DIC PROJECT PHASE 3:

Team Members:
Sujith Gollamudi – CSE 587
Bahalul Khan Pathan – CSE 587
Sai Krishna Aditya Gorthi – CSE 587

1. Problem Statement:

Click fraud is a prevalent issue in the online advertising industry, leading to misleading click data and wasted resources. TalkingData, China's largest independent big data service platform, handles an overwhelming volume of mobile ad clicks, a significant portion of which is potentially fraudulent. The current approach involves measuring user click journeys and flagging IP addresses with suspicious click behavior. However, to stay ahead of fraudsters, TalkingData seeks a predictive model to forecast app downloads after mobile ad clicks.

1.1. Background:

With China being the largest mobile market globally, and TalkingData covering over 70% of active mobile devices, the scale of fraudulent traffic is enormous. This poses a significant challenge for app developers relying on accurate click data for advertising decisions. The existing preventive measures, such as IP and device blacklists, are effective but not fool proof. The challenge is to develop an advanced algorithm that can predict whether a user will download an app after clicking a mobile ad, thereby enhancing the effectiveness of fraud detection.

1.2 Objectives:

1. Build a predictive model to identify users likely to download an app after clicking a mobile ad.
2. Improve fraud detection efficiency beyond the current blacklist approach.
3. Enhance the accuracy and reliability of click data for app developers.

1.3 Significance:

Mitigate financial losses due to click fraud for app developers and advertisers. Provide a proactive solution to stay ahead of evolving click fraud strategies. Contribute to a more reliable and trustworthy online advertising ecosystem.

2. Data Sources

The dataset for this project has been sourced from **Kaggle's TalkingData AdTracking Fraud Detection competition**. The dataset consists of click records, including features such as IP address, app ID, device type, OS version, channel ID, click timestamp, attributed time (if app was downloaded), and the target variable indicating app downloads.
Dataset Source: TalkingData Ad Tracking Fraud Detection - https://www.kaggle.com/competitions/talkingdata-adtracking-fraud-detection/data

3. Model Selection and Optimization Strategy

During Phase 2 of our product development, we strategically employed Gradient Boosting and Random Forest algorithms as our primary machine learning models. These choices were informed by their proven effectiveness in handling diverse datasets and delivering robust predictions.

3.1 Hyperparameter Tuning with Optuna:

To enhance the performance of these models, we leveraged the capabilities of Optuna for hyperparameter tuning. Optuna is an open-source hyperparameter optimization framework that employs a sequential model-based optimization (SMBO) strategy. The primary goal of hyperparameter tuning is to find the optimal set of hyperparameters that maximizes the model's predictive accuracy.

Optuna efficiently explores the hyperparameter space, intelligently sampling different configurations and iteratively narrowing down the search based on the models' performance. By doing so, it automates the tedious process of manual hyperparameter tuning, saving time and resources.

3.2 Decision to Prioritize Untuned Models:

Despite the concerted effort invested in hyperparameter tuning using Optuna, a comprehensive evaluation revealed a noteworthy observation: the original, untuned versions of the Gradient Boosting and Random Forest models consistently outperformed their tuned counterparts.

3.3 Prioritizing Untuned Models:

The decision to prioritize the untuned models was rooted in their inherent superior predictive power. The untuned models demonstrated robust performance across various datasets without the need for fine-tuning specific hyperparameters.

This approach aligns with our commitment to delivering reliable and accurate predictions to our users. By opting for the untuned models, we strike a balance between simplicity and effectiveness. This decision allows for seamless integration into our website and ensures a high standard of predictive accuracy suitable for real-time applications.

.

3.4 Implications and Considerations:

While hyperparameter tuning is a valuable practice for many machine learning scenarios, our specific context led us to favor the simplicity and consistently strong performance of the untuned models. This decision is influenced by the nature of the click fraud prediction

task, where the untuned models demonstrated remarkable efficacy without the need for additional fine-tuning.

Moving forward, this approach provides flexibility for future model updates. If necessary, we can revisit hyperparameter tuning using Optuna or other methods to adapt to evolving data patterns or incorporate new features.

3.5 Conclusion:

In conclusion, our use of Optuna for hyperparameter tuning reflects a proactive strategy to extract the best possible performance from our chosen models. While the untuned models emerged as the preferred choice in our specific scenario, the decision-making process is a testament to our dedication to delivering optimal predictive accuracy and maintaining the adaptability of our system.

4. Creating Pickle Files of Trained Models

During the model training phase, we employed a strategic approach to save our trained models using the pickling technique. Pickling is a method for serializing Python objects, enabling us to store the entire model state, including its architecture and learned parameters. This approach offers several advantages, such as quick and efficient model deployment and the ability to reproduce predictions on new data without retraining.

By pickling our trained models, we ensure a smooth transition from the training environment to real-world applications, upholding our dedication to delivering reliable and efficient predictive solutions.

So, by importing pickle saved the random forest classifier model as “final_random_forest_model.pkl” file which is imported further and used in the application and saved cleaned and balanced data to “Cleaned_data.pkl” file in order to save time on rerunning the data.

5. Building a Webpage using “streamlit” : Streamlit Web App for Click Fraud Prediction

5.0 Installing Streamlit and Setting Up the Environment

To create the Streamlit web app for click fraud prediction, the following steps were undertaken:

5.0.1 Installing Streamlit

Streamlit was installed by following the instructions provided on the Streamlit website, using the command `pip install streamlit`.

5.0.2 Creating a Working Directory

A dedicated working directory named "sample" was established to organize and store the Streamlit app files.

5.0.3 Environmental Variable Setup

Environmental variables were configured to ensure seamless execution of Streamlit in the local environment.

5.1 Writing Code for Streamlit Webpage

5.1.1 Model Loading and Preprocessing

The web app efficiently loads a pre-trained Random Forest model (final_random_forest_model.pkl) and its corresponding preprocessed data (cleaned_data.pkl). During preprocessing, the 'click_time' column is transformed into datetime format, and the 'hour' is extracted.

5.1.2 Streamlit Sidebar and Options

Utilizing Streamlit's sidebar, the app provides user-friendly options. The sidebar incorporates a distinctive company logo (Picture1.jpeg) and offers selections for 'Home,' 'Predictions,' and 'Visualizations.'

5.1.3 Home Section

The 'Home' section is adorned with a visually appealing background image (images.jpeg). Users can seamlessly navigate to other sections by choosing options from the sidebar.

5.1.4 Predictions Section

In the 'Predictions' section, users input features such as 'IP,' 'App,' 'Device,' 'OS,' 'Channel,' and 'Click Time.' By clicking 'Validate,' the app triggers preprocessing and predictions using the loaded Random Forest model. The results are then displayed, indicating whether the app is downloaded or not.

5.1.5 Visualizations Section

The 'Visualizations' section enriches user experience by providing a variety of visualization options through a dropdown menu in the sidebar. These options include visualizations for top apps, attributed clicks per hour, top apps with attribution, top devices, OS, channels, and IPs.

5.1.6 Visualization Plots

Visualizations are crafted using Seaborn and Matplotlib, seamlessly integrated with Streamlit's st.pyplot() function. Bar charts are employed to showcase top apps, clicks per hour, top devices with attribution, top OS with attribution, top channels, and top IPs.

5.1.7 User Interaction

The app actively promotes user interaction through intuitive input fields, buttons, and engaging visualizations. Users can delve into predictions and visualizations based on their inputs. Additionally, **users have the flexibility to upload their datasets**, allowing the app to generate visualizations tailored to the provided data.

5.1.8 Overall Structure

The overall structure of the app is seamlessly designed, leveraging Streamlit's functionalities to seamlessly integrate user interfaces, visualizations, and predictive modeling. The code is thoughtfully commented, enhancing clarity, and each section is encapsulated within conditional statements based on user options.

This Streamlit web app stands as a testament to its interactive and user-friendly interface, providing an immersive platform for exploring click fraud predictions and related visualizations. Users can effortlessly input data, receive predictions, and gain valuable insights through a rich array of charts and plots. The inclusion of user-unloadable datasets further extends the app's versatility, allowing users to tailor visualizations to their specific data.

5.2 Running the Streamlit Web App And Webpage Visualization

To launch the Streamlit web app for click fraud prediction, follow these steps:

1. Open a terminal or command prompt.
2. Navigate to the working directory "sample" where the Streamlit app files are located.
3. Execute the following command to run the Streamlit app:
   streamlit run DICwebsite1.py
4. After running the command, Streamlit will initiate the app, and a local development server will start.
5. Look for the output in the terminal, which will include a local URL (usually starting with `http://localhost`).
6. Open a web browser and enter the provided URL to access the Streamlit web app.
7. Explore the different sections, including 'Home,' 'Predictions,' and 'Visualizations,' through the user-friendly interface.
8. Input data in the 'Predictions' section to see real-time predictions based on the pre-trained Random Forest model.

9. Navigate through various visualizations in the 'Visualizations' section to gain insights into different aspects of the click fraud dataset.
10. To stop the Streamlit app, return to the terminal and press `Ctrl + C`.

By following these steps, you can seamlessly run and interact with the Streamlit web app locally, exploring click fraud predictions and visualizations.

6. Streamlit Web App: A User-Friendly Exploration

Explore the World of Click Fraud Predictions and Visualizations

1. **Interactive Interface:**
   Dive into an engaging experience with our "Heimdall 'NOBODY GOES THROUGH ME" web app. Easily input data, witness real-time predictions, and uncover valuable insights through intuitive charts and plots.

2. **Effortless Predictions:**
   In the 'Predictions' section, effortlessly input key features such as 'IP,' 'App,' 'Device,' 'OS,' 'Channel,' and 'Click Time.' Click 'Validate,' and watch as our pre-trained Random Forest model provides instant predictions—discerning whether an app will be downloaded or not.

3. **Insightful Visualizations:**
   Navigate to the 'Visualizations' section to unlock a treasure trove of visual insights. Choose from a variety of options, including top apps, attributed clicks per hour, top apps with attribution, and more. Our charts, crafted with Seaborn and Matplotlib, offer a visually appealing journey through the dataset.

4. **User-Friendly Navigation:**
   Explore seamlessly through the 'Home,' 'Predictions,' and 'Visualizations' sections, each designed for easy interaction. The app structure ensures clarity and enables users to delve into predictions or delve into the dataset's visual nuances effortlessly.

5. **Transparent User Interaction:**
   Encouraging user participation, our web app features input fields, buttons, and interactive visualizations. Tailor your exploration based on input, allowing for a personalized and engaging experience.

6. **Tailored Visualizations according to the User Dataset:**
   Encouraging user participation, our web takes the dataset from the user and the user can see the visualizations according to the given dataset which helps user to gain insights from the data.

7. **Local Deployment:**
   Running the app is a breeze—follow simple steps to initiate it locally. Gain access to predictions and visualizations via a local URL, bringing the power of click fraud analysis to your fingertips.

Overall, This Heimdall webpage offers an interactive and user-friendly platform for exploring click fraud predictions and related visualizations. Users can input data, view predictions, and gain insights through various charts and plots.

7. Recommendations

7.1 Scalability and Deployment:

- Scalability Planning: If the system is expected to handle a growing volume of data, ensure that the infrastructure is scalable. This involves optimizing code, utilizing cloud resources effectively, and planning for increased computational demands.
- Deployment Best Practices: Establish best practices for model deployment, considering factors such as version control, rollback mechanisms, and monitoring.

7.2 Real-time Monitoring and Adaptive Learning:

To stay ahead of evolving fraud strategies, consider implementing real-time monitoring and adaptive learning mechanisms:

- **Continuous Data Updates:** Regularly update the model with fresh data to adapt to changing patterns in click behavior. Implement a system for seamless integration of new data without disrupting the prediction pipeline.
- **Automated Retraining:** Set up automated retraining processes triggered by significant shifts in click patterns. This ensures that the model remains effective and up to date in detecting emerging fraud tactics.

7.3 User Interaction and Education:

- **User Feedback Mechanism:** Implement a feedback mechanism allowing users to provide input on predicted outcomes. This user feedback loop can serve as valuable data for model refinement and improvement.

7.4 Ethical Considerations:

In deploying predictive models for fraud detection, it is crucial to address ethical considerations:

- **Fairness and Bias Mitigation:** Continuously assess the model for potential biases and take measures to mitigate them. Fairness in predictions is essential to prevent unintended consequences or discrimination.
- **Transparency and Accountability:** Maintain transparency regarding the model's limitations and uncertainties. Establish clear accountability for the model's predictions and actively communicate these aspects to end-users.

These recommendations aim to guide the evolution of the predictive model, ensuring it remains adaptive, transparent, and aligned with the evolving landscape of click fraud detection.