Click Fraud Detection Using TabNet

Mitchell Ultee

2081742

Master's Thesis

Data Science and Society

Tilburg University, Tilburg

Supervisor: Dr. M. De Sisto

Second reader: Dr. S. Collin

Januari 2024

Wordcount: 7363

Abstract

Due to the rise of online advertising, novel forms of fraud have emerged, posing new challenges to organizations' business activities. One of those novel forms of fraud is click fraud, in which advertisers are disadvantaged by publishers or competitors who perform insincere clicks on the advertisers' ads. This form of fraud is exploited when the pay-per-click business model is applied. This leads to depletion of the advertisers' budgets, without contributing to their business goals. To counteract this issue, machine learning has been employed to build classifiers that can prematurely detect a fraudulent click, and possibly prevent it. The current study aimed to investigate the classification capabilities of TabNet for click fraud detection, as this model had, at present, not been utilized for this purpose. Additionally, the performance of TabNet was compared to the two best-performing methods as proposed in the current literature on detecting click fraud. The TalkingData dataset was used, in which roughly 185 million clicks were collected in a period of four days. The data was first split into a train and test set. Feature selection, feature engineering, and SMOTE were performed separately on the train and test set to prevent leakage. Afterward, the train set was split into a train and validation set. Modeling and hyperparameter optimization were performed on the training data and examined on the validation set. The final model with optimized hyperparameters yielded an accuracy score of 95.7%. However, the precision score of 4.2% illustrated a poor performance in predicting the positive class. Although the classification capability of TabNet showed decent performance on the TalkingData dataset, the model was unable to outperform the best-performing models. Limitations and future directions are discussed in the final chapter.

Keywords: online advertising, pay-per-click, click fraud detection, deep learning, TabNet

Technology statement

The dataset was retrieved from Kaggle and has been offered by the leading big data service platform in China, *TalkingData*. The data contains clicking information on a mobile app advertisement. IP addresses and other information that could identify the used have been label encoded. Hence, the dataset does not contain any personal information. As stated on the competition page on Kaggle, under competition Rules, permission has been granted to use the dataset for academic purposes.

The tables in this study have been created by the author of this study, however, table 1 has been adapted from Byun & Batool (2022). Furthermore, code has been adopted from a Kaggle submission for the feature engineering stage, and various Stack Overflow posts have been employed for debugging purposes. Lastly, three writing tools have been employed, namely, the thesaurus from Merriam-Webster to help with paraphrasing, the Writing Center from the University of Richmond to aid in suitable transitional words and phrases, and Grammarly for a spelling, grammar, and punctuation review of the final version of the present study.

Click Fraud Detection Using TabNet

With the rise of online advertising, unprecedented forms of fraud have arisen, posing problems in the domain of online advertising. One of those fraudulent techniques is *click fraud*, which involves generating synthetic clicks on advertisements, depleting the budget of advertisers, and leaving dishonest publishers with more revenue (Minastireanu & Mesnita, 2019; Thejas et al., 2021). According to a global PPC click fraud report, small and medium-sized businesses (SMBs) lose US$14,900 on average per year due to click fraud (PPC Protect, 2021). Furthermore, professor Cavazo from the University of Baltimore estimated that the total number of losses due to digital ad fraud amounted to US$35 billion in 2020 (University of Baltimore, 2020). Additionally, cybersecurity company *Cheq* estimated a total loss of US$61 billion in online advertising fraud in 2022 (Trajcheva, 2023). This illustrates the continuing and rapid growth of digital fraudulent behavior. Therefore, detecting and preventing fraudulent clicks can be of great benefit to the marketing activities of

companies that employ online advertising, as their budget will be administered as intended, and positively impact their business goals.

Various machine learning methods have been developed to detect click fraud. For instance, one study found the random forest algorithm to be the best machine learning algorithm, yielding an accuracy of 84% (Aljabri & Mohammad, 2023). A different study employed a novel machine learning method for the task of click fraud detection, the LightGBM method, yielding an accuracy of 98% (Minastireanu & Mesnita, 2019). Furthermore, research in this field has employed deep learning methods, namely, Artificial Neural Networks (ANN) and Generative Adversarial Networks (GAN), realizing 94.7% accuracy (Thejas et al., 2019). Lastly, an ensemble of methods, comprised of a Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory network (BiLSTM), and Random Forest was applied, detecting click fraud with 99.6% accuracy (Batool & Byun, 2022). Amongst the aforementioned studies, many other studies have applied a variety of methods, either alone standing or hybrid, to detect false clicking behavior.

Although many methods have been used to detect click fraud, to the best of my knowledge, no studies have employed TabNet for this purpose. TabNet is a novel deep learning algorithm developed for tabular data and has been used for a variety of tasks (Arik & Pfister, 2020). For instance, TabNet, combined with SMOTE, was employed for a binary classification task detecting card fraud, obtaining an AUC-ROC score of 89.2% and an accuracy score of 97.2%, outperforming all other methods used during this study (Meng et al., 2022). Moreover, TabNet has also been employed in a medical context, namely, to classify Alzheimer's disease, realizing an accuracy score of 88.3% for multi-class classification, and 93.8% for binary classification (Jin et al., 2023). These studies exemplify the potential of TabNet for classification tasks.

Given that TabNet can be effective for fraud detection and binary classification tasks, it is reasonable to suggest that TabNet can be an effective method for detecting click fraud. Therefore, the central research question of this study is: How accurately can click fraud be detected using TabNet? In addition, one sub-question will be investigated, namely, how does TabNet perform compared to the two currently most accurate methods on detecting click fraud? These methods will be discussed in the next chapter. Societally, investigating this research question can be of relevance to businesses making

use of online advertising, as it might allow for more accurate detection of click fraud, which, in turn, can reduce budget losses. Shown by the scarcity of literature on TabNet, scientifically, exploring this research question broadens our current understanding of TabNet and its functioning, by applying it to a task that, at present, has not utilized the model.

Literature review

Click fraud

Online advertising involves four parties: publishers, advertisers, end-users, and advertising networks. Publishers possess websites or apps that facilitate advertising opportunities in the form of advertising space. For example, on websites, those ads include banners, display ads, or video ads. The types of in-app advertising opportunities entail, amongst others, playable ads, offerwalls, and rewarded video ads. Advertisers purchase advertising space from publishers via advertising networks, which, in turn, display their ads to end-users. Advertising networks are large companies such as Google and Meta that facilitate the advertising process between advertisers and publishers (Wilbur & Zhu, 2009; Aljabri & Mohammad, 2023).

The process commences with an online advertising campaign devised by the advertiser. The advertiser determines a budget for the campaign and submits it to the advertising network's interface. Publishers inform the advertising networks of the advertising space that they offer on their website or in their app. The advertising network takes the ad from the advertiser and displays it on the website or in the app of the publisher to reach the end-users (i.e., the target audience). Each time an end-user clicks on the advertisement, the publisher generates revenue, the advertising network earns a portion of the publisher's revenue, and the budget of the advertiser depletes. This business model is called pay-per-click (PPC; Thejas et al., 2021; Aljabri & Mohammad, 2023).

However, this construction allows for fraudulent activities to unfold. Malignant publishers or competitors of the advertiser can generate illegal clicks to, generate more revenue for the publisher, or deplete the advertising budget of the advertiser, instigated by competitors. Naturally, parties who carry out these illegal clicks have no interest in the ad, causing the advertiser's budget to decline, without benefitting the business goals intended for the advertising campaign (Aljabri & Mohammad, 2023). In

addition to publishers and competitors, click fraud can be committed by other sources. Bots, for example, generate the majority of internet traffic and therefore the largest source of fraudulent clicks. Furthermore, proxy clicks are often illegal as the IP address and geolocation are hidden, contaminating the marketing data of advertisers. Lastly, click farms created for fraudulent interactions with websites and social media, and dissatisfied customers, have shown to be a source of click fraud (Gohil & Meniya, 2021).

Click fraud detection

Given that there are various sources and methods for click fraud, it can be difficult to detect this illegal behavior. Additionally, the lack of quality control in the world of online advertising, specifically regarding PPC, leads to a continuous increase in budget losses for advertisers (Gohil & Meniya, 2021). To combat the issue of click fraud, various machine learning methods have been developed to detect ungenuine clicks. Doing so can diminish the problem of click fraud, allowing advertisers to use their budgets as originally intended. For instance, if it is likely that an end-user with a certain IP address and device is performing fraudulent clicks, detecting their presence in the app or on the website prematurely can prevent them from performing a fraudulent click by hiding the advertisement. However, the publisher's compliance is required to do so as their traffic information is integral to this solution. If the fraudster is the publisher, this solution does not work. Nevertheless, other solutions can be constructed with the accurate prediction of click fraud to potentially overcome or reduce this challenge.

The methods in the current literature consist of machine learning, deep learning, or hybrid/ensemble models for click fraud detection. To see an overview of the models and their performances, see Table 1. As can be seen in Table 1, no two studies use the same method for the task of click fraud detection. This complicates determining the current state-of-the-art method for click fraud detection, as the robustness of one method can be questioned, i.e., performance on a dataset different from the one applied in the adhering studies to investigate if the performance of the method is comparable across multiple datasets.

Table 1

Machine learning methods for click fraud detection

Author(s)	Year	Model(s)	Dataset	Accuracy
Thejas et al.	2019	ANN-GAN	TalkingData	94.7%
Minastireanu & Mesnita	2019	LightGBM	TalkingData	98%
Viruthika et al.	2020	XGBoost	-	91%
Liu et al.	2020	Cost-Sensitive CNN	BuzzCity Dataset	93%
Li & Jia	2020	Random Forest	Real-Click Fraud Data	93.6%
Thejas et al.	2021	Cascaded Forest and XGBoost	TalkingData, Avazu, Kad	94.5%
Gohil & Meniya	2021	XGBoost Gradient Boosting	Online Advertising Data	96%
Aberathne	2021	Hidden Markov Scoring Model	Mobile Advertising Company	94%
Batool & Byun	2022	CNN-BiLSTM-RF	TalkingData	99.6%
Aljabri & Mohammad	2023	Random Forest	Beacon	84%

Note: this graph was adapted from Batool & Byun (2022). Furthermore, this graph is not exhaustive of all studies that have been conducted on click fraud detection using machine learning techniques. Studies that have not reported accuracy score have been left out.

Nonetheless, at present, the CNN-BiLSTM-RF method seems to be the most accurate method to detect fraudulent clicks, with an accuracy score of 99.6%. The architecture is comprised of a convolutional neural network for automatic implicit feature extraction, a bidirectional long short-term memory network used for learning long-term data sequences in a forward and backward fashion, and a random forest for (binary) classification (Batool & Byun, 2022). The second best-performing method for the task of click fraud detection is LightGBM, which is a gradient boosting decision tree method, using leaf-wise growth to build a decision tree. The method yielded an accuracy score of 98% (Minastireanu & Mesnita, 2019).

Although both studies aimed to accurately detect click fraud, and realized a good performance on this task, there are several differences between the ensemble method applied by Batool & Byun

(2022) and the machine learning method applied by Minastireanu and Mesnita (2019). Firstly, the former looked at a mobile ad system's relationship with users, publishers, and advertisers, while the latter assessed instances based on the click portfolio of a single user (i.e., how many clicks were performed without downloading the app). Secondly, the CNN-BiLSTM-RF method was trained on one million random samples, maintaining the class ratio of the full dataset, and utilizing the original features of the dataset. Contrastingly, the LightGBM model was trained on the whole dataset and applied feature engineering, training the model on a total of 19 features. Furthermore, Batool & Byun (2022) applied SMOTE on the train set to counteract the problem of imbalance, while Minastireanu & Mesnita (2019) did not apply an over or under sampling method to solve this issue. Lastly, the performance of CNN-BiLSTM-RF was evaluated on the Area Under the Curve (AUC), accuracy, precision, sensitivity, F1-score, and specificity, while LightGBM was evaluated solely on accuracy.

TabNet

Where deep learning for data types like images, audio, and text is widely researched and has shown to be successful in comparison to other machine learning models, tabular data is relatively underrepresented in the field of deep learning. However, tabular data is the largest encountered datatype in today's world. To realize similar success of deep learning for tabular data, as has been the case for other datatypes, TabNet was developed (Arik & Pfister, 2020).

TabNet is a novel canonical deep neural network algorithm for tabular data. The algorithm uses gradient descent to find the minimum value of the loss function, and sequential attention for feature selection, making feature selection before training nonessential (Arik & Pfister, 2020). TabNet has been applied to a variety of tasks like, credit card fraud detection yielding an accuracy score of 96.5% (Zhang et al., 2022), rainfall prediction yielding an RMSE score of 0.619 (Yan et al.., 2021), and diabetes detection realizing an accuracy score of 99.4% (Joseph et al., 2022). In the aforementioned research, TabNet outperformed all other benchmark models used during these studies.

As previously stated, TabNet has, at present, not been utilized for the task of click fraud detection. Given the high accuracy scores of the currently best performing methods, CNN-BiLSTM-RF and LightGBM, for the task at hand, one can argue the relevance of applying TabNet to this context, as there is little room for improving the accuracy of click fraud detection that the currently

best performing methods provide. However, the market size of online advertising in 2022 was valued at US$236.9 billion and is expected to realize annual growth of 15.7% up to and including 2030 (Grand View Research, 2022). In 2022, the total loss to ad fraud was estimated at US$61 billion, which is more than a quarter of the total market size (Trajcheva, 2023). Because of the context of the current study, concerning enormous amounts of money, a small improvement in the accuracy of click fraud detection can lead to a large decrease in economic losses caused by click fraud. Not only does this help commercial organizations with the optimization of their advertising budgets, but it can also help government agencies with tackling this criminal activity. When ad fraud becomes less lucrative due to even the higher accuracy of fraud detection and prevention, ad fraudsters might lose interest and abandon their criminal activities, decreasing the threat of this type of cybercrime.

Methodology

Dataset Description

In this study, the *TalkingData* dataset was used, offered by the leading big data service platform in China. TalkingData provided a separate train and test file in CSV format for competition purposes. However, the test file does not contain a target feature and, therefore, only the train dataset was used in the current study. This dataset contains information on roughly 185 million clicks collected between 06-11-2017 and 10-11-2017. The dataset was retrieved from Kaggle and has been used in other studies that aimed to predict click fraud (see Table 1). Eight features are included in the dataset: *ip_address*, *app*, *device*, *os*, *channel*, *click time*, *downloads time*, and *is_attributed*. See Appendix A for a description of each feature. The features *ip_address*, *app*, *device*, *os*, and *channel* have already been label encoded by the provider of the data. The target feature is the *is_attributed* feature, indicating if an app was downloaded after clicking a mobile app advertisement. There is a major imbalance in the data as the target feature pertains for 99.75% to value 0, meaning that the app was not downloaded, and 0,25% pertains to the class that did download the app denoted as value 1.

Algorithms and Software

During this study, all code was written in Python language (version 3.11.5; Van Rossum & Drake, 2009). The TabNet model is trained using the open-source PyTorch_tabnet package, developed

by French software company DreamQuark (2023). The model uses gradient descent for optimization and sequential attention for soft feature selection. A single encoding step consists of three processes: a feature transformer, an attentive transformer, and a mask. The feature transformer process includes a fully connected layer, batch normalization layer, and a gated linear unit to select the features for each decision step. The attentive transformer encompasses a fully connected layer, a batch normalization layer, prior scales indicating prior use of a feature, and a sparsemax activation function, which results in the selection of salient features. The mask indicates what features to use for the next decision step. The decoder architecture, aimed at reconstructing tabular features from the encoding step, consists of a feature transformer followed by a fully connected layer at each step.

Next to Pytorch_tabnet, various other packages were used for preprocessing the data, visualizing the data, hyperparameter optimization, and calculating the evaluation metrics. These processes are discussed in more detail in the next subparagraph. Firstly, Pandas (McKinney, 2010) and NumPy (Harris et al., 2020) were used for data manipulation, structuring the data in such a manner that it could be used for feature engineering, oversampling, and training the model. Furthermore, Pandas was used to convert the *click_time* feature to a time object. Secondly, the Python module *datetime* was employed to extract individual time features (i.e., minute, hour, etc.) from the *click_time* feature. Furthermore, Imblearn was used for the Synthetic Minority Oversampling Technique (SMOTE; Lemaître, 2017), and the Python module *collections* to count the instances of the majority and minority classes before and after the application of SMOTE. Additionally, Scikit-learn was used for the three-way splitting of the data, hyperparameter optimization with randomized search, and calculating the evaluation metrics (Pedregosa et al., 2011). Also, the built-in attribute of Scikit-learn, *feature_importances_,* was used to calculate the feature importance. Lastly, Matplotlib was utilized to visualize the exploratory data analysis, confusion matrices, and feature importance (see Figures 4 and 5; Hunter, 2007). See Appendix B for an overview of all packages and their versions used for the current study. The aforementioned methods are discussed in more detail in the following subparagraph.

The performance of TabNet on detecting click fraud was compared to the two currently best-performing methods for click fraud detection, namely, CNN-BiLSTM-RF and LightGBM (see Table

1). As previously mentioned, TalkingData provided separate train and test files. The study by Batool & Byun (2022) used the train file provided by TalkingData, containing roughly 185 million rows, and trained the CNN-BiLSTM-RF method on one million random samples because of computational limitations, which was also applicable to the current study. Therefore, the CNN-BiLSTM-RF method was not retrained in the current study. Contrastingly, Minastireanu and Mesnita (2019) trained LightGBM on a combined dataset of the train and test file provided by TalkingData, which amounted to approximately 200 million. Therefore, in the present study, LightGBM was retrained to explore the performance of the model when trained on the same data as was used by Batool & Byun (2022), and in the current study. By using the exact same data for all three models, a more impartial comparison could be made between, CNN-BiLSTM-RF and LightGBM, and TabNet. The LightGBM package was used to retrain LightGBM in the present study (Ke et al., 2017).

Workflow

TabNet

A graphical representation of the workflow can be found in Figure 1. Firstly, exploratory data analysis was performed to gain a better understanding of the data and visualize its patterns. Next, the data was split into a train and test set, with 80% of data reserved for the training set, and 20% for the test set. The training and test set consisted of roughly 148 million, and 37 million rows respectively, after the train/test split. This was done to avoid data leakage from occurring. Due to computational limitations and time restrictions, the training set was reduced to one million random samples, reducing run time on modeling and hyperparameter optimization. Next, preprocessing was implemented for both the training and test sets separately. Feature selection was performed by dropping the *attributed_time* feature. Because TabNet does not allow the input of missing values, and 99.75% of this column consisted of missing values, the feature was dropped from both the train and test set.

Next, feature engineering was performed. First, the *click_time* feature was used to add time-related features to the subsets. The day in the week and year on which the click was performed were added as separate features. Furthermore, the hour, second, and minute of the click timestamp were extracted and added as individual features. Secondly, features were created by grouping the IP address with other features and counting the occurrences. For instance, *ip_count* was added, showing the

number of clicks one IP address performed in the course of four days. Another addition is the feature *ip_hour_channel*, indicating the number of times a click was performed from one IP address, in a specific publisher app, at a certain hour. For all added features based on IP grouping and time-related features, see Appendix C. Again, the aforementioned methods were performed on both the train and test set, separately.

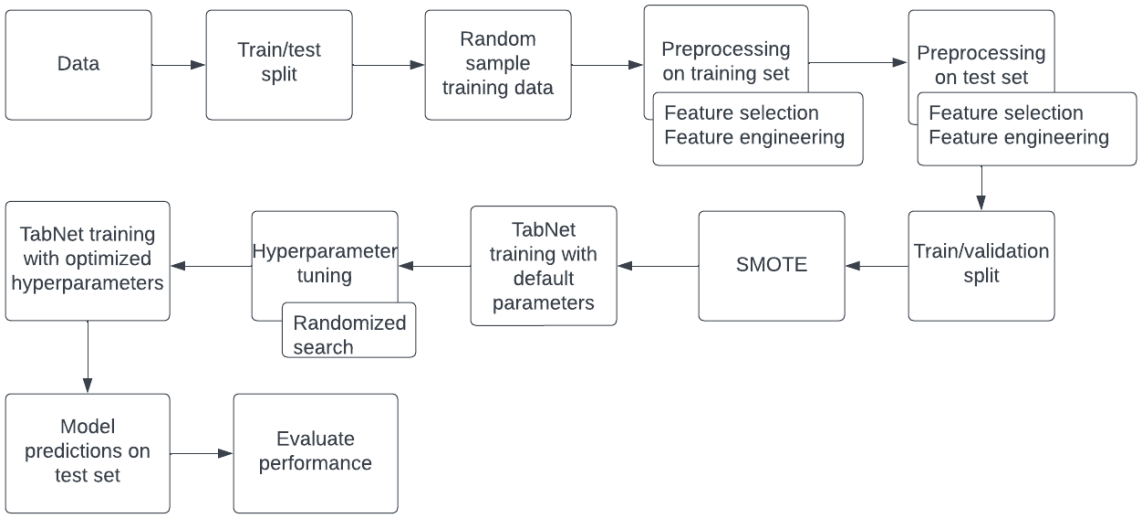
After preprocessing, the data train data was split into 80% train data and 20% validation data, which amounted to 800.000 and 200.000 rows respectively. All subsets housed 17 columns. Hold-out cross-validation (i.e., three-way split) was applied in this study, as opposed to k-fold cross-validation, because of the large dataset providing a sufficient amount of data for modeling (Yadav & Shukla, 2016). On the remaining training data, SMOTE was performed to counteract the sizable imbalance in the dataset, leaving both classes with 800.000 rows. SMOTE was chosen for the current dataset as it has shown to be the most effective technique for highly imbalanced data compared to other oversampling, undersampling, or hybrid techniques (Bach et al., 2017; Vimalraj & Porkodi 2018). Next, with the synthetically oversampled training dataset, the first TabNet model was trained, employing the default hyperparameters. The maximum number of epochs was 100 and for each epoch, the performance was evaluated on balanced accuracy. This metric was chosen because of the imbalanced data. Early stopping was performed when the performance did not show improvement within 10 epochs. The original paper introducing TabNet recommended a batch size of up to 10% (Arik & Pfister, 2020). Accordingly, the batch size was set at 64000 training examples, which was 10% of the training data after the train validation split.

To tune the hyperparameters of the model, due to time restrictions, randomized search was applied for six hyperparameters. For randomized search, the number of iterations to find the optimal hyperparameters can be specified, as opposed to grid search which exhaustively searches for the optimal hyperparameters. The number of iterations was set at 130, exploring about a quarter of all possible hyperparameter combinations (486) specified in the parameter grid. See Appendix D for all hyperparameters and values examined during randomized search. Note that, because randomized search does not search for the optimal hyperparameters exhaustively, the true combination of optimal

hyperparameters might have never been found. With the improved hyperparameters, the TabNet model was optimized, predicted on the test set, and evaluated on its performance.

Figure 1

Methodology workflow TabNet



Note: this workflow only applies in whole to the training of TabNet, not LightGBM, which has mostly been retrained based on the data science techniques reported in the original paper (Minastireanu & Mesnita, 2019).

LightGBM

The workflow for retraining LightGBM is relatively the same for TabNet up to and including the preprocessing of the training and test set. Doing so allows for retraining the model on the same dataset as was done for TabNet. SMOTE was not applied because the original paper did not make use of such a technique (Minastireanu & Mesnita, 2019). Furthermore, the exact hyperparameters that were used in the original paper were also used in the current study to retrain LightGBM for click fraud detection. See Appendix E for an overview of all the hyperparameters and theirs values. For exploratory purposes, the retrained LightGBM model was also trained on the full train set (i.e., 148 million rows) to investigate the dependence of LightGBM on the amount of data that the model is trained on. LightGBM is less computationally demanding and could therefore also be trained on the full train dataset, instead of only one million random samples.

Evaluation Method

The evaluation metrics for the performance of TabNet and benchmark models will be accuracy, as this is the only evaluation metric that was reported in both the original papers of the benchmark models. Additionally, the study by Batool & Byun (2022) also evaluated the CNN-BiLSTM-RF method on precision, sensitivity, specificity, and the F1-score, and will therefore also be reported in the current study for TabNet and the retrained LightGBM model. Not all evaluation metrics that will be reported in the current study, were reported in the original study by Minastireanu & Mesnita (2019), and can therefore only be compared based on accuracy score.

Furthermore, to give a better overview of TabNet's and retrained LightGBM's predictive performances on the positive and negative classes, the confusion matrices for both models will be reported. Moreover, feature importance for both TabNet and the retrained LightGBM was calculated and visualized to examine the individual contribution of the original features and engineered features, to the classification performance of both models. This was primarily done to examine the contribution of the feature engineering phase to the performance of the models.

Accuracy is not effective for highly imbalanced data, as is the case for the TalkingData dataset (Thölke, 2023). Therefore, the balanced accuracy score, which is the arithmetic mean of specificity and sensitivity, developed for imbalanced data, was used to as the evaluation metric for each epoch during the training of TabNet (Garcia et al., 2009).

Results

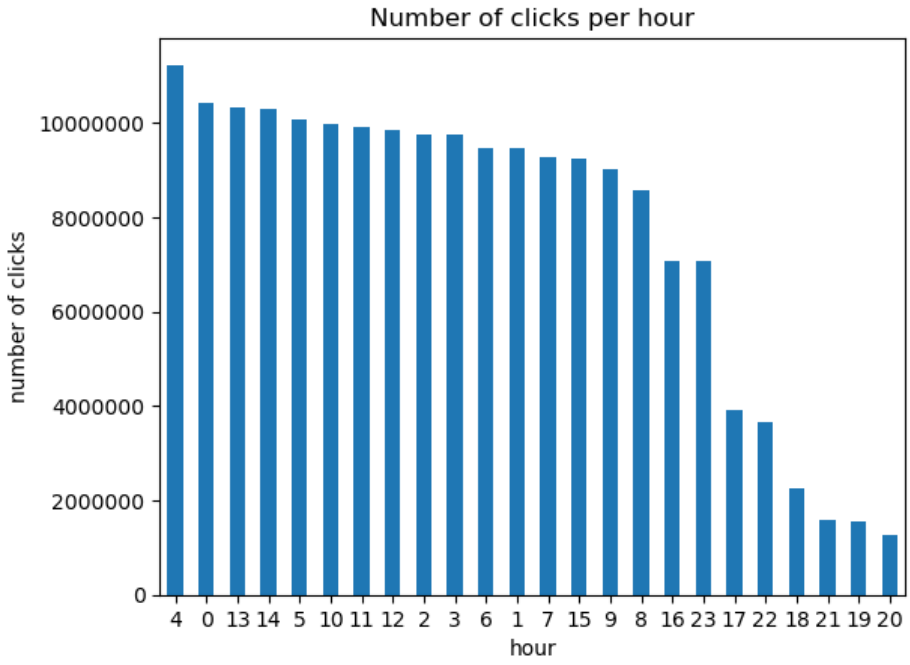
In this section, EDA, data preprocessing, hyperparameter optimization, and classification performance of TabNet and the retrained LightGBM on the TalkingData dataset, as described in the methodology section, will be reported.

Exploratory data analysis

Firstly, the number of clicks performed per hour was examined. Figure 2 shows that the most clicks were performed at 4 a.m. This might illustrate the activity of clicks performed by bots, as this is an unusual time for actual end users to be active, and demonstrates the challenge of click fraud.

Figure 2

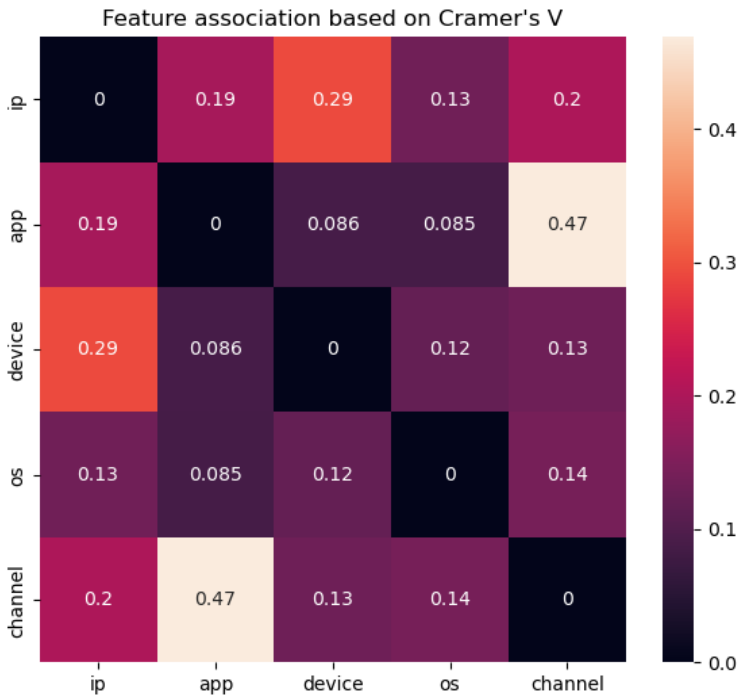
Number of clicks per hour



Furthermore, associations between the categorical features, *ip*, *app*, *device*, *os*, and *channel* were investigated to see if there were highly associated features that could be removed to potentially improve the model's performance and enhance computational efficiency. The effect size measurement *Cramer's V* was used, which is a statistic for measuring the association between categorical variables (Kearney, 2017). Figure 3 shows a moderate association between *ip* and *device*, indicating that clicks from unique users are often performed on the same device, and *app* and *channel*, implying that ads of advertisers are often displayed to end-users on one certain app. Because the associations were only moderate, no features were removed from the dataset.

Figure 3

Feature associations



TabNet

Initial training of TabNet with default hyperparameters achieved an accuracy score of 88.5%, a balanced accuracy score of 84.7%, a precision score of 1.8%, a sensitivity score of 80.9%, an F1 score of 3.5%, a specificity score of 88.5%, and an AUC score of 84.7% on the validation set. After hyperparameter tuning, the model showed improved performance, achieving an accuracy score of 94.0%, a balanced accuracy score of 87.8%, a precision score of 3.3%, a sensitivity score of 88.0%, an F1 score of 6.4%, a specificity score of 94%, and an AUC score of 87.8% on the validation set. See Appendix F for the hyperparameter values after the execution of randomized search to find optimal parameters. The final predictions of TabNet on the test set achieved an accuracy score of 95.7%, a balanced accuracy score of 85.3%, a precision score of 4.2%, a sensitivity score of 74.9%, an F1 score of 7.9%, a specificity score of 95.7%, and an AUC score of 85.3%. See Table 2 for an overview of all aforementioned performance scores at each stage.

Considering the aforementioned evaluation metrics, TabNet seems to perform relatively well based on accuracy score. From all instances, the model predicts 95.7% correctly. However, when considering precision and the F1 score, it becomes evident that the model performs very poorly on the

Table 2

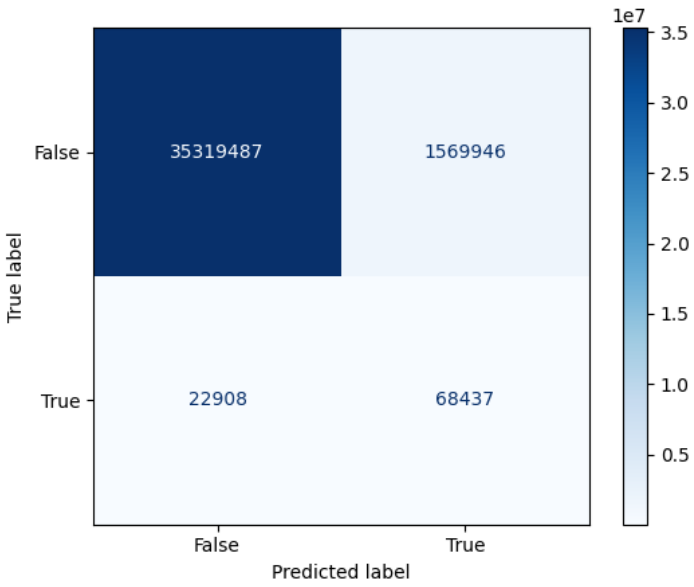
Performance of TabNet

Phase	Accuracy	Balanced accuracy	Precision	Sensitivity	F1	Specificity	AUC
Initial training	88.5%	84.7%	1.8%	80.9%	3.5%	88.5%	84.7%
Optimized training	94.0%	87.8%	3.3%	88.0%	6.4%	94.0%	87.8%
Evaluation on test set	95.7%	85.3%	4.2%	74.9%	7.9%	95.7%	85.3%

positive classes, i.e., the instances leading to a download of the app. This can also be observed from the confusion matrix in Figure 4, showing that the model predicted 1.56 million instances as positive when they were, in fact, negative, out of all 36.9 million instances (4.2%), while 22.908 out of 91.345 positive instances were predicted incorrectly (25.1%). This illustrates that the error rate for prediction in positive instances is considerably higher compared to the error rate in the negative instances.

Figure 4

Confusion matrix TabNet



Interestingly, the AUC score of 85.3% suggests that the model is capable of distinguishing the classes fairly well, and thus, the predictive performance can potentially be improved by moving the classification threshold. Nevertheless, the original LightGBM and CNN-BiLSTM-RF methods achieved a higher accuracy score. Therefore, TabNet was not able to outperform the best performing models (see Table 3).

Lastly, the contribution to the model's prediction, of the original features and the features from the engineering stage, are considered. Figure 5 shows that the original features *app* and *device* contribute the most to the model's prediction, contributing 33.9% and 21.2%, respectively. The original feature *channel* shows adequate importance, contributing for 15.6%. The features from the engineering stage did not contribute much to the model's predictions.

Table 3

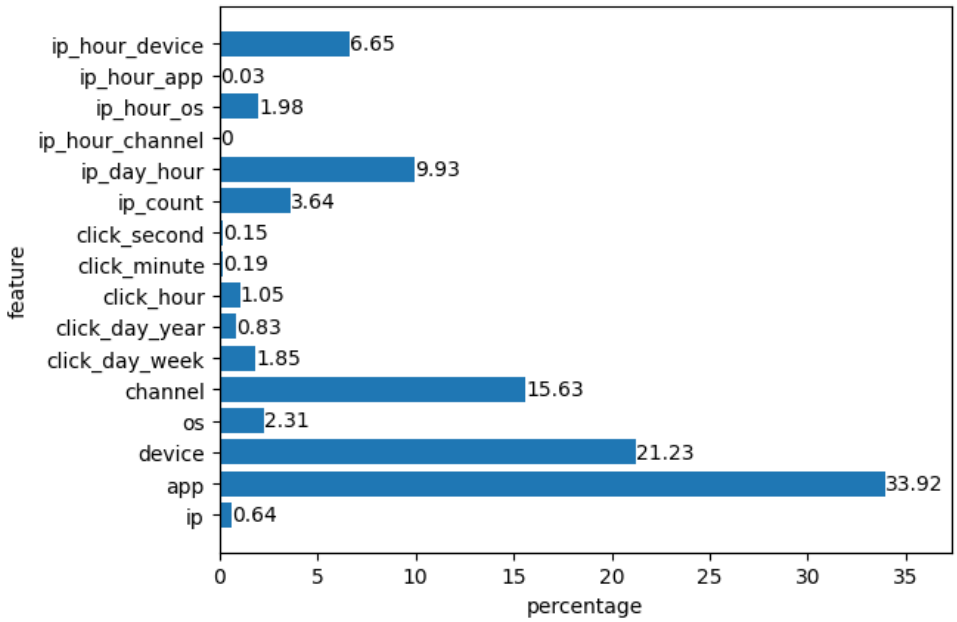
Comparison of TabNet to the best-performing models

Model	**Accuracy**	**Balanced accuracy**	**Precision**	**Sensitivity**	**F1**	**Specificity**	**AUC**
CNN-BiLSTM-RF	99.6%	-	99.6%	99.6%	99.6%	99.6%	99.6%
LightGBM	98.0%	-	-	-	-	-	-
TabNet	94.6%	85.3%	4.2%	74.9%	7.9%	95.7%	85.3%

Note: in this graph, the performance of LightGBM from the original paper by Minastireanu and Mesnita (2019) is reported, not the performance of the LightGBM model retrained in the present study.

Figure 5

Feature importance TabNet



Note: the values on the x-axis of this graph display percentages. For an explanation of each feature exhibited on the y-axis, please see Appendix A and C.

Retrained LightGBM

First, LightGBM was retrained using the reduced dataset of one million samples. The model achieved an accuracy score of 72.2%, a balanced accuracy score of 58.1%, a precision score of 0.2%, a sensitivity score of 43.9%, an F1 score of 1.0%, a specificity score of 72.2%, and an AUC score of 58.1%. Next, LightGBM was retrained using the full train dataset in the current study, achieving an accuracy score of 87.1%, a balanced accuracy score of 52.3%, a precision score of 0.3%, a sensitivity score of 17.3%, an F1 score of 1.0%, a specificity score of 87.2%, and an AUC score of 52.3%. For an overview of the aforementioned metrics, see Table 4.

These performance metrics and the confusion matrices in Figures 7 and 8, demonstrate some interesting findings. Firstly, the improvement in accuracy shows that LightGBM is dependent on the amount of data that the model is trained on, as the accuracy score improves when the amount of data increases (see Table 4). Therefore, it might be preferable to use a different model for click fraud detection when there is only little data available for training.

Table 4

Comparison of retrained and original LightGBM models

Model	Accuracy	Balanced accuracy	Precision	Sensitivity	F1	Specificity	AUC
Retrained LightGBM small dataset	72.2%	58.1%	0.2%	43.9%	1.0%	72.2%	58.1%
Retrained LightGBM large dataset	87.1%	52.3%	0.3%	17.3%	1.0%	87.2%	52.3%
Original LightGBM	98.0%	-	-	-	-	-	-

Note: the model named Retrained LightGBM small dataset refers to the model being retrained on one million random samples, while Retrained LightGBM large dataset refers to the model being retrained on the complete train dataset.

Figure 7

Confusion matrix retrained LightGBM one million samples

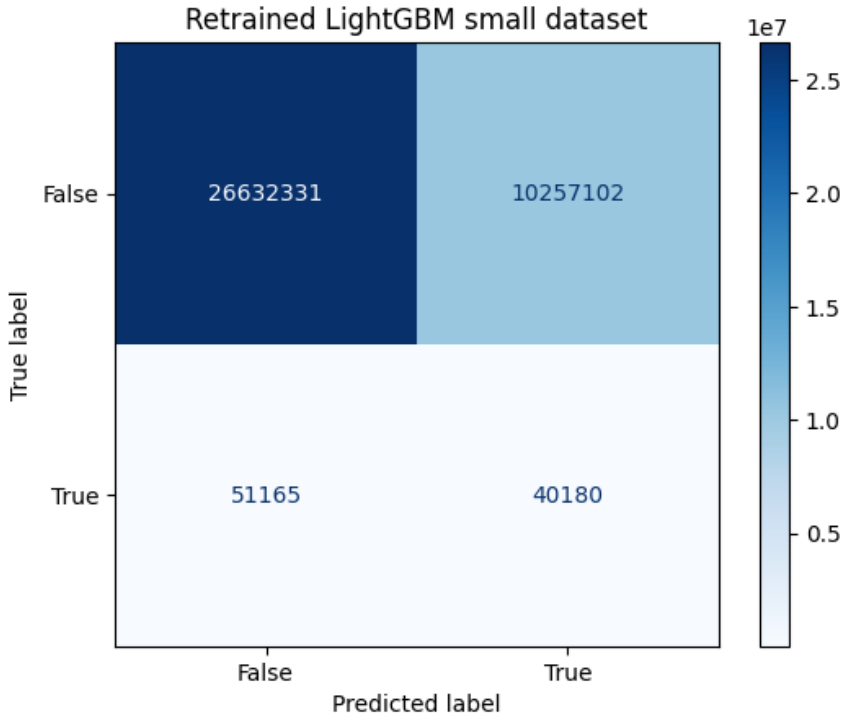
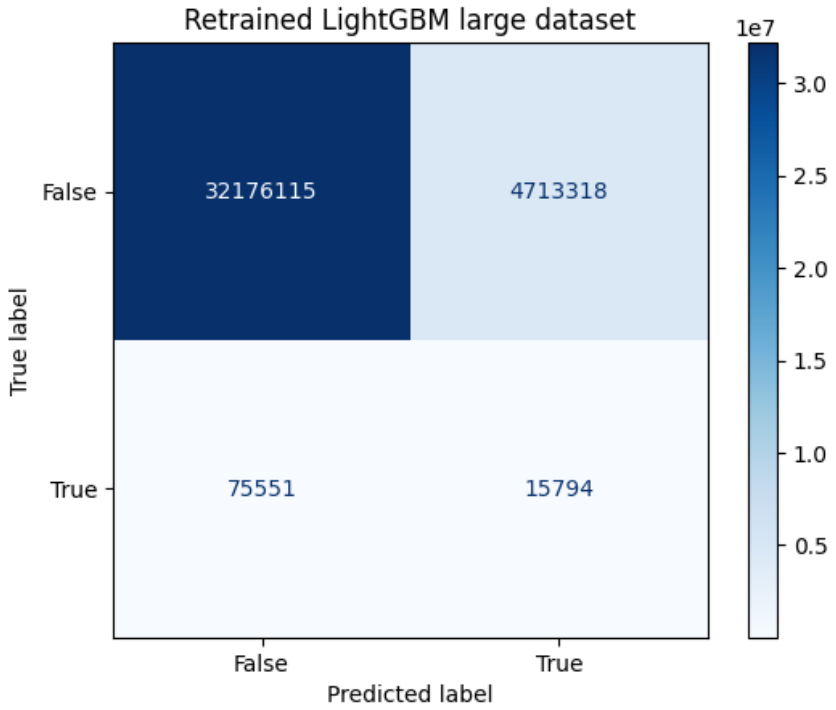


Figure 8

Confusion matrix retrained light GBM full train set

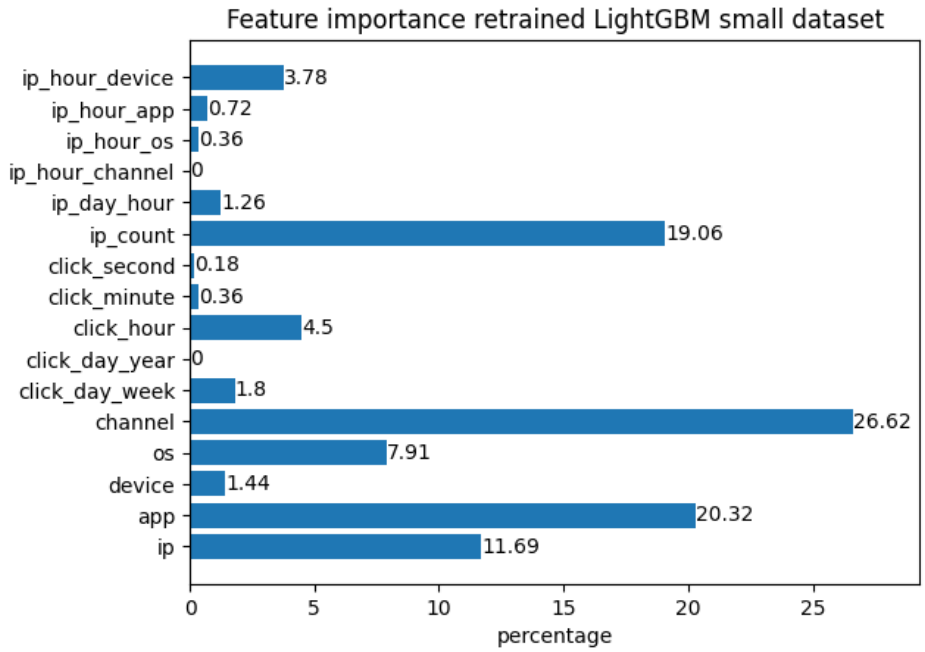


Secondly, although accuracy scores are relatively high, the balanced accuracy score and F1 score are not, especially the precision and F1 scores. When looking at the confusion matrices, and the sensitivity and specificity scores (Figures 7 and 8), it becomes evident that the model performs better in predicting the negative classes as opposed to the positive classes. When trained on the small dataset, the model predicted 27.8% of all negative instances incorrectly, and 12.8% when trained on the large dataset. Contrastingly, the model predicted 56.0% of all positive instances incorrectly when trained on the small dataset, and 82.7% when trained on the large dataset, illustrating the poor predictive performance in the positive class. It is likely that the unbalanced data, without applying a technique to counteract this problem, is the cause for this poor predictive performance. Moreover, the AUC scores (58.1% and 52.3%, respectively) illustrate that the model has poor discriminatory capability between the classes and that the model is not capable of improving considerably when moving the classification threshold.

Lastly, the feature importance charts in Figures 9 and 10 show that, when LightGBM is retrained on the small dataset, the features that contribute the most are the original features *app* and *channel*, which contribute to the model's performance for 20.3% and 26.6%, respectively. Additionally, the feature *ip_count*, from the feature engineering phase, contributes 19.1%, which is a considerable amount to the model's performance in comparison to the other features. Interestingly, when the model is retrained on the large dataset, more features from the feature engineering phase increase in importance, while the contribution of the *channel* feature substantially decreases from 26.6% to 4.8%. The original feature *app* remains amongst the most important features, contributing 16.0% to the model's performance. In addition, *ip_day_hour*, *ip_hour_os,* and *ip_hour_device* are the most important features when LightGBM is retrained on the large dataset, contributing 16.8%, 15.2%, and 15.8%, respectively.

Figure 9

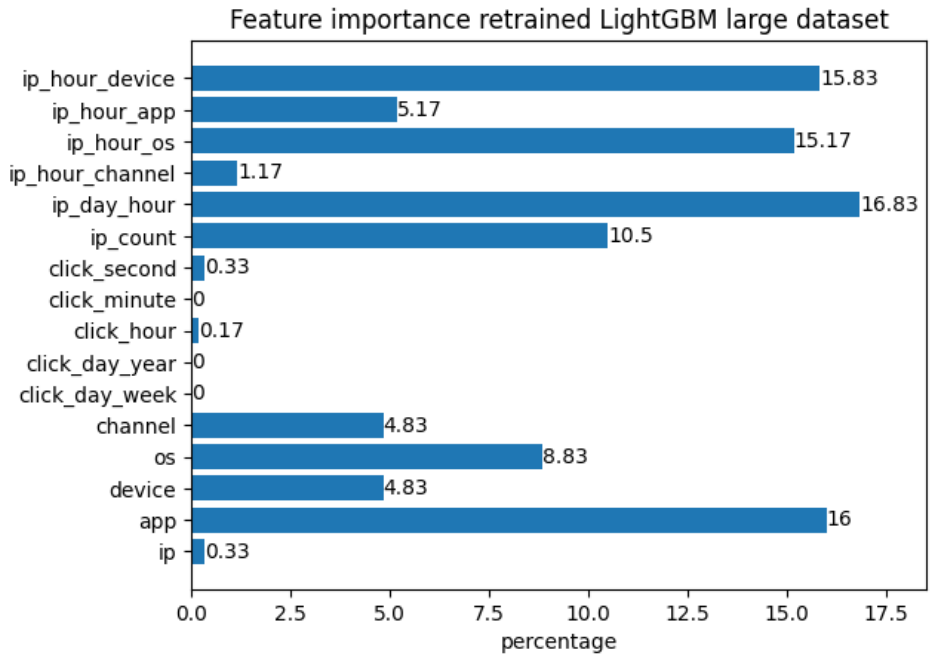
Feature importance retrained LightGBM one million samples



Note: the values on the x-axis of this graph display percentages. For an explanation of each feature exhibited on the y-axis, please see Appendix A and C.

Figure 10

Feature importance retrained light GBM full train set



Note: the values on the x-axis of this graph display percentages. For an explanation of each feature exhibited on the y-axis, please see Appendix A and C.

Discussion

Results and current literature

The goal of this study was to investigate to what degree TabNet was capable of accurately detecting click fraud. At first glance, the model showed an adequate performance when looking at the accuracy score (95.7%). In addition, TabNet has shown the ability to detect click fraud more accurately than other machine learning methods in the current literature (see Table 1). These findings are in line with the previous literature, stating the notable classification capabilities of TabNet on tabular data (Zhang et al., 2022; Joseph et al., 2022). However, to answer the sub-question investigated in the present study, TabNet was not able to outperform the two best-performing models, CNN-BiLSTM-RF (99.58%) and LightGBM (98%), on detecting click fraud, when considering their accuracy scores, which contradicts the findings in the current literature on TabNet, which displayed

superiority to all other benchmark models investigated in these studies (Zhang et al., 2022; Yan et al., 2021, Joseph et al., 2022).

Furthermore, the precision (4.2%) and F1 (7.9%) score illustrate that TabNet performs very poorly on predicting the positive class, predicting only 4.2% of all positive classes correctly. This also contradicts the current findings in the literature, as literature has shown a good performance of TabNet on both classes of an unbalanced dataset (Joseph et al., 2022). Therefore, it is reasonable to suggest that the poor performance in the minority class is a result of the methods used in the current study, not the classification capability of the model itself. It can be concluded that the model was still biased towards the prediction of the negative class due to the imbalanced data, even though SMOTE was applied to counteract this issue. Comparing the workflow of the current study to the workflow from Batool & Byun (2022), one considerable difference in the techniques is the application of k-fold cross-validation, in addition to SMOTE. Hence, the application of k-fold cross-validation might have positively affected the classification performance of TabNet on the minority class, making a less biased model.

Lastly, LightGBM was retrained in the current study for exploratory purposes, and to improve the comparative potential between LightGBM, CNN-BiLSTM-RF, and TabNet. The reason being is that the authors of the original paper used a larger dataset to train the model, compared to the study by Batool & Byun (2022) and the current study, which used one million randomly sampled rows for training. The results showed that the retrained LightGBM model was dependent on the amount of data that the model was trained on, achieving lower accuracy scores (i.e., 72.2% and 87.1%) compared to the original model (i.e., 98.0%). Likewise to TabNet, the retrained LightGBM showed poor performance in predicting positive instances, achieving a precision score of 0.2% on the small dataset (i.e., one million random samples) and 0.3% on the large dataset (i.e., the full training dataset in the current study). This raises the possibility that the model in the original paper by Minastireanu and Mesnita (2019) was equally poor in correctly predicting the positive class. Important to note is that this notion is based on the fact that the authors solely provided the accuracy score, giving no further insight into the model's classification performance in the positive and negative classes individually.

Interestingly, the AUC scores of the retrained LightGBM model on the small and large datasets were 58.1% and 52.3%, respectively, illustrating that the model performs inadequately in the discrimination between classes. However, TabNet achieved an AUC score of 85.3%, demonstrating that the performance of TabNet could be improved by moving the classification threshold. Considering this, one might argue that TabNet is preferred over LightGBM for the task at hand. At any rate, Batool & Byun's (2022) CNN-BiLSTM-RF model for click fraud detection remains superior to LightGBM, TabNet, and other benchmark models.

Limitations and future directions

Although the performance based on accuracy was sufficient, changes to some techniques that were applied during this study might positively affect the classification capability of TabNet on the task at hand. Firstly, due to time constraints and computational limitations, a subset of only one million rows was used for both training and hyperparameter tuning. Increasing this number enables the model to learn from a larger body of data which can improve its predictive performance (Banko & Brill, 2001). Moreover, using more data can potentially show superior hyperparameters to the ones unveiled during the hyperparameter optimization stage in the current study. For future research, it is advised to utilize all the data available and opt for GPU employment, so that an enhanced predictive performance of TabNet on detecting click fraud might be attained.

Secondly, random search was used to find the optimal hyperparameters for the employed model. Again, this technique was chosen over grid search to reduce run time. However, this raises the possibility that the true optimal hyperparameters were never found. Therefore, the use of grid search might yield different results, showing superior hyperparameters to the ones used in the current study, because this method searches for optimal hyperparameters exhaustively. Consequently, this might enhance TabNet's classification performance. Investigating the impact on the classification performance when using grid search for hyperparameter optimization in future research might yield improved results.

Lastly, a different evaluation metric for updating the model for each epoch might have yielded better results. In the current study, balanced accuracy was used because of the imbalanced data, with the intention of achieving a good performance in both the majority and minority classes. However, this

metric favors classifiers that perform better on the positive class instead of the negative class. In the current study, however, illustrated by the specificity and sensitivity scores, performance in the negative class was superior to performance in the positive class. Contrastingly, the metric F1 gives no weight to the number of correctly predicted negative instances (Wegier & Ksieniewicz, 2020). Therefore, to improve prediction on the positive instances, F1 might have been a better metric for updating the model at each epoch, resulting in a less biased model. To possibly reduce the bias even further, as mentioned in the previous sub-paragraph, it is advised to use k-fold cross-validation and move the classification threshold to improve classification performance in the minority class.

Yet, in the context of the current study, one might argue that predicting the positive classes is less important than the negative classes, because the positive instances indicate a download of the app, and are therefore unlikely to concern click fraud. At any rate, applying the aforementioned alterations in methods to future research might result in a superior classification performance of TabNet on the TalkingData dataset, compared to the currently best-performing models on click fraud detection.

Conclusion

The current study aimed to investigate how accurately click fraud could be detected using a novel canonical network for tabular data, TabNet, with the central research question being: How accurately can click fraud be detected using TabNet? The sub-question investigated during this study was: how does TabNet perform compared to the two currently most accurate methods on detecting click fraud? These models were CNN-BiLSTM-RF and LightGBM (Batool & Byun, 2022; Minastireanu & Mesnita, 2019).

The rise of online advertising marked the onset of click fraud, in which malicious publishers and unsportsmanlike competitors deplete the budgets of advertisers without contributing to their business goals. Many machine learning methods have been developed to detect click fraud, which can possibly prevent these illegal clicks from taking place. However, a deep learning model for tabular data called TabNet, had not been exploited for the task of click fraud detection to this date, even though the current literature has unveiled its promising classification potential. The model achieved an adequate performance with an accuracy score of 95.7%. Although the model outperformed other

methods investigated in the current literature, it was not able to outperform the two best-performing models for detecting click fraud. In addition, the model performed poorly on classifying instances in which an actual download of the app occurred (i.e., the minority class).

Nevertheless, the current study has important implications on both a societal and scientific level. Scientifically, this study demonstrates the ability of TabNet to accurately predict click fraud and, therefore, further supports the current literature showing that TabNet is an adequate model for classification tasks. Although the model was not able to outperform the best-performing models, implementing the changes proposed in the discussion might improve the performance to such an extent that the model is able to outperform the other methods established in the current literature. As for the implications on a societal level, the current study did not show a superior performance in comparison to the best-performing models. Therefore, organizations that are trying to encounter the problem of click fraud might want to opt for a better performing model to predict click fraud as accurately as currently possible.

References

Aljabri, M., & Mohammad, R.M.A. (2023). Click fraud detection for online advertising using machine learning. *Egyptian Informatics Journal*, *24*(2), 341-350 https://doi.org/10.1016/j.eij.2023.05.006

Arik, S. Ö., & Pfister, T. (2020). *TabNet: Attentive Interpretable Tabular Learning*. ArXiv. https://doi.org/10.48550/arXiv.1908.07442

Bach, M., Werner, A., Zywiec, J., & Pluskiewicz, W. (2017). The study of under- and over-sampling methods' utility in analysis of highly imbalanced data on osteoporosis. *Information Sciences*, *384*, 174-190. https://doi.org/10.1016/j.ins.2016.09.038

Banko, M., & Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. *Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference, 39*, 26-33. https://doi-org.tilburguniversity.idm.oclc.org/10.3115/1073012.1073017

Batool A., & Byun, Y-C. (2022). An Ensemble Architecture Based on Deep Learning Model for Click Fraud Detection in Pay-Per-Click Advertisement campaign *IEEE Access*, *10*, 113410-113426. https://doi.org/10.1109/ACCESS.2022.3211528

DreamQuark. (2021). PyTorch-TabNet: PyTorch implementation of TabNet, *version 4.1.0*. https://github.com/dreamquark-ai/tabnet

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3-5), 75-174. https://doi.org/10.1016/j.physrep.2009.11.002

Garcia, V., Mollineda, R. A., & Sánchez, J. S. (2009). Index of Balanced Accuracy: A Performance Measure for Skewed Class Distributions. *Pattern Recognition and Image Analysis*, 441-448. https://doi.org/10.1007/978-3-642-02172-5_57

Gohil, N. P., & Meniya, A. D. (2021). Click Ad Fraud Detection Using XGBoost Gradient Boosting Algorithm. *Computer and Information Science*, *1416*, 67-81. https://doi.org/10.1007/978-3-030-76776-1_5

Grand View Research. (2022). *Online Advertising Market Size & Trends*. https://shorturl.at/jBCI3

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wies, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H., Brett, M., Haldane, A., Fernandez del Rio, J., Wiebe, M., Peterson, P., …Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357-362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90-95. DOI: 10.1109/MCSE.2007.55

Jin, Y., Ren, Z., Wang, W., Zhang, Y., Zhou, L., Yao, X., & Wu, T. (2023). Classification of Alzheimer's disease using robust TabNet neural networks on genetic data. *Mathematical Biosciences and Engineering*, *20*(5), *8358-8374*. https://doi.org/10.3934/mbe.2023366

Jianzhuo, Y., Tianyu, X., Yonchuang, Y., & Hongxia, X. (2021). Rainfall Forecast Model Based on the TabNet Model. *Water*, *13*. https://doi.org/10.3390/w13091272

Joseph, L. P., Joseph, E. A., & Prasad, R. (2022). Explainable diabetes classification using hybrid Bayesian-optimized TabNet architecture. *Computers in Biology and Medicine*, *151*. https://doi.org/10.1016/j.compbiomed.2022.106178

Ke., G., Meng, A., Finley, T., Wang, T., Chen, W., Ma, Ye, Q., & Liu, T-Y. (2017). LightGBM: A highly efficient gradient boosting descision tree. *Advances in Neural Information Processing Systems*, *30*, 3146-3154. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Kearney, M. W. (2017). Cramér's V. In *The SAGE Encyclopedia of Communication Research Methods*. https://doi.org/10.4135/9781483381411

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, *18*(17), 1-5. http://jmlr.org/papers/v18/16-365.html

McKinney, W. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, *445,* 51-56. http://dx.doi.org/10.25080/Majora-92bf1922-00a

Meng, C. C., Lim, K. M., Lee, C. P., & Lim, J. Y. (2023). Credit Card Fraud using TabNet. *International Conference on Information and Communication Technology (ICoICT)*, 394-399. https://doi.org/10.1109/ICoICT58202.2023.10262711

Minastireanu, E.A., & Mesnita, G. (2019). Light GBM Machine Learning Algorithm to online Click Fraud Detection. *Journal of Information Assurance & Cybersecurity*. DOI: 10.5171/2019.263928.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825-2830. https://doi.org/10.48550/arXiv.1201.0490

PPC protect. (2021). *The Global PPC Click Fraud Report 2020-21*. https://www.searchenginejournal.com/the-global-ppc-click-fraud-report-2020-21/391493/

Spelman, V. S., & Porkodi, R. (2018). A Review on Handling Imbalanced Data. *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, 1-11. https://doi-org.tilburguniversity.idm.oclc.org/10.1109/ICCTCT.2018.8551020

Thejas, G. S., Boroojeni, K., Chandna, K., Bhatia, I., Iyengar, S. S., & Sunitha, N. R. (2019). Deep Learning-based Model to Fight Against Ad Click Fraud. *ACM SE '19: Proceedings of the 2019 ACM Southeast Conference*, 176-181. http://dx.doi.org/10.1145/3299815.3314453

Thejas, G. S., Dheeshjith, S., Iyengar, S. S., Sunitha, N. R., & Badrinath, P. (2021). A hybrid and effective learning approach for Click Fraud detection. *Machine Learning with Applications*, *3*. https://doi.org/10.1016/j.mlwa.2020.100016

Thölke, P., Mantilla-Ramos, Y-J., Abdelhedi, H., Maschke, C., Dehgan, A., Harel, Y., Kemtur, A., Berrada, L. M., Sahraoui, M., Young, T., Pépin, A. B., Khantour, C. E., Landry, M., Pascarella, A., Hadid, V., Combrisson, E., O'Byrne, J., & Jerbi, K. (2023). Class imbalance should not throw you off balance: Choosing the right classifiers and performance metrics for brain decoding with imbalanced data. *NeuroImage*, *277*. https://doi.org/10.1016/j.neuroimage.2023.120253

Trajcheva, S. (2023). *The ultimate list of click fraud statistics 2023*. Cheq. https://shorturl.at/hsv34

University of Baltimore. (2020). *Prof. Cavazos's Latest Report: Digital Ad Fraud Costs Will Rise to $35B Globally This Year*. University of Baltimore. http://www.ubalt.edu/news/news-releases.cfm?id=3621

Van Rossum, G., & Drake, F. L. (2009) *Python 3 Reference Manual*. Scotts Valley, CAL CreateSpace.

Wegier, W., & Ksieniewicz, P. (2020). Application of Imbalanced Data Classification Quality Metrics as Weighting Methods of the Ensemble Data Stream Classification Algorithms. *Entropy (Basel)*, *22*(8), 849-849. https://doi.org/10.3390/e22080849

Wilbur, K.C., & Zhu, Y. (2009). Click fraud. *Marketing Science*, *28*(2), 293-308. https://doi-org.tilburguniversity.idm.oclc.org/10.1287/mksc.1080.0397

Yadav, S., & Shukla, S. (2016). Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification. *2016 IEEE 6th International Conference on*

Advanced Computing (IACC), Bhimavaram, India, 2016, pp. 78-83, doi: 10.1109/IACC.2016.25.

Zhang, L., Ma, K., & Fang, W. (2022). A TabNet based card Fraud detection Algorithm with Feature Engineering. *2022 2nd international conference on consumer electronics and computer engineering (iccece)*, *911-914*. https://doi.org/10.1109/ICCECE54139.2022.9712822.

Zou, M., Gan, Z., Cao, R.C., Guan, C., & Leng, S. (2023). Similarity-navigated graph neural networks for node classification. *Information Sciences*, *633*, 41-69. https://doi.org/10.1016/j.ins.2023.03.057

Appendix

Appendix A

Feature name	Feature description
IP	IP address of click
App	App ID for marketing
Device	Device type ID of user mobile phone (e.g., iPhone 6 plus, iPhone 7, Huawei Mate 7, etc.)
Os	Os version ID of mobile ad publisher
Channel	Channel ID of mobile ad publisher
Click_time	Timestamp of click (UTC)
Attributed_time	If the user downloads the app after clicking an ad, this is the time for the app download
Is_attributed	The target that is to be predicted, indicating the app was downloaded

Appendix B

Package	Version
Pandas	1.5.3
NumPy	1.24.3
DaTetime (Python)	3.10.9
PyTorch_tabnet	4.1.0
Scikit-learn	1.2.1
Imblearn	0.10.1
Matplotlib	3.7.0
LightGBM	4.2.0

Appendix C

Feature name	Feature description
Click_day_week	The day in the week on which the click was performed
Click_day_year	The day in the year on which the click was performed
Click_hour	The hour on which the click was performed
Click_minute	The minute in which the click was performed
Ip_count	The number of times a unique IP address is found in the dataset
Ip_day_hour	The number of times a unique IP address has clicked on an advertisement within one hour on the same day
Ip_hour_channel	The number of times a unique IP address has clicked on the advertisement of a specific publisher within one hour
Ip_hour_os	The number of times a unique IP address, using a specific operating system, has clicked on an advertisement within one hour
Ip_hour_app	The number of times a unique IP address has clicked on an advertisement in a specific app within one hour.
Ip_hour_device	The number of times a unique IP address, using a specific device, has clicked on an advertisement within one hour

Appendix D

Hyperparameter	Examined values
n_d	8, 30, 64
n_steps	1, 5, 10
gamma	1, 1.5, 2
n_shared	1, 3 ,5
lambda_sparse	1e-6, 1e-5, 1e-3
mask_type	sparsemax, entmax

Appendix E

Hyperparameter	Value
max_depth	3
learning_rate	0.20
num_leaves	7
min_child_samples	100
max_bin	100
subsample	0.7
subsample_freq	1
colsample_bytree	0.9
scale_pos_weight	200
gamma	0.9
min_child_weight	0

Appendix F

Hyperparameter	Value
n_d	30
n_a	8
n_steps	5
gamma	2
n_independent	2
n_shared	1
epsilon	1e-15
seed	42
momentum	0.02
clip_value	2
lambda_sparse	1e-06
optimizer_fn	torch.optim.Adam
scheduler_fn	torch.optim.lr_scheduler.StepLR
device_name	‘cpu’
mask_type	sparsemax