EthiScan Documentation

A Comprehensive Browser Extension, Mobile App, and Admin Panel

CodeSage

Dark Patterns Buster Hackathon 2023

G.L. Bajaj Institute of Technology & Management

Greater Noida, Uttar Pradesh

Team Members

Aman Kumar Singh

Aaditya Raj

Pragati Singh

Rishabh Gupta

Nitin Pilkhwal

Team Mentor

Ms. Attiuttama Mishra

Table of Contents

Chapter 1

EthiScan

1.1 **Problem Statement** 5

1.2 **Solution Overview** 5

1.3 **Technologies Used** 6

Chapter 2

Detailed Description of EthiScan

2.1 **Browser Extension** 6

Features of Browser Extension 7

Working of Browser Extension 8

Demonstration of Browser Extension 9

2.2 **Mobile Application** 9

Features of Mobile Application 10

Working of Mobile Application 10

Demonstration of Mobile Application 11

2.3 **Admin Panel** 12

Chart Specifications 12

Demonstration of Admin Panel 13

2.4 **Validation** 13

2.5 **Backend** 15

Endpoints to access the detection and validation services: 15

Chapter 3

Machine Learning/Large Language Models

3.1 **Data Handling** 16

3.2 **XLnet Model** 17

Model Overview 17

Architecture 17

Working 18

3.3 **Dark Pattern Detection ML Model** 19

Architecture 19

Preprocessing Steps 20

Training Pipeline 20

Improvements 22

3.4 **Fake Review Detection ML Model** 22

Architecture 22

Data Handling 23

Pre-processing Steps 23

Training Pipeline 23

Improvement 24

3.5 **EthiBot - Chatbot Llama-7b** 25

Overview 25

Data Handling 25

Chapter 4

Cloud Deployment

4.1 **Cloud Deployment** **25**

Chapter 5

Installation and setup

5.1 **Backend & Admin Panel on Localhost** 26

5.2 **Browser Extension** 27

Chrome 27

Mozilla Firefox 27

Microsoft Edge 27

5.3 **Mobile Application** 27

Prerequisites 27

Installation Steps 28

6. Conclusion 28

List of Figures

Figure 1: Extension Working **9**

Figure 2: App Working 12

Figure 3: Admin Panel Working 13

Figure 4: Validation Working 15

Figure 5: Dataset Information 17

Figure 6: XLNET Architecture **18**

Figure 7: XLNET Report **19**

Figure 8: Radom Forest Architecture 20

Figure 9: Random Forest Report 22

Figure 10: LGBM Report 24

Figure 11: Cloud Architecture 26

Problem Statement

Design and prototype innovative app or software-based solutions that can detect dark patterns' use, type, and scale on e-commerce platforms.

Solution Overview

Our project, EthiScan, offers a comprehensive solution to address Dark Patterns on E-Commerce platforms. Leveraging cutting-edge technologies and innovative approaches, our solution aims to provide an approach that can Safeguard the users and create awareness about the Manipulative Tactics used by E-Commerce Platforms.

At its core, EthiScan consists of a Browser Extension, a Mobile Application, and an Admin Panel. It offers a comprehensive approach that Detects and Validate different Dark Patterns in E-Commerce with a great focus on User Interface and User Experience. Through the use of Large Language Models and advanced Machine Learning Models, our solution is capable of detecting 14 types of Dark Patterns which are Activity Notification, Confirm Shaming, Countdown Timers, Forced Enrollment, Hard to Cancel, Hidden Subscriptions, High-demand Messages, Limited-time Messages, Low-stock Message, Pressured Selling, Testimonials of Uncertain Origin, Trick Questions, Visual Interference Fake Review.

EthiScan recognizes that there is a fine line between Dark Patterns and Legitimate Marketing Strategies, so it moves forward by performing Validation on Major Marketing Strategies, which are Activity Notification, Limited-time Messages, Low-stock Messages, and Countdown Timers.

Furthermore, it covers the manipulative tactics that are frequently used on E-commerce Platforms such as Hidden Cost, Bait & Switch, Pre-Checked Box, Hidden Terms and Conditions, and Subscription Trap.

Another feature that EthiScan covers is Monitoring the trends of Dark Patterns on E-Commerce, as a result, provides an Admin Panel to monitor all trends and gain insights from them by representing the information in the form of graphs.

Hence, EthiScan with its Multi-Disciplinary approach targets every possible Dark Pattern with the highest concern for User Privacy and User Experience.

Technologies Used

- ➔ Browser Extension
  - ◆ HTML
  - ◆ CSS
  - ◆ Tailwind CSS
  - ◆ JavaScript
- ➔ Mobile Application
  - ◆ Flutter
  - ◆ Kotlin
  - ◆ Java
- ➔ Backend
  - ◆ Django
  - ◆ SQLite3
- ➔ Machine Learning Model
  - ◆ RandomForest model
  - ◆ LightGBM
- ➔ Large Language Models
  - ◆ Laama-7b
  - ◆ XLNet
- ➔ Admin Panel
  - ◆ Chart.js
  - ◆ Django-jazzmin
- ➔ Cloud Deployment
  - ◆ Google Cloud Platform
  - ◆ Compute

Detailed Description of EthiScan

1. Browser Extension

We have developed a browser extension called EthiScan, which scans e-commerce websites for dark patterns and highlights any text elements that contain them. The extension is capable of identifying the severity and type of pattern. Additional features are also available, which we will discuss in the following section. EthiScan has been designed to work on all popular browsers and has been built using HTML/CSS, Tailwind for popup design, and Javascript to handle DOM changes and connect to the backend. The extension uses Machine Learning/Large Language models to accurately check scanned texts.

Features of Browser Extension

- Our extension has been developed to detect 14 different types of dark patterns accurately including Activity Notification, Confirm Shaming, Countdown Timer, Forced Enrollment, Hard to Cancel, Hidden Subscriptions, High-demand Messages, Limited-time Messages, Low-stock Message, Pressured Selling, Testimonials of Uncertain Origin, Trick Questions, Visual Interference, and Fake Review. Each of these patterns is highlighted by the extension once detected. Among these, dark patterns like Limited-time messages, Activity Notification, Countdown Timers, and Low Stock messages get validated to determine if they are truly dark patterns or just a marketing strategy.

- The features of the extension are incredibly efficient and work in real time. This implies that the extension continuously monitors the webpage for any dark patterns and reports them promptly. For example, the extension automatically scans any pop-ups or ads that appear on the screen, not only during website opening. Additionally, the extension can operate simultaneously on multiple tabs, not just on a single one.

- The extension popup organizes dark patterns by type and displays website dark pattern content, making it easy to identify.

- Apart from highlighting dark patterns, the extension can detect hidden costs, nagging links, and disguised URLs present on the website.

- The extension effectively identifies potential hidden costs during the payment process by cross-referencing the payment page with the product page to address any discrepancies in pricing.

- Our extension streamlines the user experience by automatically unchecking pre-selected checkboxes on the product page. This ensures that users do not inadvertently add unnecessary items to their cart, enhancing transparency and minimizing extraneous costs.

- Introducing a valuable crowdsourcing feature, our extension empowers users to report discrepancies effortlessly. By enabling users to select and right-click on suspicious text, they can contribute to maintaining the integrity of the platform.

- Seamlessly navigating the terms and conditions page, the extension autonomously scans for language that may pose risks to the user. This proactive approach allows users to stay informed and protected from potential harm.

- The extension can provide users with guidance on how to cancel their membership and provide Admins with the depth of the cancel button.

- Our extension is designed to monitor outgoing requests from the website and notify the user if any sensitive data is being shared with third-party domains. This helps prevent cases where websites share user information with unauthorized parties or websites.

Working of Browser Extension

- The browser extension systematically scans all elements within a webpage, extracting their inner text for sequential transmission to the backend server for dark pattern analysis. Employing a custom API tailored for text detection, the browser awaits the server response to determine the presence of dark patterns, identifying their type and severity level for subsequent highlighting.

- Upon receiving the server response, the browser distinguishes between validated non-dark pattern text (highlighted in green with a red border) and identified dark patterns (highlighted with a yellow background and a red border). This process is extensive, covering various aspects such as user feedback, terms and conditions, and subscription checks through dedicated APIs.

- Utilizing Browser storage, the extension efficiently retains product links for ongoing monitoring of nagging product promotions on websites. This feature allows the extension to assess and report potential manipulative tactics related to specific product links.

- The extension seamlessly integrates with browser web tools to monitor the network tab, continuously checking for outgoing POST requests on the website. This proactive approach ensures real-time surveillance of potential data transmissions, contributing to a comprehensive assessment of the website's behavior.

- All these functionalities are executed autonomously without requiring user interaction, demonstrating a user-friendly experience with built-in error-handling mechanisms. The extension operates in the background, enhancing the user's online experience by proactively identifying and addressing potential dark patterns.

- While the extension works seamlessly without user intervention, it provides a convenient user interface through its popup. Users can interact with the extension popup to effortlessly remove all checkboxes on the website or review a consolidated list of detected dark patterns for transparency and control over their online interactions.

Demonstration of Browser Extension

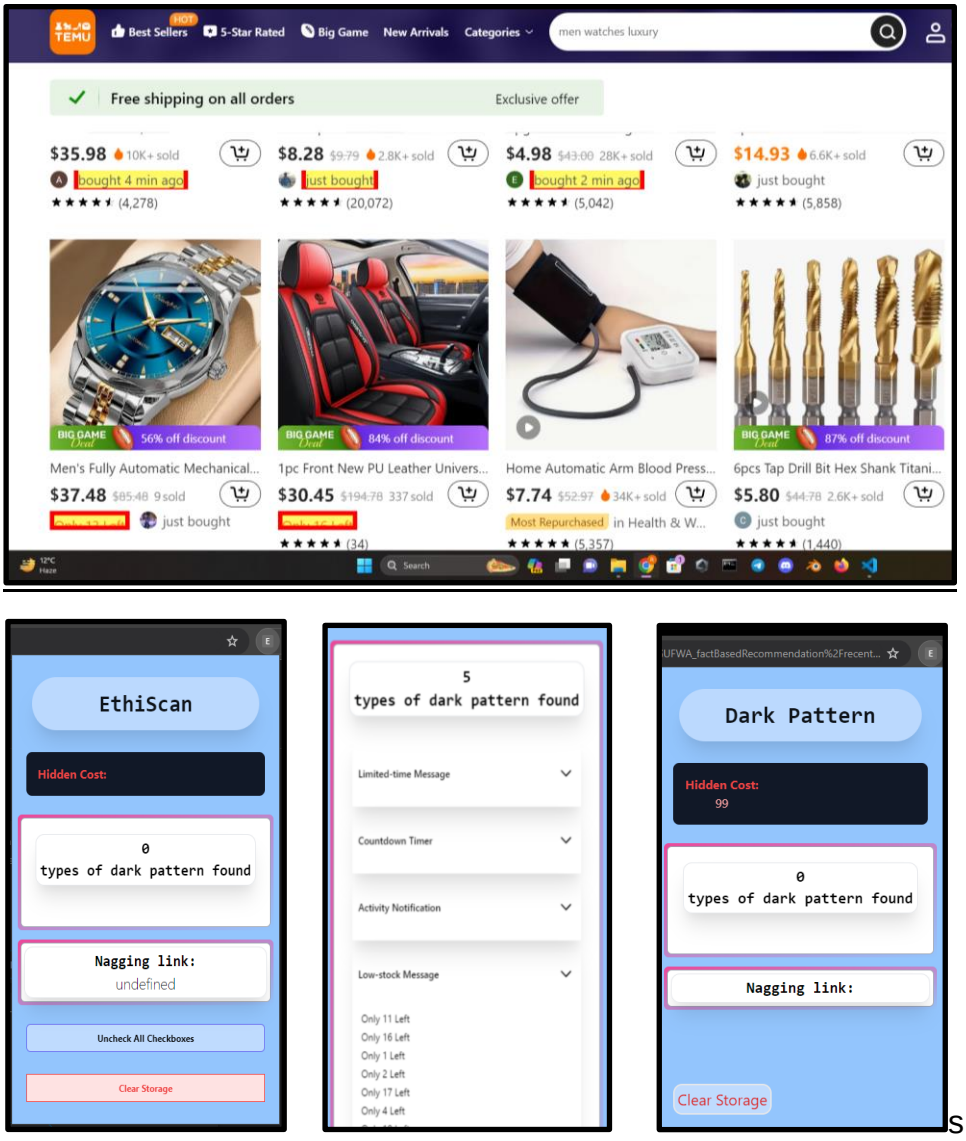


Figure 1: Extension Working

2. Mobile Application

The EthiScan Mobile Application is constructed with Flutter, an open-source framework renowned for building cross-platform applications. Flutter was selected because it provides a unified codebase for developing applications that are compatible with both the Android and iOS platforms.

Our mobile application seamlessly integrates with a variety of large language models and machine learning models using the Django server.

Our application improves user experience by using an overlay button to create a virtual screen, allowing users to extract data with a single click while browsing a website. Additionally, it employs voice assistance to verbalize results, thus minimizing the need for navigating between multiple apps and avoiding unnecessary storage consumption from screenshots.

Features of Mobile Application

- Our application is capable of successfully detecting 14 types of Dark Patterns, including Activity Notification, Confirm Shaming, Countdown Timer, Forced Enrollment, Hard to Cancel, Hidden Subscription, High-demand Message, Limited-time Message, Low-stock Message, Pressured Selling, Testimonials of Uncertain Origin, Trick Questions, and Fake Review. It highlights these patterns within the application, providing users with insights and awareness.

- Our application performs validation on major Dark Patterns, such as Activity Notification and Countdown Timer, to ensure that these are not merely marketing strategies.

- Our application features a Knowledge Hub and a Chatbot built using Laama-7b, aimed at providing awareness about the various manipulative tactics employed by E-Commerce platforms.

- Facilitate Crowdsourced Data and provide in-app rewards for every unique Dark Pattern found by the User.

- It includes Voice Assistance which is used to enhance user experience.

Working of Mobile Application

Here, we'll delve into the detailed workings of our innovative application

- To initiate the services, users can simply click on the shield icon, which activates the overlay button displayed on the screen. As users navigate through any e-commerce website or application, detecting potential dark patterns is as simple as clicking on this overlay button.

- Upon activation, the application prompts for permission to create a virtual screen of the mobile display. Upon agreement, the application seamlessly generates a virtual screen of the e-commerce site, sending it to the backend for analysis.

- In the backend, text extraction using optimal character recognition techniques takes place. This extracted text is then subjected to various machine learning models specifically designed to detect dark patterns.

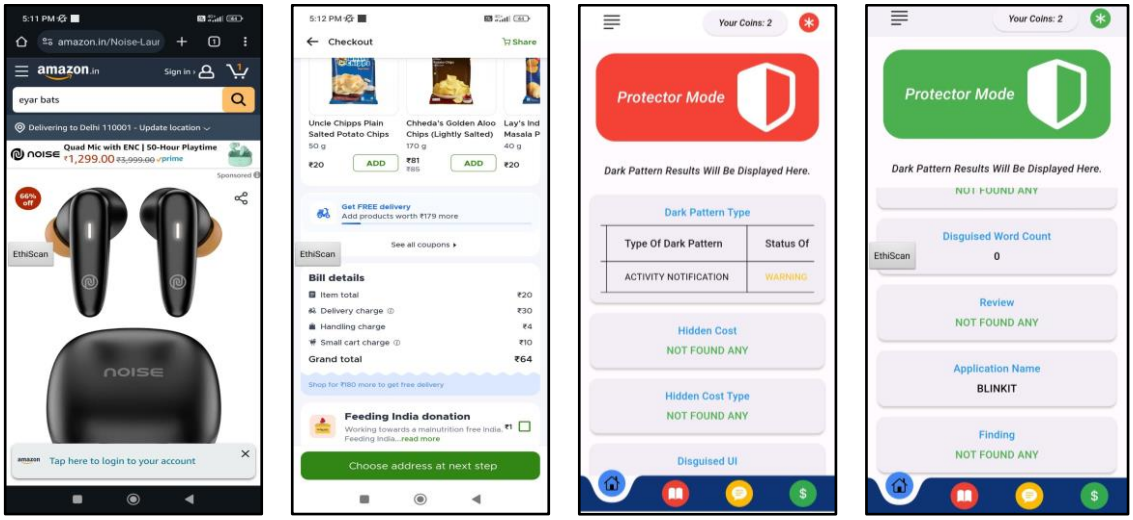
- For dark patterns categorized under validation, a validation process ensues to determine whether they are confirmed dark patterns or merely marketing strategies. Confirmed dark patterns are promptly displayed to the user and updated in the database, while those awaiting full validation are shown with a warning severity.

- Dark patterns not requiring validation are immediately marked as confirmed and conveyed to the user. Additionally, the models diligently detect the presence of fake reviews, disguised UI elements, or hidden costs, promptly alerting users when such patterns are identified.

- Users are provided with the option to receive alerts via voice assistance, allowing them to audibly hear all detected dark patterns or choose to read them themselves. Furthermore, all identified dark patterns are cross-referenced with previously detected patterns. If a new type of dark pattern is discovered, users are rewarded with in-app incentives as acknowledgment of their contribution.

This comprehensive approach ensures that users are empowered with real-time insights into manipulative tactics employed by e-commerce platforms, enhancing transparency and consumer awareness.

Demonstration of Mobile Application



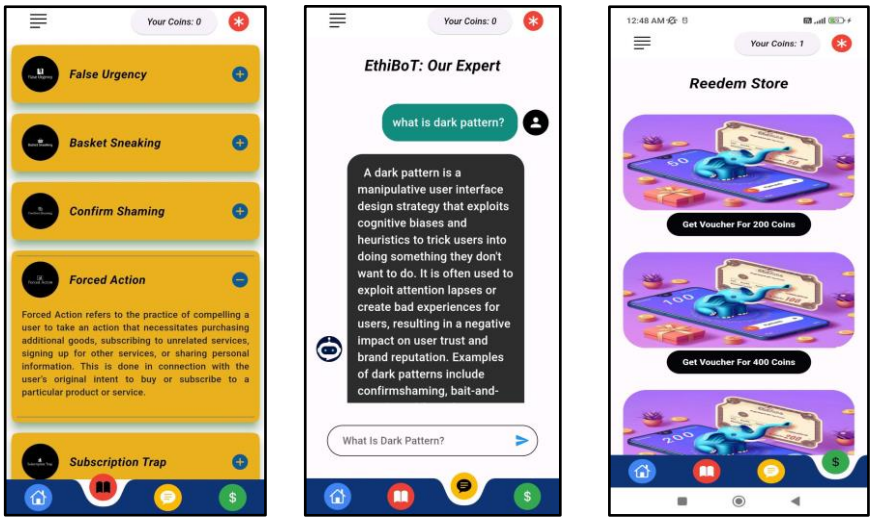


Figure 2: App Working

3. Admin Panel

Proper monitoring plays a crucial role in the fight against the use of these dark patterns. Therefore, to empower proper monitoring we have developed an admin panel. This Admin panel has full control over the central database, allowing the admin to perform CRUD operations easily on the database. The admin panel also provides various charts to empower proper visualization of diverse data recorded across the sites. All these charts are fully customizable and can easily be extended with more visuals. The central database provides a well-organized repository for records of all the deceptive practices that have been encountered so far.

Chart Specifications

The visualization of data present in the database is done using Chart.js, a JavaScript Library, and Ajax. Following are the different distributions we have incorporated inside the admin panel.

- Site vs Count: This distribution is a bar chart distribution. It shows the e-commerce sites with the count of different dark patterns it is using.
- Share distribution: This distribution is a pie chart distribution that shows the percentage share of different dark patterns in the market based on recordings.
- Monthly trend distribution: It is a line chart that describes the monthly trend of selected websites in the selected year. The trend here means the recording of the dark patterns on the website in different months.
- Overall monthly trend: This distribution describes the trend of the market related to recordings of the dark pattern monthly in the current year.

Demonstration of Admin Panel

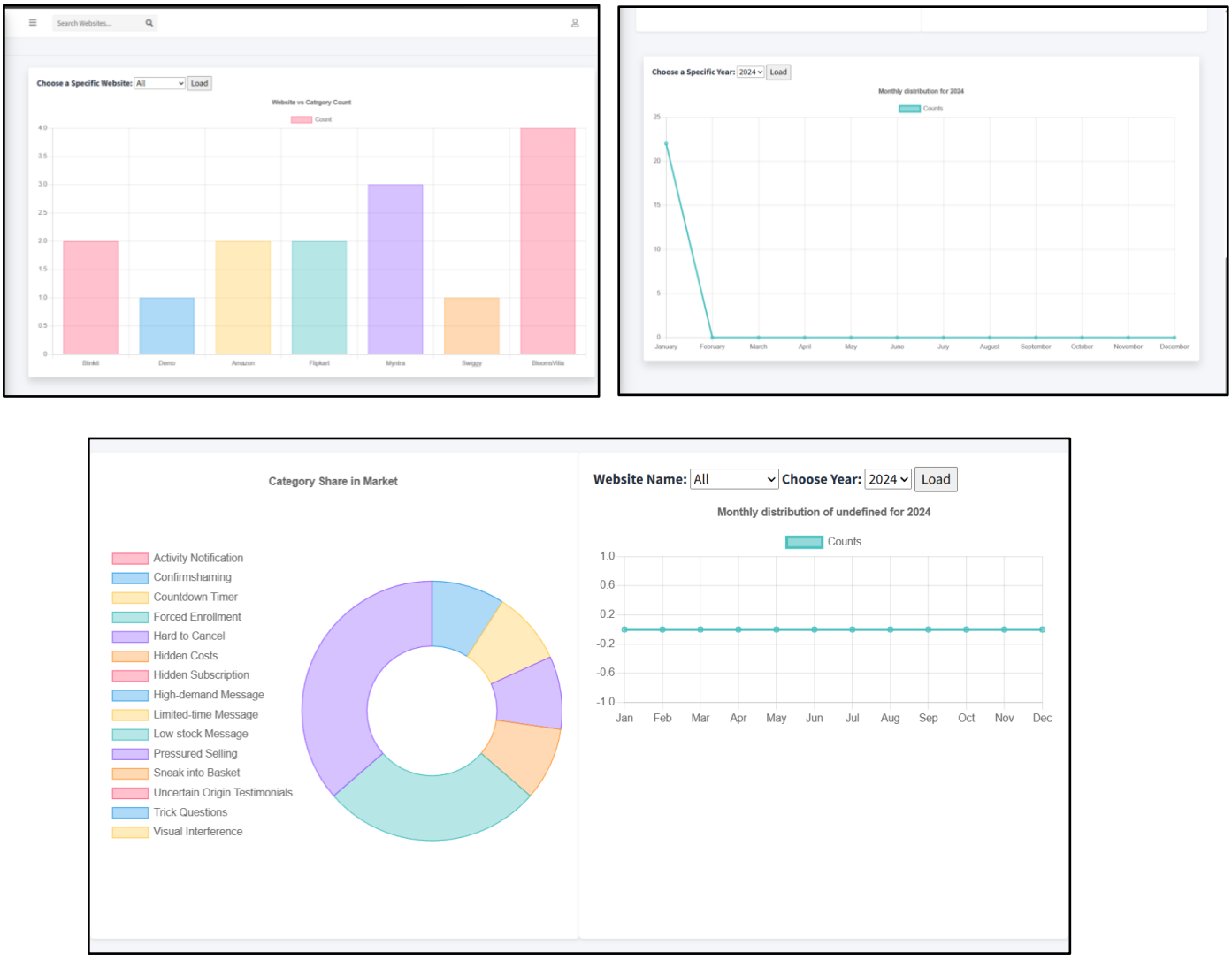


Figure 3: Admin Panel Working

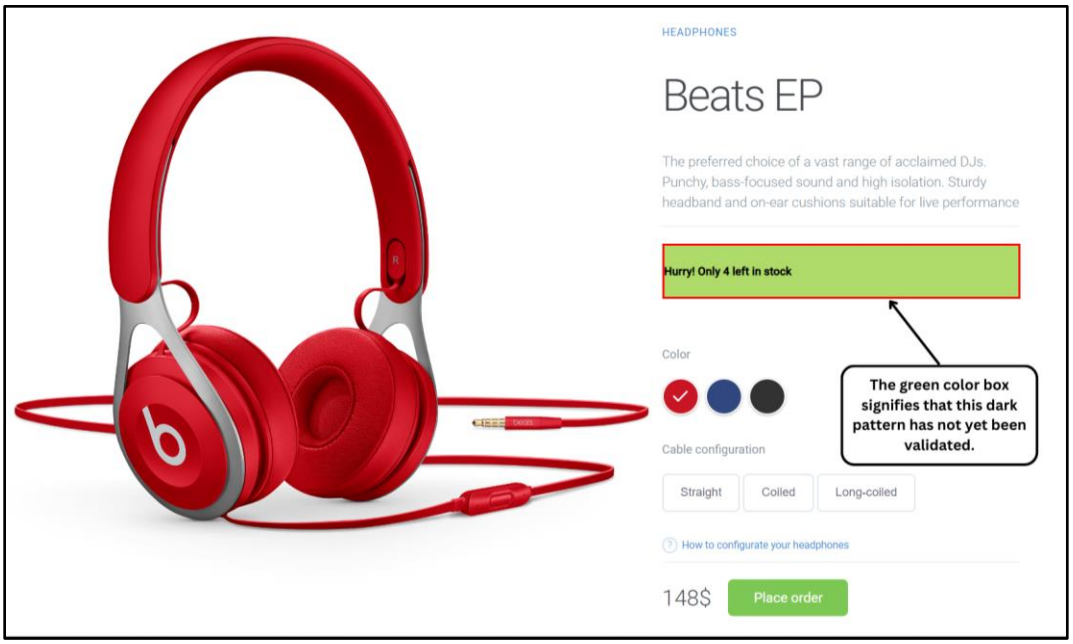
4. Validation

- Dark patterns such as Low-Stock messages, Activity notifications, Limited-time messages, and Countdown timers need to be validated first to determine whether they are meant to manipulate the intention of the user or just a marketing strategy of the site.
- For example, a message something like “Hurry, Only 4 left in Stock!” falls under the category of a Low Stock Message dark pattern but it can also represent the true quantity for that product.

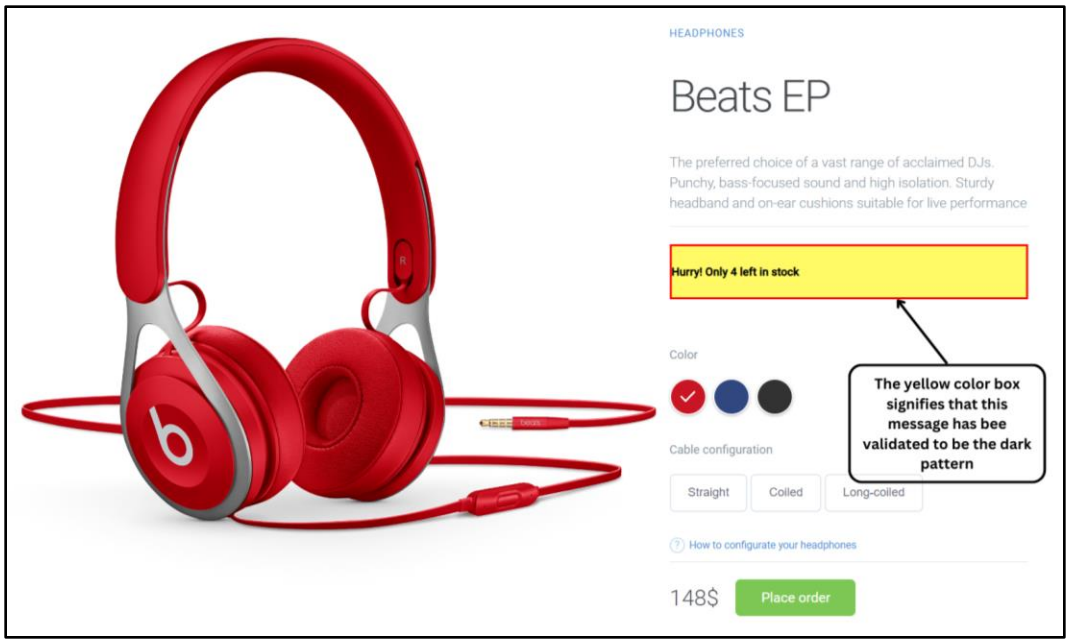
- To validate low-stock messages such as “Only 3 left” and Activity Notification messages such as “7 People bought in the last hour”, when we first encounter such messages on any specific page then we record this into our database along with a time stamp and the count of the item left or the number of people bought the item, respectively. If any user purchases this item, then the count of this item must decrease. But if any user again encounters the same message as it was earlier then it implies that the purchase of that item is not causing any effect on the limited stock message.

- This gives the validation that the message present on the page for that website has been put there intentionally by the developers to manipulate users.

- For validating dark patterns like High Demand, Limited Time, and Countdown Timers, we record their presence for a specific product along with the current timestamp in the database. If any other user also encounters the same stuff for the same product at a different time stamp then we perform a comparison of the timestamps concerning the detected pattern and through our algorithm, we validate the detected dark pattern.



The limited stock message is marked with a green box, signifying that the message has not yet been validated to be the dark pattern.



The limited stock message is marked with a yellow box, signifying that the message has been validated to be the dark pattern.

Figure 4: Validation Working

5. Backend

- The backend of our solution has been written using the Django framework.
- We do not rely on any third-party API. We have developed our endpoints using Django and our frontend parts communicate with our detection and validation models using these endpoints only.
- All the data which are sent by our frontend applications, get stored inside a SQLite database. This database is manageable and accessible using the admin panel discussed above.

Endpoints to access the detection and validation services:

- *SERVER_PORT*/txtanalyzer/: This endpoint has been used in the web extension. It takes the textual content as one of the parameters and feeds it to our detection and then validation models. The result from the model is then sent back to the caller in the form of a JSON response. The method used is the POST method while calling the endpoint.

  Request: JSON({"text": “”, webName": “”, baseUrl": “”, "prodName": “”, "isProdPage": “”, "count": “”})

- *SERVER_PORT/*imganalyzer*/:* This endpoint has been used in the mobile application. It takes a base64 encoded image as one of the parameters and feeds it to our detection and then validation models. The result from the model is sent back to the caller in the form of a JSON response. The method used is the POST method while calling the endpoint.

  Request: JSON({"base64_img": ""})

- *SERVER_PORT/*ethibot/: This endpoint has been used to get responses for our chatbot. This endpoint triggers an LLM model trained to answer the queries related to dark patterns. This endpoint accepts query text as a parameter and returns the response of the LLM model as the JSON response.

  Request: JSON({"user_message": ""})

6. ML & LLM Models

Data Handling

- Our dataset consists of over 5000+ datasets with 36% curated from open source data from 11,000 websites, predominantly from the e-commerce domain, and 64% scrapped from our own sets from tested with recent Indian E-commerce websites.

- We meticulously scraped labeled data from these websites to ensure diversity in training examples.

- Our dark pattern detection model is trained on a diverse dataset comprising instances of user interactions with online interfaces. Pre-processing involves tokenization, and special attention is given to maintain the sequential order of user actions. The training pipeline involves the permutation-based training objective of XLNet, leveraging the full contextual information present in the input sequences.

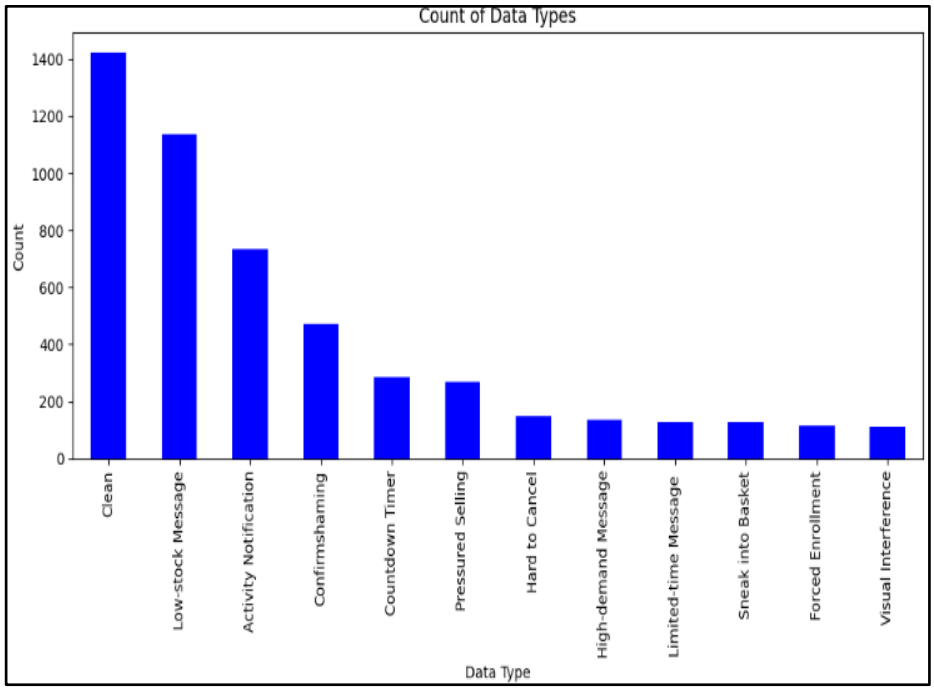


Figure 5: Dataset Information

6.1. XLnet Model

Model Overview

- XLNet is an advanced natural language processing model known for its accuracy, reaching 85%. Developed by Google Brain and Carnegie Mellon University, XLNet excels in understanding language semantics and detecting patterns, particularly in e-commerce platforms. Its transformer architecture and permutation-based training objective make it highly proficient in capturing bidirectional context, ensuring robust language comprehension. XLNet excels in capturing bidirectional context and understanding language semantics, surpassing its predecessor BERT on various benchmarks.

Architecture

- XLNet utilizes the transformer architecture, featuring self-attention layers and feedforward neural networks to model word relationships effectively. Unlike BERT's masked language model objective, XLNet employs a permutation-based approach, considering all token permutations to predict their order and capture bidirectional context information. During training, it predicts token order by assigning probabilities to permutations. This bidirectional context understanding allows XLNet to excel in tasks like sentiment analysis, text classification, and question answering during inference.

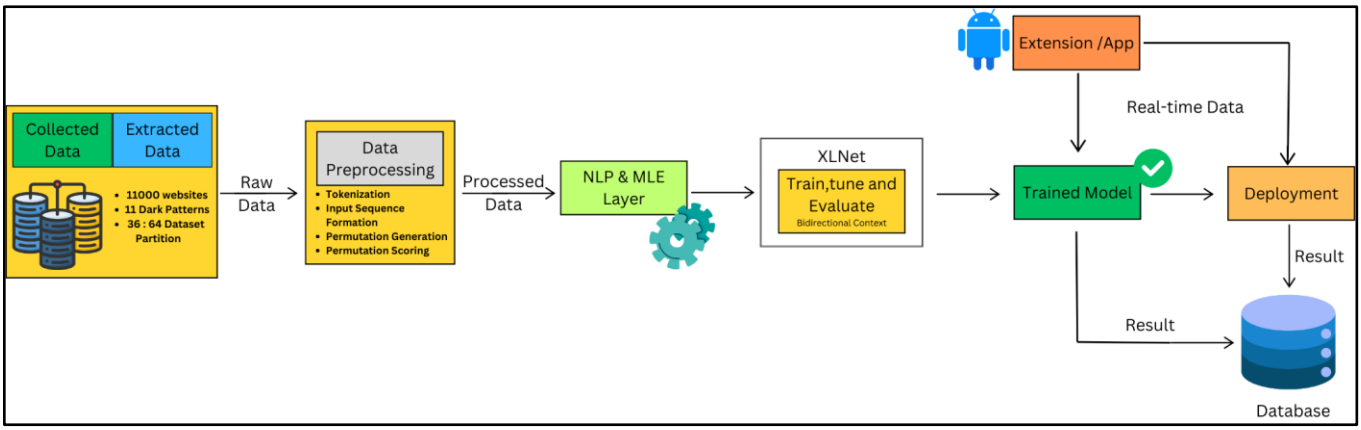


Figure 6: XLNET Architecture

Working

- Transformer Architecture: XLNet uses the transformer architecture, which consists of multiple layers of self-attention mechanisms and feedforward neural networks. This architecture allows the model to efficiently model the relationships and dependencies between words in a sequence.

- Permutation-Based Training Objective: Unlike some other models like BERT, XLNet employs a permutation-based training objective. Instead of using a masked language model (MLM) objective, XLNet considers all possible permutations of the input tokens and predicts the order in which they appear. This allows XLNet to capture bidirectional context information during training, as it does not mask any tokens and learns from the entire input sequence.

- Training Process: During training, XLNet takes a sequence of tokens as input and predicts the order of those tokens. It considers all possible permutations of the tokens and learns to assign a probability to each permutation based on how likely it is to represent the correct order of tokens. XLNet is trained using a variant of maximum likelihood estimation (MLE) objective.

- Bidirectional Context: By considering all possible permutations of the tokens, XLNet can effectively capture bidirectional context information. This means that the model understands the relationships between words in a sentence, even when they are separated by other words or phrases. This ability helps XLNet comprehend the semantic meaning and context of natural language text.

- Inference: During inference, XLNet can be used for various natural language processing tasks, such as sentiment analysis, text classification, named

entity recognition, question answering, and more. Given a sequence of text, XLNet can generate predictions or classifications based on its learned representations and understanding of language semantics.

Step	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
50	2.262000	1.790305	0.303862	0.189095	0.313833	0.246356
100	0.928100	0.759899	0.750000	0.377407	0.394706	0.388617
150	0.458100	0.513487	0.869919	0.443270	0.464163	0.450694
200	0.346400	0.759232	0.785569	0.433329	0.459274	0.435135
250	0.347500	0.533567	0.863821	0.401732	0.420076	0.412480
300	0.216900	0.574969	0.837398	0.386442	0.418912	0.374095
350	0.226300	0.476079	0.848577	0.362490	0.398986	0.348927

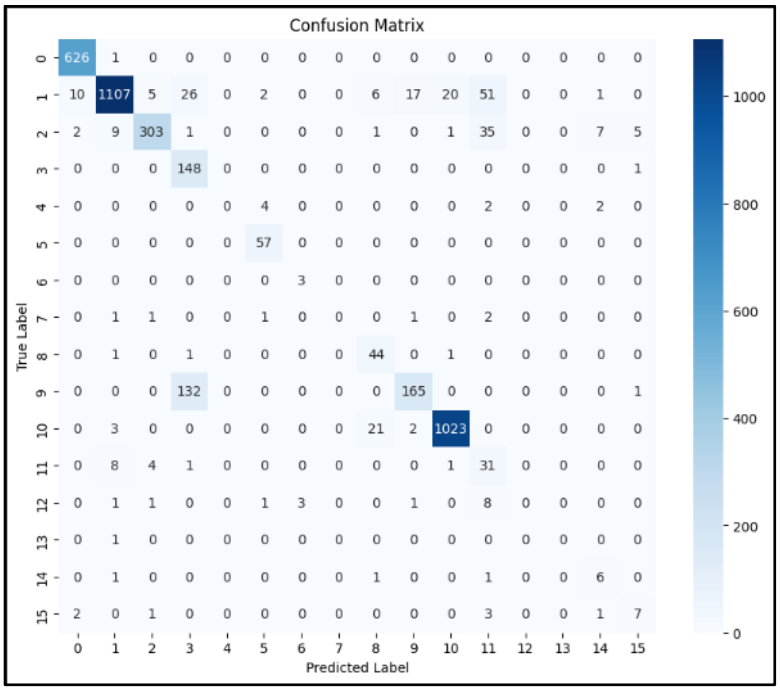


Figure 7: XLNET Report

6.2. Dark Pattern Detection ML Model

Architecture

- Our ML model integrates the Random Forest Classifier with Natural Language Processing (NLP) techniques to construct a powerful framework for detecting various dark patterns.
- Random Forest Classifier is a versatile ensemble learning method known for its robustness and effectiveness in handling various machine learning tasks. It operates by constructing multiple decision trees during training

and then aggregating their predictions to produce the final result. Random Forest is characterized by its ability to handle high-dimensional data, resistance to overfitting, and capability to provide insights into feature importance. It is widely used for classification and regression tasks due to its simplicity, scalability, and ability to handle large datasets efficiently.

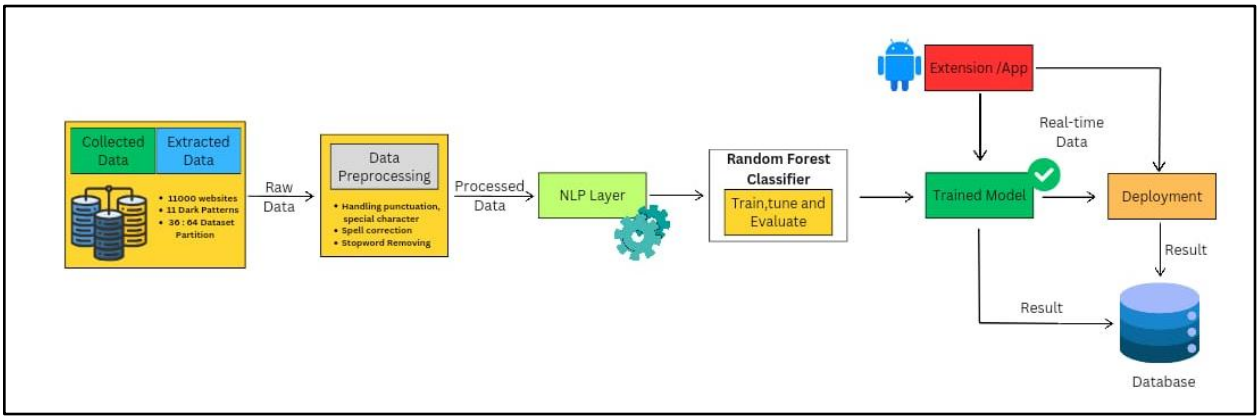


Figure 8: Radom Forest Architecture

Preprocessing Steps

Before training our model, we need to preprocess the raw textual data to standardize it and make it suitable for analysis. Preprocessing steps typically include:

- ❖ Text Cleaning: Handling special characters, punctuation, and any other noise that does not contribute to the semantics of the text.
- ❖ Tokenization: Breaking down the text into individual words or tokens.
- ❖ Lowercasing: Converting all text to lowercase to ensure consistency and prevent duplication of words with different cases.
- ❖ Stopword Removal: Eliminating common words that do not carry significant meaning.
- ❖ Stemming: Reducing words to their base or root form to normalize variations.

Training Pipeline

Once the data is preprocessed, it is divided into training and validation sets. The training pipeline involves the following steps:

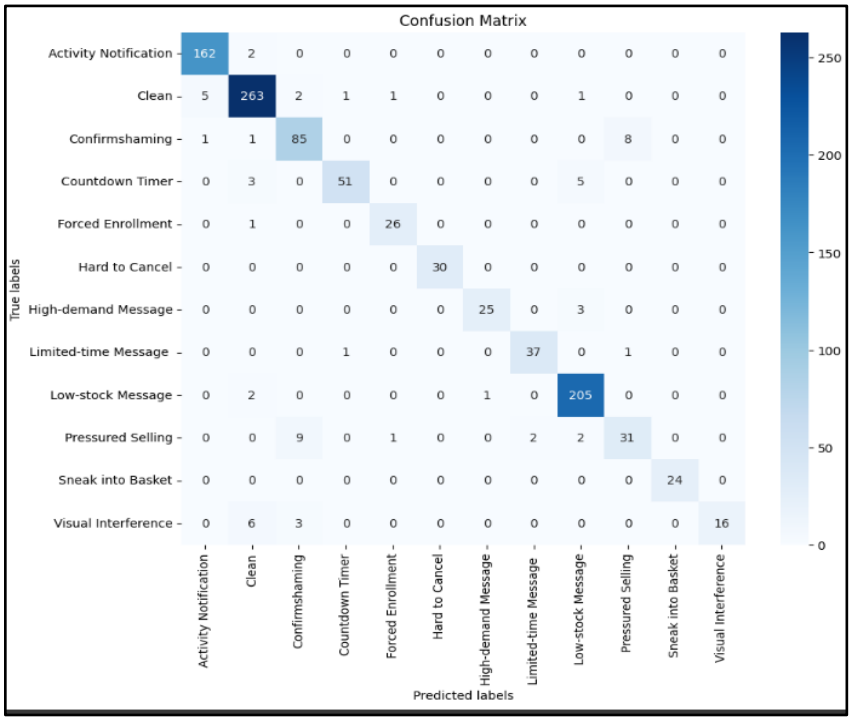
- ❖ Feature Extraction: Transforming the preprocessed textual data into numerical representations suitable for machine learning algorithms.

In our architecture, this is achieved through TF-IDF vectorization, which assigns weights to terms based on their frequency and importance.

- ❖ Model Training: The preprocessed and vectorized data is used to train the Random Forest Classifier. During training, the classifier learns the underlying patterns in the data and constructs an ensemble of decision trees.

- ❖ Hyperparameter Tuning: Fine-tuning the parameters of the Random Forest Classifier, such as the number of trees, maximum depth, and minimum samples per leaf, to optimize performance.

- ❖ Validation: Evaluating the trained model's performance on the validation set using appropriate metrics such as accuracy, mean-squared-error, root mean-squared-error, and R2-score.



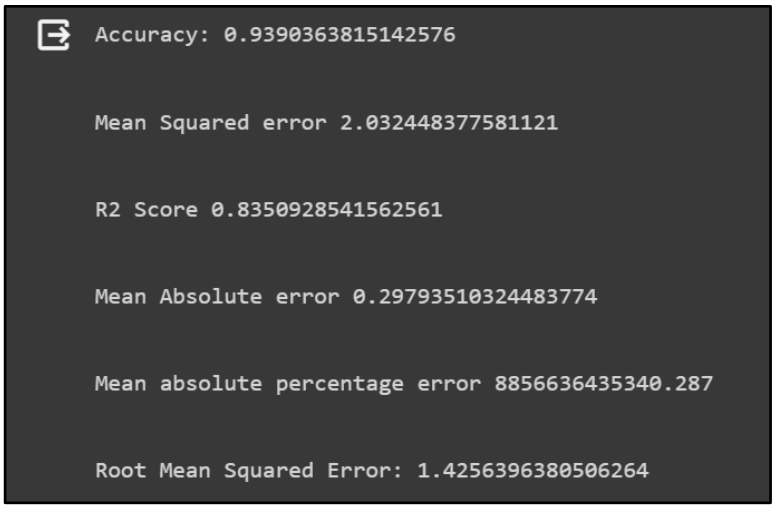


Figure 9: Random Forest Report

Improvements

- Detect more dark patterns: Till now model detects 11 dark patterns, and we can improve and add effective data related to other dark patterns. Fine-tune Parameters: Adjust the settings of the Random Forest model to find the best combination for your data.
- Fine-tune Parameters: Adjust the settings of the Random Forest model to find the best combination for your data.
- Prevent Overfitting: Add techniques like dropout or regularization to prevent the model from memorizing the training data too much.

6.3. Fake Review Detection ML Model

- Dataset we used, is the publicly available and well suited for our needs. It consists of prompt from different AI bots and some genuine reviews from the verified customers. In addition to this, we also manually added data and make it more diverse and ultimately it will help our model to gain insights.

Architecture

- Our ML model harnesses the capabilities of the LGBMClassifier, seamlessly integrating it with advanced Natural Language Processing (NLP) techniques. This fusion creates a robust framework designed specifically for the detection the AI-generated text. LightGBM (LGBM) is a highly efficient and scalable gradient-boosting framework known for its fast training speed, low memory usage, and support for parallel and distributed computing. It stands out for its ability to handle categorical features directly, streamlining

preprocessing. LGBM's innovative decision tree construction and leaf-wise growth strategy contribute to its excellent performance in classification, regression, and ranking tasks.

Data Handling

- The data utilized for this text classification task is sourced as online dataset. Each prompt is accompanied by a label indicating whether it was generated by AI or not.

Pre-processing Steps

- ❖ Dataset : It consists of 1000+ prompts with evenly distributed classes.

- ❖ Feature Engineering: Additional features are engineered from the text data to provide more information for model training. These features include: Text Length, Word Count, Average Word Length, Unique Word Count, Punctuation Count, Digit Count, Uppercase Count, and Average Word Length:

Training Pipeline

- ❖ TF-IDF Vectorization: The TfidfVectorizer from scikit-learn is utilized to convert the text data into numerical features.

- ❖ Target Preparation: The target variable for classification, indicating whether the prompt was generated by AI, or not.

- ❖ Model Training: The LGBMClassifier is chosen for its efficiency and performance in text classification tasks. The model is trained using the TF-IDF transformed features.

- ❖ Model Evaluation: The trained model is evaluated using validation techniques such as cross-validation or held-out test data to assess its generalization ability.

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       275
           1       0.00      0.00      0.00         1

    accuracy                           1.00       276
   macro avg       0.50      0.50      0.50       276
weighted avg       0.99      1.00      0.99       276
```

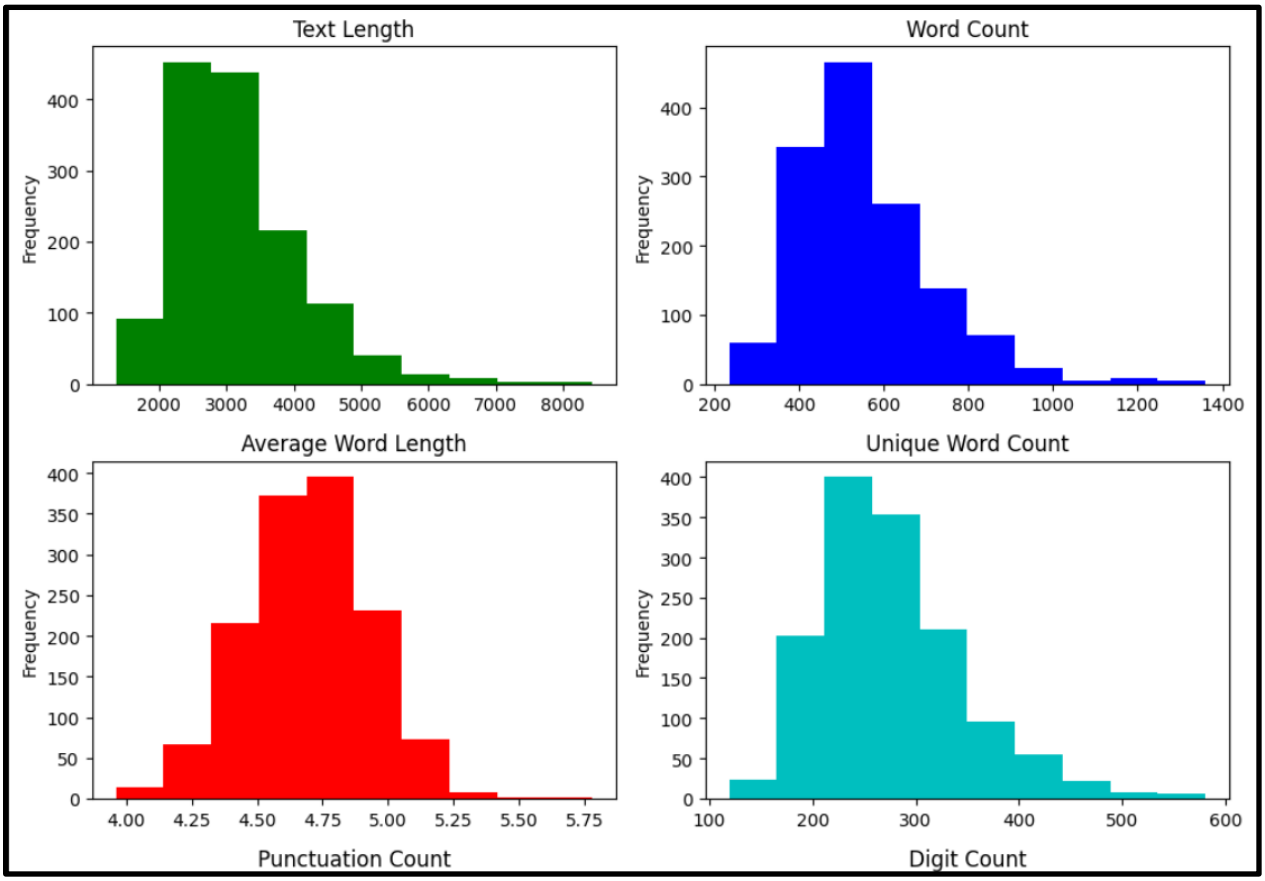


Figure 10: LGBM Report

Improvement

- ❖ Fine-tune Model Parameters: Adjust the LGBMClassifier's settings to optimize its performance on the dataset, finding the best configuration for improved accuracy.
- ❖ Prevent Overfitting: Incorporate techniques like dropout or regularization to prevent the LGBMClassifier from memorizing the

training data excessively, improving its ability to generalize to unseen examples.

6.4. EthiBot - Chatbot Llama-7b

Overview

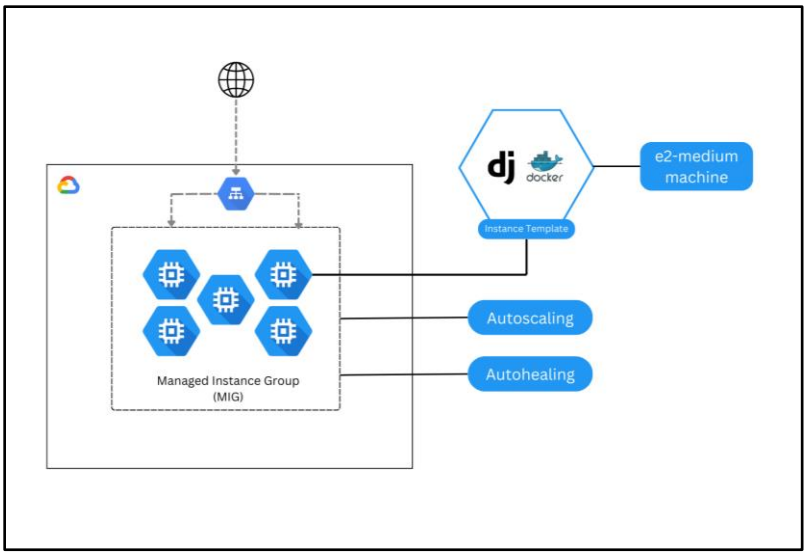
- Model Architecture Llama 2 is an auto-regressive language model that uses an optimized transformer architecture. The tuned versions use supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) to align to human preferences for helpfulness and safety.

Data Handling

- The dataset utilized in this model comprises a substantial volume of information meticulously extracted pertaining to dark patterns, meticulously organized within a text file. This dataset encompasses diverse categories, including but not limited to news articles, definitional content, recent updates, and other relevant sources. Each entry within the dataset is carefully curated to provide comprehensive coverage of the intricacies associated with dark patterns, ensuring a robust foundation for training and enhancing model proficiency.

7. Cloud Deployment

- We have deployed our backend on the Google Cloud Platform (GCP).
- We have created a Managed Instance Group of e2-medium type machines. The reason for going with MIG is that it provides automatic scaling which means that it can automatically add VM or remove VM as per the need of traffic and thus more suitable for solutions with dynamic traffic.
- The instance template is developed using a custom image from Bitnami Package for Django Stack.
- This MIG is also attached with a load balancer service to handle a dynamic amount of traffic from the mobile application and browser extensions.



Managed instance group in GCP for our solution

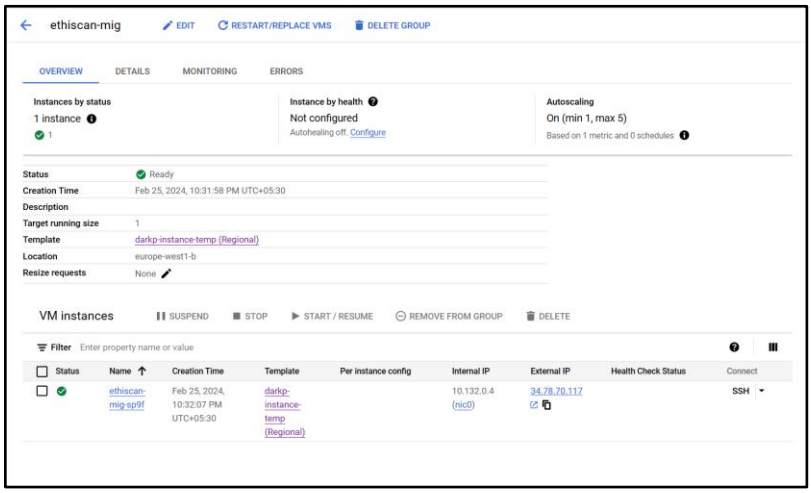


Figure 11: Cloud Architecture

8. Installation and Setup

Backend & Admin Panel on Localhost

- Install Python
- Move to the “CodeSage > Backend” folder and start your cmd/terminal.
- Type
  - cd core_api
  - pip install -r requirements.txt

- ➢ python manage.py runserver

- ❖ These commands will run the backend server on 8000 port number
- ❖ To access the admin panel, on any browser go to https://localhost:8000/admin

❖ Browser Extension

Chrome

- ➢ On chrome browser paste “chrome://extensions/” onto the address field
- ➢ Select the “Load unpacked” button and choose the “CodeSage > Extension” folder
- ➢ The extension will be added to Chrome extensions, and you can directly use it on any website

Mozilla Firefox

- ➢ On Firefox browser paste “about:debugging#/runtime/this-firefox” onto the address field
- ➢ Select the “Load Temporary Add-on” button and choose the “CodeSage > Extension > Extension.zip” file
- ➢ The extension will be added to Firefox extensions, and you can directly use it on any website

Microsoft Edge

- ➢ On Edge browser paste “edge://extensions/” onto the address field
- ➢ Select the “Load unpacked” button and choose the “CodeSage > Extension > Extension.zip” file
- ➢ The extension will be added to Edge extensions, and you can directly use it on any website

❖ Mobile Application

Prerequisites

- ➢ Flutter SDK
- ➢ Flutter-compatible IDE
- ➢ Android Studio or Xcode
- ➢ Git

Installation Steps

- ➢ Unzip the EthiScan application folder
- ➢ Open the folder in VS Code
- ➢ Type in terminal: flutter pub get
- ➢ Type in terminal: flutter run

9. Conclusion

In conclusion, EthiScan represents a comprehensive approach to combatting Dark Patterns within E-Commerce platforms. By leveraging advanced technologies and innovative methodologies across its Browser Extension, Mobile Application, and Admin Panel, EthiScan effectively identifies and validates a wide array of manipulative tactics while ensuring user privacy and enhancing the overall experience. Through trend monitoring and insightful analysis, EthiScan empowers stakeholders to stay ahead of emerging issues, fostering a more transparent and equitable online shopping environment for all users.

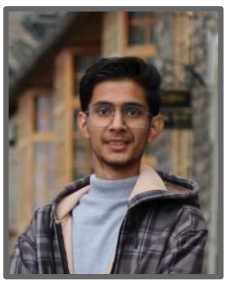
10. About Team

CodeSage

Aman Kumar Singh
Team Leader

Aaditya Raj
Team Member

Pragati Singh
Team Member



Rishabh Gupta
Team Member

Nitin Pilkhwal
Team Member