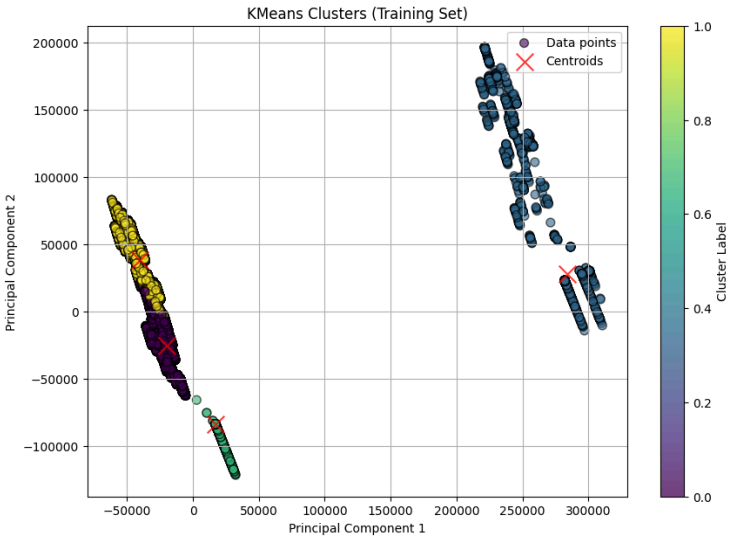


Figure 4.1: Training dataset grouped in clusters

In this setup, the number of clusters is a critical parameter and was determined to be 4 using the Elbow method shown in Figure 4.1. This method determines the optimal number of clusters by plotting the sum of squared distances from each point to its assigned cluster center (inertia) against the number of clusters, and identifying the point where the inertia begins to decrease more slowly, forming an "elbow" shape (Celebi et al., 2013).

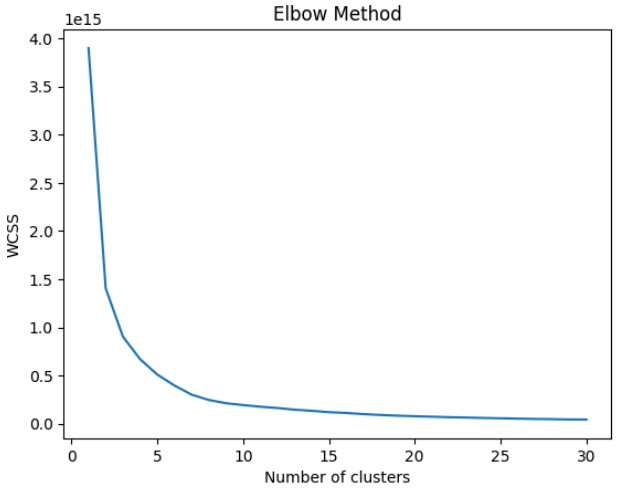


Figure 4.2: Elbow Method to determine optimal number of clusters.

Note: *The x-axis represents the number of clusters, while the y-axis shows the Within-Cluster Sum of Squares (WCSS), which measures the sum of squared distances of samples to their nearest cluster center.*

Once the data was organized into 4 clusters, each cluster is then analyzed using a specific predictive model. Using K-means clustering as a precursor to predictive modeling is an

effective strategy to enhance model performance by tailoring specific models to more homogenous subsets of data. As reviewed in the literature, Kumar et al. (2019) used a clustering approach for CTR prediction obtaining promising results. This targeted approach allows for the application of the most appropriate modeling techniques to different data characteristics.

In this scenario, four distinct models are used to train in each of the clusters: Logistic Regression, Random Forest, XG-Boost and Catboost are trained for each cluster. Therefore, the most suitable model is chosen for each cluster. Finally, when the model is exposed to unseen data, first the datapoint will be assigned to its corresponding cluster, and the prediction will be done taking into account the corresponding trained model for the specific cluster it belongs to.

4.2. Evaluation Metrics

The main metrics to evaluate the performance of the models are Logaritmic loss and F1-Score.

Log-Loss

Log-loss, also known as cross-entropy loss, is an important metric in evaluating machine learning classification algorithms. It is based on prediction probabilities, where a lower log-loss value indicates better predictions (Aggarwal et al., 2021). The metric is extensively used in industries where understanding the model's confidence in its predictions is as important as the predictions themselves (Ferri et al., 2009). It measures the uncertainty of the model's predictions based on how much the predicted probabilities deviate from the actual class labels. It is calculated as the negative log of the probability assigned to the true class for each instance. A lower log loss value indicates a model with better accuracy, as it reflects smaller differences between the predicted probabilities and the actual class labels. Log loss is particularly useful because it penalizes not just incorrect classifications, but also the confidence of the predictions.

F1-Score

The F1-score is a crucial metric for evaluating the performance of classification models, especially in the context of imbalanced datasets. It calculates the harmonic mean of precision and recall, effectively balancing the importance of both metrics. It is calculated as follows

$$F_1 = \frac{2 \,.\, precision \,.\, recall}{precision + recall}$$

where precision measures the accuracy of positive predictions, whereas recall assesses how well the model identifies all relevant instances (i.e., the actual positives). The resulting number is between zero and one, being the higher the better (Lindholm et al., 2022). The F1-score is preferred over the misclassification rate for imbalanced problems, as it considers both precision and recall, offering a balanced metric between the two. In ads click prediction is advantageous because it ensures a balanced evaluation of precision (the proportion of predicted clicks that are actual clicks) and recall (the ability to capture all potential clicks). This metric is significant in advertising, where both avoiding false positives (wasting resources on non-clickers) and maximizing true positives (identifying all potential clickers) are key for optimizing campaign effectiveness and ROI.

5. Empirical Analysis

In this section, the model settings and results are presented and discussed.

5.1. Modeling settings

In the empirical analysis, various models were employed to evaluate their performance in predicting if an ad will be clicked or not. The dataset was initially split into training and testing subsets, maintaining 80% for training and 20% for testing purposes.

Logistic Regression

For the Logistic Regression model, feature scaling was applied using StandardScaler to normalize the data, enhancing the training process. Hyperparameter tuning was conducted through cross-validation using k equal to 5, and targeting optimal regularization strength (C)

and penalty type. The tested parameters included a range of C values from 0.001 to 100. The optimal parameters determined were 'C' equal to 0.1 for L2 Regularization and

Table 5.1: Parameters for Cross-Validation for Logistic Regression

Penalty type	**Hyperparameter**	**Range**	**Best Parameter**
L2	C	[0.0001, 0.001, 0.01, 0.1, 1, 10]	0.1
L1	C	[0.0001, 0.001, 0.01, 0.1, 1, 10]	10

Random Forest

The Random Forest model was configured with a grid search to explore different combinations of the number of trees in the forest (n_estimators), the maximum depth of the trees (max_depth), the minimum number of samples required to split an internal node (min_samples_split), the minimum number of samples required to be at a leaf node (min_samples_leaf), and the number of features to consider when looking for the best split (max_features). After evaluating the configurations using cross-validation of k equal to 5, the optimal parameters were determined to be the ones shown in Table 5.2.

Table 5.2: Parameters for Cross-Validation for Random Forest

Hyperparameter	Range	Best Parameter
n_estimators	[100, 200, 300]	200
max_depth	[None, 10, 20]	None
max_features	['auto', 'sqrt']	sqrt
min_samples_split	[2, 5, 10]	10
min_samples_leaf	[1, 2, 4]	4

XG-Boost

The XGBoost model was configured with a grid search to explore different combinations of the learning rate (learning_rate), the maximum depth of the trees (max_depth), the number of trees in the forest (n_estimators), the fraction of samples used for fitting individual trees (subsample), and the fraction of features used for fitting individual trees (colsample_bytree).

After evaluating the configurations using cross-validation with k equal to 5, the optimal parameters were determined to be the ones shown in Table 5.3.

Table 5.3: Parameters for Cross-Validation for XG-Boost

Hyperparameter	**Range**	**Best Parameter**
learning_rate	[0.01, 0.1, 0.2]	0.1
max_depth	[3, 5, 7, 9]	9
n_estimators	[50, 100, 200]	200
subsample	[0.6, 0.8, 1.0]	0.8
colsample_bytree	[0.6, 0.8, 1.0]	0.8

CatBoost

The CatBoost model, specifically designed to handle categorical features efficiently, was finely tuned through a grid search. The optimization focused on adjusting parameters such as iterations, learning rate, and depth to significantly enhance model performance. After evaluating the configurations using cross-validation with k equal to 5, the optimal parameters were determined to be the ones shown in the following table.

Table 5.4: Parameters for Cross-Validation for CatBoost

Hyperparameter	Range	Best Parameter
depth	[6, 8, 10]	8
iterations	[300, 500]	300
learning_rate	[0.01, 0.05, 0.1]	0.1
l2_leaf_reg	[1, 3, 5, 7]	7

Feedforward Neural Network

The neural network model was configured with a random search to explore different combinations of the weight initializer (RandomNormal, HeNormal, GlorotNormal), the number of units in the first dense layer, the number of units in the second dense layer, the L2 regularization term for the first dense layer, the L2 regularization term for the second dense layer, the optimizer (Adam, RMSprop, SGD), and the learning rate. After evaluating the

configurations using a validation split of 0.1, the optimal parameters were determined to be the ones shown in Table 5.5.

Table 5.5: Parameters for Cross-Validation for Feedforward Neural Network

Hyperparameter	Range	Best Parameter
Weight initializer	['RandomNormal', 'HeNormal', 'GlorotNormal']	RandomNormal
Units in the first dense layer	[32, 64, 96, ..., 512] (step 32)	160
L2 regularization (first layer)	[1e-5, 1e-4, 1e-3, 1e-2] (log scale)	1e-2
Units in the second dense layer	[16, 32, 48, ..., 256] (step 16)	128
L2 regularization (second layer)	[1e-5, 1e-4, 1e-3, 1e-2] (log scale)	1e-4
Optimizer	['adam', 'rmsprop', 'sgd']	adam
Learning rate	[1e-5, 1e-4, 1e-3, 1e-2] (log scale)	1e-4

K-Means Clustering Approach

For the K-Means Clustering approach, prior to model training, the dataset was subjected to clustering into four groups, as explained in the methodology of this study. This preliminary segmentation aimed to uncover inherent groupings within the data, hypothesizing that different groups might exhibit unique behaviors that could affect the effectiveness of the predictive models. In this regard, for each cluster, the models are trained and evaluated using 5-fold cross-validation to determine the best-performing model based on the F1-score as shown in Table 5.6. The best model for each cluster is then used to make predictions on the test data, which will be clustered using the same K-Means model.

Table 5.6: Best Models for Clusters

Cluster	Best Model
0	Random Forest
1	Random Forest
2	CatBoost
3	Random Forest

5.2. Performance Overview

Among the tested models, whose results are shown in Table 5.7, the CatBoost model emerged as the top performer, achieving the lowest log-loss (0.5836) and the highest F1-score (0.7093), indicating a robust ability to predict positive instances with a good balance between precision and recall. In addition, the ensemble methods, including Random Forest and XG-Boost, also showed strong performance, particularly in terms of the True Positive Rate (TPR), with values above 0.73, which suggests their effectiveness in correctly identifying positive cases. The Feed Forward Neural Network, while not surpassing the ensemble models, still demonstrated commendable performance with a TPR of 0.7428 and an F1-score of 0.6924, reflecting its capability to handle complex patterns in data. On the other hand, the K-Means Clustering method displayed a log-loss of 0.6390 and a moderate F1-score (0.6327), showing improved efficacy but still lagging behind the other models.

Table D.1: Model Performance Results

Model	Log-Loss	TPR	FPR	F1-Score
Logistic Regression L2	0.6202	0.7151	0.4053	0.6745
Random Forest	0.5938	0.7329	0.3650	0.6987
XG-Boost	0.5940	0.7362	0.3704	0.6989
CatBoost	0.5836	0.7580	0.3793	0.7093
Feed Forward Neural Network	0.6079	0.7428	0.4029	0.6859
K-Means Clustering	0.6390	0.6171	0.3336	0.6327

Additionally, confusion matrices were plotted and shown in Appendix B. For instance, Figure 5.1 shows the confusion matrix for Logistic Regression using L2 regularization. We can interpret that this model is effective in identifying actual positives, with a TPR of 0.7151 and a relatively low FPR of 0.4053. It also achieves a decent F1-Score of 0.6745, indicating a good balance between precision and recall while maintaining acceptable overall probability prediction accuracy.

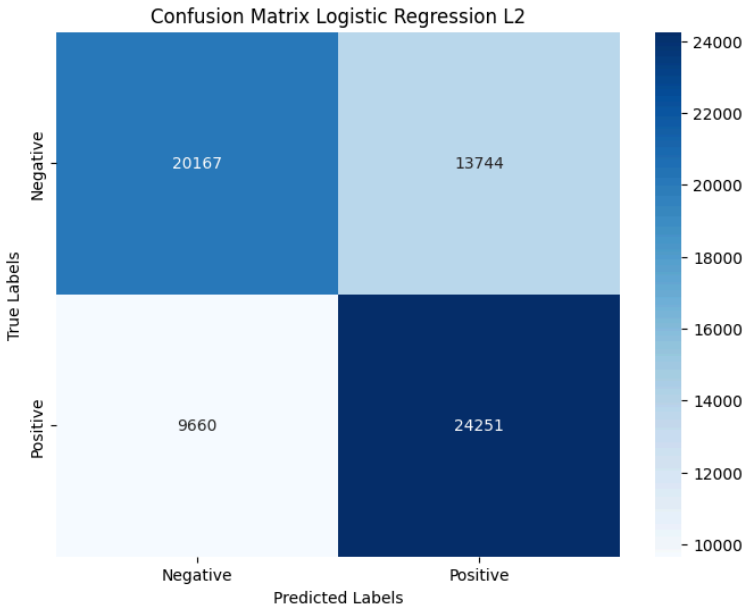


Figure 5.1: Confusion Matrix of Logistic Regression L2

While CatBoost stands out with the lowest log-loss and the highest F1-Score, it also exhibits a significantly higher FPR (0.7580), indicating a tendency to predict non-clicks as clicks more frequently. Despite this, CatBoost maintains a respectable TPR of 0.7580, demonstrating its effectiveness in correctly identifying actual ad clicks. Finally, in case of K-Means Clustering, while not leading in most metrics, presents an interesting profile with the lowest FPR (0.3336) among all models. This suggests that K-Means makes fewer errors in predicting negatives as positives.

5.3. Feature Importance

The feature importance analysis from the CatBoost model, which was the one with best performance, highlights that the features site_id, app_id, device_model, C21, and site_domain, dominate the importance chart, suggesting their critical role in the predictive

performance. The prominence of site_id and app_id underscores the impact of the specific website or application context on user engagement, as different sites and apps inherently attract varying user demographics and interests, which in turn affect CTR. Similarly, device_model serves as a strong predictor, reflecting variations in user experience and interface interaction that can influence ad engagement. The feature C21, potentially representing an ad-related attribute, also plays a vital role in influencing user responses. Additionally, the engineered feature num_previous_clicks, is the eighth most influential feature indicating that a user's propensity to engage with ads based on past interactions might be significant for predicting if a future ad will be clicked.

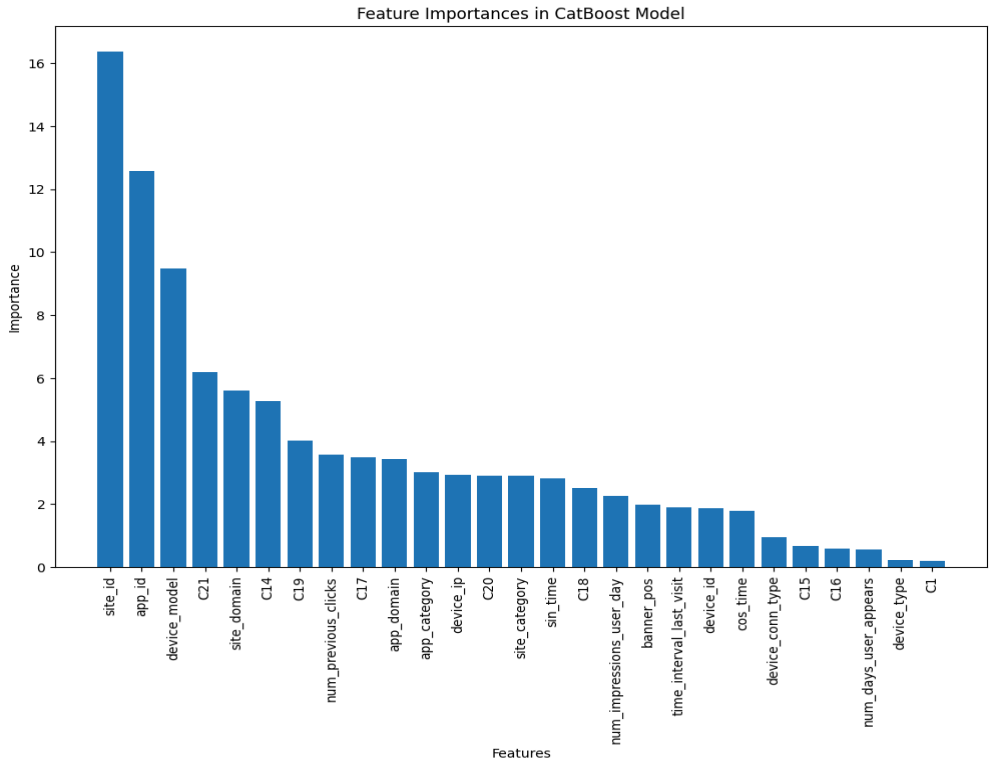


Figure 5.2: Feature Importance Bar Plot

5.4. Discussion

The empirical findings from this study revealed that CatBoost, XG-Boost, and Random Forest demonstrate robust performance. Specifically, CatBoost performed as the superior model, achieving the lowest log-loss (0.5836) and the highest F1-score (0.7093), reflecting its potent capability to predict positive instances accurately while maintaining a balance between precision and recall. Additionally, to evaluate how feature hashing impacts model

performance, CatBoost was tested both with and without encoded categorical variables, as this model can inherently manage categorical data. Interestingly, CatBoost without encoding showed a slightly higher F1-Score (0.6958) compared to when using encoded categorical variables, aligning with Kulkarni's (2022) observations about CatBoost's adeptness at managing categorical features prevalent in advertisement datasets. This ability significantly contributes to its efficiency and accuracy. On the other hand, Random Forest showcased strong performance, particularly in terms of the True Positive Rate (TPR), with a value of 0.7329. This finding is consistent with Sahllal & Souidi (2023), who reported substantial improvements in CTR prediction when using Random Forest. Such effectiveness is crucial in digital advertising for enhancing the allocation and optimization of advertising resources. Moreover, Shi and Li (2016) demonstrated Random Forest's precision in estimating CTR and CPC, reinforcing its suitability for predictive tasks in advertising. XG-Boost also demonstrated notable performance with a TPR of 0.7362 and an F1-score of 0.6989. This supports the findings of Çakmak et al. (2019), who found XGBoost outperformed other models in predicting clicks for advertisements. Similarly, AlAli Moneera et al. (2021) highlighted XGBoost's superior performance compared to Logistic Regression and Random Forest in predicting CTR with reduced computational power and fewer features.

6. Conclusion, Limitations and Future Research

In conclusion, this thesis has demonstrated the potential of machine learning models in predicting click-through rates for mobile digital advertisements. Among the various tested models, CatBoost outperformed the others by achieving the highest F1-score and the lowest log-loss, highlighting its exceptional capability in handling the complexities of categorical data which is very common to handle in digital advertising. Particularly influential in this model were the features site_id, app_id, and device_model. Random Forest also demonstrated notable strengths, particularly in managing data imbalance and achieving high true positive rates. These models notably outperformed not only traditional logistic regression but also other approaches such as XG-Boost and Feed Forward Neural Networks. Furthermore, the study also incorporated a K-Means Clustering approach, which, while showing improved performance over previous iterations, still lagged behind the singular predictive models.

On the other hand, a limitation of this study is related to the size of the dataset utilized. While the original dataset comprised over 40 million rows, a subset of 1 million rows was

utilized for analysis. This downsizing was necessary due to memory constraints, as handling such a large dataset posed computational challenges. Another limitation of this thesis is that many of the variables were anonymized, which can made it challenging for the interpretability and analysis of those variables in order to suggest actionable insights for real contexts.

Finally, future research could explore advanced feature engineering techniques to uncover new insights from user behavior data and investigate the implications of repeated device IDs in the current dataset. Additionally, with increased computational power, a broader range of features can be tested in the feature hashing process to enhance performance. Moreover, employing ensemble techniques to combine the strengths of different models, along with deep learning approaches for high-dimensional data, could further refine the prediction process.

References

Agarwal, A., Gupta, A. & Ahmad, T. (2015). A Comparative Study of Linear Learning Methods in Click-through Rate Prediction, in *2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)*, 2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI), Faridabad, India, October 2015, Faridabad, India: IEEE, pp.97–102, Available Online: http://ieeexplore.ieee.org/document/7489611/

Aggarwal, A. (2021). Label Inference Attacks from Log-Loss Scores

AlAli, M., AlQahtani, M., AlJuried, A., AlOnizan, T., Alboqaytah, D., Aslam, N. & Ullah Khan, I. (2021). Click Through Rate Effectiveness Prediction on Mobile Ads Using Extreme Gradient Boosting, *Computers, Materials & Continua*, vol. 66, no. 2, pp.1681–1696

AppsFlyer. (n.d.). Device ID | AppsFlyer Mobile Glossary, *AppsFlyer*, Available Online: https://www.appsflyer.com/glossary/device-id/

Çakmak, T., Tekin, A., Şenel, Ç., Çoban, T., Uran, Z. & Sakar, C. (2019). Accurate Prediction of Advertisement Clicks Based on Impression and Click-Through Rate Using Extreme Gradient Boosting:, in *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, 8th International Conference on Pattern Recognition Applications and Methods, Prague, Czech Republic, 2019, Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, pp.621–629, Available Online: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007394306210629

Celebi, M. E., Kingravi, H. A. & Vela, P. A. (2013). A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm, *Expert Systems with Applications*, vol. 40, no. 1, pp.200–210

Chen, J.-H., Zhao, Z.-Q., Shi, J.-Y. & Zhao, C. (2017). A New Approach for Mobile Advertising Click-Through Rate Estimation Based on Deep Belief Nets, *Computational Intelligence and Neuroscience*, vol. 2017, pp.1–8

Ferri, C., Hernández-Orallo, J. & Modroiu, R. (2009). An Experimental Comparison of Performance Measures for Classification, *Pattern Recognition Letters*, vol. 30, no. 1, pp.27–38

Fortune Business Insights. (2024). Mobile Advertising Market Size, Share, Trends | Growth [2032], Available Online: https://www.fortunebusinessinsights.com/mobile-advertising-market-102496

Fubel, E., Groll, N. M., Gundlach, P., Han, Q. & Kaiser, M. (2023). Beyond Rankings: Exploring the Impact of SERP Features on Organic Click-through Rates, arXiv:2306.01785, Available Online: http://arxiv.org/abs/2306.01785

Gudipudi, R., Nguyen, S., Bein, D. & Kurwadkar, S. (2023). Improving Internet Advertising Using Click – Through Rate Prediction, 14th International Conference on Applied Human Factors and Ergonomics (AHFE 2023), 2023, Available Online: https://openaccess.cms-conferences.org/publications/book/978-1-958651-70-4/article/978-1-958651-70-4_8

Haans, H., Raassens, N. & Van Hout, R. (2013). Search Engine Advertisements: The Impact of Advertising Statements on Click-through and Conversion Rates, *Marketing Letters*, vol. 24, no. 2, pp.151–163

He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S. & Candela, J. Q. (2014). Practical Lessons from Predicting Clicks on Ads at Facebook, in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, KDD '14: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York NY USA, 24 August 2014, New York NY USA: ACM, pp.1–9, Available Online: https://dl.acm.org/doi/10.1145/2648584.2648589

Hornik, K., Stinchcombe, M. & White, H. (1989). Multilayer Feedforward Networks Are Universal Approximators, *Neural Networks*, vol. 2, no. 5, pp.359–366

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2022). An Introduction to Statistical Learning with Applications in R, *Statistical Theory and Related Fields*, vol. 6, no. 1.

Kaggle. (2014). *Kaggle*, Available Online: https://kaggle.com/competitions/avazu-ctr-prediction

Kuhn, M. & Johnson, K. (2013). Applied Predictive Modeling, [e-book] New York, NY: Springer New York, Available Online: http://link.springer.com/10.1007/978-1-4614-6849-3

Kulkarni, C. S. (2022). Advancing Gradient Boosting: A Comprehensive Evaluation of the CatBoost Algorithm for Predictive Modeling, *Journal of Artificial Intelligence, Machine Learning and Data Science*, vol. 1, no. 5, pp.54–57

Kumar, A., Nayyar, A., Upasani, S. & Arora, A. (2020). Empirical Study of Soft Clustering Technique for Determining Click Through Rate in Online Advertising, in N. Sharma, A. Chakrabarti, & V. E. Balas (eds), *Data Management, Analytics and Innovation*, Vol. 1042, [e-book] Singapore: Springer Singapore, pp.3–13, Available Online: http://link.springer.com/10.1007/978-981-32-9949-8_1

Kumar, R., Naik, S. M., Naik, V. D., Shiralli, S., Sunil V.G & Husain, M. (2015). Predicting Clicks: CTR Estimation of Advertisements Using Logistic Regression Classifier, in *2015 IEEE International Advance Computing Conference (IACC)*, 2015 IEEE International Advance Computing Conference (IACC), Banglore, India, June 2015, Banglore, India: IEEE, pp.1134–1138, Available Online: http://ieeexplore.ieee.org/document/7154880/

Liang, Q., Liu, X., Na, Z., Wang, W., Mu, J. & Zhang, B. (eds). (2020). Communications, Signal Processing, and Systems: Proceedings of the 2018 CSPS Volume III: Systems, Vol. 517, [e-book] Singapore: Springer Singapore, Available Online: http://link.springer.com/10.1007/978-981-13-6508-9

Lindholm, A., Wahlström, N., Lindsten, F. & Schön, T. B. (2022). Machine Learning: A First Course for Engineers and Scientists, 1st edn, [e-book] Cambridge University Press, Available Online: https://www.cambridge.org/highereducation/product/9781108919371/book

Ma, Y. & Zhang, Z. (2020). Travel Mode Choice Prediction Using Deep Neural Networks With Entity Embeddings, *IEEE Access*, vol. 8, pp.64959–64970

Panda, A. R., Rout, S., Narsipuram, M., Pandey, A. & Jena, J. J. (2024). Ad Click-Through Rate Prediction: A Comparative Study of Machine Learning Models, in *2024 International Conference on Emerging Systems and Intelligent Computing (ESIC)*, 2024 International Conference on Emerging Systems and Intelligent Computing (ESIC), Bhubaneswar, India, 9 February 2024, Bhubaneswar, India: IEEE, pp.679–684, Available Online: https://ieeexplore.ieee.org/document/10481562/

Potdar, K., S., T. & D., C. (2017). A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers, *International Journal of Computer Applications*, vol. 175, no. 4, pp.7–9

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V. & Gulin, A. (2019). CatBoost: Unbiased Boosting with Categorical Features, arXiv:1706.09516, Available Online: http://arxiv.org/abs/1706.09516

Sahllal, N. & Souidi, E. M. (2023). A Comparative Analysis of Sampling Techniques for Click-Through Rate Prediction in Native Advertising, *IEEE Access*, vol. 11, pp.24511–24526

Shah, A. & Nasnodkar, S. (2021). The Impacts of User Experience Metrics on Click-Through Rate (CTR) in Digital Advertising: A Machine Learning Approach

Shi, L. & Li, B. (2016). Predict the Click-Through Rate and Average Cost Per Click for Keywords Using Machine Learning Methodologies

Statista. (2024). Global Smartphone Penetration 2016-2022, *Statista*, Available Online: https://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita-since-2005/

Time-Related Feature Engineering. (2024). *Scikit-Learn*, Available Online: https://scikit-learn/stable/auto_examples/applications/plot_cyclical_feature_engineering.html

Wah, Y. B., Rahman, H. A. A., He, H. & Bulgiba, A. (2016). Handling Imbalanced Dataset Using SVM and K-NN Approach, ADVANCES IN INDUSTRIAL AND APPLIED MATHEMATICS: Proceedings of 23rd Malaysian National Symposium of Mathematical Sciences (SKSM23), Johor Bahru, Malaysia, 2016, Johor Bahru, Malaysia, p.020023, Available Online: https://pubs.aip.org/aip/acp/article/586631

Wang, X. (2020). A Survey of Online Advertising Click-Through Rate Prediction Models, in *2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA)*, 2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA), Chongqing, China, 6 November 2020, Chongqing, China: IEEE, pp.516–521, Available Online: https://ieeexplore.ieee.org/document/9277337/

Weinberger, K., Dasgupta, A., Langford, J., Smola, A. & Attenberg, J. (2009). Feature Hashing for Large Scale Multitask Learning, in *Proceedings of the 26th Annual*

International Conference on Machine Learning, ICML '09: The 26th Annual International Conference on Machine Learning Held in Conjunction with the 2007 International Conference on Inductive Logic Programming, Montreal Quebec Canada, 14 June 2009, Montreal Quebec Canada: ACM, pp.1113–1120, Available Online: https://dl.acm.org/doi/10.1145/1553374.1553516

Yang, Y. & Zhai, P. (2022). Click-through Rate Prediction in Online Advertising: A Literature Review, *Information Processing & Management*, vol. 59, no. 2, p.102853

Yi, J. & Chang, B. (2021). Efficient Click-Through Rate Prediction for Developing Countries via Tabular Learning, arXiv:2104.07553, Available Online: http://arxiv.org/abs/2104.07553

Zeng, G. (2023). On the Analytical Properties of Category Encodings in Logistic Regression, *Communications in Statistics - Theory and Methods*, vol. 52, no. 6, pp.1870–1887

Zhang, Y., Dai, H., Xu, C., Feng, J., Wang, T., Bian, J., Wang, B. & Liu, T.-Y. (2014). Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks, arXiv:1404.5772, Available Online: http://arxiv.org/abs/1404.5772

Zhou, S. (2022). Analyzing Factors of Users Click Behavior on Ads Based on Logistic Regression and Machine Learning, in G. Ali, M. C. Birkök, & I. A. Khan (eds), *Proceedings of the 2022 6th International Seminar on Education, Management and Social Sciences (ISEMSS 2022)*, Vol. 687, [e-book] Paris: Atlantis Press SARL, pp.2538–2549, Available Online: https://www.atlantis-press.com/doi/10.2991/978-2-494069-31-2_299

Appendix A

CTR Distribution across variables

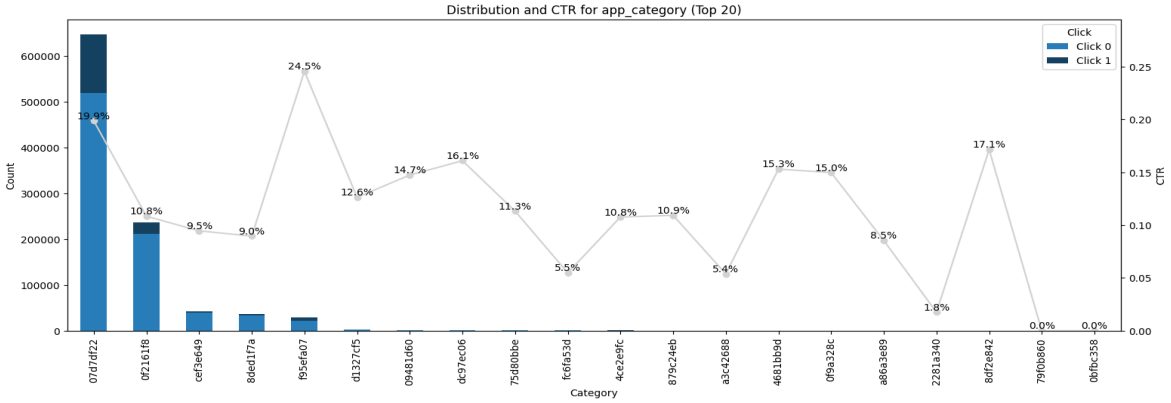


Figure B.1: Distribution of variable app_category

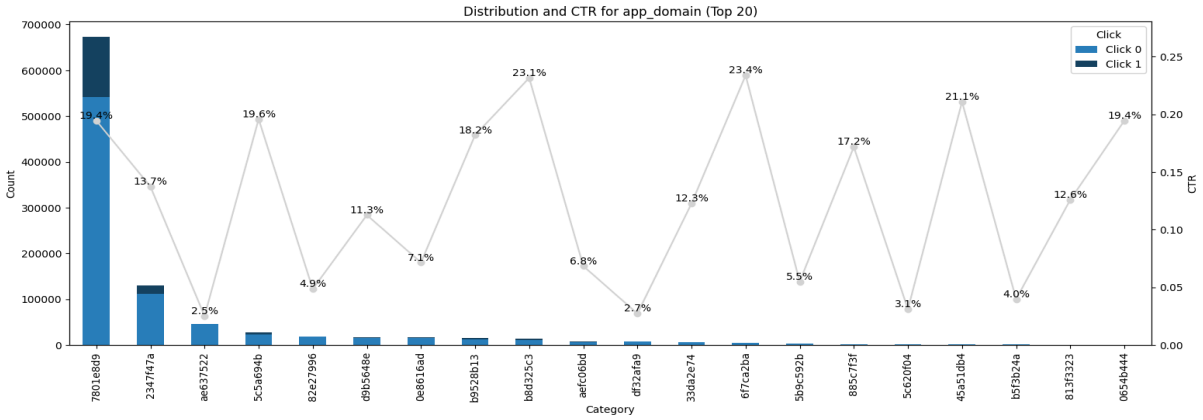


Figure B.2: Distribution of variable app_domain

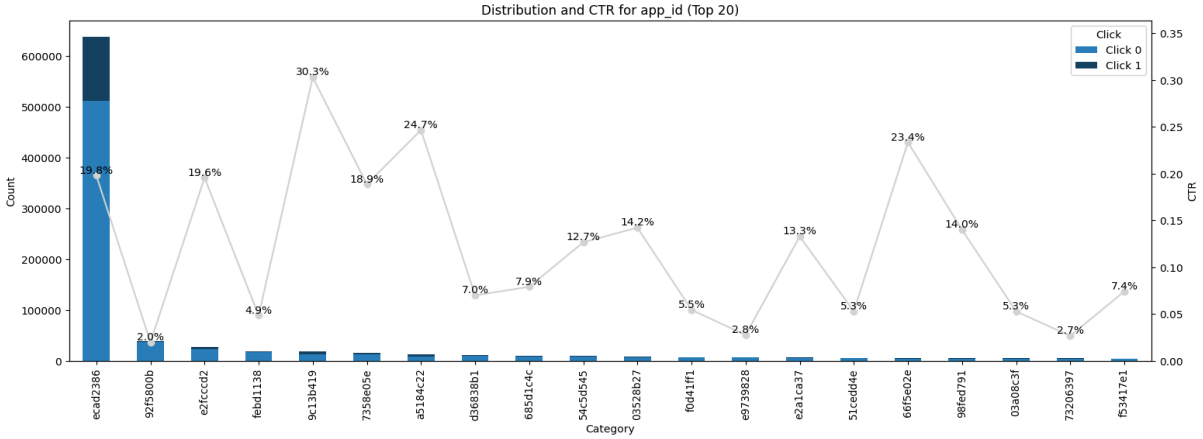


Figure B.3: Distribution of variable app_id

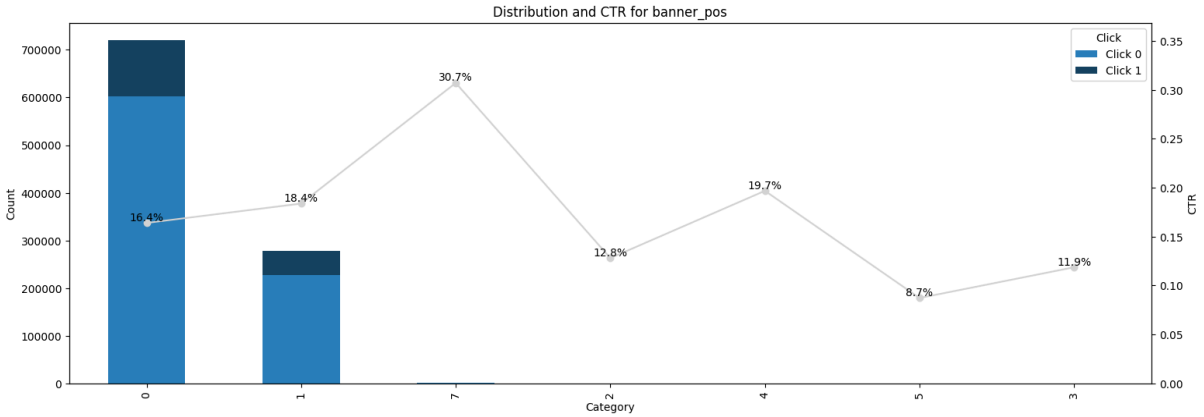


Figure B.4: Distribution of variable banner_pos

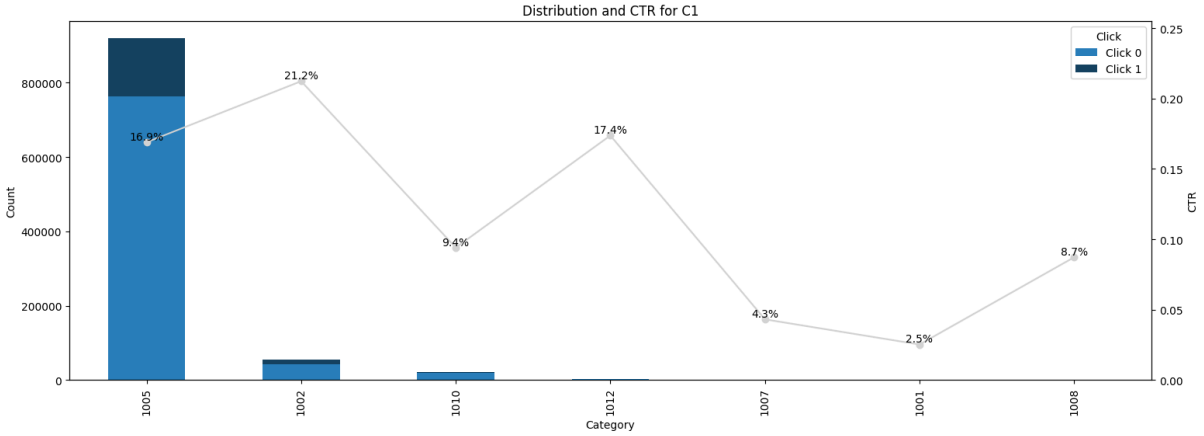


Figure B.5: Distribution of variable C1

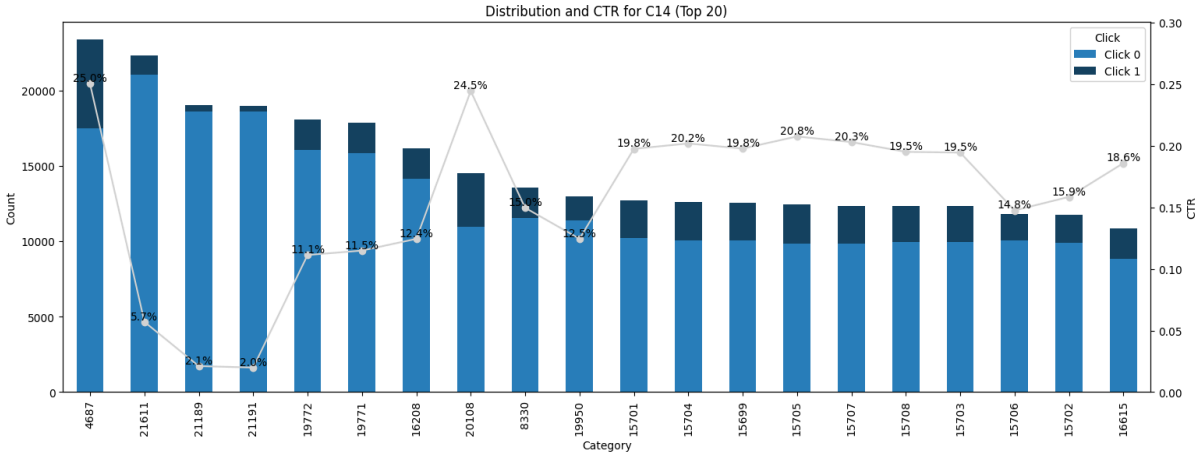


Figure B.6: Distribution of variable C14

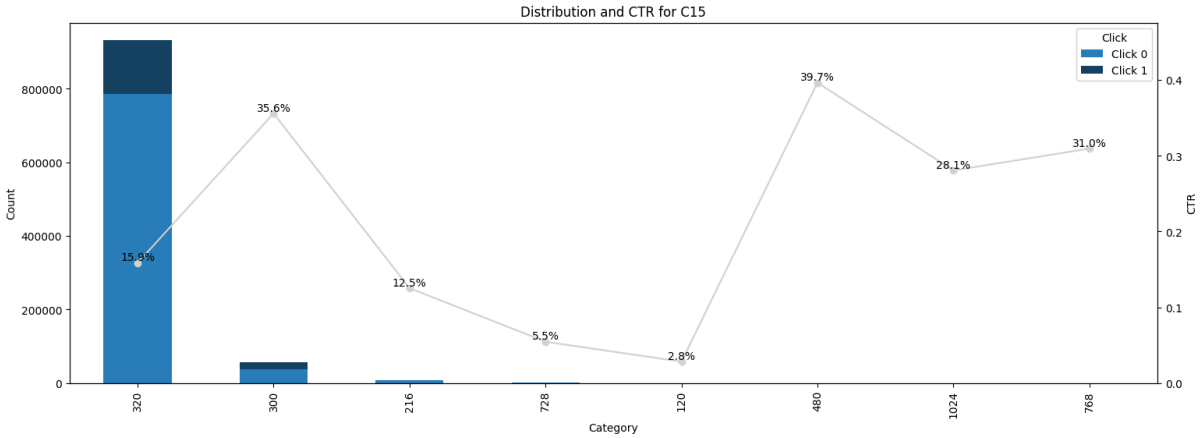


Figure B.7: Distribution of variable C15

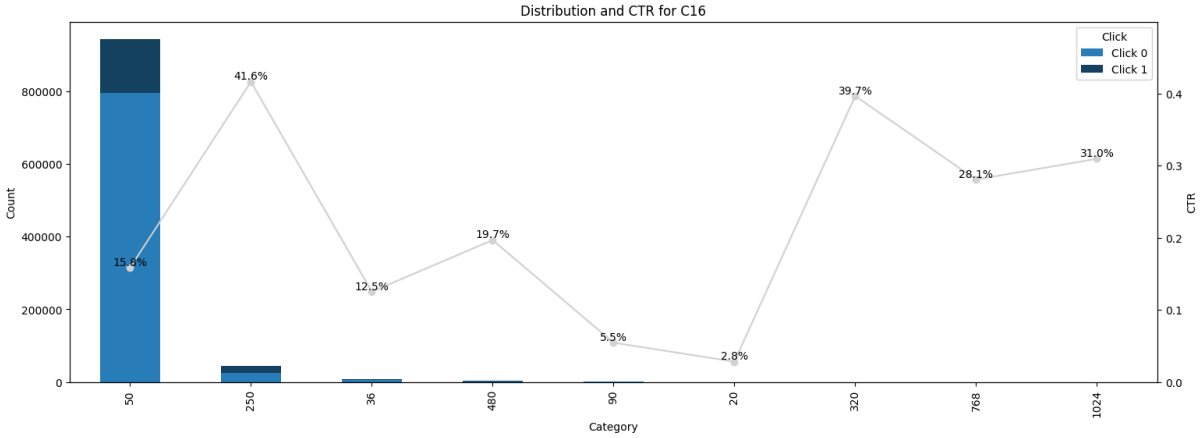


Figure B.8: Distribution of variable C16

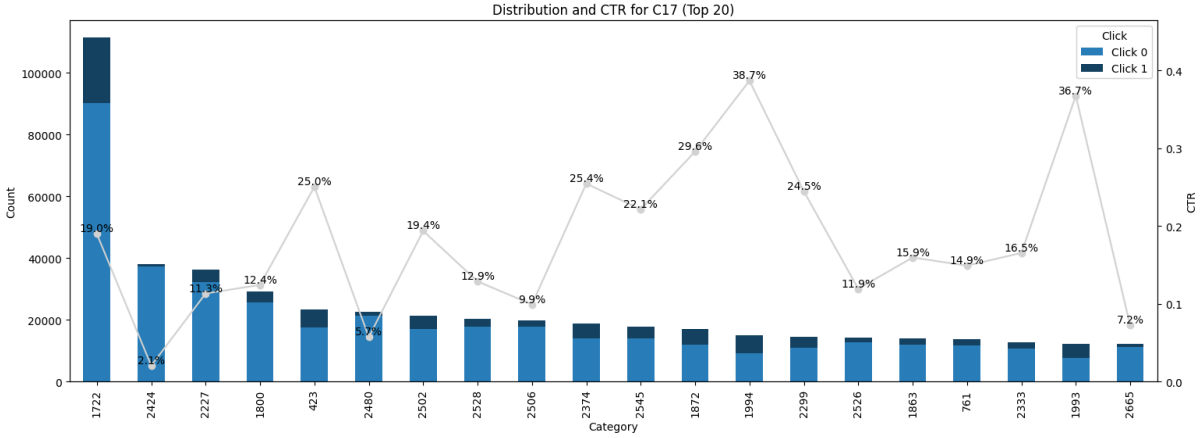


Figure B.9: Distribution of variable C17

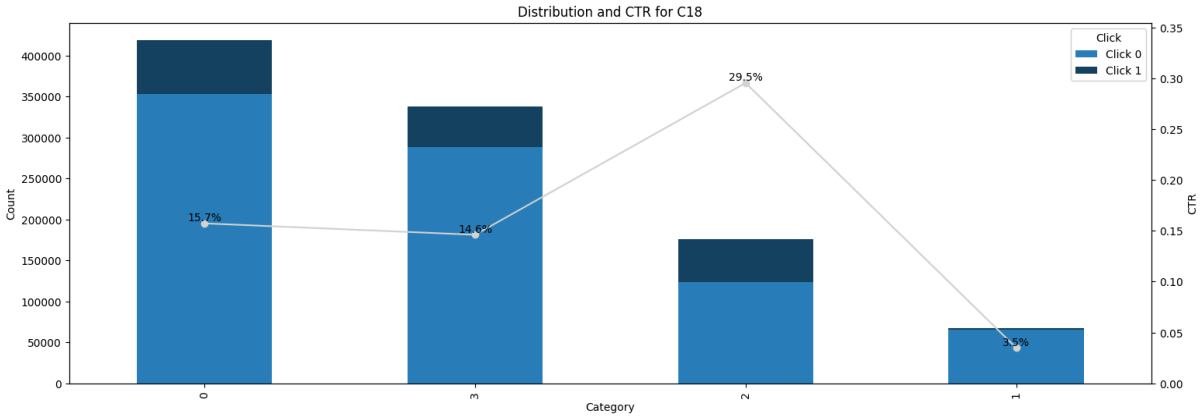


Figure B.10: Distribution of variable C18

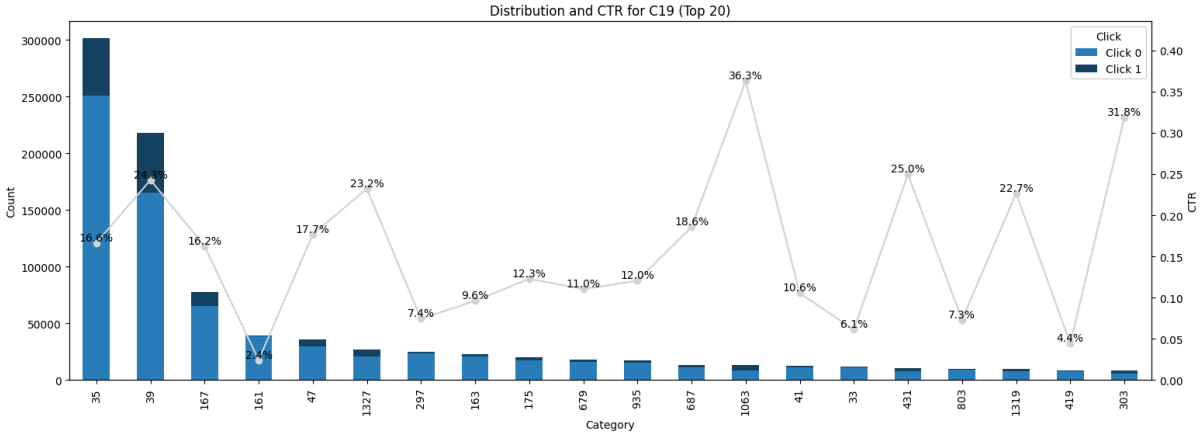


Figure B.11: Distribution of variable C19

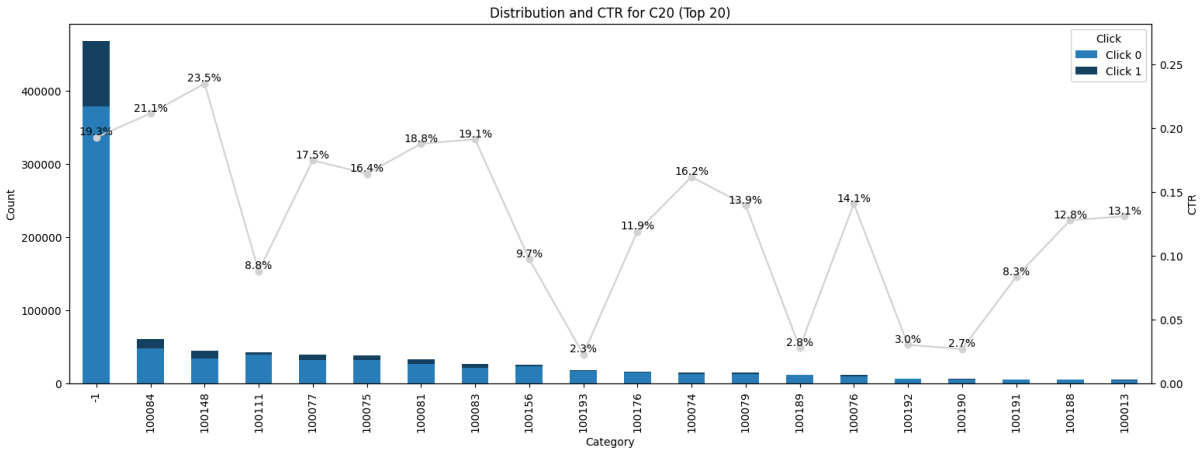


Figure B.12: Distribution of variable C20

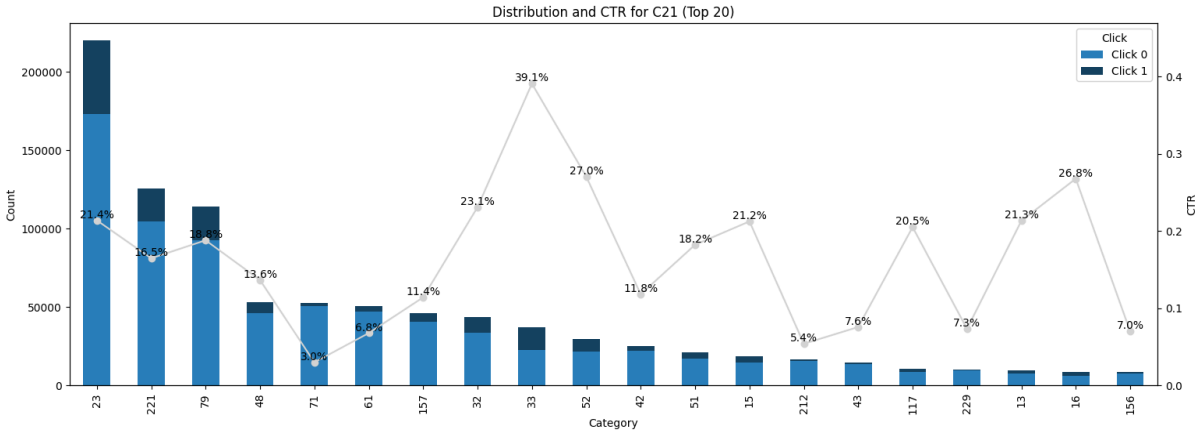


Figure B.13: Distribution of variable C21

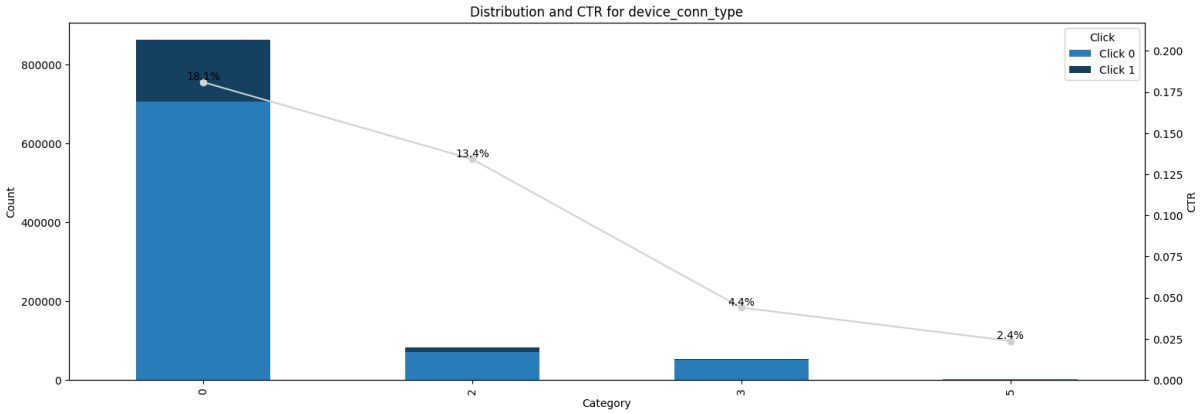


Figure B.14: Distribution of variable device_conn_type

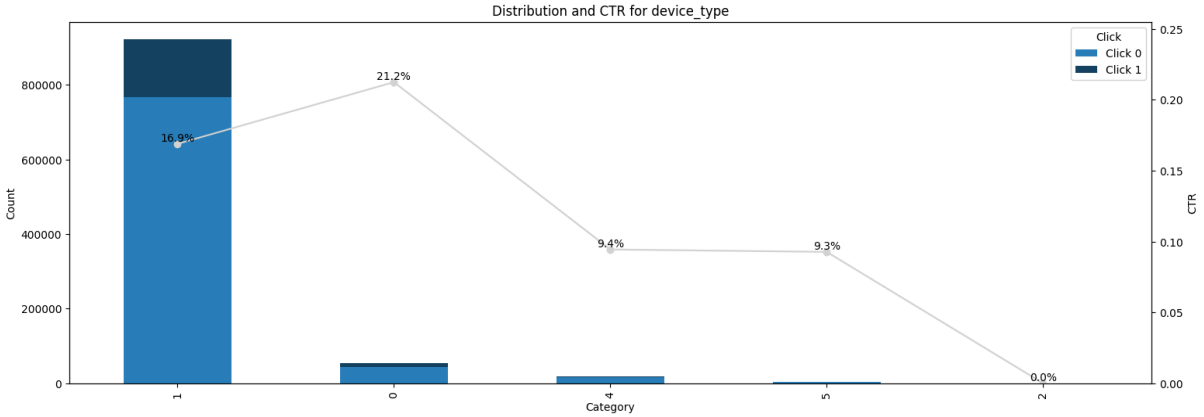


Figure B.15: Distribution of variable device_type

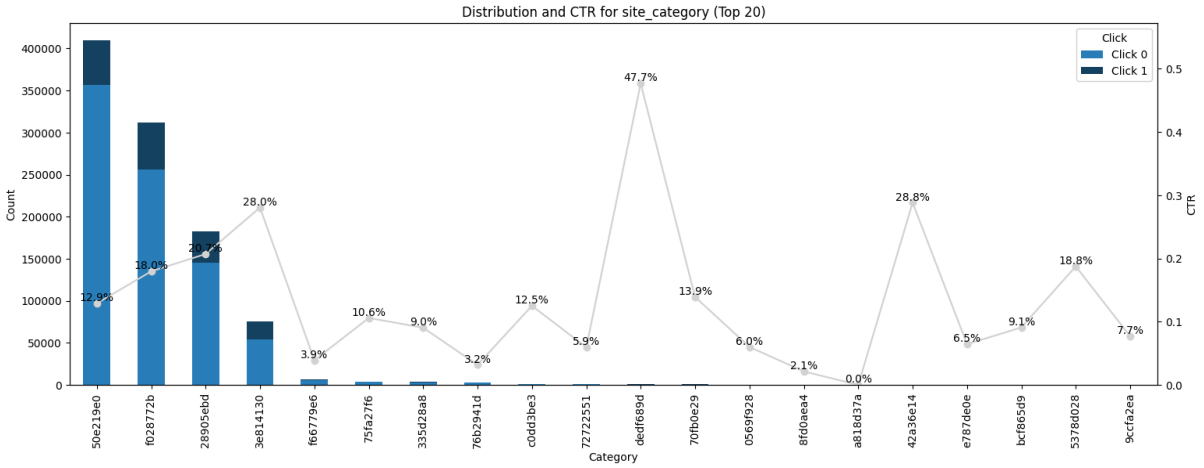


Figure B.16: Distribution of variable site_category

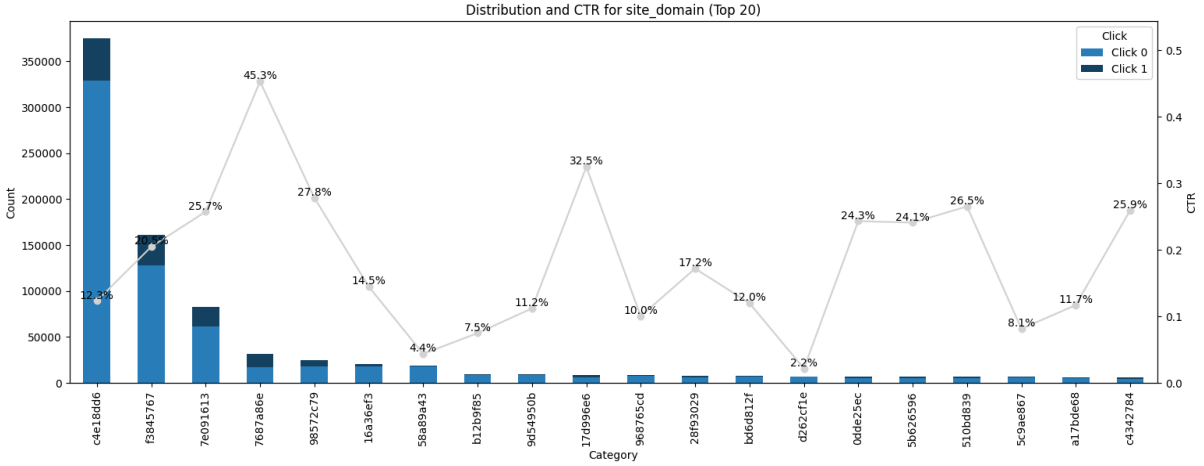


Figure B.17: Distribution of variable site_domain

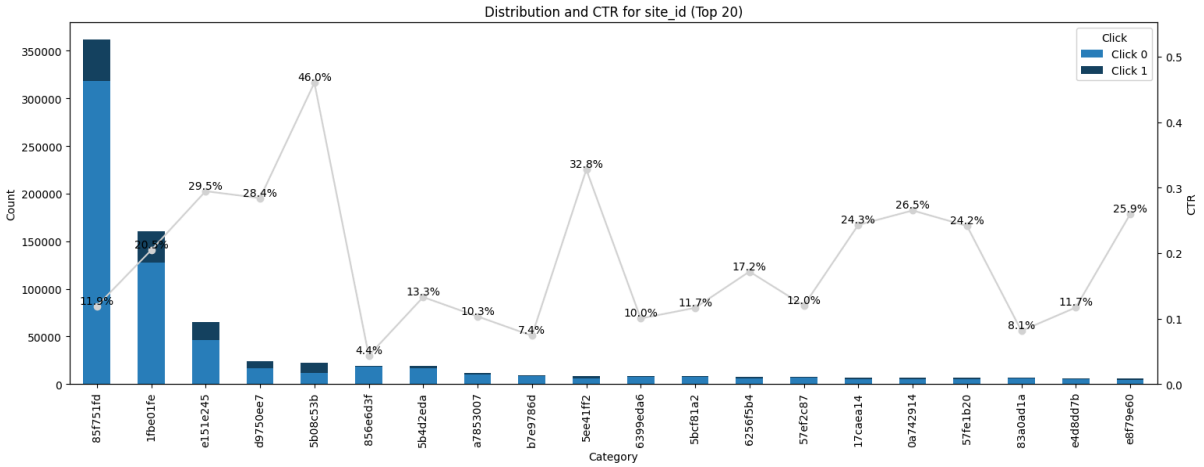


Figure B.18: Distribution of variable site_id

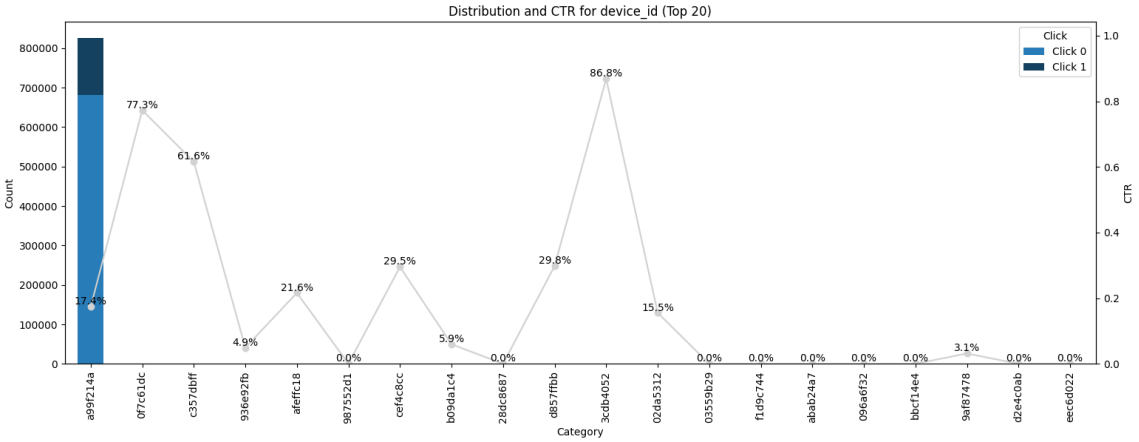


Figure B.19: Distribution of variable device_id

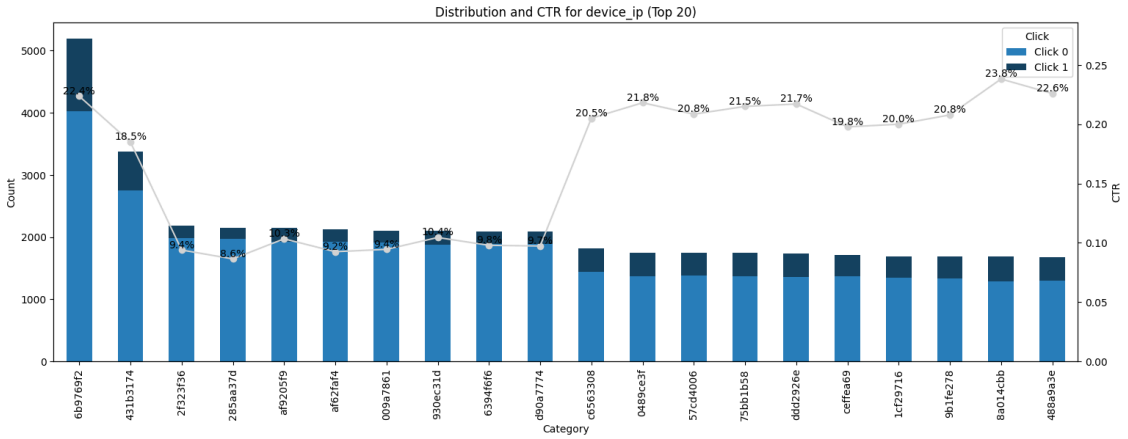
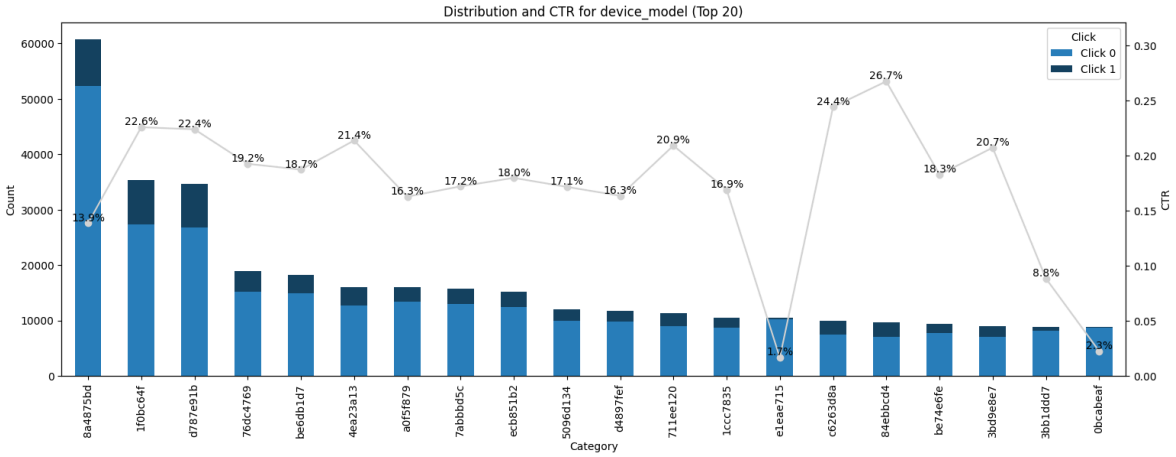


Figure B.20: Distribution of variable device_ip



Appendix B

Resulting Confusion Matrices

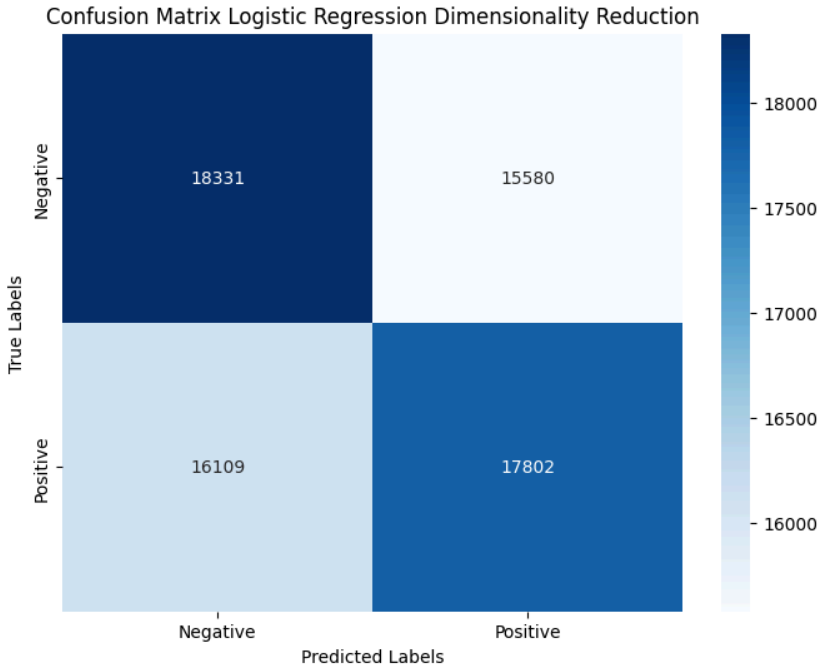


Figure E.1: Confusion Matrix Logistic Regression Dimension Reduction

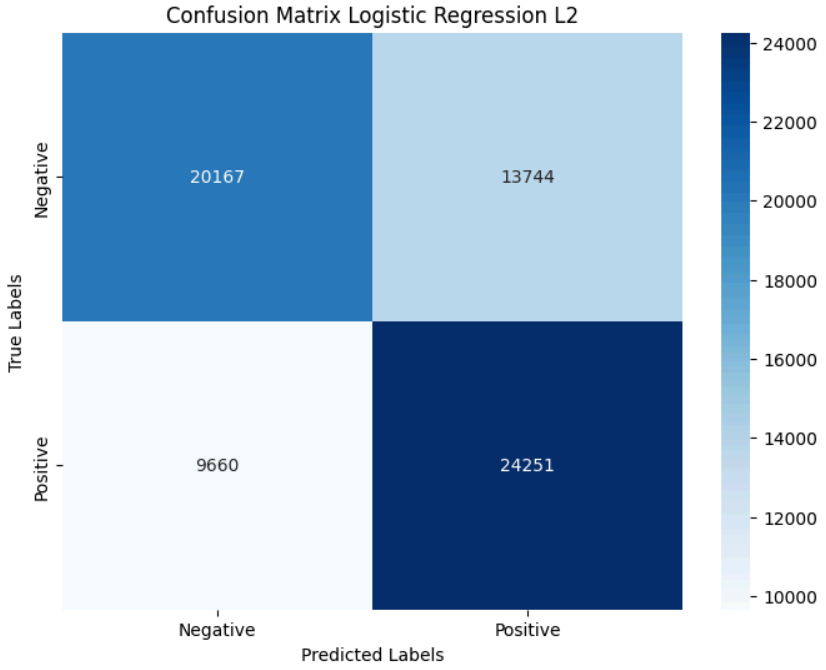


Figure E.2: Confusion Matrix Logistic Regression L2

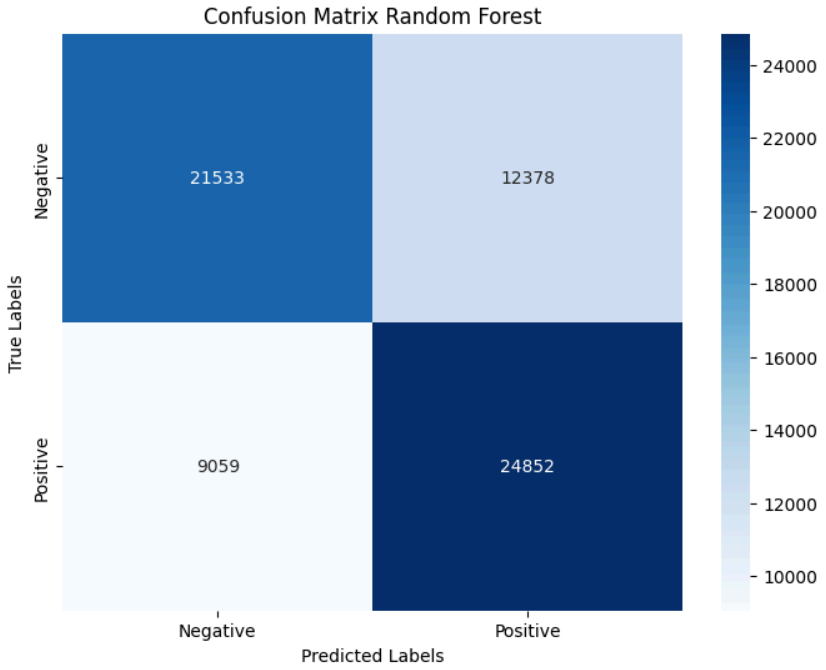


Figure E.3: Confusion Matrix Random Forest

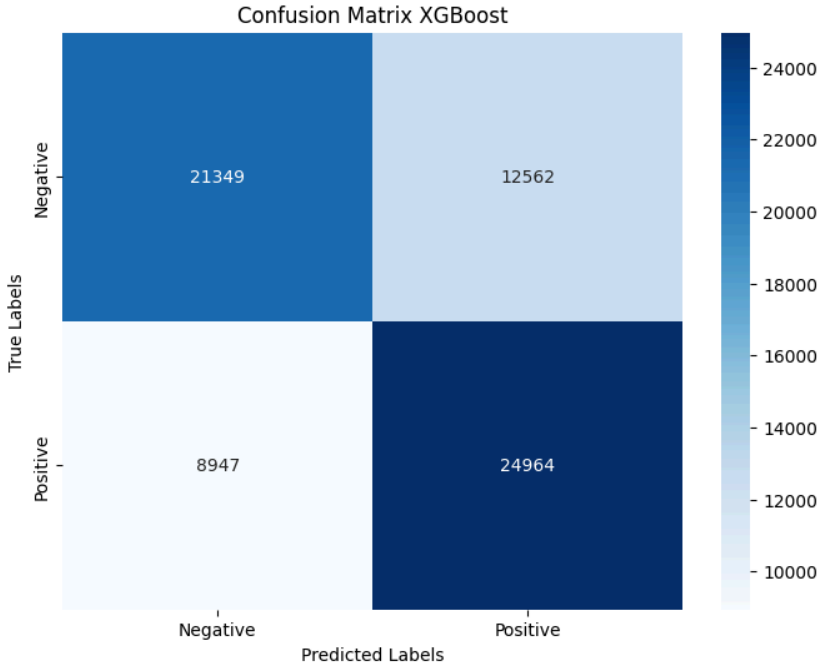


Figure E.4: Confusion Matrix XGBoost

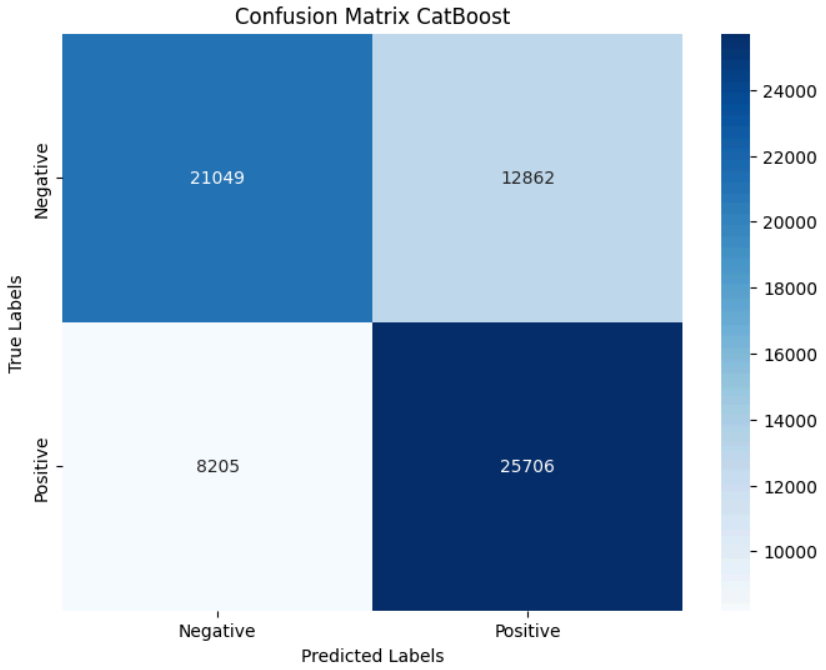


Figure E.5: Confusion Matrix CatBoost

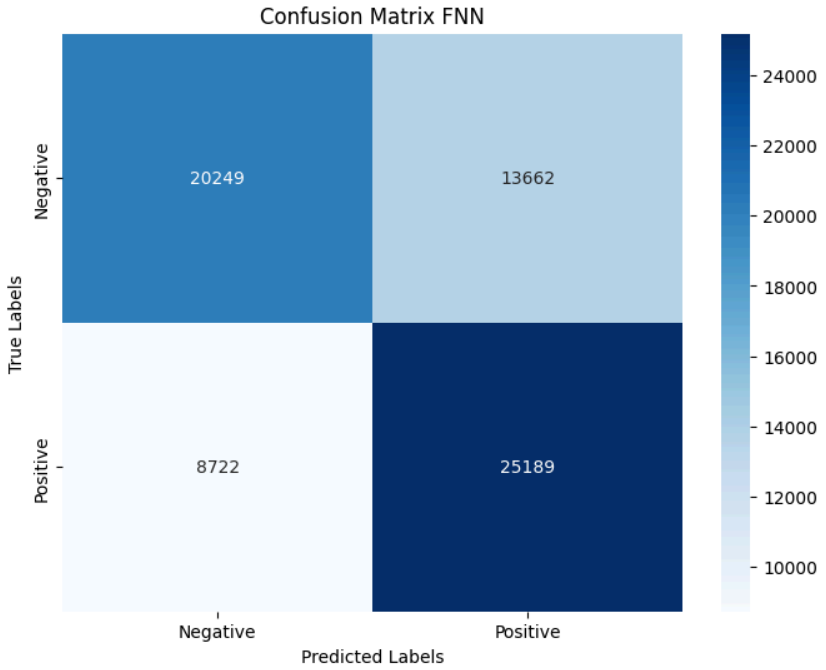


Figure E.6: Confusion Matrix Feedforward Neural Network

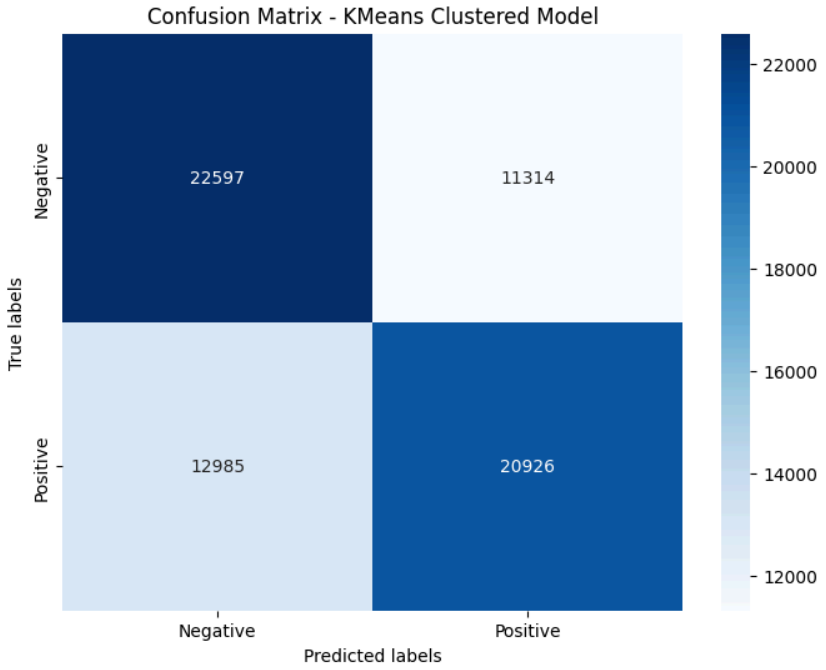


Figure E.7: Confusion Matrix KMeans