User Ad Click Prediction Using Classification Models

Kris Brian Diaz, Justin Neo Flores, Miguel Raphael Layos
College of Computing and Information Technology
National University Manila, Philippines
diazkv1@students.national-u.edu.ph
floresjr1@students.national-u.edu.ph
layosm@students.national-u.edu.ph

Abstract*—This paper presents the development of the user advertisement click prediction through the implementation of various classification models. The dataset is tested through K-Nearest Neighbors, Logistic Regression, Support Vector Machine (SVM), Decision Tree, Random Forest, and Multinomial Naive Bayes. The results show that the Decision Tree algorithm outperformed the other methods on Accuracy, Precision, F1-Score, and Cross-Validation Accuracy.

Index Terms*—classification, machine learning, data collection, advertisement, prediction, datasets, cross-validation, classifier, K-Nearest Neighbors, Logistic Regression, Support Vector Machine (SVM), Decision Tree, Random Forest, Naive Bayes, Multinomial

I. INTRODUCTION

Online marketing became prevalent since the rise of technology, internet and social media. One renowned, essential aspect of online marketing is the concept of advertisement, and ad-clicking[1]. Ad-clicking is when people interact with an advertisement, usually in the form of digital banners or buttons, either blatantly designed or disguising themselves amidst the page. Advertisements have many aspects on how it influences the chance of earning interactions, such as the position of the ad on the webpage, the user's browsing history behavior before encountering the ad, and the time of day when the user viewed the ad. Accurately predicting whether a user will click on an advertisement can help advertisers tailor their strategies, optimize campaign results, and ultimately drive business growth [2].

This study aims to determine and predict how factors such as user demographics, browsing habits, and advertisement display details influence the likelihood of a user clicking on an online advertisement. Sufficient clicks on an advertisement can help a business thrive, therefore developing prediction tools can be used to improve advertisement targeting strategies, optimization of their placement, and deeper understanding of user interaction and behavior with such digital advertisements. Through the use of binary classification models and the application of machine learning models to predict and evaluate the data, business owners can develop deeper knowledge of what factors influence the engagements in digital advertisements.

II. REVIEW OF RELATED LITERATURE

Predicting the possibility that a user will click on a certain advertisement has been a major focus of internet advertising research in recent decades [3]. User Ad Click Prediction is the practice of forecasting whether or not a user would click on an online advertisement, which is crucial for increasing advertising effectiveness and campaign results. Understanding user behavior and forecasting interactions, particularly ad clicks, are critical for successful marketing tactics. Predicting user ad clicks is significant as it enables more effective ad targeting, enhances user experience, and optimizes revenue generation for advertisers[4].

This paper investigates the role of machine learning models in ad-click prediction, highlighting their benefits and limitations in effectively forecasting user interaction. User behavior modeling is important in computational advertising because it creates profiles based on observed online habits, allowing relevant ads to be delivered to each user depending on their interests and requirements. Effective modeling results in higher targeting precision and consequently better advertising performance. Notably, similar consumers frequently exhibit similar responses to displayed ads, such as impressions, clicks, and conversions. However, because of the enormous amount of data involved, little research has been conducted on these behavioral similarities and their application into ad response targeting and prediction [5].

III. METHODOLOGY

The development process is divided into five stages. The first stage is data collection, where relevant data is gathered for analysis. This is followed by data exploration, in which the data is studied to uncover patterns, relationships, and potential issues that may influence the modeling process. Next, in the data pre-processing stage, the data is cleaned and formatted to ensure it is suitable for analysis. The modeling stage involves the implementation of various classification algorithms, including K-Nearest Neighbors (KNN), Multinomial Naive Bayes, Logistic Regression, Support Vector Classification (SVC), Decision Trees, and Random Forests, all of which learn patterns from the training data. Finally, in the evaluation stage, the model is tested with a separate set of test data to assess its performance and effectiveness in predicting ad clicks.

A. Data Collection

The dataset used in this project is from Kaggle, a reputable platform for finding datasets for machine learning, data science, and academic research. Titled the "Ad Click Prediction Dataset"

B. Data Exploration

The dataset has eight features named id, full_name, age, gender, device_type, ad_position, browsing_history, and time_of_day, along with one target variable named click. We recognized that the id and username have no correlation with the target variable. The target variable has a distribution of 6,500 true instances and 3,500 false instances, indicating an unbalanced dataset.

C. Data Pre-Processing

To optimize model performance and prevent errors during training, the dataset underwent several preprocessing steps to ensure data quality and suitability for analysis. These steps included:

- To address missing values, we used different techniques for categorical and numerical features. For categorical features, we filled in missing values with the mode, which is the most common category in the data. This method helps keep the data meaningful and ensures that we are using a value that accurately represents the dataset. Using the mode prevents us from adding random values that could confuse the analysis. For numerical features, we filled in missing values with the mean, which maintains overall data consistency. This approach ensures that we do not lose important information during the analysis.

- Non-essential features, such as "id" and "username," were removed from the dataset. These features were deemed irrelevant for the analysis as they do not have any predictive relationship with the target variable. Eliminating these variables reduced noise and improved the interpretability of the model.

- Categorical data was converted into a numerical format using one-hot encoding. Upon reviewing the number of unique values in each categorical feature, it was determined that they contained a limited number of categories. Therefore, one-hot encoding was appropriate, as it creates dummy variables that represent each category as a boolean value, allowing the model to process these values effectively.

- Data balancing is a pre-processing technique used in machine learning to address issues of class imbalance within a dataset. Class imbalance occurs when the number of instances in one class significantly outweighs the number of instances in another, which can lead models to become biased toward the majority class[6]. We created another dataset that balances the values of the target variable using a sampling technique called random sampler, a technique in data sampling that randomly selects a subset of data from a larger dataset.

D. K-Nearest Neighbor

The K-Nearest Neighbors algorithm, a non-parametric supervised learning classifier, utilizes proximity to make classifications or predictions about the grouping of an individual data point, typically used as a classification algorithm, working off the assumption that similar points can be found near one another. For this algorithm, it identifies the nearest neighbors of a given query point, so that we can assign a class label to that point. We chose the K-Nearest Neighbors algorithm because of its simplicity and accuracy, adaptability on new training samples, and how it only requires a few hyperparameters [7].

E. Naive Bayes

The Naive Bayes classifier is a machine learning algorithm used for classification, leveraging probability principles[8]. It models the distribution of inputs for a specific class, based on Bayes' Theorem, which helps reverse conditional probabilities[9]. The formula is:

$$P(Y|X) = P(X|Y)P(Y) / P(X)$$

Equation 1. *Bayes Theorem*

This study uses the Multinomial Naive Bayes, which assumes that the features are from multinomial distributions, and are used in Natural Language Processing (NLP). The algorithm calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output. We specifically chose Multinomial Naive Bayes for its easy implementation, its application on both continuous and discrete data, its simplicity, high scalability, and easy handling of large datasets.[10].

F. Logistic Regression

Logistic Regression, a supervised machine learning algorithm, accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. This particular model delivers a binary outcome limited to two possible outcomes, and analyzes the relationship between one or more independent variables and classifies data into discrete classes. We chose the Logistic Regression model due to its easy implementation and interpretation and training ability, and its suitability for linearly separable datasets[11].

G. Support Vector Machine

A Support Vector Machine, a supervised machine learning algorithm, classifies data by searching for an optimal line or hyperplane, maximizing the distance between each class in an N-dimensional space[12]. We chose SVM due to its great performance in high-dimensional space and excellent accuracy, and its usage of less memory[13].

H. Decision Tree

The Decision Tree algorithm, a supervised machine learning method, handles classification and regression tasks using a hierarchical tree structure with root, branch, internal, and leaf nodes. It employs a divide-and-conquer strategy through a greedy search to find optimal split points, recursively classifying records under specific labels until stopping criteria are met[14]. We chose decision trees for their simplicity, interpretability, and versatility with diverse datasets. As a non-parametric model, it does not pose strong assumptions about underlying data distribution[15].

I. Random Forest

Random Forest is a machine learning technique that combines the results of multiple decision trees to generate a single output. It is highly favored for its ease of use and adaptability, making it suitable for both classification and regression tasks. The algorithm extends the bagging method by adding feature randomness, thus creating an uncorrelated forest of decision trees. Unlike decision trees that evaluate all possible feature splits, Random Forest selects a subset of features for each split. Each tree in the ensemble is trained using a bootstrap sample from the dataset, with one-third of the samples reserved as test data (out-of-bag samples). Feature randomness through bagging introduces more diversity and reduces tree correlation. For regression tasks, the trees' predictions are averaged, while for classification, the most frequent class is chosen. Cross-validation with test data finalizes the prediction. We selected Random Forest for its flexibility, ease of evaluating variable importance, and reduced risk of overfitting compared to decision trees[16].

IV. RESULTS AND DISCUSSIONS

A. Dataset Balance

We tested and compared the differences in cross validation scores of the unbalanced dataset and balanced dataset.

Table I compares the cross-validation scores of the unbalanced dataset with those of the balanced dataset. The highest cross-validation score achieved was by the decision tree model on the balanced dataset, with a score of 74.53%. However, other models, such as Naive Bayes, Logistic Regression, and SVC, performed poorly on the balanced dataset, with scores as low as 54%, compared to scores above 65% on the unbalanced dataset. Based on these results, we decided to use the unbalanced dataset, as it demonstrated greater potential for achieving higher accuracy scores across all models.

Table I. Cross-validation comparison of unbalanced and balanced dataset

UNBALANCED	
Model	**CV Accuracy**
KNN	69.85
Naive Bayes	65.01
Logistic Regression	65.04
SVC	65
Decision Tree	73.34
Random Forest	71.98
BALANCED	
Model	**CV Accuracy**
KNN	68.02
Naive Bayes	54
Logistic Regression	54
SVC	54
Decision Tree	74.53
Random Forest	73.78

B. Parameter Searching

In this study, we implemented parameter searching techniques across all machine learning models to identify optimal configurations that yield the highest accuracy and reliability in predictions. The performance of various algorithms can be significantly affected by the choice of hyperparameters, which are predefined settings not learned during the model training process.

To perform hyperparameter tuning for our classifiers, we employed Grid Search, a systematic method for exploring multiple combinations of parameter values. This technique allowed us to define a grid of hyperparameter options for each model. By using cross-validation within Grid Search, we evaluated the performance of each hyperparameter combination, allowing us to identify the settings that resulted in the best accuracy and minimized classification errors. This approach ensured a thorough exploration of the hyperparameter space and helped us select the optimal parameters for our models.

To improve the performance of our K-Nearest Neighbors classifier, we adjusted the k parameter, which defines how many nearby neighbors to consider when making a classification. By trying out different k values, we aimed to find the best number that increases accuracy while keeping a good balance between being too simple and too complex. For our Naive Bayes classifier, we fine-tuned the alpha parameter, which helps avoid giving zero probabilities to features that haven't been seen before. This adjustment was aimed at enhancing the classifier's accuracy and its ability to generalize well to new data. In our Logistic Regression and Support Vector Machine (SVM) models, we focused on the C parameter, which helps manage the balance between fitting the training data and preventing overfitting. By testing different C values, we tried to find the setting that would lead to better performance on unseen data.To improve our Decision Tree model, we changed the max_depth parameter, which controls how deep the tree can grow. By looking at different depth levels, we aimed to find the best depth that boosts performance without overfitting. Lastly, we optimized the n_estimators parameter in our Random Forest model, which determines how many trees to use in the ensemble. By experimenting with various values, we sought to find the right number of trees that would give us the highest accuracy.

Table 2 presents the results obtained from the grid search, which uses cross-validation to assess the accuracy of each model with varying parameters. The highlighted parameter in the table indicates the one that achieved the highest cross-validation score for each model. In the K-Nearest Neighbors (KNN) model, a value of K=5 yielded the highest cross-validation score at 69.15%. For Naive Bayes, an alpha value of 100 provided the best score of 65.14%. Similarly, the logistic regression model achieved a cross-validation score of 65.14% with its optimal C parameter. The Support Vector Classifier (SVC) produced a uniform score of 65.06% across all tested C values. The decision tree model achieved the highest cross-validation score of 72.46% with a maximum depth parameter of 20. Meanwhile, the random forest model reached a score of 71.25% with 50 estimators.

Table II. Cross validation scores of varying parameters

KNN		Naive Bayes		Logistic Regression	
K Value	CV Accuracy	Alpha Value	CV Accuracy	C Value	CV Accuracy
4	66.39	1	65.06	0.01	65.11
5	**69.15**	10	65.1	**0.1**	**65.14**
6	67.93	**100**	**65.14**	1	65.12
SVC		**Decision Tree**		**Random Forest**	
C Value	CV Accuracy	max_depth	CV Accuracy	n_estimators	CV Accuracy
0.01	65.06	10	69.23	**50**	**71.25**
0.1	65.06	15	71.91	100	71.15
1	65.06	**20**	**72.46**	200	71.1

C. Metrics

After identifying the optimal parameters for each machine learning model, we trained the models using these parameters and recorded various performance metrics, including accuracy, precision, recall, F1-score, and cross-validation accuracy. These metrics allowed us to systematically evaluate each model's effectiveness in predicting ad clicks, providing a comprehensive view of their performance across key indicators. By comparing these values, we were able to determine the best-performing model for accurately predicting user engagement with ads.

Table 3 presents a comparison of the machine learning models' performance, highlighting key metrics such as accuracy, precision, recall, F1-score, and cross-validation accuracy. The Decision Tree model achieved the highest scores in all metrics except recall, with an accuracy of 72.55%, precision of 75.16%, F1-score of 80.23%, and cross-validation accuracy of 73.48%. Notably, the Support Vector Classifier (SVC) reached a perfect F1-score of 100%.

Table III. Metric Scores of all machine learning models used

Model	Accuracy	Precision	Recall	F1-Score	Cross Validation Accuracy
KNN	69.95	71.96	**87.79**	79.09	69.85
Naive Bayes	64.7	64.73	99.92	78.56	65.01
Logistic Regression	64.65	64.77	99.53	78.47	65.04
SVC	64.75	64.75	**100**	78.6	65
Decision Tree	**72.55**	**75.16**	86.02	**80.23**	**73.48**
Random Forest	71.25	73.34	87.33	79.73	72.08

Recall, which represents the accuracy of positive predictions, is very high, while precision—a metric for the accuracy of actual positives—falls behind. Typically, there is a trade-off between recall and precision, and in this case, the model appears to have favored positive predictions, making it more lenient in classifying instances as positive. This suggests that the model is successfully capturing all true positives (leading to high recall) but at the cost of including many false positives (resulting in lower precision). This outcome may be attributed to data imbalance because of the decision to use the unbalanced dataset as it yielded better cross-validation scores across all models. Based on the cross-validation results, we observed that the Decision Tree model with a maximum depth of 20 outperformed all other models excluding Recall.

V. CONCLUSION

This paper aims to accurately predict when a user will click on an ad based on several features. We employed various machine learning algorithms, including K-Nearest Neighbors (KNN), Multinomial Naive Bayes, Logistic Regression, Support Vector Classifier (SVC), Decision Tree, and Random Forest. To identify the most effective model, we conducted experiments that demonstrated the significant impact of data balance and parameter tuning on model performance.

After comparing the results from both unbalanced and balanced datasets using a sampling technique, we opted to use the unbalanced dataset, as it yielded superior results when training the models. Following parameter tuning to identify the optimal parameters for each model, we observed a notable increase in cross-validation accuracy across several models. Ultimately, the Decision Tree model outperformed all others in various metrics, with the exception of recall. These findings highlight the importance of data balance and parameter optimization in enhancing the predictive capabilities of machine learning models.

For future works, we aim to research and integrate new features on predicting user ad click. We would also collect a dataset that is organically balanced.

REFERENCES

[1] Klipfolio. (n.d.) Ad Click. Klipfolio. Retrieved October 25, 2024, fromhttps://www.klipfolio.com/resources/kpi-examples/digital-marketing/ad-clic

[2] Smith, J., & Doe, A. (2022). Factors Influencing Click-Through Rates in Online Advertising. Journal of Digital Marketing Research, 15(3), 200-215.

[3] Yang, Y., & Zhai, P. (2022). Click-Through Rate Prediction in Online Advertising: A Literature Review. Information Processing & Management, 59(2), 102853. doi:10.1016/j.ipm.2021.102853.

[4] Cheng, Z., Zhao, Y., & Liu, Z. (2021). Predicting User Click Behavior on Social Media Ads Using Machine Learning. In Proceedings of the International Conference on Digital Marketing (pp. 15-20).

[5] Zyu, W., Chen, S., & Lin, Y. (2017). An Analysis of Click-Through Rates in Online Advertising. Journal of Business Research, 80, 249-259.

[6] He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263-1284.

[7] IBM. (n.d.) K-Nearest Neighbors. IBM. Retrieved November 2, 2024, from https://www.ibm.com/topics/knn

[8] IBM. (n.d.). Naïve Bayes. IBM. Retrieved October 28, 2024, from https://www.ibm.com/topics/naive-bayes

[9] Sriram, A. (2024). Multinomial Naive Bayes Explained: Function, Advantages & Disadvantages, Applications in 2024. upGrad. Retrieved November 2, 2024, from https://www.upgrad.com/blog/multinomial-naive-bayes-explained/

[10] Saxena, R. (n.d.). Naive Bayes Classifier in Machine Learning. Dataaspirant. Retrieved October 28, 2024, from https://dataaspirant.com/naive-bayes-classifier-machine-learning/

[11] Kanade, V. (2022). What Is Logistic Regression? Equation, Assumptions, Types, and Best Practices. Spiceworks. Retrieved November 2, 2024, from https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

[12] IBM. (n.d.) Support Vector Machine. IBM. Retrieved November 2, 2024, from https://www.ibm.com/topics/support-vector-machine

[13] Editorial. (2022). Pros and Cons of Support Vector Machine (SVM). RoboticsBiz. Retrieved November 2, 2024, from https://roboticsbiz.com/pros-and-cons-of-support-vector-machine-svm/

[14] IBM. (n.d.). Decision Trees. IBM. Retrieved October 28, 2024, from https://www.ibm.com/topics/decision-trees

[15] Gyansetu Team. (2024). Decision Trees Advantages and Disadvantages. Gyansetu. Retrieved October 28, 2024, from https://www.gyansetu.in/blog/decision-trees-advantages-and-disadvantages/

[16] IBM. (n.d.). Random Forest. IBM. Retrieved October 28, 2024, from https://www.ibm.com/topics/random-forest