

# Configuration Manual

MSc Research Project  
Cyber Security

Sundar Ayyappan Muthukumarasamy  
Student ID: X23180749

School of Computing  
National College of Ireland

Supervisor: Michael Pantridge

**National College of Ireland**  
**MSc Project Submission Sheet**  
**School of Computing**



**Student Name:** Sundar Ayyappan Muthukumarasamy  
**Student ID:** X23180749  
**Programme:** MSc Cyber Security **Year:** 2024  
**Module:** MSc research Practicum-II  
**Lecturer:** Michael Pantridge  
**Submission Due Date:** 12/12/2024  
**Project Title:** Automated Detection of Dark Patterns in Website Design: Enhancing User Trust and Online Transparency  
**Word Count:** 768 **Page Count:** 5

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** Sundar Ayyappan Muthukumarasamy  
**Date:** 12/12/2024

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Sundar Ayyappan Muthukumarasamy  
Student ID: X23180749

## 1. Introduction

This configuration document will detail in steps how to set up and run Automated Detection of Dark Patterns in Website Design project. This paper outlines the system requirements, installation of software, preparation of data, training of models and finally deployment needed for real-time analysis. The tutorial is planned to make the replication of projects for whatever practical purpose easier.

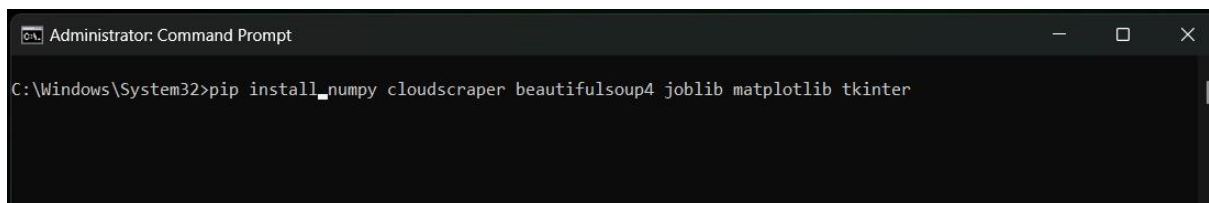
## 2. System Requirements

The system requires the following hardware and software configurations:

- Processor: Minimum Dual-Core CPU .
- RAM: At least 4GB.
- Storage: Minimum 10GB of free space.
- Operating system: windows.
- IDE: Thonny IDE .
- Python version: 3.7 or higher.

## 3. Installing Dependencies

As a first step, we should install the required libraries through Thonny or pip. Open the Thonny terminal or use the python environment to install the following dependencies:



```
Administrator: Command Prompt
C:\Windows\System32>pip install numpy cloudscraper beautifulsoup4 joblib matplotlib tkinter
```

## 4. Dataset collection

The system uses datasets collected through web scraping or manually curated CSV/JSON files. Collecting the dataset is an integral part of the project, as it provides the very raw data inputs necessary for training and testing the system to be used in the detection of dark patterns on websites. It scrapes textual content from different websites and manually labels them as "Bait and Switch," "Hidden Costs," or "Forced Continuity." This is because the methodology mixes web scraping with manual curation to make the dataset representative and diversified for real-world scenarios.

## 1. Web Scrapping:

- For web scrapping , please use the cloudscraper and BeautifulSoup libraries to scrape website content.
- And Extract textual data and save it as a structured CSV file for analysis.

```
import cloudscraper
from bs4 import BeautifulSoup
```

## 2. Manual Dataset:

- Prepare a dataset containing columns like url, html content and category.(*Dark-Pattern-Dataset*, n.d.)
- And make sure the data is clean and formatted correctly.

```
# Step 1: Load the dataset
dataset = pd.read_csv('dataset.csv')
```

# 5. Preprocessing and Feature Extraction

After text preprocessing, text data will be converted into numerical features by TF-IDF vectorization. In this approach, the importance of each term in a document is judged concerning the whole dataset. It helps the model to differentiate between the frequently used terms and the notable ones.

- Convert text into numerical features using the TfidfVectorizer from scikit-learn.

```
16
17 # Step 3: Text vectorization using TF-IDF
18 tfidf_vectorizer = TfidfVectorizer(max_features=5000)
19 X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
20 X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

# 6. Model Training

- First , Split the dataset into training (80%) and testing (20%) subsets.

```
14 # Step 2: Split the dataset into training and testing sets
15 X_train, X_test, y_train, y_test = train_test_split(dataset['text'], dataset['Pattern Category'], test_size=0.2, random_state=42)
16
```

- After splitting , Train the Naive bayes model using the TFIDF-transformed training data.

```
22 # Step 4: Train the Naive Bayes model|
23 naive_bayes_model = MultinomialNB()
24 naive_bayes_model.fit(X_train_tfidf, y_train)
25
```

- Then save the trained model and vectorizer using joblib.

```
32 # Step 6: Save the model and TF-IDF vectorizer
33 model_filename = 'dark_pattern_detection_model_naive_bayes.pkl'
34 vectorizer_filename = 'tfidf_vectorizer.pkl'
35 joblib.dump(naive_bayes_model, model_filename)
36 joblib.dump(tfidf_vectorizer, vectorizer_filename)
```

## 1. Model Output Files:

After successful execution we could see the model output files as shown below:

- dark\_pattern\_detection\_model.pkl
- tfidf\_vectorizer.pkl

 dark_pattern_detection_model_naive_bay...	06-12-2024 14:18	PKL File	263 KB
 tfidf_vectorizer.pkl	06-12-2024 14:18	PKL File	195 KB

## 7. Real-Time Detection with GUI

Develop a GUI using the tkinter library for user interaction.

- Add input URL field.
- Add button to trigger analysis.
- Add scrollable text box for displaying results.

```

Users > sundar > Downloads > dark 2 > templates > <> index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
7      <title>Dark Web Analysis</title>
8  </head>
9  <body>
10     <div class="container">
11         <h1>Dark Web Analysis</h1>
12         <form method="POST">
13             <input type="text" name="url" placeholder="Enter URL" required>
14             <button type="submit">Analyze</button>
15         </form>
16         <div class="results">
17             {% if results %}
18                 <h2>Analysis Results</h2>
19                 <ul>
20                     {% for line, category in results %}
21                         <li>
22                             <strong>Text:</strong> {{ line }}<br>
23                             <strong>Predicted Category:</strong> {{ category }}
24                         </li>
25                     {% endfor %}
26                 </ul>
27             {% endif %}
28         </div>
29     </div>
30 </body>
31 </html>
32

```

Run the GUI application via Thonny to analyze websites in real time.

## 8. Model Evaluation

This is quite an important step in understanding how well the model has performed and whether the goals set for the proper classification are attained. Strengths are measured by a wide range of metrics and visualizations, including the ability of the model to correctly identify and classify Dark Patterns.

Some of the performance metrics of the model are accuracy, precision, recall, and F1-score. Accuracy is the share of correctly predicted labels regarding the total amount of prediction.

### 1. Evaluation Metrics used:

- Accuracy, Precision, Recall, and F1-Score.
- Confusion Matrix for detailed insights into misclassifications.

```

46
47 # Step 8: Plot the ROC curve
48 # Binarize the output
49 y_test_binarized = label_binarize(y_test, classes=naive_bayes_model.classes_)
50 y_pred_proba = naive_bayes_model.predict_proba(X_test_tfidf)
51

```

2. Use matplotlib to generate visualization charts like ROC curves.

## 9. Deployment

### 1. Script Execution:

- Load all required files (model, vectorizer, dataset) in the same project directory.
- Execute the main script (e.g., main.py) in Thonny IDE.

### 2. Output:

- Input a URL and analyze results in the GUI.
- View classifications and details on detected dark patterns.

## 10. Future Enhancements

Several enhancements are advised based on improving performance, usability, and realistic usefulness of the system. First, the integration of state-of-the-art models, specifically transformers, with BERT, would increase the system's performance by a high margin. The integration of the system with browser extensions makes possible to easily operate interoperability for live navigation in the internet. In this manner, any given user can judge in real time a website, having the system automatically observe dark patterns while navigating on the net, which improves both accessibility and user experience. Finally, by applying some visualization to the GUI, the results would be more understandable to the users.

## 11. Conclusion

This configuration document enables the effective installation and deployment of the "Automated Detection of Dark Patterns" project. With the sound structure, real-time analytics, and scalability thus imparted, this system serves in great service of responsible web design for better user safety. The described processes will further enable easy replication and scaling of this innovative undertaking.

## 12. References

*dark-pattern-dataset*. (n.d.). <https://www.kaggle.com/datasets/adarshm09/dark-pattern-dataset>.