India AI Cyberguard

Project Title: Crime Report Classification Using NLP

Institute Name : Monark University

Team Members:

1. <u>**Amit Gurjar:**</u>
    - **Role:** Team Leader
    - **Profession:** BE-IT Student
    - **Expertise:** AI/ML, Computer Vision
    - **Contact:** amitgurjar8155@gmail.com
2. <u>**Ronak Gohil:**</u>
    - **Role:** Team member
    - **Profession:** BE-IT Student
    - **Expertise:** AL/ML, Python
    - **Contact:** ronakgohil240@gmail.com
3. <u>**Harshil Sangani:**</u>
    - **Role:** Team member
    - **Profession:** BE-IT Student
    - **Expertise:** AL/ML, Data Visualization
    - **Contact:** harshilsangani07@gmail.com
4. <u>**Yash Sapra:**</u>
    - **Role:** Team member
    - **Profession:** BE-IT Student
    - **Expertise:** AL/ML, Python
    - **Contact:** yprajapati276@gmail.com

Date: 2024-11-05

SR NO.	***TABLE OF CONTENT***
1.	**Executive Summary**
2.	**Introduction**
3.	**Methodology**
4.	**Results**
5.	**Challenges Faced**
6.	**Conclusion**
7.	**References**
8.	**Appendices**

1.Executive Summary

Project Overview
Our project tackles the challenge of subcategory prediction in a crime-related dataset for the India AI Cyberguard Hackathon. This dataset contains raw, unstructured text information and is classified into three primary fields: **Category**, **Subcategory**, and **Crime Additional Info**. The primary task is to build a robust NLP model that can accurately predict the subcategory based on the crime additional info, which often includes messy text descriptions.

Objective
The goal is to effectively classify these text entries into predefined subcategories to enable more streamlined and actionable insights into crime-related data. Accurate subcategory classification is vital for aligning outputs with Government of India regulations and provides a foundation for improved data analysis and reporting.

Approach
Our approach to this challenge includes experimenting with various machine learning models, with a primary focus on **XGBoost** due to its effectiveness in handling complex data. Additional models explored include traditional machine learning methods such as **Naive Bayes** and **SVM**, along with advanced techniques like **BERT** for capturing contextual information in crime-related descriptions.

To optimize model performance, we implemented a detailed pipeline that includes:

- **Text Preprocessing**: Handling noisy text data with typos, abbreviations, and inconsistent phrasing.
- **Feature Engineering**: Transforming text data into meaningful vectors, primarily using TF-IDF and embeddings.
- **Model Tuning**: Fine-tuning model parameters to maximize classification accuracy and balance across categories.

Results
After extensive model testing, XGBoost demonstrated a strong balance of accuracy and efficiency, with further refinements yielding improved precision and recall for most subcategories. Our experiments with BERT and other complex models also highlighted areas where deeper semantic understanding significantly boosted subcategory prediction, particularly in ambiguous cases.

2. Introduction

Theme and Context
This project is part of the India AI Cyberguard Hackathon, centered on developing an NLP model to assist citizens in filing cybercrime reports more accurately on the **National Cyber Crime Reporting Portal (NCRP)**. By enabling real-time analysis of text descriptions and incident-related media files uploaded by citizens, the project aims to guide users in correctly categorizing their reports, making the reporting process more efficient and less error-prone.

Problem Statement
One of the primary challenges in reporting cybercrimes is ensuring that users accurately describe incidents, including selecting the correct subcategories for streamlined investigation and action. Misclassification or incomplete information in reports can lead to inefficiencies and delays in response. This project addresses this challenge by focusing on **automatic classification of crime descriptions** into appropriate subcategories based on raw, unstructured text data.

Dataset Overview
The dataset provided contains approximately **1.56 lakh rows** of raw, unstructured text entries. Each row includes:

- **Category**: The main classification of the crime.
- **Subcategory**: More specific classifications within each category.
- **Crime Additional Info**: Descriptive text detailing the incident.

The dataset is unprocessed, containing significant noise in the form of typos, inconsistencies, and ambiguities. This unstructured nature adds complexity, as the text requires extensive preprocessing to enable accurate classification.

Objectives
Our objective is to build a reliable NLP model that:

- **Classifies each crime description into appropriate subcategories** to ensure the report aligns with Government of India guidelines.
- Provides real-time classification to assist users in choosing correct subcategories, thereby reducing the potential for misreporting.

3. Methodology

Data Preprocessing
Given the unstructured nature of the dataset, extensive preprocessing was crucial to prepare the text data for classification. Key steps included:

1. **Normalization**: Standardizing text to improve consistency and model performance. This included converting text to lowercase and removing extraneous symbols and punctuation that did not contribute to semantic meaning.

2. **Tokenization**: Splitting text into individual words or tokens, enabling analysis of linguistic structure and making it easier to apply transformations like TF-IDF.

3. **Regular Expressions (RegEx)**: RegEx patterns were used to identify and clean common text issues such as excessive whitespace, repeated characters, and informal language often seen in user-generated content.

4. **EDA (Exploratory Data Augmentation)**: To account for the imbalances in the dataset, EDA techniques were applied to expand certain categories. This included generating synthetic examples for underrepresented subcategories, which improved model performance on less frequent classes.

Feature Engineering
Transforming text data into numerical vectors is essential for feeding it into machine learning models. The **TF-IDF (Term Frequency-Inverse Document Frequency)** approach was chosen as the primary feature engineering method because it effectively captures the importance of words within the context of the dataset:

- **TF-IDF Vectors**: Each crime description was converted into a TF-IDF vector. This method assigns weights to words based on their frequency in a given document relative to their frequency across the dataset, highlighting distinctive terms for each description.

- **N-grams**: Both unigrams and bigrams were used to capture individual words and word pairs, adding context that single words alone might miss. This was especially useful for identifying patterns in the crime descriptions that were indicative of specific subcategories.

Model Selection
For classification, multiple models were tested, with **XGBoost** selected as the primary model due to its speed, interpretability, and ability to handle complex patterns in the data.

1. **XGBoost (Primary Model)**: XGBoost is a gradient-boosting algorithm that combines decision trees to improve classification accuracy iteratively. It was chosen

for its robustness in handling unstructured text data and for achieving high precision and recall in initial testing.

  - **Parameter Tuning**: Key hyperparameters like learning rate, max depth, and the number of estimators were fine-tuned to improve accuracy, especially for rare subcategories. This iterative tuning minimized overfitting and ensured better generalization on unseen data.

2. **Additional Models for Benchmarking**: Several other models were tested for comparison, including:

   - **Naive Bayes**: Chosen for its simplicity and speed in handling text classification tasks, it served as a baseline model.

   - **BERT (Bidirectional Encoder Representations from Transformers)**: Applied to assess the benefits of using a contextual language model. While BERT yielded promising results in capturing contextual nuances, it was computationally intensive and challenging to integrate within the hackathon timeframe.

Training and Validation
The dataset was split into training and validation sets to evaluate model performance. Cross-validation was employed to assess stability, particularly for imbalanced subcategories. The model was evaluated based on accuracy, precision, recall, and F1-score to ensure balanced performance across all classes.

4. Results

Model Performance Metrics (With XGBoost and LogisticRegression):

- **Category Accuracy : 0.98**
- **Category F1-score: 0.98**
- **Subcategory Accuracy: 0.93**
- **Subcategory Macro Precision: 0.93**
- **Subcategory Macro Recall: 0.93**
- **Subcategory Macro F1-score: 0.93**

Precision and Recall for Each Subcategory:

SUBCATEGORY	PRECISION	RECALL	F1-SCORE	SUPPORT
ANY OTHER CYBER CRIME	0.9195	0.7408	0.8205	43771
ATTACKS ON APPLICATIONS (E.G., E-GOVERNANCE, E-COMMERCE)	0.9730	0.9774	0.9752	22055
BUSINESS EMAIL COMPROMISE/EMAIL TAKEOVER	0.9900	1.0	0.9950	26514
CHEATING BY IMPERSONATION	0.9076	0.7951	0.8476	21668
CHILD PORNOGRAPHY/CHILD SEXUAL ABUSE MATERIAL (CSAM)	0.9872	1.0	0.9935	24471
CRYPTOCURRENCY CRIME	0.9896	1.0	0.9947	26615
CYBER BULLYING/STALKING/SEXTING	0.8959	0.7533	0.8184	21596
CYBER TERRORISM	0.9980	1.0	0.9990	21682
DAMAGE TO COMPUTER SYSTEMS	0.9994	1.0	0.9997	21633
DATA BREACHES.	0.9613	0.9808	0.9710	22055
DEBIT/CREDIT CARD FRAUD	0.8248	0.8457	0.8351	26446

DEFACEMENT/HACKING	0.9835	0.9725	0.9779	43756
DEMAT/DEPOSITORY FRAUD	0.9513	0.9789	0.9649	26342
DENIAL OF SERVICE (DOS) AND DISTRIBUTED DENIAL OF SERVICE (DDOS) ATTACKS.	0.9749	0.9913	0.9831	22055
E-WALLET RELATED FRAUDS	0.9085	0.8903	0.8993	26410
EMAIL HACKING	0.9836	0.9971	0.9903	21832
EMAIL PHISHING	0.9963	1.0	0.9981	21757
FAKE/IMPERSONATING PROFILE	0.8990	0.8920	0.8955	21578
FRAUD CALL/VISHING	0.7601	0.8004	0.7797	26353
IMPERSONATING EMAIL	0.9999	1.0	0.9999	22055
INTERNET BANKING-RELATED FRAUD	0.8484	0.8083	0.8279	26657
INTIMIDATING EMAIL	1.0	1.0	1.0	22055
MALWARE ATTACKS.	0.9802	0.9902	0.9852	22055
ONLINE CYBER TRAFFICKING	0.9933	1.0	0.9966	21795
ONLINE GAMBLING/BETTING FRAUD	0.9865	1.0	0.9932	26441
ONLINE JOB FRAUD	0.9111	0.9936	0.9505	21754
ONLINE MATRIMONIAL FRAUD	0.9976	1.0	0.9988	21084
PROFILE HACKING/IDENTITY THEFT	0.9244	0.9202	0.9223	21630
PROVOCATIVE SPEECH OF UNLAWFUL ACTS	0.9825	0.9977	0.9900	20944
RANSOMWARE	0.9836	0.9782	0.9809	44110
RAPE/GANG RAPE-SEXUALLY ABUSIVE CONTENT	0.9995	0.9976	0.9985	24791
SALE, PUBLISHING AND TRANSMITTING OBSCENE MATERIAL/SEXUALLY EXPLICIT MATERIAL	0.7954	0.9357	0.8599	49017
TAMPERING WITH COMPUTER SOURCE	0.9857	0.9716	0.9786	22055

DOCUMENTS				
UPI-RELATED FRAUDS	0.7242	0.7130	0.7186	26370
UNAUTHORIZED ACCESS/DATA BREACH	0.9214	0.9813	0.9504	21614

Category results table:

CATEGORY	**PRECISION**	**RECALL**	**F1-SCORE**	**SUPPORT**
FINANCIAL FRAUD CRIMES	0.98	0.98	0.98	47,530
OTHER CYBER CRIME	0.99	0.99	0.99	113,173
WOMEN/CHILD RELATED CRIME	0.98	0.97	0.98	19,901
ACCURACY			**0.98**	**180,604**
MACRO AVG	0.98	0.98	0.98	180,604
WEIGHTED AVG	0.98	0.98	0.98	180,604

5. Challenges Faced

1. Handling Messy Text
The dataset presented a significant challenge with its raw and unstructured text, which included:

- **Typos and Inconsistencies**: Frequent typographical errors and varying phrasing made it difficult to standardize descriptions across similar cases.
- **Abbreviations and Slang**: The use of abbreviations and informal language required careful handling to avoid loss of meaning during preprocessing.
- **Special Characters and Unwanted Symbols**: These were removed using regular expressions, although some instances required further fine-tuning to retain meaningful content.
- **Ambiguous Terminology**: Certain terms and phrases were open to interpretation, which added complexity to subcategory classification. Techniques like normalization and tokenization were used to create a more uniform data structure, allowing the model to interpret the input text more reliably.

2. Managing Imbalanced Data
The dataset was highly imbalanced, with some categories and subcategories underrepresented. This imbalance posed two main issues:

- **Biased Model Predictions**: The model tended to favor the majority classes, reducing its accuracy for underrepresented subcategories. To address this, we employed **class weighting** and **sampling techniques**. For example, we used **oversampling** of minority classes to increase their representation in the training set and **undersampling** of dominant classes to balance the data.
- **Challenge in Precision-Recall Balance**: Imbalanced data often leads to a trade-off between precision and recall, particularly for minor subcategories. To address this, we focused on tuning hyperparameters to achieve a more balanced F1-score, ensuring fair representation across all subcategories.

3. Handling Null Values
While the dataset contained mostly complete records, some fields were missing, impacting the model's learning process. We addressed this by:

- **Removing Rows with Significant Missing Data**: For cases where missing values significantly affected description quality, we excluded those rows from the dataset.
- **Imputation for Minor Null Cases**: For minor missing values, we used simple imputation techniques to fill in missing data, particularly for critical fields required by the model.

4. Model-Related Challenges
Due to the size and complexity of the dataset, additional challenges arose related to model training and performance optimization:

- **Overfitting**: Given the high variability in text, overfitting was a concern, especially with models like XGBoost. To mitigate this, we used techniques such as **early stopping**, **regularization** (L2 regularization), and **cross-validation**. These methods helped prevent the model from memorizing training data, improving its generalizability.
- **Optimization for Large Datasets**: The dataset's large size posed constraints on computational resources and processing time. We tackled this by:
    - **Batch Training**: Breaking down the training process into manageable batches to reduce memory load.
    - **Dimensionality Reduction**: Applying TF-IDF selectively and capping the maximum features to prevent the feature space from becoming too large, which optimized both memory usage and model efficiency.

6. Conclusion

Summary of Accomplishments
In this project, we developed an NLP model capable of accurately classifying crime-related text data into specific subcategories based on raw, unstructured input. Through extensive preprocessing, feature engineering, and experimentation with various machine learning techniques, our final model (XGBoost) achieved strong performance despite significant data challenges, such as imbalance and high variability in text descriptions. The model demonstrated effective handling of complex and noisy data, showing robust accuracy across categories and subcategories.

Key Learnings
This project reinforced the importance of:

- **Thorough Data Preprocessing**: Managing unstructured and messy text data through normalization, tokenization, and regex proved essential to obtaining reliable results.
- **Class Imbalance Solutions**: Addressing data imbalance with sampling and class weighting contributed to fairer, more accurate classification, even for underrepresented subcategories.
- **Optimization for Large Datasets**: Efficient handling of large datasets was crucial for maintaining computational performance while preventing overfitting.

Impact and Real-World Potential
The developed model is a step toward enhancing the National Cyber Crime Reporting Portal (NCRP) by supporting citizens in filing accurate, well-categorized cybercrime reports. By guiding users in real-time, this model could help streamline the reporting process and facilitate more responsive action from authorities. The model's framework can also be adapted to similar tasks involving classification of unstructured data, making it a valuable asset in other government and legal applications.

Future Directions
To further improve model performance and adaptability, we propose:

- **Incorporating Advanced NLP Models**: Future iterations could integrate more complex NLP architectures, such as fine-tuned BERT or other transformer-based models, to enhance contextual understanding.
- **Expanding Feature Engineering**: Exploring additional feature engineering techniques, like entity recognition or sentiment analysis, could improve the model's ability to discern nuanced differences in crime descriptions.

- **Exploring Real-Time Deployment**: To maximize impact, the model could be optimized for deployment as a real-time classification service, with updates based on newly reported cases to keep classification criteria current.

Closing Statement
This project highlights the feasibility and utility of machine learning in addressing societal needs through intelligent automation. With continued refinements, our model holds the potential to improve the NCRP's operational efficiency and better support citizens in reporting cybercrimes, aligning with broader goals for data-driven governance and citizen engagement.

7. References

Data Processing:

- NumPy: Oliphant, T. E. (2006). A guide to NumPy. T. E. Oliphant.
- Pandas: McKinney, W. (2011). pandas: a foundational library for data analysis and manipulation. Python for Data Analysis.
- Scikit-learn: Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
- NLTK: Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python. O'Reilly Media, Inc.

Model Training:

- XGBoost: Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.
- Scikit-learn: Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Data Balancing:

- Scikit-learn Resampling: Scikit-learn documentation. (n.d.). Resampling strategies in scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/classes.html#module-sklearn.utils

Exploratory Data Analysis (EDA):

- Matplotlib: Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95.
- Seaborn: Waskom, M. L., et al. (2020). Seaborn: statistical data visualization. Journal of Open Source Software, 5(51), 2411.

8. Appendices

Appendix A: Data Preprocessing Steps

- Detailed outline of preprocessing techniques, including:
    - **Normalization Techniques**: Explanation of how text normalization was applied and examples of typical transformations.
    - **Tokenization Details**: Information on the tokenization approach, such as word-based or subword tokenization.
    - **Regex Patterns**: Common regular expressions used to clean unwanted symbols and examples of text transformations.
    - **Handling Null Values**: Documentation of methods used to handle missing values.

Appendix B: Exploratory Data Analysis (EDA) Results

- **Category and Subcategory Distributions**: Visualizations or tables showing the imbalance among categories and subcategories.
- **Word Frequency Analysis**: Summary statistics on the most common words and phrases across different subcategories.
- **Data Imbalance Visuals**: Plots depicting the distribution of data across classes to illustrate the degree of imbalance.

Appendix C: Model Hyperparameters and Tuning

- **Hyperparameter Tuning Logs**: Detailed listing of parameters tested for XGBoost, including learning rate, maximum depth, and other key settings.
- **Best Parameter Configuration**: The final set of hyperparameters chosen for the primary model, along with rationale for each choice.
- **Grid Search or Cross-Validation Details**: Information on the validation techniques used for selecting optimal parameters.

Appendix D: Evaluation Metrics and Confusion Matrices

- **Confusion Matrices for Key Categories**: Confusion matrices displaying model performance across major subcategories.
- **Precision, Recall, and F1-Score Metrics**: Detailed performance metrics for each category and subcategory, offering insight into model accuracy and potential areas for improvement.

Appendix E: Code Snippets

- **Key Functions for Preprocessing**: Code snippets for custom functions used in data preprocessing, tokenization, and text cleaning.
- **Feature Engineering with TF-IDF**: Sample code illustrating how TF-IDF was applied to convert text into feature vectors.
- **Model Training Script**: Essential portions of the code used for training and tuning the XGBoost model.

Appendix F: Additional Experiments and Observations

- **Comparison of Model Performances**: Summary tables showing how different models (e.g., Naive Bayes, SVM, BERT) performed relative to XGBoost.
- **Insights from Failed Approaches**: Brief discussion of models or methods tested that did not yield significant improvements, along with reasons for their limitations.