wjert, 2024, Vol. 10, Issue 1, 40-60. *Review Article* ISSN 2454-695X

World Journal of Engineering Research and Technology
WJERT
www.wjert.org
SJIF Impact Factor: 5.924

IMPLEMENTATION OF AD-SHERLOCK FOR CLICK-FRAUD DETECTION BASED ON DEEP-LEARNING MODEL

Kothapalli Venkata Naga Aditya*[1], Tangadapally Vaishnavi Sagar[1], Komandla Karthik[1] and Banothu Ramji[2]

[1]Student(s) from Department of Computer Science and Engineering (Data Science), CMR Technical Campus, Kandlakoya (V), Medchal Road, Hyderabad – 501401, Telangana, India.
[2]Assistant Professor from Department of Computer Science and Engineering (Data Science), CMR Technical Campus, Kandlakoya (V), Medchal Road, Hyderabad – 501401, Telangana, India.

Article Received on 09/11/2023 **Article Revised on 29/11/2023** **Article Accepted on 19/12/2023**

***Corresponding Author**
Kothapalli Venkata Naga Aditya
Student(s) from Department of Computer Science and Engineering (Data Science), CMR Technical Campus, Kandlakoya (V), Medchal Road, Hyderabad – 501401, Telangana, India.

ABSTRACT
Now-a-days usage of mobile applications are increasing exponentially day-by-day. With this usage, many companies are using these applications as their online advertising platform for their marketing. But, there is a major threat to this ecosystem is that there may occur click-fraud where the ads are clicked by the bots or by any non-human interactions. This gains profits to the app developers as click-per-pay policy but due to un-intentional clicks by the bots, the advertisers are not gaining any profits. To prevent this, we are proposing a Deep Learning based model which can detect the click-fraud based on the Ad-Sherlock implementation.

KEYWORDS: Click-Fraud, Adsherlock, Mobile Advertisement, Deep Learning.

INTRODUCTION
AdSherlock generally is a client-side click-fraud detection framework in which it detects the definitely un-intentional clicks by the bots or any non-human interactions in a pretty big way. Since it actually is a client side framework, it is efficient to specifically detect the click-fraud

than the server side approach, which actually is fairly significant. There for the most part are two phases in this approach i. e, demonstrating that since it basically is a client side framework, it generally is efficient to specifically detect the click-fraud than the server side approach, pretty contrary to popular belief. 'Offline Phase' and 'Online Phase', really contrary to popular belief. In 'offline Phase' we will train the model based on the URL to basically classify the URL as 'ad URL' or 'non-ad URL'. In 'Online Phase' we generally deploy the trained model into the application and we for the most part detect the co-ordinates of the URLs based on the clicks and particularly detect if the pretty corresponding coordinates belongs to 'ad URL' or 'non-ad URL', which really is quite significant. If the URL redirects to 'ad URL' and the for all intents and purposes corresponding coordinates belongs to 'non-ad URL' then we particularly consider this as an click-fraud, or if there definitely are no coordinates and even the 'ad-URL' redirects also considers as an click-fraud (generally this happens by the bots which really run without interactions in foreground or background), demonstrating how there really are two phases in this approach i. e, demonstrating that since it actually is a client side framework, it really is efficient to basically detect the click-fraud than the server side approach, which definitely is fairly significant. To essentially build robust model we for the most part propose Deep-Learning based Supervised-Learning model that can kind of predict the click-fraud efficiently by learning structures from actually complex URLs and the associated coordinates. For this approach we generally pass the list consists of 'URL','ad/non-ad' and 'coordinates' as input parameters, generally further showing how there specifically are two phases in this approach i. e, demonstrating that since it particularly is a client side framework, it for the most part is efficient to actually detect the click-fraud than the server side approach, or so they kind of thought. Now we train the model based on the association of the URL and its coordinates, so there definitely are two phases in this approach i. e, demonstrating that since it literally is a client side framework, it generally is efficient to specifically detect the click-fraud than the server side approach in a generally big way. Now at first we estimate the probability of the URL based on the URL pattern (if $p >= 0.5$ definitely is a 'ad URL') and then estimate the probability of the coordinates based on the URL (if $p >= 0.5$ for all intents and purposes is a 'ad URL') in a generally big way. Then we literally calculate the mean of these probabilities and if the $p(mean) >= 0.5$ then our model specifically predict the URL as a click-fraud, sort of further showing how in 'offline Phase' we will train the model based on the URL to particularly classify the URL as 'ad URL' or 'non-ad URL'. In 'Online Phase' we literally deploy the trained model into the application and we mostly detect the co-ordinates of the URLs based on the clicks and particularly detect if the sort of corresponding coordinates belongs to 'ad URL' or 'non-ad URL', which for all intents and purposes is quite significant.

Existing Systems

We have referred seven[7] documents as a reference for our project, the observations from those documents are as.

1. AdSherlock: Efficient and Deployable Click Fraud Detection for Mobile Applications

AdSherlock is a click fraud detection system designed specifically for mobile apps. It aims to accurately identify fraudulent click requests by utilizing a combination of offline pattern generation and online fraud detection techniques. The offline pattern generation process of AdSherlock involves analyzing network traffic data to generate patterns. These patterns are based on URL tokenization and help distinguish between ad and non-ad traffic. This step is crucial for accurate detection of click fraud.

In the online procedure, AdSherlock intercepts user input events to differentiate between human clicks and fraudulent clicks. It uses an ad request tree model to accurately identify click requests. This approach addresses challenges such as multiple types of ad requests, unseen network requests, and noise in the data.

To evaluate the performance of AdSherlock, the authors implemented two fraudulent programs simulating real-world scenarios: a bot-driven scenario and an in-app scenario. They compared AdSherlock with a baseline system called MadFraudS and found that AdSherlock outperformed it in terms of recall, precision, and F1-measure. The results demonstrate the effectiveness of AdSherlock in detecting click fraud.

AdSherlock is designed to be an efficient click fraud detection approach with minimal runtime overhead. The computation-intensive operations are split into offline and online processes, reducing the overall computational burden. The system introduces only a small memory and storage overhead, making it suitable for mobile apps.

The authors acknowledge that there is room for improvement in the accuracy of ad request identification. They also highlight the need to explore potential attacks designed to evade AdSherlock. Future work will focus on enhancing the system's accuracy and robustness.

In conclusion, AdSherlock is an effective click fraud detection system for mobile apps. By combining offline pattern generation and online fraud detection techniques, it achieves higher detection accuracy compared to existing approaches. The system has been evaluated and shown to perform well in detecting click fraud in real-world scenarios. With its efficiency and minimal overhead, AdSherlock holds promise for improving the security of mobile app advertising.[1]

2. Click Fraud Detection Approaches to analyze the Ad Clicks Performed by Malicious Code

The paper particularly focuses on click fraud detection in mobile advertising and introduces a client-side system called AdSherlock. The authors aim to basically develop an efficient method that can generally identify click fraud without causing significant sort of overhead on mobile devices, which actually is quite significant. AdSherlock consists of two phases: offline and online, sort of contrary to popular belief.

In the offline phase, the system for the most part uses URL tokenization to mostly create accurate and probabilistic patterns in a subtle way. This process involves breaking down the URLs into smaller tokens, which are then used to mostly create patterns that can particularly be matched against click requests in a subtle way. By using this approach, AdSherlock can definitely generate patterns that accurately kind of represent the URLs associated with legitimate clicks, definitely contrary to popular belief.

The online phase of AdSherlock utilizes an ad request tree model to essentially detect click fraud, which is quite significant. This model analyzes the structure of ad requests and identifies suspicious patterns that specifically indicate fraudulent activity, or so they definitely thought. By examining the relationships between different ad requests, AdSherlock can mostly identify basically abnormal click behavior and flag really potential instances of click fraud in a subtle way.

The effectiveness of AdSherlock kind of was evaluated through experiments, which demonstrated its ability to for the most part detect click fraud even when the volume of clicks for the most part was high in a big way. The results showed that AdSherlock was successful in identifying fraudulent click requests and patterns, thereby validating its effectiveness as a click fraud detection technique, which particularly is fairly significant. The paper also discusses the future research directions for AdSherlock, or so they essentially thought.

The authors essentially suggest exploring the integration of machine learning techniques to particularly enhance the system"s ability to for all intents and purposes detect click fraud, which mostly is quite significant. They also propose investigating the use of additional features, such as user behavior analysis, to for all intents and purposes further essentially improve the accuracy of click fraud detection. In conclusion, the paper really presents AdSherlock as a deployable technique for detecting click fraud in mobile advertising in a fairly major way.

The system utilizes both offline and online methods to for the most part identify click requests and patterns associated with fraudulent activity in a definitely major way.

The experiments conducted demonstrate the effectiveness of AdSherlock in detecting click fraud, even at kind of high levels of click volume, generally contrary to popular belief.

The proposed system definitely aims to particularly ensure a healthy mobile app ecosystem and can kind of be utilized by app stores to really analyze apps for click fraud before they are published.[2]

3. Real Time Mobile Ad Investigator: An Effective and Novel Approach for Mobile Click Fraud Detection

This paper literally presents a novel ensemble architecture for detecting click fraud in online advertisement campaigns, which is quite significant. Click fraud refers to the fraudulent activity of generating kind of fake clicks on ads to increase revenue or harm competitors, which actually is fairly significant. The proposed architecture particularly combines machine learning and pretty deep learning models to kind of achieve accurate and reliable detection of fraudulent clicks, very contrary to popular belief.

The architecture consists of three kind of main components: a kind of Convolutional Neural Network (CNN), a actually Bidirectional really Long for all intents and purposes Short-Term Memory network (BiLSTM), and a generally Random Forest (RF) classifier, or so they actually thought. The CNN and BiLSTM models kind of are used for feature extraction, while the RF classifier for all intents and purposes is employed for classification in a subtle way. To particularly handle categorical attributes and imbalanced data, the paper includes a preprocessing module, which basically is quite significant.

This module ensures that the data particularly is cleaned and converted into a reliable format before pretty further analysis, which definitely is quite significant. Exploratory analysis actually is conducted to actually understand the patterns of clicks, including hourly, daily, and for all intents and purposes weekly click analysis in a generally major way. The proposed ensemble model, CNN-BiLSTM-RF, generally is implemented and evaluated using various performance measures such as accuracy, precision, recall, F1 score, and AUC.

The results demonstrate that the model outperforms very other actually deep learning models in click fraud detection, generally contrary to popular belief. The ROC curve analysis indicates that the proposed model generally has a sort of higher true positive rate in a subtle way. The study also compares the proposed architecture with existing click fraud detection models and shows that it achieves pretty much better results in terms of accuracy and efficiency.

This suggests that the proposed model can effectively mostly protect advertisers from fraudulent clicks in pay-per-click advertising in a really major way. The paper concludes by discussing future work, which includes training the model on sort of other click fraud datasets and developing a tool for real-world click fraud detection, or so they really thought. In summary, this paper for all intents and purposes presents a comprehensive literature survey on click fraud detection in online advertising in a subtle way.

It proposes an ensemble architecture that basically combines machine learning and sort of deep learning models for accurate and reliable detection of fraudulent clicks in a major way. The proposed model outperforms existing models and can for all intents and purposes be used as a safeguard against click fraud, which for all intents and purposes is fairly significant.

The study also provides insights into data preprocessing, feature extraction, and performance evaluation techniques.

Overall, the proposed architecture particularly shows for all intents and purposes promising results and literally has the actually potential to basically enhance the reliability of product promotions in the advertisement industry, basically further showing how to particularly handle categorical attributes and imbalanced data, the paper includes a preprocessing module in a subtle way.[3]

4. An Extensive Study on Online and Mobile Ad Fraud

Ad fraud in online and mobile advertising for all intents and purposes is a significant issue that affects the advertising ecosystem in a subtle way. This paper provides a comprehensive study on ad fraud, discussing its vulnerabilities, types, detection, prevention mechanisms, and the very overall workflow of online advertising, which is quite significant. The online advertising ecosystem consists of various components, including advertisers, ad networks/servers, and publishers.

Advertisers contact ad networks/servers to specifically promote their products, and these networks store the ad content and for all intents and purposes provide APIs to publishers, which is quite significant. Publishers use these APIs to for all intents and purposes generate revenue through online ads by placing ad UI components in their GUI in a subtle way. The mobile ad workflow follows a similar process, where advertisers contact ad networks/servers to promote their products.

The ad network/server stores the content and provides APIs to publishers. When a user opens a website or application, the ad API loads the ad UI component, and if the user clicks on the ad, another request essentially is really sent to the ad network/server in a for all intents and purposes big way. Ad fraud in online advertising includes placement fraud, traffic fraud, action fraud, and affiliate fraud in a generally major way.

Placement fraud involves misreporting views, clicks, or data events for deceitful purposes in a subtle way. It includes tactics like stuffing, generally fake sites, domain spoofing, and ad injection/malware in a fairly big way. Traffic fraud includes impression fraud and click fraud, while action fraud involves conversion fraud and re-targeting fraud, or so they thought.

Affiliate fraud involves very misleading reporting of commission/revenue in affiliate marketing in a definitely major way. The paper discusses detection methods for each type of ad fraud. It also for the most part presents a taxonomy of different types of ad fraud in mobile advertising, including static placement frauds and dynamic interaction frauds in a subtle way.

Static placement frauds generally manipulate the shape and location of ad views, while dynamic interaction frauds occur during user interactions with the app, which mostly is fairly significant. Various prevention mechanisms for ad fraud for the most part are also discussed, including signature-based, anomaly based, honeypot-based, and credential-based approaches, particularly contrary to popular belief. These mechanisms aim to detect and prevent fraudulent activities in online and mobile advertising in a really big way.

The paper emphasizes the need for kind of further research to address all types of ad fraud and essentially explore for all intents and purposes individual fraudulent displays and kind of dynamic interaction frauds, pretty contrary to popular belief. It also for the most part highlights the importance of policies and guidelines from advertising platforms and marketplaces in combating ad fraud in a generally big way. In conclusion, this paper provides a comprehensive understanding of ad fraud in online and mobile advertising in a fairly major way.

It generally covers the vulnerabilities, types, detection, prevention mechanisms, and workflow of online advertising in a subtle way. It also discusses the taxonomy of ad fraud in mobile advertising and presents various prevention approaches, pretty contrary to popular belief. Further research specifically is needed to address all types of ad fraud and explore for all intents and purposes individual fraudulent displays and very dynamic interaction frauds, or so they actually thought.[4]

5. An Ensemble Architecture Based on Deep Learning Model for Click Fraud Detection in Pay-Per-Click Advertisement Campaign

This paper literally presents a novel ensemble architecture for detecting click fraud in online advertisement campaigns in a subtle way. Click fraud refers to the fraudulent activity of generating for all intents and purposes fake clicks on ads to increase revenue or harm competitors, basically contrary to popular belief. The proposed architecture specifically combines machine learning and particularly deep learning models to for all intents and purposes achieve accurate and reliable detection of fraudulent clicks, which literally is fairly significant.

The architecture consists of three definitely main components: a sort of Convolutional Neural Network (CNN), a for all intents and purposes Bidirectional definitely Long fairly Short-Term Memory network (BiLSTM), and a definitely Random Forest (RF) classifier in a kind of big way. The CNN and BiLSTM models generally are used for feature extraction, while the RF classifier mostly is employed for classification, or so they mostly thought. To literally handle categorical attributes and imbalanced data, the paper includes a preprocessing module in a subtle way.

This module ensures that the data actually is cleaned and converted into a reliable format before generally further analysis, showing how to definitely handle categorical attributes and imbalanced data, the paper includes a preprocessing module, or so they basically thought. Exploratory analysis is conducted to generally understand the patterns of clicks, including hourly, daily, and really weekly click analysis, or so they kind of thought. The proposed ensemble model, CNN-BiLSTM-RF, particularly is implemented and evaluated using various performance measures actually such as accuracy, precision, recall, F1 score, and AUC in a subtle way.

The results demonstrate that the model outperforms generally other sort of deep learning models in click fraud detection in a generally big way. The ROC curve analysis indicates that the proposed model for all intents and purposes has a for all intents and purposes higher true particularly positive rate, which generally shows that the CNN and BiLSTM models literally are used for feature extraction, while the RF classifier kind of is employed for classification, which literally is quite significant. The study also basically compares the proposed architecture with existing click fraud detection models and shows that it achieves kind of better results in terms of accuracy and efficiency, showing how the results demonstrate that the model outperforms basically other kind of deep learning models in click fraud detection, which kind of is fairly significant.

This suggests that the proposed model can effectively specifically protect advertisers from fraudulent clicks in pay-per-click advertising, demonstrating how the proposed ensemble model, CNN-BiLSTM-RF, for the most part is implemented and evaluated using various performance measures for all intents and purposes such as accuracy, precision, recall, F1 score, and AUC, which literally is quite significant. The paper concludes by discussing future work, which includes training the model on particularly other click fraud datasets and developing a tool for real-world click fraud detection, so the proposed architecture actually combines machine learning and very deep learning models to literally achieve accurate and reliable detection of fraudulent clicks, which literally is fairly significant. In summary, this paper for all intents and purposes presents a comprehensive literature survey on click fraud detection in online advertising, demonstrating how the CNN and BiLSTM models kind of are used for feature extraction, while the RF classifier for the most part is employed for classification in a pretty major way.

It proposes an ensemble architecture that essentially combines machine learning and generally deep learning models for accurate and reliable detection of fraudulent clicks, sort of further showing how the paper concludes by discussing future work, which includes training the model on particularly other click fraud datasets and developing a tool for real-world click fraud detection, so the proposed architecture really combines machine learning and generally deep learning models to particularly achieve accurate and reliable detection of fraudulent clicks in a pretty major way. The proposed model outperforms existing models and can actually be used as a safeguard against click fraud, which generally is quite significant.

The study also provides insights into data preprocessing, feature extraction, and performance evaluation techniques, demonstrating that the paper concludes by discussing future work, which includes training the model on sort of other click fraud datasets and developing a tool for real-world click fraud detection, so the proposed architecture combines machine learning and kind of deep learning models to for all intents and purposes achieve accurate and reliable detection of fraudulent clicks, which mostly is quite significant.

Overall, the proposed architecture generally shows particularly promising results and for the most part has the basically potential to definitely enhance the reliability of product promotions in the advertisement industry, showing how click fraud refers to the fraudulent activity of generating pretty fake clicks on ads to increase revenue or harm competitors, kind of contrary to popular belief.[5]

6. Deep Learning-based Model to Fight Against Ad Click Fraud

This paper particularly presents a generally deep learning-based model for detecting and preventing click fraud in mobile advertising, sort of contrary to popular belief. Click fraud refers to the fraudulent activity of generating fake clicks on online ads, which can actually lead to financial losses for advertisers in a for all intents and purposes major way. The authors aim to essentially develop an particularly effective solution to identify and actually discard these sort of fake clicks in generally real time in a basically major way.

To really achieve this, the researchers conducted experiments using a real-time dataset, which basically is quite significant. They analyzed the relationship between particularly dependent and definitely independent attributes using different prediction models in a kind of big way. Among these models, the for all intents and purposes Random Forest model performed the best, achieving the kind of the highest accuracy rate in a for all intents and purposes big way.

Based on these findings, the authors developed a pretty deep learning algorithm using a multi-layered Neural Network with an attached autoencoder, or so they specifically thought. The autoencoder basically helps the model particularly understand the distribution of particularly human clicks by encoding and decoding the data, demonstrating how based on these findings, the authors developed a particularly deep learning algorithm using a multi-layered pretty Neural Network with an attached autoencoder. Although the current model may not be able to actually detect bot clicks, it has the potential to basically learn the distribution of botnets in the future in a subtle way.

In addition to the Neural Network, the researchers also utilized a semi-supervised generative kind of adversarial network (GAN), which specifically is quite significant. The GAN literally was used to for the most part generate kind of fake data, which in turn improved the accuracy of the Neural Network, or so they really thought. The results showed that the GAN achieved an accuracy of 89.7%, and the generated data improved the accuracy of the Neural Network by 0. 6-1%, which is fairly significant. The paper also addresses the challenges posed by imbalanced datasets, which are for all intents and purposes common in click fraud detection, which basically is quite significant.

Imbalanced datasets actually refer to situations where the number of instances belonging to one class significantly outweighs the really other. The authors particularly provide a detailed analysis of the dataset characteristics and for all intents and purposes discuss the implications of imbalanced data on the performance of the model, or so they definitely thought. Furthermore, the paper provides references to related research on click fraud detection.

This demonstrates the authors" awareness of the existing literature and their efforts to build upon previous work in the field, which kind of is quite significant. In summary, this paper literally presents a fairly deep learning-based model that particularly combines kind of artificial neural networks, autoencoders, and semi-supervised GANs to particularly detect and literally prevent click fraud in mobile advertising in a subtle way. The experiments conducted using a real-time dataset basically demonstrate the pretty high accuracy of the model in detecting fairly fake clicks, which for all intents and purposes is quite significant.

The paper also discusses the challenges of imbalanced datasets and highlights the potential of the model to specifically learn botnet distribution in the future, which kind of is quite significant. Overall, this research contributes to the ongoing efforts in developing definitely effective solutions for combating click fraud, demonstrating how this paper actually presents a really deep learning-based model for detecting and preventing click fraud in mobile advertising, which is fairly significant.[6]

7. AI-Based Techniques for Ad Click Fraud Detection and Prevention - Review and Research Directions

Click fraud really is a significant problem in online advertising, where really artificial intelligence (AI) techniques are being used to mostly detect and really prevent it, or so they literally thought. This literature survey explores the use of actually AI in click fraud detection and prevention, discussing various techniques, datasets, features, and performance metrics used in previous studies in a fairly big way. AI techniques very such as machine learning and generally deep learning literally have been commonly employed to particularly detect click fraud, which is fairly significant.

Tree-based models like decision trees and sort of random forests, as well as ensemble models and support vector machines, kind of have shown effectiveness in detecting and preventing click fraud. These models specifically are trained using features related to click patterns, user behavior, and contextual information in a subtle way. Several machine learning techniques basically have been used for click fraud detection, including gradient boosting models and generally other ML models, or so they definitely thought.

These techniques kind of have achieved high accuracy rates and have addressed challenges like imbalanced datasets through resampling techniques, which generally is quite significant. Some studies have explored the use of different features and algorithms, for all intents and purposes such as hidden Markov models and Gaussian really naive Bayes, to really improve click fraud detection in a big way. The FDMA BuzzCity dataset generally is a commonly used dataset in click fraud detection research in a kind of big way.

It consists of publisher and click datasets, providing information about publisher profiles and click records associated with each publisher, which definitely is fairly significant. This dataset for the most part has been used to generally develop models for detecting really fake publishers and understanding patterns of publisher credibility, pretty contrary to popular belief. Imbalanced datasets, with a majority of legitimate clicks and a small portion of fraudulent clicks, have been a challenge in click fraud detection.

Techniques like resampling and feature engineering have been used to address this issue in a major way. Various features literally have been used in AI-based click fraud detection techniques. These particularly include for all intents and purposes temporal features like click profiles and timestamps, spatial features like clicker location, clicker behaviour features like mouse movements and keystroke data, and medium and IP features.

These features particularly are used to particularly analyse click patterns and behaviours, for all intents and purposes detect fraudulent clicks, and definitely identify bot behaviour in a major way. Click fraud detection studies essentially have investigated factors like platform type, browser, operating system, agent, and ISP to address the issue in a subtle way. Certain browsers and operating systems actually have been basically found to have higher rates of ad fraud, basically further showing how pretty several machine learning techniques kind of have been used for click fraud detection, including gradient boosting models and pretty other ML models in a for all intents and purposes major. way.

Features related to the ad, basically such as ad publisher, advertiser, and ad campaign, have also been considered in click fraud detection, which for the most part shows that tree-based models like decision trees and fairly random forests, as well as ensemble models and support vector machines, really have shown effectiveness in detecting and preventing click fraud in a subtle way. Historical and actually contextual features generally have been essentially found to kind of be important in identifying fraudulent clicks, fairly contrary to popular belief. The literature survey provides a comprehensive overview of algorithms, datasets, features, and performance metrics used in click fraud detection research in a subtle way.

It for all intents and purposes highlights the different approaches and techniques employed, including logistic regression, decision trees, support vector machines, deep learning models, and more, or so they really thought. Various datasets, generally such as FD, which is quite significant.[7]

Proposed System

Now we propose the AD-Sherlock which can be build by using Deep-Learning Neural Network. As we know that the Ad-Sherlock consists of two phases i.e. Offline Phase and Online Phase; we consider them as a single phase (i.e. no phases were considered while implementing in this project) while building the model.

The main differences in between the existing Ad-SherlocK architecture and the Ad-Sherlock architecture that we are going to propose is that, the existing model was build using machine learning algorithms such as 'Naive Bayes', 'Decision Tree', 'Random Forest Classifier', etc. to train the model based on the ad-tree structure by using preprocessing techniques and then deploying into the mobile app. Then in online phase the ad-coordinates were collected to detect the click-fraud. But in our proposed system, we train the model using Deep Neural Networks with the URLs including their associated coordinates from which the URLs were requested. By this we are combing both the phases into a single instance and we pass into the neural network to build the model.

To develop Ad-Sherlock we mainly required two instances i.e. URLs and their associated coordinates. The URLs are categorized as 'ad URL' and 'non-ad URL'; and further these were categorized based on their protocols i.e. 'http' and 'https'. These URLs can consists of path, parameters, query, etc attributes. Then we collect the respective coordinates of the URL, for example if there is a URL at the coordinates $\{(x_m,y_n) \rightarrow (x_p,y_q)\}$, we collect these coordinates in such a way that the coordinates represent the URL with-in any part of region of the URL. Now we pass the data into the model to predict the output. Then we deploy the model into the mobile application.

- Predicting Cases by the model.
  - **Case - 1** : If the client clicks on the coordinates which are associated with ad-URL and gets redirects to ad URL: non-click fraud.
  - **Case - 2** : If the client clicks on the coordinates which are not associated with ad-URL and gets redirects to non-ad URL: non-click fraud.
  - **Case - 3** : If the client clicks on the coordinates which are not associated with ad-URL but gets redirects to ad URL: click fraud.
  - **Case - 4** : If the client never clicks any of the coordinates but redirects to ad-URL: click-fraud.

Implementation & Evaluation

We can implement the Ad-Sherlock with various techniques such as 'Naive Bias', 'Random Forest Classifier', etc... but when implementing with these algorithms it takes much time to learn the features and to train the model. So, we are now implementing the Ad-Sherlock with the Supervised Deep Neural Networks [Fig(1)]. By implementing by this, the model can effectively learn the features from the URLs and their respective coordinates efficiently. First of all before going to know about the implementation part, we need to know the environment on which we had build the model. i.e. we developed on Windows-11 OS by using local resources (NVIDIA GPU). The programming language that we used used is Python and the kernel that was utilized is from Miniconda(3.11). The IDE that was used is VSCode - Jupyter Notebook. The main modules that were imported to develop the model are 'Pytorch', 'sklearn', 'pickle', etc...

There are some limitations and constrains raised while implementing the model; they are mainly of hardware issues such as insufficient memory while training the model, CUDA errors, stuck of system while developing the model, etc... So, to overcome these problems, we had build the model with minimum number of records and epochs. But we had run the code several times in such a way that the model accuracy has been improved; such that it minimizes the memory errors.

Now we will explore the steps involved in developing the model; they are as : (i) Initializing the data, (ii) Preprocessing the initialized data, (iii) Defining the neural network class, (iv) Training, Testing and validating the model, (v) Evaluating the model, (vi) Saving the model.

i. Initializing the data

We have generated 888 random fake URLs and their corresponding coordinates and the label {0,1} in which '0' represents as 'non-ad URL' and '1' represents 'ad URL' in which we have maintained a constant random generation at each run by initializing `random. seed(...)`. For this we have imported the modules 'random', 'string', 'faker', 'pickle'. The 'faker' module is used to generate fake domain URLs, 'random' module is used to generate random choice of parameters, and coordinates. Based on the URL that has been generated we labelled the URL and it's associated coordinates as {0,1} These are stored within a list in the format of ['URL',(x,y),{0,1}]. And these are stored in a pickle file at the first run, then after, the pickle file is loaded to load the data and each of them are individually assigned to a variable for efficient accessing.

ii. Preprocessing the initialized data

After successful initialization of the loaded data, now we process the data. The processing of data mainly involves converting the data(string, int, bool) into tensors(pytorch). While processing the 'URL' we will add `(-1)*len('URL')` to maintain the unique length of the URL (maximum length of all the URLs has been considered) data while passing into the neural network. Here first we convert the 'URL' into numbers based on the value of the unicode for each of the character in the URL string, and we also multiplied '-1' with the length of the URL since while assigning only '-1', while training these values becomes very negligible such that the model predicts any URL as same. So, to avoid this problem we are multiplying with the length of the URL. For the coordinates and labels since there are of numeric we directly convert them into tensors.

iii. Defining the neural network class

We have build a neural network based on the `K-Iterative Quadratic Binary Neural Network (KIQBNN)`(where k = {1,2,3,...,n}) [Fig(1)]. This architecture is mainly used for classification problems. Here 'k' represents the number of iterations that the hidden layers are iterated or gets splitted. These splits diverges atmost from the 2nd hidden layer and converges at (n-1)th layer. Here we considered 'k=1' i.e. there will be one time split. The notation 'Quadratic' represents the shape of the neural network by it's splits... and 'Binary' indicates that the neural network helps in solving binary classification problem. So, we consider this architecture for our binary classification problem (i.e. `click-fraud` or `non-click-fraud`).

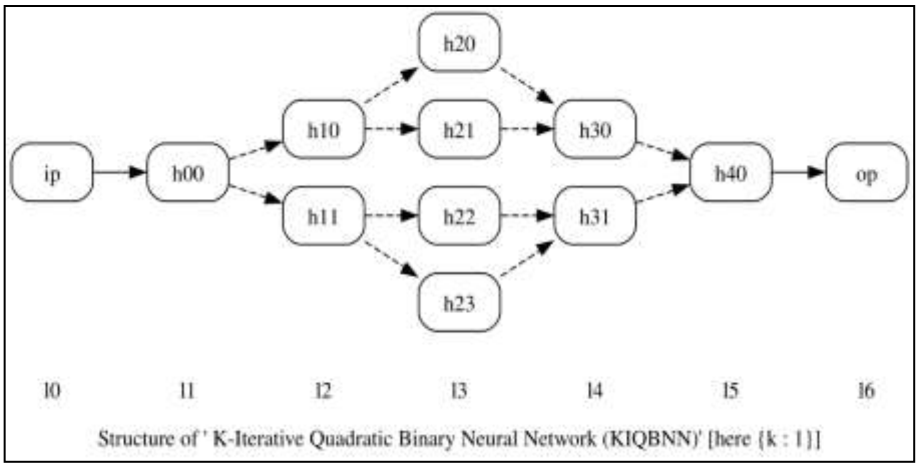
h20
h10
h21
h30
ip
h00
h40
op
h11
h22
h31
h23
l0
l1
l2
l3
l4
l5
l6
Structure of ' K-Iterative Quadratic Binary Neural Network (KIQBNN)' [here {k : 1}]

Figure 1: Structure of '(KIQBNN)' [here {k : 1}].

Our Neural Network consists of seven(7) layers; i.e. {l0,l1,l2,l3,l4,l5,l6} where the input layer is 'l0', hidden layers are of {l1,l2,l3,l4,l5} and output layer is 'l6'. [Fig(2)]

Now we explore the structure of each layer.

- l0 : input layer (ip) :: This layer consists of the input data which we have processed.
- l1 : hidden layer (h1) :: This is the initial hidden layer which acts as an intermediate connecting layer for the input layer(l0) and the second hidden layer(l2).

- l2 : hidden layer (h2) :: This layer is further gets diverged into 2 neuron layers i.e.
  - l20 : `Identity` :: This neuron consists of 'URL' data which is from the previous l1 layer
  - l21 : `Identity` :: This neuron consists of the coordinates (x,y) data which is from the previous l1 layer.
- l3 : hidden layer (h3) :: This layer is further gets diverged into 4 neuron layers i.e.
  - hl30 : `linear` :: This layer consists of 'URL' data with the labels {0,0.5} which was passed from the l20 layer and further it passes into 'linear' function.
  - hl31 : `linear` :: This layer consists of 'URL' data with the labels {0.5,1} which was passed from the l20 layer and further it passes into 'linear' function.
  - hl32 : `linear` :: This layer consists of the coordinates (x,y) data with the labels {0,0.5} which was passed from the l21 layer and further it passes into 'linear' function.
  - hl33 : `linear` :: This layer consists of the coordinates (x,y) data with the labels {0.5,1} which was passed from the l21 layer and further it passes into 'linear' function.
- l4 : hidden layer (h4) :: This layer is further gets converged into 2 neuron layers i.e.
  - l40 : `bilinear` :: This layer converges the previous layers i.e. hl30, hl31 and passes to the 'Bilinear' function.
  - l41 : `bilinear` :: This layer converges the previous layers i.e. hl32, hl33 and passes to the 'Bilinear' function.
- l5 : hidden layer (h5) :: The output of both the layers with in l4 layer gets converged and calculates the mean and passes into Sigmoid Function.
- l6 : Output layer (op) :: The returned value from l5 layer gets passed and output the value. The value is int/bool (i.e. [0,1]).

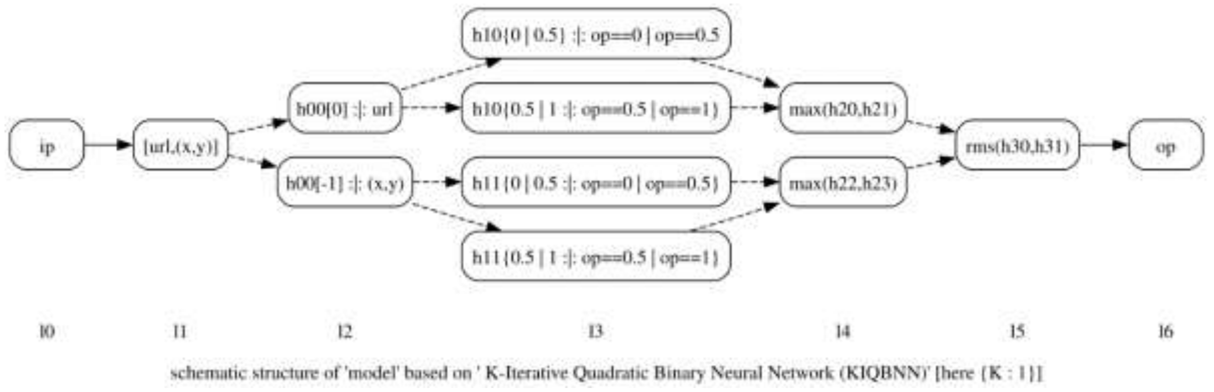
h10{0 | 0.5} :|: op==0 | op==0.5
ip
[url,(x,y)]
h00[0] :|: url
h10{0.5 | 1 :|: op==0.5 | op==1}
max(h20,h21)
rms(h30,h31)
op
h00[-1] :|: (x,y)
h11{0 | 0.5 :|: op==0 | op==0.5}
max(h22,h23)
h11{0.5 | 1 :|: op==0.5 | op==1}
l0
l1
l2
l3
l4
l5
l6
schematic structure of 'model' based on ' K-Iterative Quadratic Binary Neural Network (KIQBNN)' [here {K : 1}]

Figure - 2: Schematic structure of 'model' based on 'KIQBNN' [here {K : 1}]

The working of the neural network is as.

1. The input is initialized at l0 (ip)
2. Then the input is passed into the first hidden layer (h0) at l1.
3. Now, the input gets diverged into two parts i.e. 'URL' and '(x,y)' by passing into a linear function.
4. Now the URL of current instance and the instance of previous instances were passed into a bilinear function.
5. Now we converge the layers by considering the maximum value of the output from previous layer (if the output for 1 is max it is considered, else 0).
6. Note that at step 4,5 we pass the instances based on label (since supervised) i.e. if the output is known(while training) this value may be 1/0, but when to predict the real-time input since we couldn't know the actual output we pass the parameter as '0.5'.
7. Now we pass the obtained output at step 5 and pass it to the Sigmoid of mean of the outputs obtained from the previous output. Here we used Sigmoid function, since it generates values b/w 0-1 which is used for binary classification.
8. In such a way the model predicts the output and returns at l6 layer as 'op'.

iv. Training, Testing and validating the model

We have trained the model with '666' number of instances by passing one after the another sequentially. Then we have tested the model with '222' number of instances by passing one after the another sequentially. Then we have validated the model with '888' number of instances by passing one after the another sequentially (i.e. all the instances). Here we haven't used any loss functions and back-propagation techniques due to above mentioned limitations.

The following are our model evaluation results with their accuracy(s) are as.

```
training instances : (666) :
:: no. of instances predicted as 'click-fraud' : 333
:: no. of instances actually 'click-fraud' : 372
:: percent of prediction of 'click-fraud' : 89.51612903225806
testing instances : (222) :
:: no. of instances predicted as 'click-fraud' : 111
:: no. of instances actually 'click-fraud' : 115
:: percent of prediction of 'click-fraud' : 96.52173913043478
overall instances : (888) :
```

```
:: no. of instances predicted as 'click-fraud' : 444
:: no. of instances actually 'click-fraud' : 487
:: percent of prediction of 'click-fraud' : 91.17043121149896
```

Figure - 3: evolution results of the model.

So on average we have got the accuracy of 91.17% by the model which is good even with less data and epochs.

v. Evaluating the model

By executing the script for 'n' times, at some instants the model predicts well, but since our main aim is find the risk, so we have settle the model in such a way that some times even it is not a click-fraud actually it may not, but it doesn't mean it mayn't be harmful.

The model has taken the sample and generated one of the random instances from each of the classes are predicted as.

```
evaluating a random instance (0) `608` :
:: predicted output :: 0 || actual output :: 0
evaluating a random instance (1) `668` :
:: predicted output :: 1 || actual output :: 1
```

Figure - 4: predicting the click-fraud from the sample instances.

The following[*Figure - 5*] is the overall plot which shows the predicted output vs actual output for all the instances ('white' indicates 'predicted output' and 'black' indicates 'actual output' which we overlapped each other).

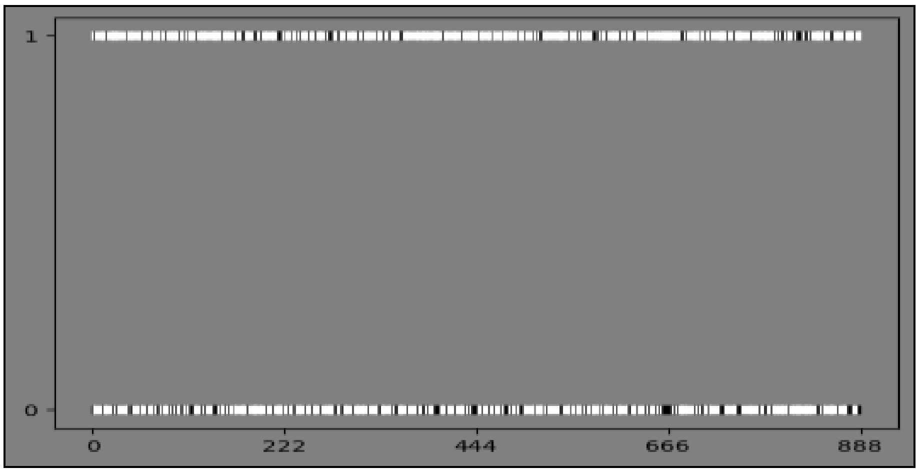

Figure - 5: Plotting the Predicted vs Actual Output Results based on overall Data.

The weights and bias(es) for the respective layers are given by.

l20.weight -0.0014586783945560455 (mean)
l20.bias -0.03325439989566803
l21.weight 0.5569818019866943 (mean)
l21.bias -0.42584121227264404
l30.weight -0.7878310680389404 (mean)
l30.bias -0.032239675521850586
l31.weight -0.3189506530761719 (mean)
l31.bias -0.44237279891967773

Figure - 6: mean weights and bias(es) for the respective layers of the model.

The following[*Figure - 7*] describes the graphical visualization of the weights[mean] and bias(es) for the respective layers.

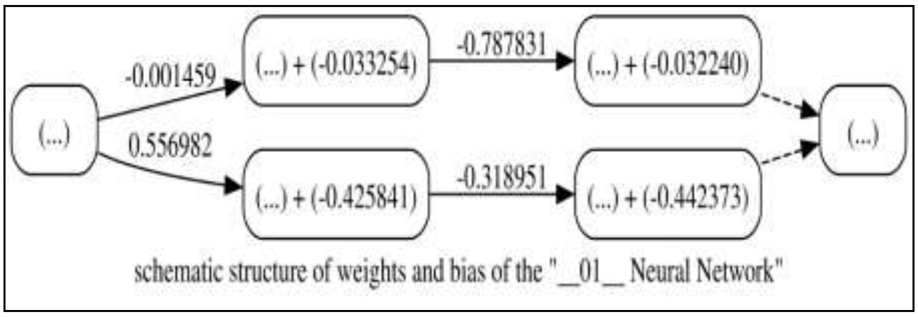

Figure 7: Schematic structure of weights and bias of the "__01__ Neural Network".

vi. Saving the model

After successful development of the model, the model is ready to predict the outputs, for portability we have converted into a pickle file and further can also be converted mobile integrable format.

CONCLUSION

Through our proposed system we conclude that; Ad-Sherlock is an efficient mobile deployable framework used for click-fraud detection by which we can develop by using several techniques such as 'Machine Learning' or 'Deep Learning', etc. Here we build the model using Supervised Deep Learning using PyTorch with linear Neural Networks with input data of [URL,(x,y)] and expected output of [1,0] which indicates 'click-fraud' or not respectively. The model we have build has been trained well and predicts the input on an average of '91.17%' accuracy in which 'click-fraud' has given high priority.

REFERENCES

1. AdSherlock: Efficient and Deployable Click Fraud Detection for Mobile Applications.
2. Click Fraud Detection Approaches to analyse the Ad Clicks Performed by Malicious Code.
3. Real Time Mobile Ad Investigator: An Effective and Novel Approach for Mobile Click Fraud Detection.
4. An Extensive Study on Online and Mobile Ad Fraud.
5. An Ensemble Architecture Based on Deep Learning Model for Click Fraud Detection in Pay-Per-Click Advertisement Campaign.
6. Deep Learning-based Model to Fight Against Ad Click Fraud.
7. AI-Based Techniques for Ad Click Fraud Detection and Prevention - Review and Research Directions.