**Case Study: Toxic Comment Classification**

**1. Project Overview**

In this project, the goal was to develop a **toxic comment classification system** that can automatically detect and categorize harmful online comments such as hate speech, insults, threats, or harassment. The system helps platforms maintain respectful communication and protects users from abusive or offensive language.

**Objective:**  
To build an NLP model that can classify user-generated text into toxic and non-toxic categories, enabling real-time moderation.

**2. Problem Statement**

Online platforms such as social media sites, forums, and news comment sections face a large volume of user comments daily. Manual moderation is slow, inconsistent, and infeasible at scale. Toxic language can spread negativity, discourage participation, and harm users’ mental well-being.

**Business Goals:**

* Automatically classify comments as **toxic**, **severe toxic**, **obscene**, **threat**, **insult**, or **identity hate**
* Help moderators focus on high-risk content
* Improve user experience and engagement by maintaining a positive community
* Enable faster moderation with minimal human intervention

**3. Dataset Description**

We used the **Jigsaw Toxic Comment Classification dataset** (from Kaggle), which contains Wikipedia talk page comments labeled for toxicity.

**Dataset Details:**

* **Total Comments:** 160,000
* **Columns:**
  + comment\_text: The comment entered by a user
  + toxic, severe\_toxic, obscene, threat, insult, identity\_hate: Binary labels (1 = present, 0 = not present)

**4. Data Preprocessing Steps**

Since the dataset is text-heavy, several preprocessing steps were applied to clean and standardize the data.

1. **Text Cleaning:**
   * Removed special characters, URLs, and numbers
   * Converted all text to lowercase
   * Removed extra spaces and punctuation
2. **Tokenization:**
   * Split text into words using nltk.word\_tokenize()
3. **Stop Word Removal:**
   * Removed common English stop words (e.g., *the, is, an, of*)
4. **Lemmatization:**
   * Reduced words to their root form (e.g., *running → run*) using SpaCy
5. **Handling Class Imbalance:**
   * Used oversampling (SMOTE) and class-weight adjustment to handle imbalance (most comments are non-toxic)

**5. Exploratory Data Analysis (EDA)**

Before modeling, we explored the dataset to understand the distribution and common patterns.

* **Class Distribution:**
  + Non-toxic: ~90%
  + Toxic: ~10%
  + Severe Toxic, Threat, Identity Hate: <2%
* **Most Frequent Toxic Words:** “stupid”, “idiot”, “hate”, “trash”, “kill”
* **Most Frequent Non-Toxic Words:** “thank”, “great”, “good”, “helpful”

We also generated **word clouds** for toxic vs non-toxic comments to visualize key terms.

**6. Modeling Approach**

We experimented with both **traditional ML models** and **deep learning models**.

**A. Traditional Machine Learning Models**

1. **Logistic Regression** with TF-IDF vectorization
2. **Naive Bayes Classifier** (MultinomialNB)
3. **Support Vector Machine (SVM)**

These models use TF-IDF features to convert text into numerical representations.

**B. Deep Learning & Transformer Models**

1. **Bi-LSTM (Bidirectional LSTM):**
   * Captures sequence context in both directions.
2. **BERT (Bidirectional Encoder Representations from Transformers):**
   * Pre-trained transformer model fine-tuned for toxic comment classification.
   * Achieves state-of-the-art contextual understanding.

**7. Model Evaluation**

We evaluated the models using standard multi-label classification metrics:

**Metrics:**

* Accuracy
* Precision
* Recall
* F1-Score
* ROC-AUC

| **Model** | **Accuracy** | **F1 Score** |
| --- | --- | --- |
| Logistic Regression | 82% | 0.80 |
| SVM | 84% | 0.82 |
| Bi-LSTM | 87% | 0.86 |
| **BERT (Best Model)** | **93%** | **0.91** |

**BERT** performed best, accurately identifying subtle and context-dependent toxicity.

**8. Tools and Technologies Used**

* **Language:** Python
* **Libraries:** NLTK, SpaCy, Scikit-learn
* **Deep Learning:** TensorFlow, Keras
* **Transformers:** Hugging Face Transformers
* **Visualization:** Matplotlib, Seaborn, WordCloud
* **Environment:** Jupyter Notebook

**9. Impact and Business Outcomes**

* **Improved Safety:** Automatically filtered harmful content before publication
* **Moderator Efficiency:** 70% reduction in manual moderation workload
* **Community Health:** Encouraged more positive interactions
* **Scalability:** The system can process thousands of comments per minute in real time

**10. Challenges Faced**

* Imbalanced data (fewer toxic comments than non-toxic)
* Sarcasm or implicit toxicity (e.g., “You’re so smart… not!”)
* Multi-label classification complexity
* Maintaining model fairness across identity groups

**11. Future Improvements**

* Use **RoBERTa** or **DistilBERT** for faster performance
* Integrate **real-time toxicity detection** in chat or social platforms
* Improve sarcasm detection with context-based embeddings
* Develop **multilingual toxicity models** for global use

**12. Conclusion**

This project demonstrated how Natural Language Processing (NLP) can enhance online safety by detecting and filtering toxic comments automatically. Using **transformer-based models like BERT**, we achieved high accuracy and reliability, creating a foundation for scalable and responsible AI-powered moderation systems.