CYBER JUSTICE
PROJECT UNDER
IndiaAI CyberGuard AI Hackathon

By Tanvi Suryawanshi

INTRODUCTION:

Cyber Justice is an individual project undertaken as part of the **IndiaAI CyberGuard AI Hackathon 2024**. The primary objective of this project is to design and develop an NLP-based model that categorizes cybercrime complaints into various predefined categories. These classifications are based on parameters such as the type of victim, nature of fraud, and other relevant features of the textual complaint data.

The hackathon is structured into multiple stages, and this submission focuses on **Stage 1**, which requires participants to demonstrate their ability to preprocess textual data, build an NLP model, and evaluate its performance on a given dataset.

The project workflow includes the following key milestones:

1. **Text Preprocessing**:
   Tasks such as tokenization, stop-word removal, stemming, and data cleaning were performed to prepare the textual data for further analysis.
2. **Model Development**:
   The project employs a combination of **TF-IDF Vectorizer** and **Logistic Regression**, which together form an effective pipeline for text classification tasks.
3. **Accuracy Measurement**:
   Performance metrics such as accuracy, precision, recall, and F1-score were used to evaluate the model’s success in categorizing complaints correctly.

By leveraging Python-based tools such as **NLTK**, **Scikit-learn**, and visualization libraries, the project has achieved notable results, including an accuracy of **98.79%**. However, the analysis also highlighted certain challenges, such as **class imbalance**, which impacted the model's precision for underrepresented categories.

This submission provides a solid foundation for further refinement and explores potential improvements, such as addressing class imbalances and adopting more advanced NLP techniques. **Cyber Justice** aims to contribute to the fight against cybercrime by enabling efficient complaint categorization and data-driven decision-making.

THE SIGNIFICANT FINDINGS FROM NLP ANALYSIS:

The **Cyber Justice** project aimed to build a robust NLP model for classifying cybercrime complaints based on victims, fraud types, and other relevant factors. The analysis conducted during the project provided several insights regarding model performance, data distribution, and areas for improvement.

1. Data Patterns and Themes

The complaint dataset highlighted clear thematic trends:

- **Recurring Topics**: Terms like *transaction*, *OTP*, *account*, and *identity* were prevalent, emphasizing the frequency of financial and identity-related frauds.
- **Victim Profiles**: Complaints often categorized victims as individuals or businesses, with financial losses and personal identity theft being dominant themes.
- **Fraud Types**: Subcategories such as *credit/debit card fraud* were repeatedly encountered, aiding classification.

2. Model Performance Metrics

The model achieved a high overall accuracy of **98.79%**, and evaluation metrics included:

- **Precision (Weighted Average)**: 97.79%
- **Recall (Weighted Average)**: 98.79%
- **F1-Score (Weighted Average)**: 98.28%

Despite these impressive results, precision for certain categories was **0**, indicating that the model failed to correctly classify instances for those categories. This issue stemmed primarily from **class imbalance** in the dataset.

3. Class Imbalance and Its Impact

Analysis of the dataset revealed a significant imbalance in category representation. For example, some subcategories had numerous entries, while others had very few.

- **Impact on Model Performance**: Categories with fewer entries were often misclassified or not classified at all, leading to a precision score of **0** for those classes.
- **Root Cause**: The lack of sufficient examples for underrepresented classes hindered the model's ability to learn meaningful patterns for accurate prediction.

4. Key Drivers Behind Model Predictions

The success of the model in well-represented categories can be attributed to:

- **Effective Preprocessing**: Tokenization, stemming, and removal of stop words helped retain only the most meaningful data.
- **TF-IDF Vectorizer**: This method effectively highlighted the most relevant terms, improving the model’s focus on important features.
- **Logistic Regression**: The algorithm handled numerical feature representation derived from text data with high efficiency.

5. Misclassification Analysis

Misclassifications were primarily seen in:

- **Underrepresented Classes**: Categories with fewer examples were often ignored or incorrectly predicted.
- **Ambiguities**: Complaints with overlapping or vague descriptions posed challenges for accurate classification.
- **Domain-Specific Language**: Limited contextual understanding of mixed-language entries reduced effectiveness in certain cases.

6. Recommendations for Improvement

To address the challenges, the following strategies are recommended:

- **Data Augmentation**: Introducing more entries for underrepresented categories will help balance the dataset and improve training. Techniques like **SMOTE (Synthetic Minority Oversampling Technique)** can be applied to generate synthetic samples for minority classes.
- **Advanced NLP Models**: Leveraging transformer-based architectures such as **BERT** or **DistilBERT** could enhance the model's contextual understanding, addressing ambiguities and domain-specific challenges.
- **Error Analysis**: Regularly analyzing misclassified entries and incorporating feedback loops can improve accuracy for difficult categories.

IMPLEMENTATION PLAN:

Our focus is to refine the NLP model for optimal text classification in Stage 1. Here's the plan:

1. **Enhancements to the System**
   - **Feature Engineering**: Refine text preprocessing by improving tokenization, handling imbalanced classes through oversampling/undersampling, and experimenting with TF-IDF and embeddings (e.g., Word2Vec).
   - **Model Improvements**: Explore additional classification algorithms (e.g., SVM, Random Forest) and advanced deep learning models (e.g., BERT). Hyperparameter tuning will optimize performance.

2. **Further Analysis**
   - Perform error analysis using confusion matrices to identify misclassification patterns and retrain with targeted fixes.
   - Visualize class distributions to understand imbalances and adjust data preprocessing.

- ○ Use TF-IDF plots to highlight key drivers of classification, ensuring model interpretability.

3. **Testing and Validation**

    - ○ Implement k-fold cross-validation for robust evaluation.
    - ○ Assess performance using metrics like accuracy, precision, recall, and F1-score, prioritizing those that align with the use case.

4. **Deployment Plan**

    - ○ Package the best-performing model as a deployable script on GitHub/Open Forge.
    - ○ Ensure reproducibility by documenting the preprocessing pipeline and providing usage instructions.

This iterative approach ensures precision, scalability, and adherence to project goals while preparing for deployment.

CREDITS:

- **Datasets**: The datasets I have used are taken from IndiaAI app.
    - ■ train.csv
    - ■ text.csv
- **Libraries and Tools Used**:
    - ■ Pandas: Data manipulation and analysis (Pandas Documentation)
    - ■ NumPy: Numerical computations (NumPy Documentation)
    - ■ Matplotlib: Data visualization (Matplotlib Documentation)
    - ■ Seaborn: Statistical data visualization (Seaborn Documentation)

    - Scikit-learn: Preprocessing and modeling (Scikit-learn Documentation)
    - NLTK: Natural Language Toolkit for text processing and analysis (NLTK Documentation)
- Additional Acknowledgments:
    - PorterStemmer from NLTK for stemming operations
    - Tokenization and stopword filtering functionalities provided by NLTK
    - Python's standard libraries such as re and string for text manipulation
- Special thanks to the contributors of these open-source libraries for making this project possible!
- **Inspiration**:
    - IndiaAI CyberGuard AI Hackathon 2024
- **External Tools**: Google Colab, GitHub, Google Search, Google Scholar
- **AI Tools and Usage:** This project leveraged OpenAI's ChatGPT, Google Gemini for:
    - Generating ideas for efficient text preprocessing techniques.
    - Clarifying concepts related to natural language processing and machine learning.
    - Debugging Python code
    - Providing optimization suggestions which were reviewed, validated, and implemented by the developer.

All outputs from ChatGPT, Gemini were critically analyzed, modified, and implemented by the developer to align with project requirements. ChatGPT, Gemini served as a productivity booster rather than a direct contributor to the code.

- **Research References**:
    - Data preprocessing by Hui Yang Department of Computer Science San Francisco State University
    - Using TF-IDF to Determine Word Relevance in Document Queries by Juan Ramos Department of Computer Science, Rutgers University
- **YouTube Reference**:
    - Text Representation Using TF-IDF: NLP Tutorial For Beginners - S2 E6 by codebasics

    - How to convert categorical data to numerical data in python | Python Basics Tutorial by Abhishek Agarrwal
    - Text Preprocessing | tokenization | cleaning | stemming | stopwords | lemmatization by Utsav Aggarwal
    - Logistic Regression | Logistic Regression in Python | Machine Learning Algorithms | Simplilearn
- **Summary of Contributions**
    - By leveraging a combination of research, open-source tools, and AI-assisted development, this project demonstrates a thoughtful and efficient approach to solving text analytics challenges.
    - Each component was carefully implemented with an emphasis on originality and transparency, ensuring adherence to ethical standards.
- **Plagiarism Declaration**
    - I hereby declare that the work submitted is original and created solely by our team for the IndiaAI CyberGuard AI Hackathon. All external resources, libraries, and concepts have been appropriately cited, and no unauthorized materials have been used.