# ORIE 5750 Applied Machine Learning

Final Project - TalkingData AdTracking Fraud Detection Challenge

**Ying Wang (yw2574), Yan Ru (yr264), Qingrong Yi (qy248)**

**Datasets & Code**

- Resample Train: https://drive.google.com/file/d/1dI0luVNrfrtayvKAoV9LSkHdhs6H7TUq/view?usp=drive_link
- Original Train: https://drive.google.com/file/d/175YzchSOR2d2Ac1izGbXM4FPR7H0YA46/view?usp=sharing
- Test: https://drive.google.com/file/d/1DehwsvPJ5WKrQQQKcCrBb-G09DyY8XQ9/view?usp=drive_link
- Final Project Code: https://colab.research.google.com/drive/1IdDicCepwTpeWWJVMPVWmOAixc4KPrfF?usp=sharing

**Contribution**
All team members are responsible for structuring the entire project, working on the baseline results, and coordinating the project timeline with effective communication. Overall, each team member makes an equal and significant contribution to the success of the project.

Cornell Tech
Dec 15, 2023

# ORIE 5750 Applied Machine Learning Final Project

Ying Wang
yw2574@cornell.edu

Yan Ru
yr264@cornell.edu

Qingrong Yi
qy248@cornell.edu

## Abstract

*The paper presents a comprehensive study on click fraud detection. Various machine learning models, including baseline logistic regression, advanced algorithms like Gradient Boosting, XGBoost, LightGBM, CatBoost, and Random Forest, and ensemble techniques like Stacking Generalization and Majority Vote Ensembling are implemented. Despite the diversity of approaches, the Majority Vote Ensembling model emerged as the top performer. The study also highlights limitations like computational constraints, the impact of dataset size, and challenges in hyperparameter tuning. These insights emphasize the complexity of click fraud detection and the effectiveness of ensemble strategies in tackling it.*

## 1. Introduction

### 1.1. Background and Motivation

According to PwC Entertainment and Media Outlook [17], the global internet advertising revenue reaches \$522 billion in 2023 and is expected to further grow to \$663 billion by 2027. Within this booming industry, the cost-per-click (CPC) model, also known as the pay-per-click (PPC) model, is a prevalent and dominant digital marketing revenue model that allows agencies or platforms such as Google Ads [7] to charge advertisers based on the number of times visitors click on a display ad attached to the advertisers' websites, applications, etc. [6].

With the rise of such digital advertising revenue models, ad fraud has become a significant issue, leading to substantial financial losses for advertisers. According to Wilbur and Zhu (2009) [13], click fraud is the practice of deceptively clicking on search ads with the intention of either increasing third-party website revenues or exhausting an advertiser's budget. To combat such click fraud, we aim to address this issue by developing a machine learning model capable of detecting fraudulent ad clicks with high accuracy in real time, thereby ensuring the integrity of ad tracking data, improving transparency in digital advertising metrics, and protecting digital advertising investments.

To develop such a predictive machine learning model, we turn to the Kaggle competition - TalkingData AdTracking Fraud Detection Challenge [1], which provides relevant data sources to address this issue using advanced data analysis and machine learning techniques. Specifically, the challenge centers around the context of click fraud in China, the largest mobile market in the world with over 1 billion mobile devices in active use every month. As indicated by the challenge, the biggest independent big data service platform in China processes 3 billion clicks daily. Alarmingly, up to 90% of these clicks are suspected to be fraudulent. Therefore, within this setup, we intend to build a machine-learning algorithm that predicts whether a user will download an app after clicking a mobile app ad.

### 1.2. Dataset Description

From the Kaggle competition - TalkingData AdTracking Fraud Detection Challenge [1], we got datasets covering approximately 200 million clicks over 4 days. Specifically, there are 3 main datasets for our predictive analysis. The primary training set, $train.csv$, contains click records integral for modeling, while $train\_sample.csv$ offers a manageable subset of 100,000 randomly selected rows from the training data, facilitating initial data inspection before accessing the entire dataset. These datasets consist of various features crucial for analysis: $ip$ denotes the click's IP address, $app$ signifies the app ID used for marketing, $device$ encompasses the user's mobile device type ID, $os$ indicates the OS version of the user's mobile phone, and $channel$ represents the mobile ad publisher's channel ID. Moreover, $click\_time$ denotes the timestamp of the click in UTC, and $attributed\_time$ indicates the app download time if the user downloads the app post-ad click, while $is\_attributed$ serves as the predictive target, signaling whether the app was downloaded. It's important to note that $ip$, $app$, $device$, $os$, and $channel$ are encoded in the dataset. The test dataset, $test.csv$, holds a similar structure, yet includes distinct features such as $click\_id$, utilized as a reference for prediction, although it excludes the $is\_attributed$ column, which needs to be predicted. These datasets collectively form a comprehensive resource for analysis and prediction in this context.

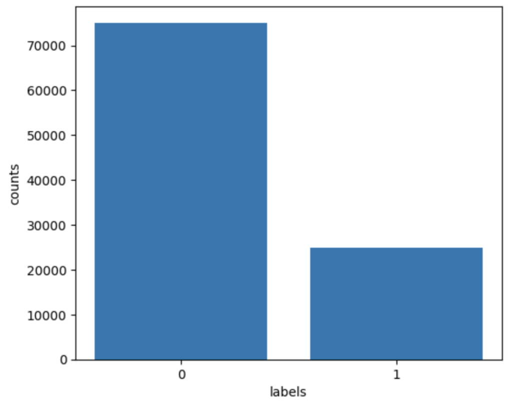

Figure 1. Label Distribution

## 2. Related Work

For this section, we reviewed prior works related to click prediction algorithms. McMahan et al. (2013) [14] discussed a machine learning technique for predicting ad click-through rates (CTR) at Google. They primarily focus on using a large-scale logistic regression model, optimized with the Follow-The-Regularized-Leader (FTRL) Proximal algorithm, which is beneficial for its sparsity and convergence properties. He et al. (2014) [10] proposed a hybrid model that fuses decision trees with logistic regression. This combined approach surpasses the performance of each individual method by over 3%, marking a significant enhancement in overall system efficiency. Iqbal, Zulkernine, Jaafar, and Gu (2016) [12] introduced FCFraud [12], a technique for detecting click fraud on the user's side, specifically targeting automated clickers used in botnets. The technique uses the RandomForest algorithm to classify ad requests and employs a series of heuristics to detect fraudulent ad clicks. Zhang, Liu, and Guo (2018) [21] suggested a click fraud detection scheme that combines a Cost-sensitive Back Propagation Neural Network (CSBPNN) with an Artificial Bee Colony (ABC) algorithm in mobile advertising. Xie et al. (2018) [19] presented a novel Graph-based Tensor Recovery (Graph-TR) model for Internet anomaly detection, which improves the accuracy of detecting traffic anomalies. This model innovatively combines tensor recovery with graph theory, specifically utilizing graph Laplacians to incorporate non-linear proximity information into tensor factorization. Haider, Iqbal, A. H. Rahman, and M. S. Rahman (2018) [9] recommended an approach for detecting impression fraud in mobile advertising, using ensemble learning methods. The model employs decision tree classifiers enhanced with ensemble techniques like Bagging and Boosting and applies oversampling to address data imbalance. Notably, the model also addresses the challenge of high-dimensional feature sets and unbalanced data, employing techniques like SMOTE for balancing the training dataset. Zhu, Zhang, Zheng, and Otaibi (2022) [22] initiated a novel click fraud detection model for online advertising, employing a tensor-based mechanism with Locality-Sensitive Hashing (LSH). This approach aims to enhance click fraud detection in the context of mobile advertising. It leverages the LSH algorithm to effectively identify fraud clicks and improve accuracy and prediction-recall rates. Our review of existing literature has provided valuable insights for the creation of our click prediction algorithm, influencing our choice of models and the specific techniques we employed.

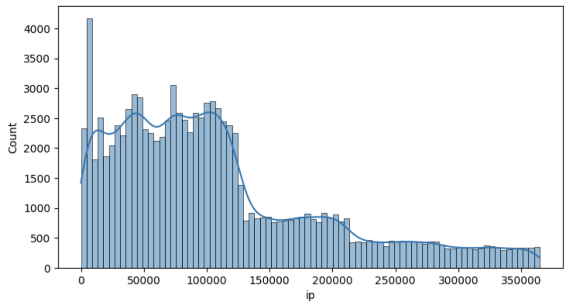

Figure 2. Feature Distribution - ip

## 3. Methods

### 3.1. Initial Preprocessing

#### 3.1.1 Missing Values & Feature Redundancy

We first checked for missing values in the train dataset. Specifically, the $attributed\_time$ column has $75,000$ missing values, indicating many clicks did not result in app downloads. Intuitively, it should not be considered as missing values as it is indeed an indication that no attributed time existed due to no app downloads. Based on the data description about $attributed\_time$ and $is\_attributed$ [1], intuitively, we believe these two features are perfectly positively correlated, and we successfully verified our assumption during the initial preprocessing in that whenever $is\_attributed$ equates to 1, there is a corresponding timestamp in $attributed\_time$.

#### 3.1.2 Distribution Exploration

**Label Distribution** As shown in Figure 1, we plotted a bar chart showing the distribution of the label $is\_attributed$ in the train dataset. The plot reveals a significant imbalance with $75,000$ non-downloads ('0') and only $25,000$ downloads ('1'), which indicates a highly skewed dataset. This skewness highlights the rarity of actual downloads in real-world scenarios, which later guided our choice of machine learning techniques and metrics that can handle such imbalanced distributions effectively.

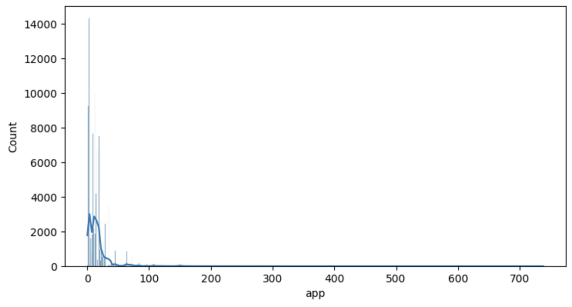

Figure 3. Feature Distribution - app

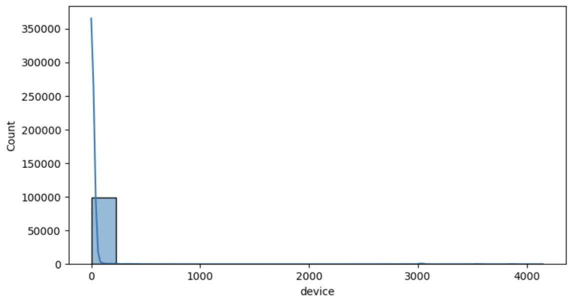

Figure 4. Feature Distribution - device

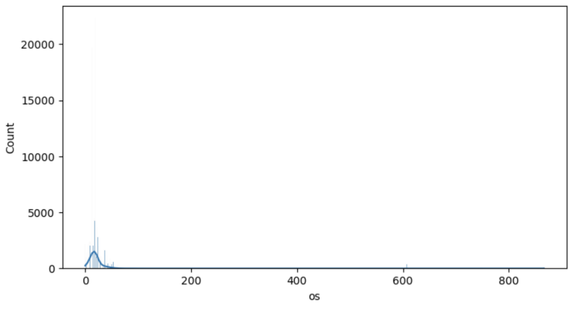

Figure 5. Feature Distribution - os

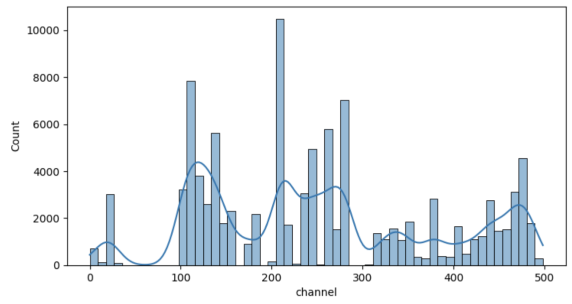

Figure 6. Feature Distribution - channel

**Feature Distribution** To better understand each feature, for the five discrete features $ip$, $app$, $device$, $os$, $channel$, we also plotted their distributions shown in Figure 2, Figure 3, Figure 4, Figure 5, and Figure 6: **(1)** The distribution for the $ip$ feature displays variability, with several peaks suggesting a non-uniform distribution of clicks across IP addresses. There is a noticeable decline in counts for the IP address on the right tail, which could indicate that certain IP addresses (on the left end) are more active or more frequently involved in clicking behavior. This pattern might be indicative of certain IPs generating higher traffic, potentially flagging them for further analysis in this context of click fraud detection, as they may represent bots or click farms; **(2)** The distribution of the $app$ feature appears to be highly skewed, which suggests that a small number of apps have a high number of clicks, while the vast majority have comparatively fewer. Such a distribution potentially implies that a few apps are extremely popular or are being targeted more in ad campaigns, which is a common pattern in real-world digital marketing; **(3)** The distribution for the $device$ feature shows a heavy concentration at the lower end of device IDs, with a steep drop-off as the device ID increases. This indicates that a small range of devices accounts for a large number of clicks. The long tail to the right suggests that while there are many unique devices present, they contribute relatively few clicks each. This kind of distribution is valuable for click fraud prediction as it might help identify specific devices that are overrepresented in the click data, which could be indicative of fraudulent behavior; **(4)** The distribution for the $os$ feature is highly concentrated around lower values, with a rapid drop in count as the OS version number increases. This indicates that only a few operating systems are commonly used among the devices involved in clicking activity, while the majority have a low incidence rate. This kind of concentrated distribution could be significant click fraud prediction, suggesting that certain operating systems might be more susceptible to fraudulent activity or more commonly used in general; **(5)** The distribution for the $channel$ feature shows multiple peaks and troughs, suggesting a multimodal occurrence across different channels. There is a noticeable variation in click counts, with some channels displaying significantly higher activity than others. This pattern could indicate varying popularity or effectiveness of channels in attracting clicks. For click fraud prediction, such a distribution can be critical as channels with unusually high or low activity might warrant further investigation for potential fraudulent behavior.

### 3.1.3 Date-Time Conversion

During the preprocessing stage, we transformed the $click\_time$ data into datetime format and then extracted day of the week, day, and hour based on the following reasoning. In the model training phase, these enriched features could offer more nuanced patterns than raw exact timestamps, which improves the model's robustness. Also, this transformation is essential for both statistical and ranking feature engineering performed in a later stage, as it allows for attribute combinations like [$ip$, $day$, $hour$] to be used effectively. These time-based features enhance our model's ability to detect patterns and anomalies in click behavior.

#### 3.1.4 Feature Removal

We then removed 2 features in this part to streamline the datasets, which ensures that only relevant features are utilized in the model, potentially enhancing the training efficiency of our models. We separated and stored the target variable $is\_attributed$ from both the train and the test dataset while removing $attributed\_time$ and $is\_attributed$ columns due to redundancy with the target variable.

### 3.2. Feature Engineering

After implementing the baseline model, we moved to two types of feature engineering: statistical and ranking features. The statistical features involve creating new metrics based on counts and unique values from various data aspects, such as IP and device type. Meanwhile, the ranking features order click events within specific groups by time, which offers a detailed view of user interactions. These techniques are important for our improved models after obtaining the baseline results.

**Statistical Features** We implemented statistical feature engineering for our improved machine learning models by creating new features based on counts and unique values within specified groups of columns (including single attribute [$ip$], triple combination [$ip$, $os$, $device$], triple combination with time factors [$ip$, $day$, $hour$]). By grouping and transforming data, we generated new columns indicating the count of $click\_time$ for different combinations of IP, OS version, device type, day, and hour of click time. Moreover, we further calculated the number of unique apps, device types, OS versions, and channels per IP. These engineered features eventually contribute to our improved models by providing detailed insights into user behavior patterns, improving its ability to distinguish between fraudulent and legitimate clicks.

**Ranking Features** We ranked click events within specific groupings (i.e., combinations of $ip$, $os$, $device$, and $app$) based on $click\_time$. There are two types of ranks: ascending and descending, indicating the order of clicks. These rankings provide insights into user click patterns, which are crucial in identifying anomalies indicative of fraudulent behavior. By integrating these rankings into our models, it becomes more adept at detecting sophisticated fraud schemes, enhancing the overall effectiveness of click fraud prediction in the vast and complex digital advertising landscape.

### 3.3. SMOTE

According to Elreedy and Atiya (2019) [5], in anomaly-type classification problems like fraud detection, imbalanced datasets are quite common. To address the data scarcity in the minority class, methods like the Synthetic Minority over-sampling TEchnique (SMOTE) are used to synthesize new samples. Specifically, as indicated by Elreedy and Atiya (2019), "it is based on sampling data from the minority class by simply generating data points on the line segment connecting a randomly selected data point and one of its K-nearest neighbors" [5]. Additionally, undersampling of the majority class can be implemented, either randomly or based on certain criteria, to further balance the dataset and improve the efficacy of classification models.

Due to the imbalanced nature of our datasets, we also utilized SMOTE to address the class imbalance issue. By artificially synthesizing new examples of the minority class (fraudulent clicks), SMOTE balances the dataset, which helps in improving the performance of machine learning models. After oversampling the minority class, we also performed downsampling of the majority class to fine-tune the balance. This balanced dataset can lead to a more generalized model that performs better on unseen data, potentially improving its fraud detection capabilities.

### 3.4. Models

#### 3.4.1 Baseline Model - Logistic Regression

According to Zou, Hu, Tian, and Shen (2019) [23], logistic regression-based classification is given an arbitrary set of inputs, and then the output is obtained by a function, which is the classification of the input data. Although effective, this method can be time-consuming due to numerous iterations, particularly when training on large datasets, making it less suitable for such applications. In our project, we implemented the logistic regression model as our baseline model particularly suited for this context (a binary classification problem). In terms of choices of parameters, we used the "sag" solver (that is suitable for large datasets), the L2 regularization (that prevents overfitting), and the regularization strength of C=1.0 (that offers a balance between complexity and accuracy) to predict click fraud efficiently.

#### 3.4.2 Gradient Boosting Classifier

After implementing the baseline model and finalizing the feature engineering, we developed several improved models within the gradient boosting framework, including the original gradient boosting classifier discussed in this section, the extreme gradient boosting (XGBoost) discussed in Section 3.4.3, the light gradient boosting machine (LightGBM) discussed in Section 3.4.4, and the categorical boosting (CatBoost) discussed in Section 3.4.5.

According to Chakrabarty et al. (2019) [2], gradient boosting is a method used to develop classification and regression models to optimize the learning process of the model, which are mostly non-linear and more widely known as decision or regression trees. Our Gradient Boosting

Classifier model for click fraud prediction demonstrates high efficacy. We chose the following parameters including "n_estimators" of 200, controlling the number of boosting stages for the model to learn from; a "learning_rate" of 0.1, affecting the step size; a "max_depth" of 5, limiting the complexity of each tree; "min_samples_split" of 4, and "min_samples_leaf" of 2, respectively, helping to prevent overfitting; and "max_features" of "sqrt", which helps in a random selection of features for a better-split decision making.

### 3.4.3 XGBoost

According to Chen and Guestrin (2016) [3], XGBoost is a scalable end-to-end machine learning system for tree boosting widely used by data scientists to achieve state-of-the-art results. Our XGBoost model is initialized with specific parameters for optimized performance. The chosen parameters include a learning rate of 0.2, a "max_depth" of 15 for complex tree structures, and "subsample" and "colsample_bytree" settings for random sampling of the training data. The model's high "scale_pos_weight" helps balance the class distribution.

### 3.4.4 LightGBM

LightGBM is a gradient-boosted decision trees (GBDT) technique, which functions as an ensemble method, combining multiple decision trees sequentially, as noted by Yıldırım in 2020 [20]. Our LightGBM model is configured with following parameters: "num_leaves" of 127 for tree complexity, "min_data_in_leaf" of 32 to prevent overfitting, an unrestricted "max_depth", a "learning_rate" of 0.1 to ensure steady learning, "min_child_samples", "feature_fraction", and "bagging_fraction" together to manage overfitting and improve model robustness.

### 3.4.5 CatBoost

According to Prokhorenkova1 et al. (2018) [16], categorical boosting (CatBoost) combines ordered boosting, an enhanced gradient boosting algorithm preventing target leakage, with a novel approach for processing categorical features. This combination allows CatBoost to surpass other gradient-boosted decision tree implementations like XGBoost and LightGBM across various machine learning tasks. Regarding our CatBoost model, we included the following parameters: a depth of 8, a learning rate of 0.1, and 200 iterations to balance the model complexity and the learning speed; the L2 regularization and border count of 32 to control overfitting; bagging temperature of 1.0 to manages model variance. Additionally, the "Logloss" loss function, iterative overfitting detector, and silent training mode are chosen for efficient binary classification.

### 3.4.6 Random Forest

According to Schonlau and Zou (2020) [18], the Random Forest algorithm, an ensemble approach in machine learning, builds multiple decision trees to enhance prediction accuracy and stability, particularly demonstrated in its performance on validation datasets. Specifically, our Random Forest combines multiple decision trees to improve the overall prediction accuracy and control overfitting. We set the following key parameters including "n_estimators" of 400 to provide a large ensemble of trees for robustness; "max_depth" of 10 to balance the depth of each tree; "min_samples_split" of 9 and "min_samples_leaf" of 3 to prevent the model from being too specific to the training data; "max_features" of "sqrt" to optimize the number of features considered for each split; and the choice of the 'gini' criterion to be more effective for classification tasks.

### 3.4.7 Stacking Generalization Ensembling

According to Healey et al. (2018) [11], stacking generalization typically involves a parametric model to combine outputs from various algorithms. The effectiveness of ensemble predictions is enhanced by adding new, informative, and uncorrelated inputs to the existing ensemble mix. Our Stacking model combines the strengths of various individual models - Gradient Boosting, XGBoost, LightGBM, CatBoost, and Random Forest, each bringing its unique predictive power. The predictions of these base models are used as input features for a Logistic Regression meta-model, which effectively integrates the individual predictions. Stacking is then performed with 5-fold cross-validation to ensure model robustness and prevent overfitting.

### 3.4.8 Majority Vote Ensembling

According to Dogan and Birant (2019) [4], majority vote ensembling evaluates individual performances of each classifier in the ensemble and adjusts their contributions to class decision. Our Majority Vote Ensembling model integrates predictions from multiple models (Gradient Boosting, XGBoost, LightGBM, and CatBoost) using the majority voting principle. By stacking the predictions and using the "mode" function, the model determines the majority vote across all models for each observation, which capitalizes on the collective wisdom of diverse algorithms, assuming that the most frequent prediction is likely the most reliable one.

## 4. Results

After training on the processed training set, we obtained a 92.63% accuracy and an AUC score of 0.721 for the baseline model, 97.57% accuracy and 0.969 AUC score for Gradient Boosting Classifier, 93.53% accuracy and 0.964 AUC score for XGBoost, 97.86% accuracy and 0.973 AUC score

| Model | Baseline | GBC | XGBoost | LightGBM | CatBoost | Random Forest | Stack | Majority |
|---|---|---|---|---|---|---|---|---|
| ACC | 0.9263 | 0.9757 | 0.9353 | 0.9786 | 0.9798 | 0.9719 | 0.9594 | **0.9817** |
| AUC | 0.721 | 0.969 | 0.964 | 0.973 | **0.975** | 0.956 | 0.972 | - |

Table 1. Summary of Model Performance (ACC AUC)

for LightGBM, 97.98% accuracy and 0.975 AUC score for CatBoost, 97.19% accuracy and 0.956 AUC score for Random Forest, 95.94% accuracy and 0.972 AUC score for Stacking Generalization Ensembling, and 98.17% accuracy for Majority Vote Ensembling. From the table 1, we observe that the **Majority Vote Ensembling model** gave us the best performance regarding accuracy while **CatBoost** provided the highest AUC among all models.

## 5. Discussion

### 5.1. Limitations

**The use of SMOTE for oversampling can lead to overfitting.** While useful for balancing datasets, SMOTE also has limitations. It can create synthetic samples that are duplicates of the minority class data points, overly general or not representative of the true minority class, potentially leading to overfitting. The technique also assumes that all points in the minority class are equally important, which might not be the case. In general, its over-specificity may hinder our models' ability to generalize to new, unseen data, reducing its overall effectiveness in real-world applications.

**The size of the original training dataset poses a computational challenge.** The original training dataset contains a massive approximately 200 million observations [1], and it is not feasible to run the entire dataset due to resource limitations. To overcome that, a random sample of 100,000 observations has been selected for training. While this sampling significantly reduces computational demands, it also introduces a limitation: the reduced dataset may not capture the full diversity and patterns present in the entire dataset, impacting the model's generalizability and accuracy when applied to broader datasets in real-world scenarios.

**The lack of hyperparameter tuning limits models' optimal performance.** Our limited computational resources restricted our ability to perform hyperparameter tuning for our models. Hyperparameter tuning is a critical process to optimize model performance, and not being able to conduct it could lead to suboptimal model results. This limitation means our models might not achieve their maximum potential performance, possibly affecting their accuracy and efficiency in practical applications.

### 5.2. Insights

**Best Model** We found the Majority Vote Ensembling model outperformed all previous models, which is consistent with our previous thoughts. The reason is that the Majority Vote Ensembling model evaluates the performance of each classifier in the ensemble and adjusts their weights to achieve a better prediction. This method's success underscores the value of combining diverse algorithms to enhance predictive capabilities, particularly in complex tasks like this one, where different models capture different aspects of the data.

**Limited Scope for Significant Improvements** In this dataset, the baseline model yields a high accuracy of 0.9263, indicating limited scope for significant improvements. This scenario is indicative of a dataset where the underlying patterns are well-captured even by simpler models, leaving little room for advanced models to achieve noticeably better performance.

**Overestimated Performance of Stacking Generalization Ensembling** Ideally, the Stacking Generalization Ensembling model should outperform all other models. However, despite combining various strong models like Gradient Boosting, XGBoost, LightGBM, CatBoost, and Random Forest, it didn't outperform the individual CatBoost model. This might be due to the meta-model (Logistic Regression) not effectively capturing the combined strengths of the base models. Additionally, CatBoost's advanced handling of categorical data and ability to model complex interactions might have given it an edge that the stacked model, with its current configuration and meta-model choice, couldn't surpass. This suggests that while stacking is powerful, its efficacy greatly depends on the synergy between base models and the meta-learner's ability to integrate their predictions.

### 5.3. Future Work

**More Feature Engineering Methods** For future enhancements, deeper feature engineering could be implemented before using the models. One area of focus could be the development of time difference features, which might provide more nuanced insights into the dataset's temporal aspects. This approach could help in capturing time-related patterns more effectively, potentially leading to improved model performance in predicting outcomes.

**More Deep Learning Models** Furthermore, we could use some deep learning models, such as DeepFM [8], to this problem to get potentially better performance. DeepFM combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture. It has a shared input to its "wide" and "deep" parts, with no need for feature engineering besides raw features [15].

## References

[1] J. K. J. E. T. Y. Aaron Yin, inversion. Talkingdata adtracking fraud detection challenge., 2018. 1, 2, 6

[2] N. Chakrabarty, T. Kundu, S. Dandapat, A. Sarkar, and D. K. Kole. Flight arrival delay prediction using gradient boosting classifier. In *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 2*, pages 651–659. Springer, 2019. 4

[3] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016. 5

[4] A. Dogan and D. Birant. A weighted majority voting ensemble approach for classification. In *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pages 1–6. IEEE, 2019. 5

[5] D. Elreedy and A. F. Atiya. A comprehensive analysis of synthetic minority oversampling technique (smote) for handling class imbalance. *Information Sciences*, 505:32–64, 2019. 4

[6] J. FRANKENFIEL. Cost per click (cpc) explained, with formula and alternatives., 2023. 1

[7] Google. Cost-per-click (cpc): Definition., 2023. 1

[8] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017. 6

[9] C. M. R. Haider, A. Iqbal, A. H. Rahman, and M. S. Rahman. An ensemble learning based approach for impression fraud detection in mobile advertising. *Journal of Network and Computer Applications*, 112:126–141, 2018. 2

[10] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. n. Candela. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, ADKDD'14, page 1–9, New York, NY, USA, 2014. Association for Computing Machinery. 2

[11] S. P. Healey, W. B. Cohen, Z. Yang, C. Kenneth Brewer, E. B. Brooks, N. Gorelick, A. J. Hernandez, C. Huang, M. Joseph Hughes, R. E. Kennedy, T. R. Loveland, G. G. Moisen, T. A. Schroeder, S. V. Stehman, J. E. Vogelmann, C. E. Woodcock, L. Yang, and Z. Zhu. Mapping forest change using stacked generalization: An ensemble approach. *Remote Sensing of Environment*, 204:717–728, 2018. 5

[12] M. S. Iqbal, M. Zulkernine, F. Jaafar, and Y. Gu. Fcfraud: Fighting click-fraud from the user side. In *2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE)*, pages 157–164, 2016. 2

[13] Y. Z. Kenneth C. Wilbur. Click fraud., 2008. 1

[14] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2013. 2

[15] OpenAI. Chatgpt, 2023. 6

[16] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018. 5

[17] PwC. Global internet advertising revenue from 2017 to 2027., 2023. 1

[18] M. Schonlau and R. Y. Zou. The random forest algorithm for statistical learning. *The Stata Journal*, 20(1):3–29, 2020. 5

[19] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, and D. Zhang. Graph based tensor recovery for accurate internet anomaly detection. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1502–1510, 2018. 2

[20] S. Yıldırım. Understanding the lightgbm, 2020. 5

[21] X. Zhang, X. Liu, and H. Guo. A click fraud detection scheme based on cost sensitive bpnn and abc in mobile advertising. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pages 1360–1365. IEEE, 2018. 2

[22] F. Zhu, C. Zhang, Z. Zheng, and S. A. Otaibi. Click fraud detection of online advertising–lsh based tensor recovery mechanism. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):9747–9754, 2022. 2

[23] X. Zou, Y. Hu, Z. Tian, and K. Shen. Logistic regression model optimization and case analysis. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, pages 135–139, 2019. 4